IN-63
007218

Fuzzy Adaptive Control for Intelligent Autonomous Space Exploration Problems

Augustine O. Esogbue, Ph. D. and Principal Investigator
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332-0205
e-mail: aesogbue@isye.gatech.edu

Completion Report
National Aeronautics and Space Administration Grant NAG 9-726
Initiated: March 1, 1994,
Completed: November 30, 1998

Project Summary

The principal objective of the research reported here is the re-design, analysis and optimization of our newly developed neural network fuzzy adaptive controller model for complex processes capable of learning fuzzy control rules using process data and improving its control through on-line adaption. The learned improvement is according to a performance objective function that provides evaluative feedback; this performance objective is broadly defined to meet long-range goals over time.

Although fuzzy control had proven effective for complex, nonlinear, imprecisely-defined processes for which standard models and controls are either inefficient, impractical or cannot be derived, the state of the art prior to our work showed that procedures for deriving fuzzy control, however, were mostly ad hoc heuristics. The learning ability of neural networks was exploited to systematically derive fuzzy control and permit on-line adaption and in the process optimize control. The operation of neural networks integrates very naturally with fuzzy logic.

The neural networks which were designed and tested using simulation software and simulated data, followed by realistic industrial data were reconfigured for application on several platforms as well as for the employment of improved algorithms. The statistical procedures of the learning process were investigated and evaluated with standard statistical procedures (such as ANOVA, graphical analysis of residuals, etc.). The computational advantage of dynamic programming-like methods of optimal control was used to permit on-line fuzzy adaptive control. Tests for the consistency, completeness and interaction of the control rules were applied. Comparisons to other methods and controllers were made so as to identify the major advantages of the resulting controller model.

Several specific modifications and extensions were made to the original controller. Additional modifications and explorations have been proposed for further study. Some of these are in progress in our laboratory while others await additional support. All of these enhancements will improve the attractiveness of the controller as an effective tool for the on line control of an array of complex process environments.

Based on identified research activities and projects at NASA, we established the following research objectives of interest to the ongoing activities relative to efficient application of fuzzy logic, neural networks, and optimization to autonomous orbital operations at JSC but specifically control of tethered satellites.

We sought to contribute to the improvement of accurate interpretation of sensor and other measurements for deriving accurate state information prior to optimal control through the design of optimal clustering and classifier algorithms. This is of fundamental importance in the design of computationally efficient fuzzy logic and neural networks. We achieved this by developing a fuzzy clustering algorithm based on fuzzy criteria which were then compared with the performance of other existing clustering methods.

Another area of contribution was in the design of computationally efficient neural networks for space systems applications through the development of efficient dynamic programming-like methods and algorithms for the efficient implementation of reinforcement learning in adaptive neural networks. This was achieved by in different fronts. The first was the development of a new type of fuzzy dynamic programming which we called fuzzy criteria dynamic programming as opposed to the classical cost dependent criteria algorithms. The second is the effort to imbue our neuro fuzzy adaptive controller with learning capabilities based on the temporal difference (TD) and Q-Learning algorithms. Each of these results tantamounts to improved optimization approaches for our first generation learning controller.

The new improved controller was then employed to simulate the performance of two important real world problems of interest or similar to those encountered in space systems. The first is the tethered satellite problem drawn from the various high fidelity simulations at JSC by the Software Technology Group (STG). We note that the tethered satellite system retrieval problem was visited as a problem of central interest to NASA for which inadequate and inefficient control methodologies had been used due to the complexities of the system. The literature showed that fuzzy logic controllers could be effective for these types of problems. Our neuro-fuzzy adaptive controller offers significant advantages over both classical and fuzzy controllers for this problem. The second problem area visited was the power systems stabilization problem faced by various real world environments including NASA. In each of the foregoing application problem areas, the performance of our controller was demonstrated. Areas of superiority, as well as those in need of improvements and extensions, were identified.

In summary, each of the proposed goals was achieved and exceeded in instances during the duration of the project. A major outcome is the reconfiguration, development, testing, and application of the statistical fuzzy associative learning controller, SFAL-C, to an array of sample problems, both prototypical and novel. An overview of the controller design, components and performance is given in Chapter 2. In Chapter 3, we provide a more detailed account of the controller and its properties as extracted from Chapter 3 of precursor project funded by the National Science Foundation. Modifications and requisite extensions to the controller are presented in Chapter 4. In Chapter 5, we discuss the development of efficient clustering methods. In Chapter 6, we outline the development of a new fuzzy dynamic programming methodology which has many ramifications for control in general and the project in particular. In Chapter 7, we consider two problems of import to NASA and applied the controller to these problems. In particular, the problems of tethered satellite system retrieval and power system stabilization are con-

sidered in greater detail.

KEY WORDS: Fuzzy Logic, Fuzzy Control, Neural Networks, Intelligent Control, Learning Algorithms, Approximate Dynamic Programming, Fuzzy Criterion Set Dynamic Programming, Inverted Pendulum, Tethered Satellite System, Fuzzy Criterion Clustering.

Author

AUGUSTINE O. ESOGBUE is a Professor and Director of the Intelligent Systems and Controls Laboratory in the School of Industrial and Systems Engineeering at the Georgia Institute of Technology. He was the Principal Investigator for this project and the author of this report. His background and expertise are operations research and intelligent control with emphasis on computational intelligence, modeling, optimization-especially dynamic programming. Professor Esogbue is Associate Editor for the Journal of Mathematical Analysis and Applications, Journal of Computational Mathematics and Applications, Journal of Advanced Computational Intelligence and a Founding Advisory Editor of Journal of Fuzzy Sets and Systems.

Acknowledgments

The contributions of several graduate and undergraduate students including James Murrell, Warren Hearnes, and Qiang Song who worked on various aspects of this project that are reported here, under funding provided principally by the National Science Foundation under Grant No. ECS 91216004 and the National Aeronautics and Space Administration Grant NAG 9-726 are gratefully acknowledged. Production assistance was also provided by Warren Hearnes.

Contents

List of Figures

Chapter 1

Introduction, Research Objectives and Major Results

1.1 Overview of the Literature Leading to the Project

Control and optimal decision-making in various systems: industrial, economic, biological, ecological, social and space exploration systems require modeling and analysis techniques that are effective for large, complex, often nonlinear and imprecisely defined systems. The analytical and statistical methods that have worked well in the "hard" sciences are often difficult to apply to these complex systems. In the "soft" sciences of these systems, where life forms or human intelligence are involved, interactions and relationships are key. Living systems, particularly the human brain, have provided the inspiration for many human-made tools useful in engineering with these complex systems.

The field of artificial intelligence (AI) is a class of such tools inspired by human processing. The most effective processing that humans do is not with numeric quantities but with concepts and symbols. AI systems manipulate "symbols" or abstract objects according to rules and relationships. Many useful systems have been engineered with this type of processing, including expert systems for decision-making and control for complex operations and processes. Unlike most human processing, however, AI systems tend to be rigidly rule-based and inflexible. Also, while precise quantities are less important with AI, the objects are very "hard and crisp". For example, an object is either red or it is not, or a proposition is either true or it is not. Despite the utility of AI in certain instances, there are situations where adaptability and reasoning with less precise concepts are required.

Humans are very poor at working with precise numerical values, precise concepts, or tedious sequential procedures. Compared to very mechanistic machines like von Neumann computers, humans are slow and imprecise.

At NASA, and in connection with the quest for cost effective and human friendly engineered systems, the need to depart from classical, hard and rigid design and control approaches has become considerably transparent. The need to design and operationalize cost effective integrated human/machine interfaces and environments has led to increased interest in telerobotics, artificial intelligence, and systems analysis techniques in various aspects of space exploration and navigation. Several groups concerned with these problems have emerged in various NASA locations. One such group at NASA's Johnson Space Center (SPC) is the Software Technology Branch (STB).

One of the main concerns of the STB is the development of intelligent control systems for space systems and robotics. Among the problem areas of interest are various issues in connection with autonomous orbital operation. The principal modeling and analysis technique used is fuzzy logic and neural networks. The problems are plagued with considerable complexities, nonlinearities, uncertain dynamics particularly of the imprecise variety, and time varying phenomena. The appeal of fuzzy logic is perhaps best summarized thus:

> ... it is this fuzzy ... logic that plays a basic role in ... one of the most important facets of human thinking, namely, the ability to summarize information–to extract from the collections of masses of data impinging on the human brain only those sub collections which are relevant to the performance of the task at hand. ... a summary is an approximation to what it summarizes. For many purposes, a very approximate characterization of a collection of data is sufficient because most of the basic tasks performed by humans do not require a high degree of precision in their execution. The human brain takes advantage of this tolerance for imprecision by encoding the 'task-relevant' (or 'decision-relevant') information into labels of fuzzy sets which bear an approximate relation to the primary data. In this way, the stream of information reaching the brain via the visual, auditory, tactile and other senses is eventually reduced to the trickle that is needed to perform a specified task with a minimal degree of precision. (Zadeh 1973, p. 28-29.)

What is needed is the development of systems which combine the speed and learning capability of neural networks with the ability to aggregate and reason with imprecision that fuzzy logic offers.

> Developing a methodology for integrating the computational paradigms of fuzzy expert systems and neural networks is of paramount importance. Harnessing their combined power in a hybrid system, which allows for imprecise information and/or uncertain environments, yields a system more powerful than either system standing alone. (Kandel, Schneider, Langholz 1991, p. 98)

While considerable success has been reported by the STB of JSC in dealing with these problems, much remains to be done. The answers to the following poignant questions posed by Lea and Jani (1992,

pp. 152-153) must be affirmatively provided before complete success can be achieved:

1. Is it possible to create control systems that do not require a high degree of redesign when systems configurations change or operating environments differ?

2. Can a fuzzy controller be used as a high-level controller to function with classical controllers in a way the human would?

3. How easy or difficult is it to design and implement a fuzzy rule base that will control a complex system as opposed to developing a classical control system to solve the same problem?

4. Particular questions of interest to NASA are, Where can hardware implementations be utilized advantageously, and how easy or difficult is it to transfer rule bases to hardware?

Our thesis is that the above can be aided by an adroit wedding of techniques of optimization, control theory, fuzzy logic and neural networks, and of course, computer science in the development of such systems. Thus, we proposed a work mission which followed this line of pursuit.

1.2 Background and Significance of the Problem

In this section, the suitability of fuzzy logic methods for the control of complex processes such as those in space exploration is discussed. We suggested that the developments most needed to increase the applicability and effectiveness of these methods could be achieved by an adroit integration of fuzzy logic and neural networks and optimization. In the sequel, we briefly review the state of the art at the beginning of our project.

1.2.1 Overview of Research in Fuzzy Control

A number of industrial processes and operations involve systems that are highly nonlinear, have a large number of interacting variables, or much of what is known about their modeling and control is imprecise or expressible mostly in linguistic terms. It is now generally agreed that the well-developed methods of control theory are not applicable because these systems are usually too imprecisely known to develop an adequate mathematical model, and even if a model could be obtained, there are no general results for controlling nonlinear systems (see, for example, discussions in Gupta 1989, p. 3019; Lee 1990a, p. 411, 432).

Yet, experienced human operators are often able to control such processes (even when they know little about the plant dynamics) by using imprecise heuristic rules. These heuristics have often been modeled successfully by "knowledge-based" or "expert" systems which consist of a set of linguistic statements or rules which specify what control actions are to be taken in response to process behavior (Patrikar and

Provence 1990, p. 359). A human decision-maker aggregates volumes of numerical data into linguistic labels that make the partition of the data space more manageable for processing (Gupta 1990, p. 3019). The imprecision of the knowledge that humans employ is effectively implemented by "if-then" rules using these linguistic labels and fuzzy logic. This methodology has been effectively used in many industrial control problems, such as blast furnace, cement kiln and steel plant processing control; robot vision and movement control; flight control (see Lee 1990a, Sugeno 1985 for many references therein).

Fuzzy control rules may sometimes be obtained by observing the actions of the human controller and asking questions about his or her procedures (Lee 1990a, pp. 411-412). In such cases, the resulting rules need to be checked for consistency and accuracy; the resulting control may still be far from optimal. In general, determining fuzzy control rules is even more involved. Many practical situations involve several human operators, a mixture of human and machine control, or no human operators at all. In designing controllers for these more complex systems, the knowledge engineer would like to capture the experience, knowledge and cognitive skill of human experts, requiring lengthy study and perhaps interviews and questionnaires (Gupta 1989, p. 3019; Lee 1990a, p. 411). These procedures are subjective, ad hoc, and require several cycles of trial, evaluation, revision and retrial (Lee 1990a, pp. 411-412). Sometimes it is simply not feasible or possible to derive the rules this way.

Determining the membership functions for the fuzzy categories also often involves the skill of experts in a "cut and trial" process, and "... this inevitably becomes a bottleneck of system design (Takagi 1990)".

More efficient and systematic methods for knowledge acquisition and fuzzy controller synthesis are needed (Lee 1990a). Adaptive fuzzy controllers capable of learning from process data as well as incorporating linguistic data could address this need and offer significant advantages. An adaptive fuzzy controller could automatically generate a set of linguistic rules and improve on them as the control process evolves (Patrikar and Provence 1990, p. III-359; Procyk and Mamdani 1979, p. 15), without interruption of current service (Bezdek 1992, p. 102).

There have been some results on adaptive fuzzy controllers, known as linguistic self-organizing controllers. Procyk and Mamdani (1979) pioneered this line of work and others have published results based on their model. Other work on "self-tuning" improvement for fuzzy controllers has been presented (e.g., Maeda 1990). Xu and Lu (1987) and Araki, et al (1991) developed adaptive rule-modification procedures for system identification. However, these methods "... suffer from some drawbacks such as high computation time, high memory storage requirements and complex rule modification procedures (Patrikar and Provence 1990, p. III-359)." Takagi and Sugeno (1985) use linear models and fuzzy parameters to perform system identification of existing human control, but their system is not adaptive. These methods do not employ neural networks, rather they are attempts to make sequential, rule-based algorithms incorporate adaptive or learning capability, which has proven to be difficult.

Neural networks have been proposed as a means to implement fuzzy controllers (Rocha 1991; Gupta and Gorzalczany 1991; Yager 1991; Czogala 1991; Gupta and Pedrycz 1989; Gupta 1989). In these studies, the linguistic labels for process state and control variables are presumed already given, and it is the "if-then" associations between these which are learned. The learning requires that all or some of the rules are already obtained by other methods, since the aim in these studies was to express knowledge derived otherwise. In Patrikar and Provence (1990), control rules are learned when none are known a priori, but a set of performance evaluation rules must be pre-established. A self-organizing neural network controller of limited applicability is reported by Nakanishi (1990). Such work (e.g., Yamaoka and Mukaidono 1991) uses feed-forward networks with pre-determined fuzzy sets to identify the fuzzy rules relation from process input-output pairs, but from such a model the control must yet be derived. All of these methods use the back-propagation learning algorithm, which is a supervised scheme and is notoriously slow. Other research has focused on feed forward networks to learn membership functions. These are supervised learning methods requiring a priori knowledge about the fuzzy sets in order to construct the training sets. Yamaguchi (1991) used an unsupervised neural network to learn membership functions directly from process data, but the number of rules and their form are pre-established and limited.

There have been some results with reinforcement learning. Berenji (1992) has reported a modified back propagation network that uses reinforcement learning in a controller which refines pre-established rules. Lee (1990) extended the work of Barto and Sutton (1983) to a fuzzy reinforcement learning scheme with adaptive critic; this work used predefined membership functions for the states and the training essentially adjusted the location but not the shape or spread of the control membership functions.

One generally-applicable adaptive fuzzy controller which learns both membership functions and an arbitrary set of control rules "from scratch" is due to Jang (1992) who reported a fuzzy controller which uses back propagation feed forward networks to learn both the fuzzy membership functions and the fuzzy rules with no a priori knowledge. However, the fact that it is based on error back-propagation means that very rich information in the form of errors as feedback rather than simply a performance measure as feedback is required for learning.

1.2.2 Neural Networks to Implement Learning

Neural networks have been used successfully in many applications to provide learning capability. A neural network is a system of simple interconnected components called neurons, each of which computes for its single output y, a generally nonlinear function g of the weighted sum of its inputs x_i, $i = 1, \ldots, p$. Each weight w_i is the strength of connection from the input source to the neuron. These weights encode the information contained in a neural network using various activation functions (see Rumelhart 1986).

Neural networks are not programmed with a list of instructions to be executed sequentially like

ordinary computers. Nor does the learning take place as in some AI methods where a collection of explicit rules are formulated using rules that adapt other rules. Learning neural networks are designed, like the human brain, to learn to perform a task by being taught with a series of examples and then to generalize to new instances. It is in this sense that the network can be said to have learned the rules and it "appears in hindsight as though the model did indeed know those rules (Khanna 1990, p. 7)." The series of examples of desired input-output relations examples make up the training set, which together with a proper learning algorithm allow the desired information to be encoded in the network (Alspector 1989, p. 30). Thus, neural systems are especially useful for problems where a compact algorithmic description does not exist but this set of data does, as is the case with most of the difficult problems of artificial intelligence (Lippman 1987, p. 4).

This learning "extracts correlation" between the inputs and outputs presented in the training set, finding the significant statistical regularities and ignoring random deviations, making it robust and insensitive to noise (Alspector 1989, p. 30). Learning can be viewed as a statistical problem in parametric estimation, where the model parameters are the connection weights (White 1989). Statistical considerations, such as the "design of experiments" problem in determining the training data set, and validity of extrapolation, are important issues in designing learning systems. The differences with ordinary statistical models are that neural networks function with massively parallel computation, are adaptive not static, and the meaning of the parameters (weights) is quite different from the usual. Recursive statistical algorithms can be adaptive but do not possess the other advantages of neural networks:

> ... neural networks also provide a greater degree of robustness or fault tolerance than conventional von Neumann sequential machines. Damage to a few neurons or connections, and minor variabilities in the characteristics of neurons, do not impair the overall performance significantly. Furthermore, neural networks also possess the ability to gracefully handle inconsistent knowledge from different experts, or some degree of contamination in incoming data, without too much degradation in its performance. (Kandel, Schneider and Langholz 1991, p. 98)

There are three basic types of learning. In supervised learning, each output of the input-output pairs of the training set represents a value that a "teacher" knows to be the correct response to a given input. In unsupervised learning, the data used to train the network does not include instruction on what the correct output should be for a given input. In learning with reinforcement, the network receives feedback from a "critic" that gives it a measure of its performance, but not the "correct answer" from the teacher.

Although many types of control strategies have been implemented with neural networks (see Werbos 1990b, 1991), the focus of this research is to explore ways in which the learning capabilities of neural networks can be used to enhance fuzzy control strategies.

1.2.3 Neural Network and Fuzzy Logic Interface

The manner in which neural networks represent information in the pattern of its weights and the consequent activation of its neurons is quite compatible with fuzzy logic. Neural networks are applicable to problems

> ... where incoming information is often slightly inaccurate or incomplete and a choice must be made from many alternatives. ... Each of the many possible solutions to a problem can be thought of as a true-or-false proposition. ... When a conventional [algorithm] solves the problem, it goes through each proposition one by one and determines whether that proposition is true. ... A neural net making a decision doesn't consider whether these individual propositions are strictly true or false. Instead, each proposition has weight, which might be characterized as the strength of the network's 'opinion' as to whether it is true or false. (Allman 1989, pp. 94-95)

The activation of a neuron can be viewed as an indication of its degree of confidence that an associated feature is present, or as the degree to which a proposition is true, rather than merely providing a yes or no indication. Alternatively, the activation may represent the proportion of a feature that is present.

> Much work on fuzzy set theory can be related to both these interpretations of activation. ... The key elements of cognition are not numbers, but labels of fuzzy sets (i.e., classes of objects in which the transition from membership to non membership is gradual rather than abrupt). This, essentially is what the notion of 'activation' is designed to capture. (Khanna 1990, p. 13)

Fuzzy logic provides a means for linking the symbolic processing of linguistic con-structs and qualitative relationships with numeric computations using precise, detailed algorithmic manipulations of quantitative information, both essential to real-world tasks (Pao 1989, p. 223). In the area of fuzzy control, neural networks are the instruments for implementing this linkage in two key ways (Takagi 1990, p. 14). First, the member-ship function gives a measure of compatibility of a numeric description with a qualitative entity, such as the linguistic object "process state x is negative big." The activation of a neuron can represent the fuzzy degree to which such a linguistic concept is true. Or, it can give a measure of the possibility that the concept corresponds to a particular set of numeric data (Pao 1989, p.23). A neural network can be trained to synthesize membership functions representing a particular linguistic con-cept. This method might be used to construct membership functions that are not as arbitrary and subjective. Second, the learning of fuzzy control rules is a situation where a complete set of rules is not available a priori, but a suitably chosen set of process data can be used to train the network to generate those rules.

Neural networks have been proposed as a means to implement fuzzy controllers (Rocha 1991; Gupta and Gorzalczany 1991; Glorennec 1991; Yager 1991; Czogala 1991; Iwata 1990; Horikawa 1990; Gupta and Pedrycz 1989; Gupta 1989). In these studies, the linguistic labels for process state and control variables are presumed already given, and it is the "if-then" associations between these which are learned. The learning requires that all or some of the rules are already obtained by other methods, since the aim in these studies was to express knowledge derived otherwise. In Patrikar and Provence (1990), control rules are learned when none are known a priori, but a set of performance evaluation rules must be pre-established. A self-organizing neural network controller of limited applicability is reported by Nakanishi (1990). Recent work (e.g., Yamaoka and Mukaidono 1991) uses feed-forward networks with pre-determined fuzzy sets to identify the fuzzy rules relation from process input-output pairs, but from such a model the control must yet be derived. All of these methods use the back-propagation learning algorithm, which is a supervised scheme and is notoriously slow. Other research (e.g., Takagi and Hayashi 1988; Furuya 1988) has focused on feed forward networks to learn membership functions. These are supervised learning methods requiring a priori knowledge about the fuzzy sets in order to construct the training sets. Yamaguchi (1991) used an unsupervised neural network to learn membership functions directly from process data, but the number of rules and their form are pre-established.

Although these results are useful and impressive, more research is needed in developing fuzzy controllers that learn unsupervised from experience the necessary controls using neural networks. It should be possible for a controller to generate effective control rules and membership functions using process data but little or no prior information.

"Learning in uncertain or unknown environments, particularly autonomous or unsupervised learning, is an essential component for any intelligent system. The ability to garner new information, process it, and increase the understanding and capability of the system is crucial to the performance of autonomous intelligent systems. Such systems should be able to learn in an unsupervised situation by experimentation, classification, and recognition of similarity, and to generalize and apply appropriate previous solutions or hypothesize new solutions to situations never before encountered by the system." (Kandel, Schneider and Langholz 1991, p. 98)

1.2.4 Overview of Fuzzy Logic Applications to Research in Space Exploration

In recent times, a considerable amount of attention and theoretical results in fuzzy control and expert systems have been directed to problems in space exploration. The reason for this interest is simple. Automation through fuzzy control is particularly well suited to this domain primarily because human involvement is currently essential in carrying out even "basic" mission functions, such as spacecraft control and data acquisition/analysis. Research into space-related applications of fuzzy controllers and expert systems in large part has two functions: i) as an aid in developing control systems that emulate human

reasoning patterns in order to minimize or eliminate the need for human participation in certain tasks, and ii) enhancing the performance of traditionally automated systems. The following selected discussion of some aspects of recent research relative to applications of fuzzy control and expert systems to space exploration is intended to provide some insight into this bristling research area, and provide appropriate background to the research activities and results reported here.

Vachtsevanos and Davey (1987) proposed the use of a fuzzy controller to locate, diagnose and correct component faults in a proposed space station thermal control system. This intelligent controller works in concert with a classical controller of the system's fluid flow in order to: i) maximize sensitivity to failure detection, while simultaneously ii) minimizing the rate of failure detection false alarms of all components comprising the thermal control system. Diagnosing a fault in a component requires the accumulation (through parity space representations, analytic redundancy and limit checking) of symptoms by the intelligent controller. The fuzzy rule base and a compositional rule of inference provide degrees of fault severity depending on the prevailing system conditions. A similar approach is taken in developing the algorithm that actually identifies the faulty component. Monte Carlo simulations on a system of 100 components demonstrate that faults are correctly isolated, but the required computation time increases considerably along with the order of the system fault.

In Lea (1988a and 1988b) a discussion of efforts to develop fuzzy expert systems to control spacecraft during unmanned missions is provided. Using established results from previous Space Shuttle-related research, the goal was to develop an effective piloting system that automatically controls fuel consumption and plume effects. The chosen membership functions are the pi and S functions because they can easily reflect varying degrees of "fuzziness" through parameter modifications and hence model a wide variety of pilots-from those who must always keep the target dead center to those who merely need to maintain visual contact with the target. Improvements were required in the fuzzy decision rules and scope of the prototype system, which only deals with proximity operations and not all phases of rendezvous operations. Consideration is given to using a fuzzy computer chip to alleviate computation, which would be particularly useful in developing rotational and translational controllers for spacecraft. Several mission profiles were to be simulated to determine requirements for sensor accuracy, system redundancy and propellants.

Additional efforts by NASA to develop fuzzy control for an autonomous navigation system that would eliminate a current data filtering problem were also provided by Lea (1988). During mission phases when the Shuttle is beyond rendezvous radar range, the star tracker (ST) is used to track a target. Angles calculated by the star tracker are used to update the navigation system, but since the ST is equally capable of tracking all bright objects, tracking false targets (e.g. stars and debris) is not uncommon. To prevent such errors, the crew edits the star tracker data before passing it on to the filtering system using, in effect, a fuzzy decision rule. Hence, an expert system is developed and is found to perform as

well as crew members. This system is then expanded to periodically examine data during ST tracking intervals and either notifies the crew or updates data using fuzzy weighing factors depending on how suspect the data is. The problem of random switching between Inertial Measurement Units (IMU's) is also examined. It appears that successful and encouraging results using fuzzy control have already been obtained by this team of researchers. Future work was to include designing an expert system (ONEX) to perform the majority of tasks currently required by engineers situated at ground control consoles.

Villareal and Shelton (1990) describe the implementation of a Space-Time Neural Network (STNN) arising from the need for a model that can automatically associate spatial information with its appropriate position in time. The network is created by replacing the single synaptic weight between two nodes of a standard backpropogation neural network with several weights representing temporal dependencies as well as association. The network is composed of two or more layers of interconnected nodes and buffered by sigmoid transfer nodes at the intermediate input and output. Network performance is based on simulations of the classic XOR problem, unsolvable by a simple two-layer network and the learning and modeling the dynamics of a chaotic system. Future research was proposed using the STNN to model more complicated dynamical systems applicable to adaptive control.

Lea, et. al. (1992) proposed using the STNN for detecting the "skiprope" phenomenon-oscillation of a tether and its payload due to the Earth's magnetic field. The STNN filters gyroscopic data from the payload to detect and predict the magnitude, phase and (x,y) coordinates of the tether's midpoint during these oscillations. Experimental results show that the STNN's accuracy in predicting skiprope amplitudes varies considerably, and does not perform well in predicting skiprope phases. However, the STNN did perform well in predicting the tether coordinates. Although current techniques using Kalman filters are not particularly accurate, there is no question of their validity, unlike the STNN which has not seen earlier applications and requires very time-consuming, detailed verification. The next goal was to configure and train the STNN with a data set and then evaluate its performance.

Continuing their research in tethered satellite control, Lea et. al. (1992) proposed a fuzzy controller to monitor tether lengths. A fuzzy rule base and membership functions to command an electric reel were defined and compiled to computer source code using fuzzy development software. The controller was evaluated by both a massless tether simulation and a finite element model, which represents the tether as beads, each assigned a mass, strung together with springs. In the massless case, the fuzzy controller exhibited better control than a conventional controller, eliminating length error spikes. Additional criteria were used in evaluating the fuzzy controller with the bead model simulation, particularly in-plane and out-of-plane liberation amplitudes. If these become too great while retrieving a payload, a phenomenon known as "wrap around" can occur, which could seriously endanger the shuttle and its crew. The fuzzy controller demonstrated its superiority over the conventional controller as it reduced the liberation amplitudes by more than half. However, because the dynamics of a tethered system are so complex,

neither fuzzy nor conventional controllers can eliminate the "skiprope" effect, and manual maneuvers with thrusters must be used. Future work was to include training fuzzy logic/neural network based control system on past data that could possibly damp and control "skiprope" activity during a mission.

Karr, Freeman and Meredith (1990) improved the performance of a docking controller using a genetic algorithm to determine performance-enhancing membership functions. Berenji, Jani and Lea (1991) applied the approximate reasoning based intelligent control architecture (ARIC) in the development of a Space Shuttle attitude controller. The two primary components of the ARIC model are the Action-state Evaluation Network (AEN) and the Action Selection Network (ASN). The AEN constantly critiques the actions recommended by the AEN and attempts to predict reinforcements associated with the various input states. The ASN includes a fuzzy controller, composed of a fuzzifier, rule base, decision making logic and defuzzifier. The controller output is the correcting torque required from the thrusters. Examining the full learning capabilities of the ARIC architecture in this context will require further study, but the initial results are said to be very positive. It is expected that this approach will be adapted for camera tracking systems, trajectory control for the Mars rover and tether control systems. We note that the ARIC, though a reinforcement learning controller, still uses backpropogation schemes in both the AEN and ASN networks.

Lea and Jani (1992) provided a detailed and didactic overview of past, present and future activities of the Software Technology Branch at the Johnson Space Center relating to fuzzy control. Past accomplishments included fuzzy control systems for sensor data processing, translational, rotational and attitude control of spacecraft and a fuzzy-based concept for a camera tracking system (See Lea, et. al. (1992)). Efforts were in progress to combine into one controller the existing translational and rotational controllers. Test plans were in the preliminary stages, with details such as initial conditions still to be defined. Considerable effort was also being devoted to the development of pattern recognition and object identification algorithms with the intent of extending the capabilities of the camera tracking system into image processing. Also under investigation was the fuzzy control of the Mars rover during sample collecting missions. To collect samples, the rover must travel between destinations among obstacles that prior to the actual excursion cannot be identified. Fuzzy control and planning provides the rover with the capability to identify hazards from imprecise sensor measurements of obstacle size and distance, as well as taking "evasive action" as hazards present themselves since communication times between Mars and Earth based control are extremely long and could jeopardize the success of the mission. The controller receives a target point and uses position errors in the x and y directions in conjunction with orientation error to command the rover in terms of steering angle and velocity from a fuzzy rule base of 112 rules. The rover is driven towards the x-axis of a control error frame while commanded to the proper orientation and slowly driven towards the target. While several test cases have yielded accurate control, it is believed control of the rover can be improved even further by altering the membership functions corresponding

to inputs and outputs. Future projects at the JSC include applying the aforementioned camera tracking system to traffic management around a space station, incorporating the reinforcement learning technique to the translational control of a spacecraft during rendezvous sequences and the development of a fuzzy rule base and membership functions for a fuzzy controller for the living quarters of the space station Freedom. (Health Monitoring System for Environmental and Life Support System for Large Volume Crew Quarters)

A fuzzy tracking controller which commands the camera gimble drive pan and tilt rates utilizing range and horizontal and vertical position data as input is discussed by Lea, et. al. (1992). Range data is fuzzified and used in conjunction with five-rule base to determine the scale factor, which is then defuzzified into a crisp value. Reported results indicated excellent controller performance with the tracked object maintained in the center of the field of view (FOV). Future goals were to: i) modify the membership functions to maintain the object in a narrow range, ii) incorporate range data directly into the fuzzy rule base, thus eliminating the intermediate scale factor, iii) implement angular velocity rules, iv) implement centroid algorithms, and v) transfer the rule base to a fuzzy chip that will interface directly with the camera motor drives.

NASA's growing interest in fuzzy control and artificial intelligence research is further evidenced by the organization's hosting of conferences, such as the NASA GSFC Fuzzy Logic Workshop and the Goddard Conference on Space Applications of Artificial Intelligence.

1.3 Research Project Objectives

From our survey of the literature dealing with the applicability of fuzzy logic and neural networks in Space Exploration and Navigation, as well as conversations with the Software Technology Branch of NASA's Lyndon B. Johnson Space Center in Houston, primarily Dr. Villareal, it appeared that several opportunities existed where beneficial use could be made of various aspects of our work including, but not restricted to, our new adaptive fuzzy neural network controller.

The goal of our proposed effort was therefore multidimensional. Over the years, we had embarked on a dynamic research mission and developed an array of modeling, optimization, and control techniques with a generality of applications. We have tested them, with considerable success, in various arena particularly in socio-technical systems such as medical decision making and water resources and environmental pollution. We were desirous of applying them to relevant in space exploration. From our review of the applications of a subset of our techniques in space exploration (sect. 1.2.5) there appeared to be a gamut of reasonably well defined problems facing NASA that have been successfully attacked by, as well as others that are susceptible to, the tools at our disposal. Further, and quite important, we had identified at least a group of researchers at the Software Technology Branch of the Johnson Space Center (JSC) interested in our

expertise as well as willing to work with us. Our goal was then to develop this relationship further in order to gain an articulate understanding of NASA's problems of interest, sharpen our tools with a view to applying them cost-effectively to NASA's problems, and then become a contributing member of NASA's space exploration efforts on a substantial basis. In view of the foregoing, we established the following research objectives of interest to the ongoing activities relative to efficient application of fuzzy logic, neural networks, and optimization to autonomous orbital operations at JSC but specifically control of tethered satellites:

1. Improve accurate interpretation of sensor and other measurements for deriving accurate state information prior to optimal control through the design of optimal clustering and classifier algorithms. This is of fundamental importance in the design of computationally efficient fuzzy logic and neural networks.

2. Contribute to the design of computationally efficient neural networks for space systems applications through the development of efficient dynamic programming-like methods and algorithms for the efficient implementation of reinforcement learning in adaptive neural networks.

3. Apply the algorithms developed from 1.4.1. and 1.4.2. above to some data and, if possible, from the various high fidelity simulations at JSC by the Software Technology Group (STG). Compare them with other clustering research using, if possible, similar data from tethered satellite system.

4. Apply the new neural network under development in our laboratory to test bed data from the STC-JSC simulations and compared to other neural network simulations, possibly Lea and Villareal, et. al. 1992.

Each of the foregoing goals was achieved and exceeded in instances during the duration of the project. A major outcome is the development, testing, and application of the statistical fuzzy associative learning controller, SFAL-C, to an array of sample problems, both prototypical and novel. An overview of the controller design, components and performance is given in Chapter 2. In Chapter 3, we provide a more detailed account of the controller and its properties as extracted from Chapter 5 of J. A. Murrell's doctoral thesis which was funded in part by this project and supervised by the PI. Modifications and requisite extensions to the controller are presented in Chapter 4. In Chapter 5, we discuss the development of efficient clustering methods. In Chapter 6, we outline the development of a new fuzzy dynamic programming methodology. In Chapter 7, we consider two problems of import to NASA and applied the controller to these problems. In particular, the problems of tethered satellite system retrieval and power system stabilization are considered in greater detail.

1.4 Resume of Principal Research Results

Prior to discussing the essential components of the Controller vis--vis the stated objectives above, let us summarize the salient properties and contributions of this research project. This controller has several unique features mentioned earlier which we summarize here. It is an adaptive controller which is well suited to the on line control of complex processes. In particular, it has been shown to be capable of learning effective control using process data and improving its control through on-line adaption.

The controller performs a fuzzy discretization of the state and control spaces and learns the fuzzy relations for these fuzzy subsets using various learning schema including a variation of the TD method with its dynamic programming inspirations. Other more sophisticated and improved learning protocols are in progress in our laboratory. While it adapts both the membership functions and the control rule state-control association, the controller primarily learns the control rule associations, unlike many other methods which fix the rules and adjust the membership functions. Most important, it does so for the entire state space. Additionally, no training data sets nor any error signals derived from knowledge of the desired plant trajectory are needed.

The properties of the controller have also been rigorously tested via well designed statistical experiments. This self-learning controller has been validated and successfully applied to a number of classic problems including the inverted pendulum problem, the DC servomotor position control problem, the switching problem in a distributed communications network, and the power system stabilization problem. Comparisons of the functioning of the controller as well as its effectiveness with those of other controllers, particularly those of the fuzzy or neuro-fuzzy variety , were made both theoretically and via some test problems. See for example the communications network problem of Chapter II.

To repeat the summarization presented in Murrell(1994), one of the principal developments of this research includes a feasible and practical tool for the application of fuzzy generalization to the discrete rules or associations of action -response utilized in reinforcement learning methods. Another important and novel characteristic of the controller is the manner in which it integrates the information derived from the fuzzy associative reinforcement learning process into the adaption mechanism for the fuzzy discretization pre-processing transformation. The resultant effect is that the location of the fuzzy clusters is both a function of the distribution of the states as well as the regions in the state space which are most significant in determining the appropriate controls. Clearly, the importance of the location of these fuzzy clusters in the optimal development of the controller can not be over-emphasized and hence the role of optimal fuzzy clustering techniques in future enhancements of the performance of the controller. For work in this arena which is beyond the scope of this project, see the Appendix where the developments of Esogbue and Liu(1996) can be found.

1.5 Complexity Analysis of Controller Algorithms

To determine the size of the problem, one must consider the number of state space dimensions p and the number of control space dimensions q. Recall that the size of the spaces must be used in deciding the number of fuzzy subsets utilized in the fuzzy discretization of these spaces. As a consequence, the size of the problem in terms of the complexity of the controller algorithms, must also be expressed in terms of the number s of state space terms (i.e., the number of state space nodes N_i) and the number r of control terms.

The state space node map approach permits increasing the dimension of the input state vectors without an exponential increase in controller complexity. We note that other fuzzy discretization schemes generate a complete set of fuzzy subset terms for each dimension of the state, thereby greatly increasing the number of fuzzy rules as the dimension increases. In general, the usual number of rules is $l^P \bullet m^q$ with l and l the number of state terms per dimension and the number of control terms per dimension respectively. In the proposed method, the number of state terms s is the number of map nodes. This is ,in general, selected according to the size of the state space p so that every dimension of the state space can be adequately covered. However, s need not be set in such a way that it grows exponentially with p, since we can elect to cover the state space more sparsely than with a full factorial lattice. Since fuzzification is of the state space vectors rather than merely within each state dimension separately, the generalizing and interpolating characteristics of the fuzzy inference can exploit any smoothness or redundancy in information in all the dimensions simultaneously. Also, the adaptation of the location vectors and spreads tends to move the coverage of the membership function to regions where it is most needed in the state space. The number of rules is determined by the product sr, which can be chosen to grow with p and q more slowly than exponentially. Whether the number of nodes needs to be increased as the dimension increases is a matter of discretion involving a trade-off between the precision of the fuzzification and an increase in the number of nodes. In discussing the size of a problem , we must also consider the number of input data points. For on-line control, there is no fixed number of input data points, rather there is a potentially infinite sequence of data. However, there are some types of algorithms, such as recursive least squares or some types of adaptive clustering algorithms which would require either the complete past history or the history for some period into the past to be stored and used in the computations performed at each time step as a new data point becomes available. The usual clustering algorithms, for example, would require a pass through a set of past data for each additional point, greatly adding to the complexity of the computations as time advanced. Incremental parametric algorithms, including many neural network algorithms and the algorithms proposed here, do not have this drawback. Let us now discuss the complexity of the computation for one time step of the SEAL controller.

We begin with the SFDN algorithms. Computation of a_i requires s sets of computations, one for each

node; each of these includes operations that must be performed on each state dimension, i.e., p times. For one step of the SFDN algorithms, the number of computations is bounded above by $Ms + Nsp + L$ (for some integer l, M, and N), and thus the order of complexity is $O(sp)$.

In the SLFCN algorithm, while in principle, the quantities γ_{ij} require, sr computations, the simplified algorithm utilized by Murrell(1994) with γ_{ij} as an indicator,needs only r computations. On the average, even this can be made smaller. Obtaining all the b_j similarly requires sr, or simply r computations, depending on whether or not γ_{ij} indicates more than one node N_i. The c_{ij} require sr, or actually $s'(k) \bullet r$ computations, where again $s'(k)$ decreases with time down from s. The SLFCN therefore requires $Mr + Nsr + L$ operations (L, M, N some integers, different from above),and as such has complexity $O(sr)$.

The IFCN requires only a few arithmetic operations performed sr times to obtain the g_{ij} from the c_{ij}, and sr computations to obtain $\boldsymbol{b} = \boldsymbol{a}^T \boldsymbol{G}$ leading to a complexity of $O(sr)$. The CAN similarly needs only r sets of simple arithmetic operations, while the PES needs only 1 small set of arithmetic operations per time step.

Thus, one iteration of the entire controller algorithm has a complexity of $O(sp + sr)$. Let us next address the problem of the number of iterations that may be required.

To learn the control law as a fuzzy relation matrix, requires that s pieces of information be learned or estimated. This is accomplished by a stochastic search of r^s combinations which could be of formidable complexity. However, the algorithm does not try each combination one at a time. Each SFDN node is a learning unit which explores only r possibilities, in parallel sequences of learning trials with all the other nodes. If the reinforcement signals were based on performance scores that were absolutely accurate and certain, then in principle, each control choice for each state need be visited exactly once, hence requiring only sr iterations of the controller algorithm. However, since the information is uncertain in a *Statistical* as well as a fuzzy sense, some multiple of sr is required to obtain an adequate sample of information. A conservative analysis shows that , with a probability greater than 0.98, the controller can generate a complete set of performance predictions in approximately $4r$ plant runs.

1.6 Research Extensions for Application to Space Systems

1.6.1 Efficient Clustering Algorithms

The importance of cluster analysis as a tool in pattern recognition is well recognized. Basically, we may view the task in hand as that of dividing a set of K data points into N clusters in an optimal fashion where the number N may be a preassigned integer. Clustering can be done both classically and via fuzzy set theory. The latter generally recognizes the following two problem classes of interest: The first one is to group fuzzy data points into some fuzzy sets. The other is to divide the crisp data points into a specified number of subsets which need not be fuzzy but utilizing fuzzy set theoretic methods in developing the

clusters.

Our focus is on fuzzy clustering whose literature in recent years has grown increasingly vast. Perhaps, the earliest reference to fuzzy cluster analysis may be traced to Bellman *et al.* [4] and Ruspini [62]. According to Yang [73], the studies of cluster analysis employing fuzzy set theory can be divided into three categories: fuzzy clustering based on fuzzy relation, fuzzy clustering based on objective function, and the fuzzy generalized k-nearest neighbor rule. The first one, fuzzy clustering based on fuzzy relation, was first proposed by Tamura *et al.* [68]. They presented a multi-step procedure by using the composition of fuzzy relations beginning with a reflexive and symmetric relation. The second and more interesting to us is fuzzy clustering based on objective function. This approach is best illustrated via the method proposed by Dunn [20] and generalized by Bezdek [7].

In the sequel, the use of the concepts of fuzzy prototype as opposed to crisp prototype as well as fuzzy criterion for optimal clustering was presented. Numerical experiments show that fuzzy criterion clustering based on fuzzy prototypes can overcome the effect of noise points. Its effectiveness was demonstrated for these and other studies reported elsewhere. The validity issue was also addressed.

This method which we have termed fuzzy criterion clustering has the capability of accurately classifying and detecting typical clusters including circles, ellipses and various shapes that have posed problems for some well known algorithms reported elsewhere in the literature. Additionally, it is robust and appealing to practitioners because of its user driven fuzzy criteria clustering objective function. Applications to the clustering of real world data arising from water pollution control studies as well as MRIs in cardiac sequence detection experiments are in progress in our laboratory.

1.6.2 Efficient Dynamic Programming Methods

Fuzzy dynamic programming is a powerful control and analysis apparatus which seeks to extend classical dynamic programming to many real life situations characterized by uncertainty, especially of the imprecise and ambiguous variety. Penetrating reviews of the developments in the field of fuzzy dynamic programming as well as an insightful discussion of possible extensions are provided by Esogbue and Bellman [23] and recently by Esogbue and Kacprzyk [27]. A particularly interesting generalization involves decision situations in which the decision, constraints, goals, and system dynamics are all fuzzy as given is treated by Baldwin and Pilsworth [2]. The details of these concepts and proofs are provided in Liu and Esogbue [55]. We present the framework for fuzzy criterion set and fuzzy criterion dynamic programming which is a general tool for dealing with many decision and control situations arising in many fields including the stochastic reservoir operation and stochastic inventory control models of operations research and engineering. Specifically, the objective is to maximize the expected fuzzy criterion function of the product of fuzzy criterion sets. We outline existence, uniqueness and stability theorems of the derived solutions to this model whose resultant optimal control is a bounded critical number policy under the

usual regular hypothesis assumptions.

1.6.3 Application: Tethered Satellite System

The tethered satellite system problem represents a highly nonlinear control problem with five state variables, which is considerably more than the usual test problems found in the literature. A tethered system is any two or more bodies connected by a long thin structure. The system focused on in this example is the deployment, station-keeping, and retrieval of a target satellite from the Space Shuttle. With a fixed-length tether for systems in the 'station-keeping' phase, the equations of motion are still complex. With a variable length tether—i.e., for systems in the deployment or retrieval phase—the equations of motion are further complicated by time-varying coefficients. We examine the tethered satellite system retrieval problem in more detail in the sequel.

1.6.4 Application: Power System Stabilization

With the lack of simulation data from STC-JSC, we elected to apply our control algorithm to a problem that may be of interest to NASA as well as one that illustrates the controllers properties and effectiveness. One of the most intriguing and frequently investigated problem areas for deploying potent and novel tools of control engineering is the power system stabilization problem. Many different control strategies as well as controllers have been tested on this problem. Part of this interest is engendered by both the challenge and intractability of power systems which are characterized by the existence of power inherently complex, nonlinear, time-varying and indeterminable elements, simple controllers which work well in one situation may not perform equally well in another. As part of the experimental investigations with the SFAL controller, we explored its ability to learn a robust control law to stabilize the power system under various operating conditions.

Chapter 2

The Neuro-Fuzzy Adaptive Controller: An Overview

2.1 Introduction

Solution to control problems in which there is considerable uncertainty or lack of knowledge about the process is the focal concern of our research mission. The class of problems used as the leitmotif of our work may be subsumed under the rubric of what is generally known as the general set-point regulator problem. This group encompasses the usual dynamical system plants as well as discrete state space problems which arise in the control of operations and manufacturing. Our work has focused on the development and demonstration of the potential of the proposed Statistical Fuzzy Associative Learning Controller in simulated applications to three representatives of this class of problems, namely, process control of particular second-order dynamical systems, message routing in communication networks and the power system stabilization and control problem . In each, there are complexities and uncertainties which call for intelligence, but intelligence which can be automated for speed and repeatability.

Classical analytical and optimization approaches generally require some type of model of the plant and specific knowledge of what constitutes desirable plant behavior. For cases where the lack of knowledge is of a high order, a variety of intelligent methods have been suggested to provide more acceptable performance than can be achieved with the classical methods. In our review of existing approaches(Esogbue and Murrell 1994), it was shown that many of them consist of off-line empirical modeling using a training set of data. There are many instances in which these approaches work well for control of uncertain systems. However, a training set generally requires some knowledge of what constitutes good control, and in uncertain systems, this is often lacking. Also, many systems are based on an explicit gradient-type of approach, such as the well-known back propagation neural networks with their attendant drawbacks.

It is also noteworthy that some of these problems are framed in terms of discrete spaces, do not have objective functions which are analytically differentiable, or any known objective function formula at all. Hence, gradient methods which proliferate in the literature may not be applicable.

An alternative approach to control which has had notable success is that of fuzzy controllers, which mathematically mimic the imprecise reasoning processes which are an important part of what makes humans so successful at solving problems with large uncertainties. However, the problem has been to find effective ways to capture or acquire the knowledge or skill required for a particular problem. Recent research has been devoted to ways of integrating learning algorithms with fuzzy control to automate the knowledge acquisition. Most of the attempts to combine the power of fuzzy reasoning with learning capability have taken either of two approaches. One approach has been to implement fuzzy reasoning with AI-type rule-based methods. The resulting controller models can become quite complicated and cumbersome as they are scaled up to larger systems. The second approach is to use gradient-type neural networks combined with fuzzy logic. Although these have been used successfully in some applications, they share the limitations of gradient methods.

It is now believed that learning systems (in the sense of mathematical learning theory) may be the most appropriate for the problems with a high order of uncertainty. A reinforcement learning trial system can learn on-line from its own experience with the process it is designed to control. Although such systems have been suggested since the 1960's, their applicability has been significantly inhibited by the almost exponential growth in complexity as it is scaled up to larger systems due to the necessity of discretizing the spaces. Combining these methods with fuzzy discretization and fuzzy logic can make learning trial methods a viable, practical approach. However, until the research first reported in Esogbue and Murrell(1993), and then furthered by Murrell(1994) and Esogbue, Hearnes and Song (1995), very little development of this idea has been seen in the literature.

These problems are second-order linear and nonlinear dynamical systems of the type which often occur in industrial process control. In these problems, uncertainty about the true plant model is often introduced by the environment, or by frequent changes in configuration necessitated for example by flexible manufacturing practices. Additionally, unknown, time-varying and nonlinear dynamics complicate the use of the plant model required by classical control theory methods. These concerns, necessitate the pursuit of novel approaches of the sort discussed in the sequel.

2.2 The Self-Learning Fuzzy-Neuro Controller

In response to the quest for an intelligent controller that can learn online, does not require an exact model of the plant, operates without the benefit of what is considered optimal, does not need a set of predefined control rules, and uses feedback in an instructive way without the attendant expensive

computational costs, Esogbue and Murrell [24] reported the development of a self-learning fuzzy-neuro controller with unique features and capabilities. This controller consists of five parts: (1) the Statistical Fuzzy Discretization Network (SFDN) which employs a variation of the Kohonen self-organizing map (SOM) to fuzzify the state space of the plant; (2) the Fuzzy Correlation Network (FCN) which implements the learned fuzzy control rules as fuzzy relation; (3) the Stochastic Learning Correlation Network (SLCN) which maps a particular fuzzy state to a set of fuzzy control actions through an adaptive stochastic algorithm; (4) the Control Activation Network (CAN) which defuzzifies the fuzzy control to a crisp control signal; and (5) the Performance Evaluation System (PES) which provides feedback reinforcement signals to the learning algorithm based on the effectiveness of the control action. A block diagram of the controller is shown in Figure 2.1.

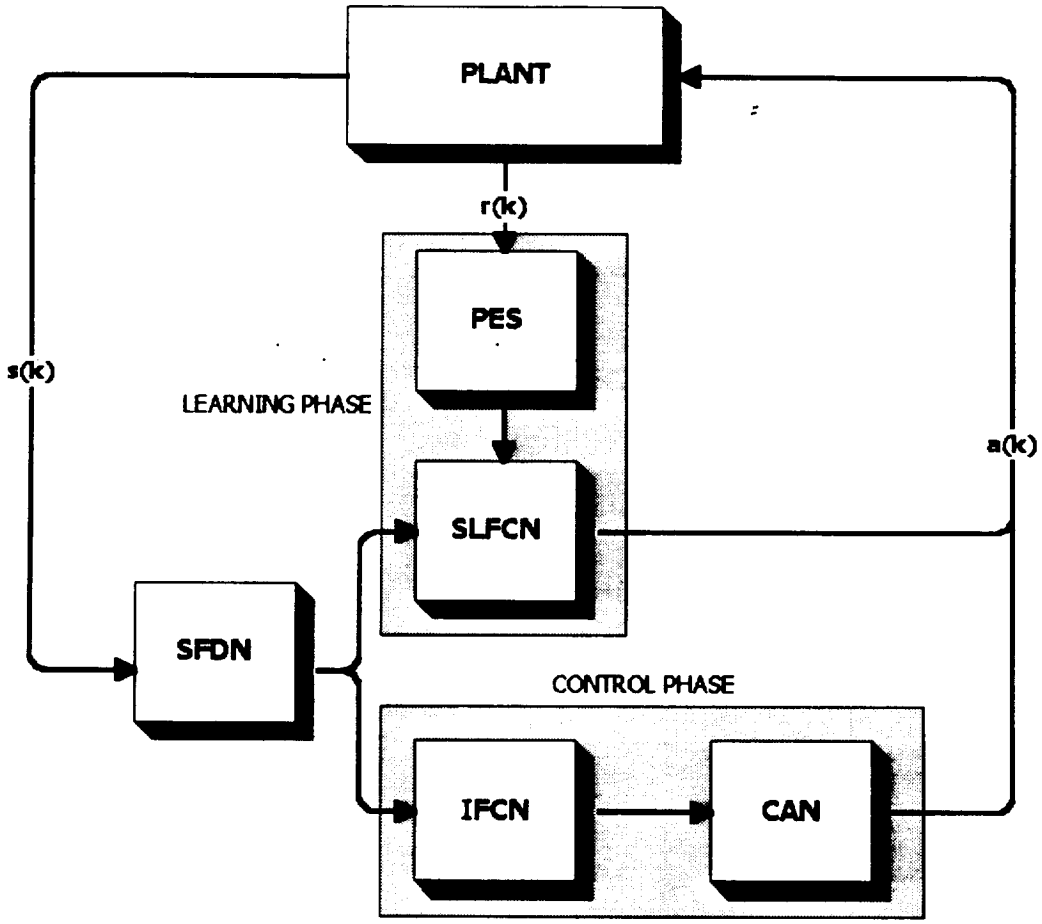

Figure 2.1: Block Diagram of Fuzzy-Neuro Controller

2.2.1 Statistical Fuzzy Discretization Network (SFDN)

This subsystem consists of a network of automata nodes ("neurons") arranged in a grid. Each node receives as its input the current process state vector. Every time a vector is input, each node computes an output that represents the degree of membership in the fuzzy subset of the input space that corresponds to that node. This output, called the node activation, is a measure of the degree of similarity of the input state vector to the ideal or prototype member of that fuzzy set. It is computed as some combination of the state vector and a location parameter vector associated with that node which represents the prototype process state for the corresponding fuzzy set. Also associated with each node is a parameter that encodes the degree of dispersion or spread of its fuzzy set membership function used to calculate the node activation. For a particular membership function form, the location parameters and spread parameter together define a fuzzy set. The SFDN thus performs a fuzzy discretization, inducing a fuzzy partition of the state space X into reference fuzzy subsets $X_1, X_2, \ldots, X_r$, each represented by a node in the grid.

The network described here is an extension of Kohonen's self-organizing feature map to fuzzy characterization of dynamic plant states. The output of the i^{th} node in the map is

$$a_i = \exp(-\| x - m_i \| / s_i) \tag{2.1}$$

where x is the state vector input, m_i is the vector of location parameters and s_i the spread parameter for the i^{th} node, and it is assumed that the choice of similarity measure is the Euclidean metric and the functional form of the membership functions is a gaussian function. Thus, the vector a of node activations is the fuzzy hyperstate due to input state vector x:

$$a = (\pi_{X_1}(x), \ldots, \pi_{X_r}(x))^T \tag{2.2}$$

where $\pi_{X_i}(\bullet)$ is the membership function for fuzzy set X_i given by the foregoing equation.

A sequence of state vectors is input to the network over time, and an adjustment algorithm adapts the location parameters to reflect the actual clustering of the state vectors by which the aggregation into fuzzy subsets is determined. A simplified version of the update rule of the j^th component of m_i for this example is

$$m_{ij,k+1} = \begin{cases} m_{ij,k} + \alpha_k(x_{j,k} - m_{ij,k}) & \text{for } i \in T_{c_k} \\ m_{ij,k} & \text{otherwise} \end{cases} \tag{2.3}$$

where k indexes the time step of the algorithm, T_{c_k} is a small neighborhood of nodes in the grid within a radius c_k around the node most activated by x_k, and α_k and c_k are decreasing functions of k. The basic concept for updating the spread parameter is given by

$$s_{i,k+1} = \begin{cases} s_{i,k} + \eta_k(|x_{j,k} - m_{ij,k}|^2 - s_{i,k}) & \text{for } i \in T_{c_k} \\ s_{i,k} & \text{otherwise} \end{cases} \tag{2.4}$$

where η_k is also a decreasing function of k.

The SFDN provides a means of aggregating similar plant states, thus permitting implementation of the control as a discrete relation. The adaptation update equations as initially configured are of the simple delta-rule type, which is more easily implemented in real time than clustering algorithms and permits the parallel distributed computation of neural networks.

2.2.2 Fuzzy Correlation Network (FCN)

The Fuzzy Correlation Network implements the fuzzy control rules as a fuzzy relation G (learned by the SLCN) which associates the collection of fuzzy sets $X_1, \ldots, X_r$ for input vectors $x \in X$ with the fuzzy sets $U_1, \ldots, U_s$ for the controls $u \in U$. This is accomplished with a fuzzy associative memory (FAM) or correlation network. The i^th node on one side represents the degree to which X_i has been selected, given by the SFDN node output a_i for state x. Each of these is linked to every node on the other side. The output b_j of the j^th node on the other side indicates the degree to which fuzzy output set U_j is the correct choice given the activations of the X_i's, or the "firing strength" of each rule for which that fuzzy control is the consequent. The connection weight parameter g_ij indicates the degree to which X_i relates to U_j. The control rule "If x is X_i then u is U_j" is represented by a strong link g_ij between node X_i and node U_j. The network connection weight matrix $G = \{g_{ij}\}$ specifies a fuzzy relation on $\{X_1, \ldots, X_r\} \times \{U_1, \ldots, U_s\}$. Thus, given input activation a, the fuzzy hyperstate, the fuzzy control vector is given by

$$b^T = \sigma(a^T G) \tag{2.5}$$

where σ is a vector-valued function whose components are $b_j = \sigma_j(a^T G\bullet, j), G\bullet, j$ is the j^th column of G, and each σ_j is some type of limiting function, (i.e., σ_i satisfies $\sigma_i(\alpha) \to 0$ as $\alpha \to -\infty$ and $\sigma_i(\alpha) \to 1$ as $\alpha \to \infty$), such as a sigmoid function. Thus, this implementation uses the product and limited-sum logic operators which is more easily implemented with a neural network associative memory than the common min-max logic.

2.2.3 Stochastic Learning Correlation Network (SLCN)

The purpose of this subsystem is to test and learn the effectiveness of pairing a particular control vector fuzzy set with each given state vector fuzzy set, using the performance evaluation provided by the Performance Evaluation System (PES), then use this knowledge base to generate the fuzzy control relation used by the FCN. Both the SLCN and the FCN receive as input the fuzzy hyperstate a output by the state map grid of the SFDN. The first phase of operation of the controller is a learning phase in which the fuzzy control vector b is generated by the SLCN. In the second phase of operation, the fuzzy relation learned by the SLCN is used by the FCN to generate b.

Initially, nothing is known about what control vector gives the best response when the process is in a given state. So, a fuzzy output is just picked and the performance measure indicates how good the

selection was. If it was not that good, the controller will be less inclined to pick that control again the next time the system enters that fuzzy state. If the selection was good, that control action is reinforced, so that it is more likely to be selected next time. The network that implements this is akin to Narendra's stochastic learning automata(SLA).

The SLCN consists of a matrix of nodes where each row corresponds to a particular fuzzy input state and each column to a particular fuzzy control action. The degree of activation of a node indicates the fuzzy degree to which it selects the control fuzzy subset to which it is assigned. Each node has a spread parameter h_ij for the i^th fuzzy state and the j^th fuzzy control. The location parameters λ_ij (scalar) are adjusted so that they always fall at the center of the spread function. In this case, a box-shaped function defined on a bounded interval is used. Thus, the output of the node for the i^th fuzzy state and the j^th fuzzy control is given by

$$c_{ij,t} = \begin{cases} 1 & \text{if} \lambda_{ij,t} - h_{ij,t} < \zeta_t < \lambda_{ij,t} + h_{ij,t} \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

$$\lambda_{ij,t} = \sum_{k=1}^{j-1} h_{ij,t} + \frac{1}{2} h_{ij,t} \tag{2.7}$$

where $h_{ij,t}$ and $\lambda_{ij,t}$ are location and spread parameters, respectively, at time t for node (i, j), and $\zeta_t \in [0, 1]$ is generated by a chaotic or pseudo-random process and serves as the input to the node. The node with the largest spread parameter is then the one that will have the maximum activation most often. The fuzzy control vector **b** is given by

$$b_{j,t} = \begin{cases} c_{ij,t} & \text{for } i = \text{ argmax } (a_i) \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

When i is the index of the most activated input fuzzy set, then the most activated node in the i^{th} row of the SLCN node matrix selects the control fuzzy set. The update algorithm for h_ij is given as

$$h_{ij,t+1} = (h_{ij,t} + r_t a_{i,t} b_{j,t} \gamma_{ij,t}) / (1 + \sum_{j^r t} a_{i,t} b_{j,t} \gamma_{ij,t}) \tag{2.9}$$

$$\gamma_{ij,t} = \begin{cases} h_{ij,t} & \text{if } r_t < 0 \\ 1 - h_{ij,t} & \text{if } r_t \geq 0 \end{cases} \tag{2.10}$$

where r_t is the reinforcement which is a function of the performance measure p_t, and $a_{i,t}$ and $b_{i,t}$ are the input and output activations, respectively, for the i^{th} fuzzy state and j^{th} fuzzy control at time t. The product $a_{i,t}\ b_{j,t}$ is the association or correlation between the state and control fuzzy sets. The quantities p_t and r_t are computed by the PES subsystem.

2.2.4 Control Activation Network (CAN)

The input to the Control Activation Network is the fuzzy control vector b. This fuzzy control is defuzzified to produce a crisp output quantity u, which is a vector for multivariable control processes. Each CAN node has its location parameter vector set to the desired control vector prototype levels. Its input is the

vector b, and its output is a crisp control vector u. The nodes can be set up as a map network to adapt the fuzzy sets according to the control vectors that are actually being output from the controller.

Using the max criterion defuzzification method, the B_i node with the largest activation (degree of truth) triggers the activation of the CAN node whose output is the prototype value $\bar{u}_j$ corresponding to the fuzzy set B_j. Alternatively, in the center of area method, u is calculated as the normalized weighted sum over the fuzzy sets B_i in which the weights are the activations b_j, given as

$$u = \left(\sum_{k=1}^{|U|} b_k s_k \bar{u}_k \right) / \sum_{k=1}^{|U|} b_k \tag{2.11}$$

where s_k are the spread parameters for the output fuzzy set membership functions. If the membership function form is symmetrical, then the effect of the spread is trivial.

2.2.5 Performance Evaluation System (PES)

The particular nature of the plant or process to be controlled and the characterization of the desired performance dictate the details of how the performance evaluation network is configured. When a performance measure is available which is an analytical function of the plant states or output, then the reinforcement signal r_t is simply a normalization of the performance measure p_t to lie in the interval [-1,1] or [0,1]. It is often the case, however, that complex processes which have no known well-defined plant model also do not have a well-defined formula for computing performance. Rather, there is a certain qualitative goal or objective to be reached, but it is not known what the values of the plant variables should be when that goal is reached. Even when a formula in terms of the variables is known, there is often an unknown delay between the control action taken and the effect on the plant variables, so that the result of the current control is not known until some future time. In such cases, various methods of estimating the performance evaluation function must be used. Our investigation into performance function estimation methods are reported elsewhere. In particular, we note the use of various dynamic programming-like algorithms such as the temporal difference (TD) algorithm by Murrell(1994) and both TD and Q-Learning by Esogbue et alia(1995).

The most straight-forward approach to the situation in which a qualitative determination of reaching the goal is given only after a period of many time intervals is described here. Reaching the goal is indicated by $p_t = 1$ (success) and reaching forbidden states (such as plant shutdown due to variables out-of-bounds) is indicated by -1 (failure). For each state entered at each time, a control action is taken. The average performance over time of this state-control pair is computed and updated whenever there is a success or failure. The reinforcement signal r_t can then simply be the current value of this average for the current state-control pair that just occurred. This method was used with success in the earliest version of this controller.

2.3 Implementation Issues for the Application of the Controller

In order to utilize this controller in its current configuration, the following conditions must be met:

1. Ability to characterize the process at any given time by its process state.
2. Ability of the control inputs to affect the sequence of input states.
3. Availability of a good performance measure which is related to the system goal
4. There must be some topological nearness measure (e.g., a metric) for both the state space $\mathbf{X}$ and control space $\mathbf{U}$ which additionally satisfies the smoothness assumptions listed below:
   (a) For every $\mathbf{x} \in \mathbf{X}$ there exists a control $\mathbf{u}^*$ such that $y(\mathbf{x}, \mathbf{u}^*) = \sup_{\mathbf{u} \in \mathbf{U}} \{y(\mathbf{x}, \mathbf{u}\}$.
   (b) If the optimal control for $\mathbf{x}_i$ is $\mathbf{u}^*$ then the optimal control for every $\mathbf{x}$ in a neighborhood of $\mathbf{x}_i$ is near $\mathbf{u}^*$.
5. There must be a direct relationship between the degree to which a control action contributes to the final computed control and the probability of that action being successful when applied purely.
6. The process is either intrinsically recurrent or can be repeatedly restarted at random initial states.

We note that the nearness structure of the spaces stipulated in the foregoing conditions 4 can be imposed or arrived at via some transformation of the original spaces. In particular, it is neither necessary for the smoothness to possess analytical differentiability property nor the nearness to be cast in terms of a metric which may be computed by a formula.

Let us finally outline the steps to be followed when trying to apply the controller to a system or process which has met the conditions stipulated in the foregoing. These are succinctly summarized in the flowchart for the algorithm in Fig. 2.2.

As usual, we begin with the set-up and initialization of the controller for the particular process as outlined below:

1. Set the state space and control space boundaries.
2. Set the state map size and initial location and spread parameters.
3. Set the control terms (control space map).
4. Set initial correlation parameters.
5. Determine the sampling interval for the process.
6. Determine process events to be used to index algorithm steps for each algorithm:

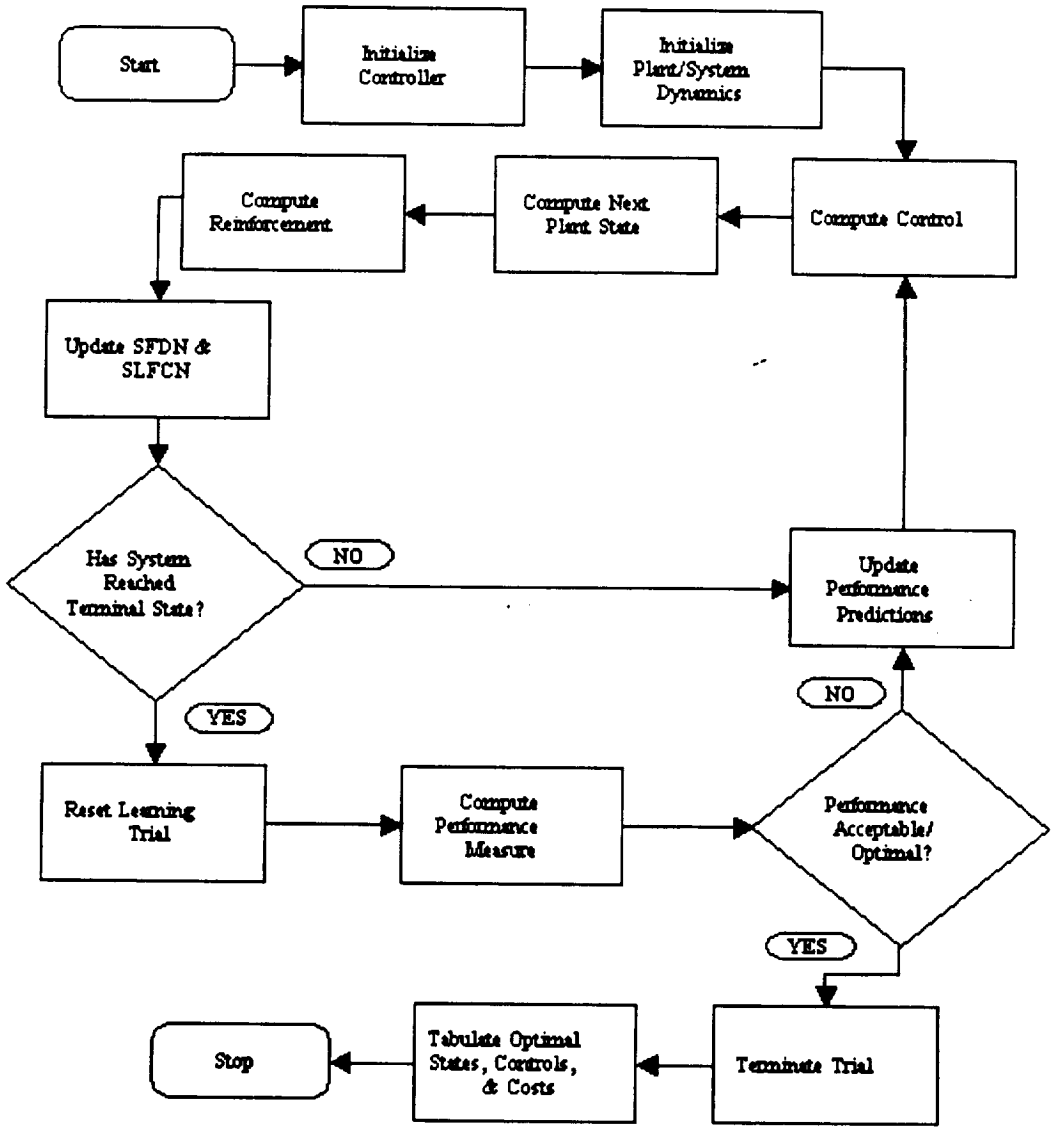

Figure 2.2: Flowchart for Implementation of Fuzzy-Neuro Controller

(a) Map node location update.

(b) Map node location neighborhood decay.

(c) Map node location update rate decay.

(d) Spreads update.

(e) Correlation parameter update.

(f) Correlation neighborhood decay.

(g) Performance evaluation update.

7. Set controller learning parameters.

8. Determine a normalized performance measure to be computed at terminal states.

As in most algorithms, it is beneficial to have some basic knowledge or information about the process such as its state and control variables as well their ranges. Using this knowledge, the state and control space limits used by the controller are determined. Determining the number of nodes, the size of the crisp neighborhoods, and the basic level of fuzziness or dispersion for the membership functions requires some reasoning about what level of precision is appropriate for the process. The size of the state space region to be covered and what level of resolution is required to resolve the control switching surfaces must be considered. Ability to partition both the state and control spaces so that they can be fuzzy discretized is essential. This relates to the issues of continuity and nearness discussed above. According to condition 4b the control behavior for a locality in the state space must be generalizable throughout the neighborhood.

When no prior information is available, the map node locations are set at the points of a uniformly spaced grid. If it is impractical to fill all the points of the grid, then design of experiments methods can be used to determine a good subset of the full factorial coverage. The nodes can be placed with greater density in regions where more coverage is needed if there is any prior information. Similarly, if there is no prior information about the control relation, then one can set the correlation parameter vectors to be uniform probability distributions. Any prior information can be encoded by biasing the distribution vectors towards favored control actions.

The operation then consists of repeated cycles of computing controls to input to the plant, plant state transitions, computing reinforcements, and performing learning updates. Whenever a terminal state is reached, such as a goal state or set-point of the problem, then a performance measure is computed and provided to the controller Performance Evaluation System in order to update its performance prediction algorithm. The system is reset—i.e., a new initial state is determined and any algorithm counters, etc., are reset. The operation terminates if it is determined that the attained performance is acceptable or near optimal.

As great as the properties of the controller are, we must emphasize that it is neither suitable nor optimal for all control environments. In general, the most ideal problem situations for applying the controller are those in which there exist a high tolerance for mistakes to be made for a brief period early in the learning phase or which allow for initial learning with a simulated on-line system.

2.4 Controller Application and Performance on Sample Problems

2.4.1 Application to Inverted Pendulum Problem

A classic testbed problem for nonlinear controllers is the inverted pendulum problem. One of the simplest inherently unstable systems known, yet it has a broad base for comparison throughout the literature. The system is shown in the following Figure 2.3.

Model of Process

Mathematically, the system is described as follows: **Inputs:** State vector—$\mathbf{x} = [\theta, \Delta\theta]$.

Outputs: Force applied (in Newtons)—$\mathbf{u} = [F]$ where $F > 0$ denotes a force applied in the positive x direction.

Equations of Motion: These equations serve only to simulate the system and are not used in the derivation of the control law:

$$\ddot{\theta} = \frac{g\sin\theta + \cos\theta\left(\frac{-F - m_p l\dot{\theta}^2\sin\theta + \mu_p \mathrm{sgn}(\dot{x})}{m_p + m_c}\right) - \frac{\mu_p\dot{\theta}}{m_p l}}{l\left(\frac{4}{3} - \frac{m_p\cos^2\theta}{m_p + m_c}\right)} \tag{2.12}$$

$$\ddot{x} = \frac{F + m_p l\left(\dot{\theta}^2\sin\theta - \ddot{\theta}\cos\theta\right) - \mu_c \mathrm{sgn}\left(\dot{x}\right)}{m_p + m_c} \tag{2.13}$$

where the variables are defined as g: acceleration due to gravity, m_c: mass of the cart (kg), m_p: mass of the pole (kg), l: half-length of pole (m), μ_c: coefficient of friction for cart (N), and μ_p is the coefficient of friction for pole (N) with values shown in Table 2.1.

$g = 9.8\mathrm{m/s}^2$	$m_c = 1.0\mathrm{kg}$
$m_p = 0.1\mathrm{kg}$	$l = 0.5\mathrm{m}$
$\mu_c = 0.0\mathrm{N}$	$\mu_p = 0.0\mathrm{N}$

Table 2.1: Parameters of Inverted Pendulum System Simulation.

Success and Failure: The setpoint for the inverted pendulum problem is $[0, 0]$. Failure occurs at either of the following conditions:

$$|\theta| > 12°$$
$$|\Delta\theta| > 25°/\mathrm{s}$$

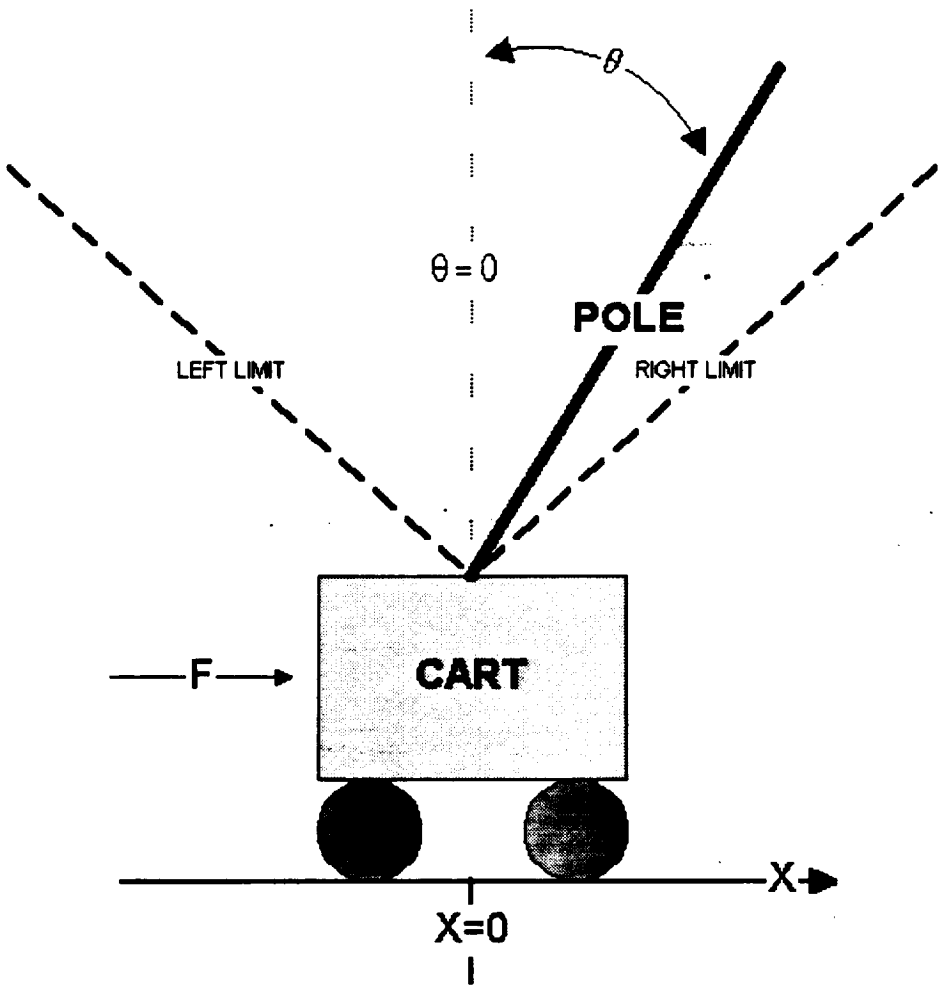

Figure 2.3: The Pole Cart System

In a dynamical system problem such as the inverted pendulum under investigation, the system is always reset upon failure (exceeding state space boundaries), and may be reset upon reaching the goal state; in all the experiments reported in this study for this type of system, reset is not performed upon reaching a goal state. Also, new initial states of the system are selected randomly. The results of a representative run of the simulations are depicted in the following figures of the trajectory (Figures 2.4 - 2.7), the phase space plot (Figure 2.8), and the control surfaces, (Figures 2.9 - 2.10).

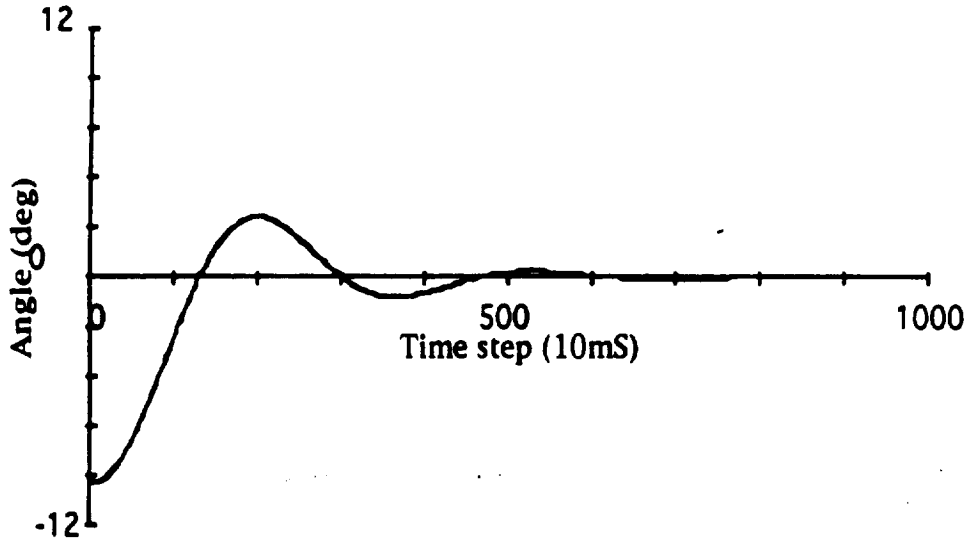

Figure 2.4: Product/limited sum inference, COA defuzzification ($15N$, Seed#1)

2.4.2 Application to Communication Networks

The controller's performance was further tested on the communications network problem using first a three node problem and then a larger ten node problem. The configuration for the later is shown in figure 2.11. Here, reset is upon arrival of the current message to its destination, and determining a new initial state consists of observing where the next message arrives in the network. No failure state is defined for the routing problem since every message eventually reaches its destination (network protocols prevent endless cycling).

Comparison with Other Network Methods

The performance of the controller in the sample networks was compared to several other message routing approaches known in the literature. A rigorous statistical analysis of the results indicate acceptable performance which is equal to and superior in instances to the approaches in question. These results are

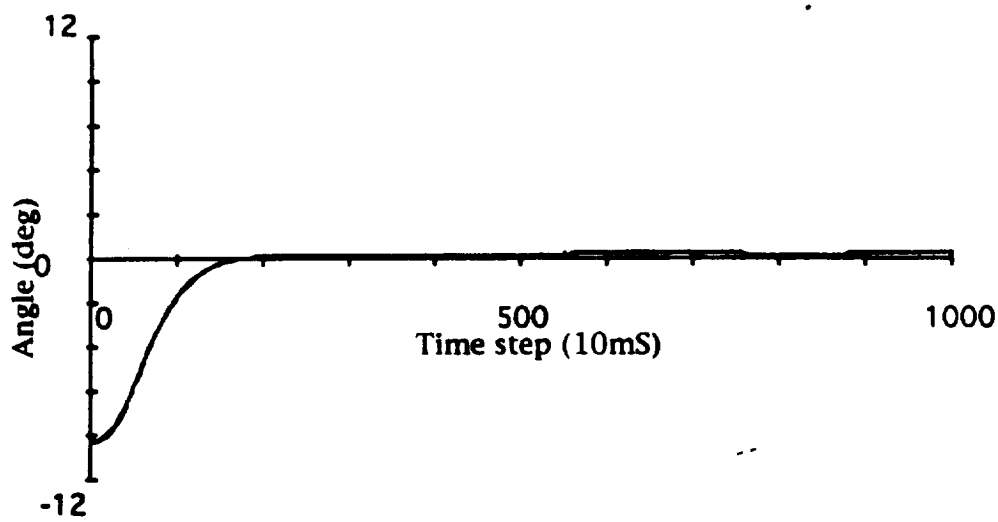

Figure 2.5: Product/limited sum inference, MAX defuzzification (15N, Seed#1)

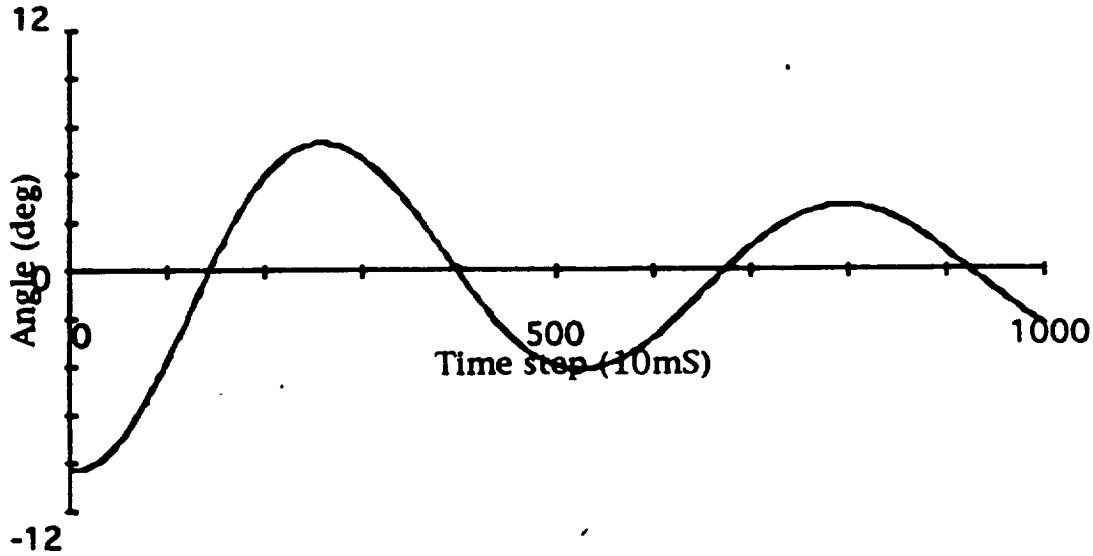

Figure 2.6: Min-max inference, COA defuzzification (15N, Seed#1)

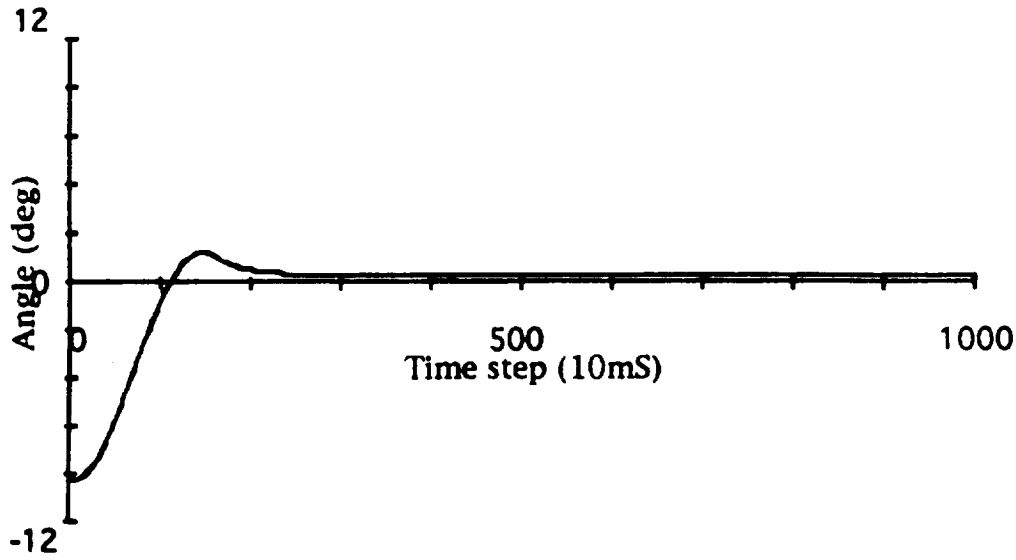

Figure 2.7: Min-max inference, MAX defuzzification (15N, Seed#1)

reported in detail in Murrell [57]. A graphical representation of these results where the 89 percent and 95 percent confidence intervals show that both the controller and the dynamic shortest path algorithms have statistically equal performance which is better than that attained by both the SLA and random controllers. This is depicted in figure 2.12

2.4.3 Application to Power System Stabilization Problems

The earliest stabilizers consisted of a lead-lag analog circuit with the speed as the input. Such a simple controller cannot satisfy the high standards of the power system. PID controllers [34] perform better than the lead-lag circuit. Yet, unless their parameters are tuned automatically as operating conditions change, PID controllers in general do not work satisfactorily within a wide range of conditions. The self-tuning controller [12, 31, 53, 52] and the adaptive controller [9, 10, 36] are designed specifically for this purpose. By continuously identifying the model of the plant, the self-tuning controller adjusts its parameters to achieve optimal performance, while the adaptive controller adjusts its parameters based on the knowledge of the plant model. These two controller paradigms are time-consuming in design but can perform well under different operating conditions. Most recently, fuzzy logic controllers [32, 33, 35, 52, 64] have been successfully applied to stabilize power systems. It has been found that the fuzzy logic controller performs as well as the self-tuning controller in power system stabilization [52] and shows great potential for application in power systems. When the system is of large scale and of high complexity, however, it is not easy to extract the control rules from human expert(s) [39] and, even if this can be done, the

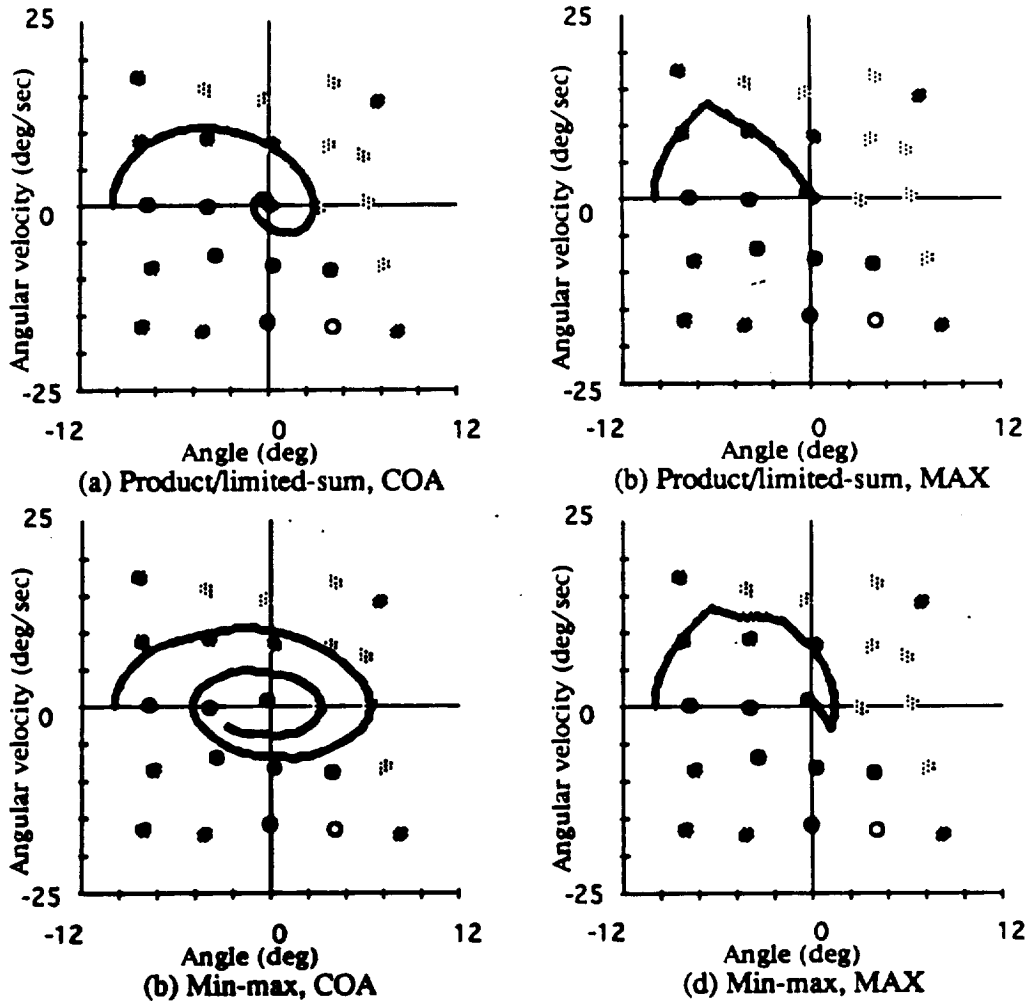

Figure 2.8: Phase space trajectories for fuzzy inference and defuzzification combinations.

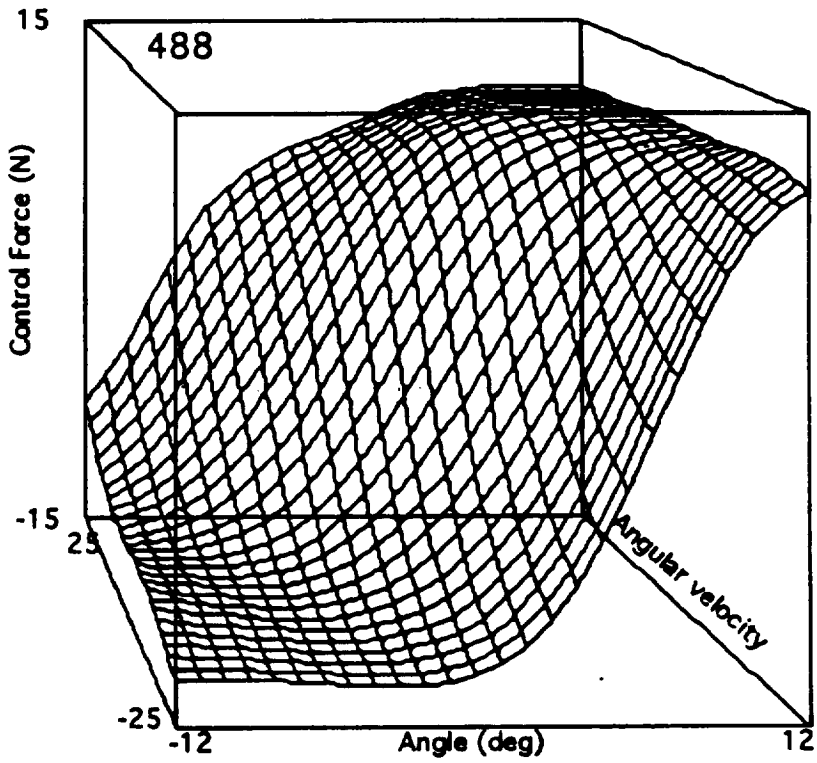

Figure 2.9: Control surface for product/limited sum and COA ($15N$, Seed#1).

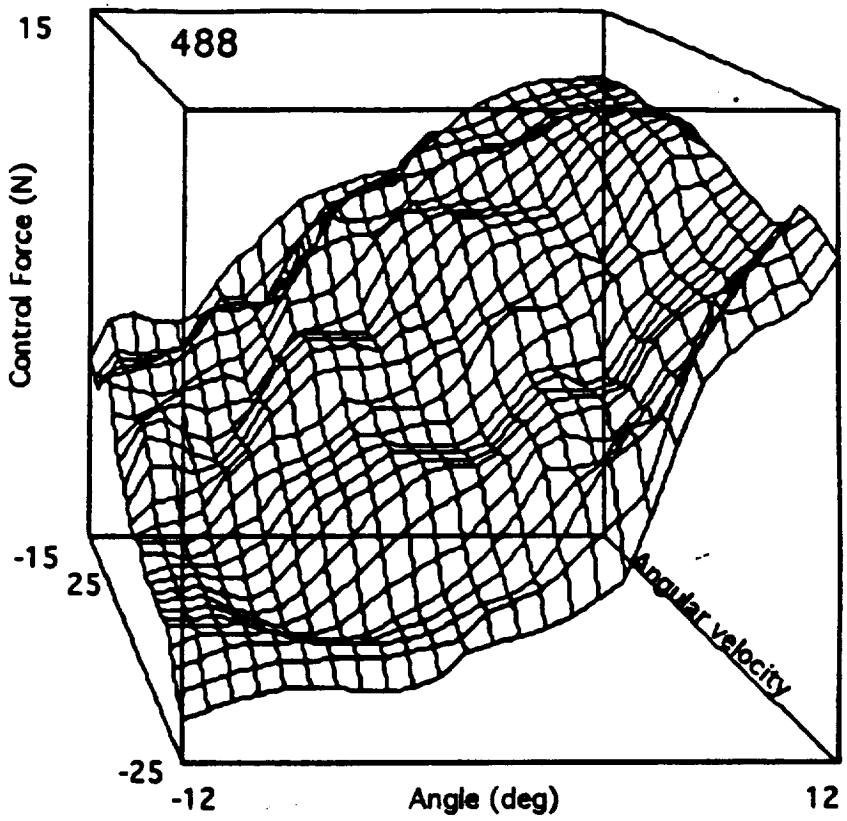

Figure 2.10: Control surface for min-max and COA ($15N$, Seed#1).

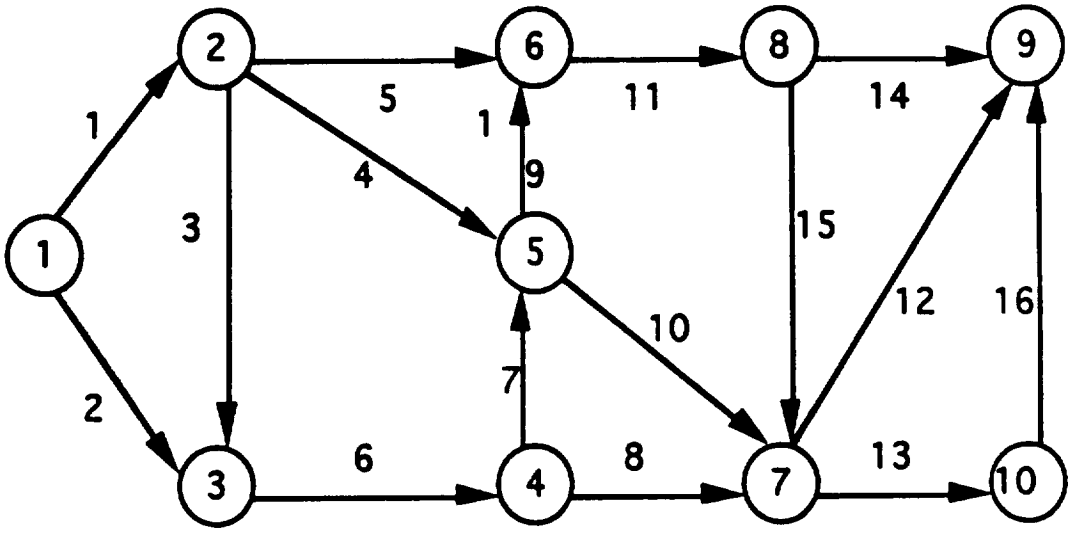

Figure 2.11: A Ten Node Communication Network Problem

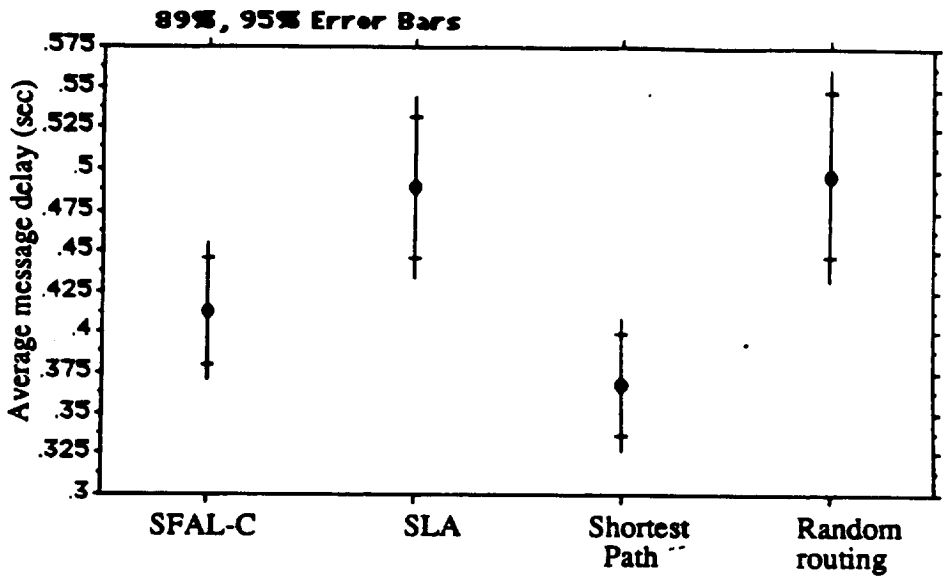

Figure 2.12: Average Delay Statistics of 4 Controllers for the Network Problem

expert's experience is still limited. Therefore, it is necessary to design a controller that can "learn" the control law via its own experience.

2.5 Mathematical Models of the Power System

The system considered here is composed of a synchronous machine with an exciter and a stabilizer connected to an infinite bus. The dynamics of the synchronous machine can be expressed as follows using the linearized incremental model [34]. These equations serve only to simulate the system and are not used in the derivation of the control law:

$$\begin{aligned}
\Delta\dot{\omega} &= \frac{1}{M}(\Delta T_m - \Delta T_e - \\
&\quad \Delta T_L - D\Delta\omega) && (2.14)\\
\Delta\dot{\delta} &= 377\Delta\omega && (2.15)\\
\Delta T_e &= K_e\Delta\delta + K_2\Delta e_q && (2.16)\\
\Delta\dot{e}_q &= \frac{1}{K_3 T_{de}}(K_3\Delta e_{fd} - \\
&\quad K_3K_4\Delta\delta - \Delta e_q) && (2.17)\\
\Delta V_t &= K_5\Delta\delta + K_6\Delta e_q && (2.18)\\
\Delta\dot{V}_f &= \frac{1}{T_F}(K_f\Delta\dot{e}_{fd} - \Delta V_F) && (2.19)
\end{aligned}$$

$$\Delta\dot{e}_{fd} = \frac{1}{T_E}(\Delta V_A - K_E \Delta e_{fd}) \tag{2.20}$$

$$\Delta\dot{V}_A = \frac{1}{T_A}(K_A \Delta V_{ref} - K_A \Delta V_F + K_A u - K_A K_6 \Delta e_q - K_A K_5 \Delta\delta - \Delta V_A) \tag{2.21}$$

$$|u| \leq u_{\max} \tag{2.22}$$

where

V_{ref}	constant reference input voltage
ΔV_t	terminal voltage change,
ΔV_o	infinite bus voltage change
Δe_{fd}	equivalent excitation voltage change
Δe_q	q-axis component voltage behind transient reactance change
ΔV_F	stabilizing transformer voltage change
u	stabilizer output
ΔT_m	mechanical input change
ΔT_e	energy conversion torque change
ΔT_L	load demand change
$\Delta\delta$	torque angle deviation,
$\Delta\omega$	angular velocity deviation
K_A, K_E	voltage regulator gains
T_A, T_E	voltage regulator time constants
K_F	stabilizing transformer gain
T_F	stabilizing transformer time constant
$K_1, \ldots, K_6$	constants of the linearized model of synchronous machine
T_{do}	d-axis transient open circuit time constant
M	inertia coefficient
D	damping coefficient
T_s	sampling period

The objective of the controller is to drive the state of the system $\vec{x}$ to $[0, 0]$ via the stabilizer output u. The values for the above parameters are given in Table 6.4 below.

$K_1 = 1.4479$	$K_2 = 1.3174$	$K_3 = 0.3072$
$K_4 = 1.8050$	$K_5 = 0.0294$	$K_6 = 0.5257$
$K_A = 400$	$T_F = 1.0$	$T_A = 0.05$
$D = 0$	$T_{do} = 5.9$	$K_E = -0.17$
$M = 4.74$	$T_E = 0.95$	$K_F = 0.025$
$\Delta T_m = 0$	$\Delta V_{ref} = 0$	$T_s = 0.01$

Table 2.2: Parameters of Simulation.

2.6 Simulation Results

We are interested in investigating the practicality and effectiveness of our newly developed controller in stabilizing the power system whose model depicted in the foregoing section was used in the simulation phase for system mimicking only but not for control purposes. The experiments run on the power system stabilization problem consisted of multiple replications of the learning phase of the controller on a simulated power system written in C. The inputs to the controller are $\Delta\omega$ and $\Delta\dot{\omega}$. Thus, the controller in effect mimics a PD-like controller with unknown structure. The state space is defined as $\Delta\omega \in [-0.012, 0.012]$ and $\Delta\dot{\omega} \in [-0.025, 0.025]$. The number of nodes for the SFDN is set at $N = 25$ and there are 5 reference control fuzzy sets defined for $u \in [-0.12, 0.12]$. Once the controller has completed the learning phase, it is used as a stabilizer in the system.

Several experiments were run and an example of the resulting controller is shown in the figures below. Figure 6.8 shows the transient process of $\Delta\omega$ when the load increases 0.05 pu and 0.3 pu, respectively. It takes about 2 seconds for the speed deviation $\Delta\omega$ to vanish for the 0.05 pu load change and about 3 seconds for the 0.3 pu load change. Figure 6.9 shows the learned control surface using product–limited sum inference and center-of-area defuzzification.

2.6.1 Comparison to Existing Controllers

The simulation results clearly showed that the controller can learn an effective control law to stabilize the system under varying load conditions. However, the results, are not optimal with regard to settling time. The settling time obtained with our controller was slightly longer than the results obtained using an existing PID controller [34] and a fuzzy controller [35], but comparable to or shorter than the settling time for other fuzzy controllers [32, 33, 64] reported in the literature. The optimality issue (see Section 2.6.2) was subsequently investigated further. The results of that inquiry are reported in Chapter V. Although not optimal, the relative ease of developing an "efficient" controller via the self-learning controller with respect to the existing methods demonstrates the potential of this approach and in fact sufficiently meets the objectives of this project.

Despite the foregoing, the advantages of our controller over other controllers reported in the literature that have been applied to the power systems stabilization problem are very significant and should be

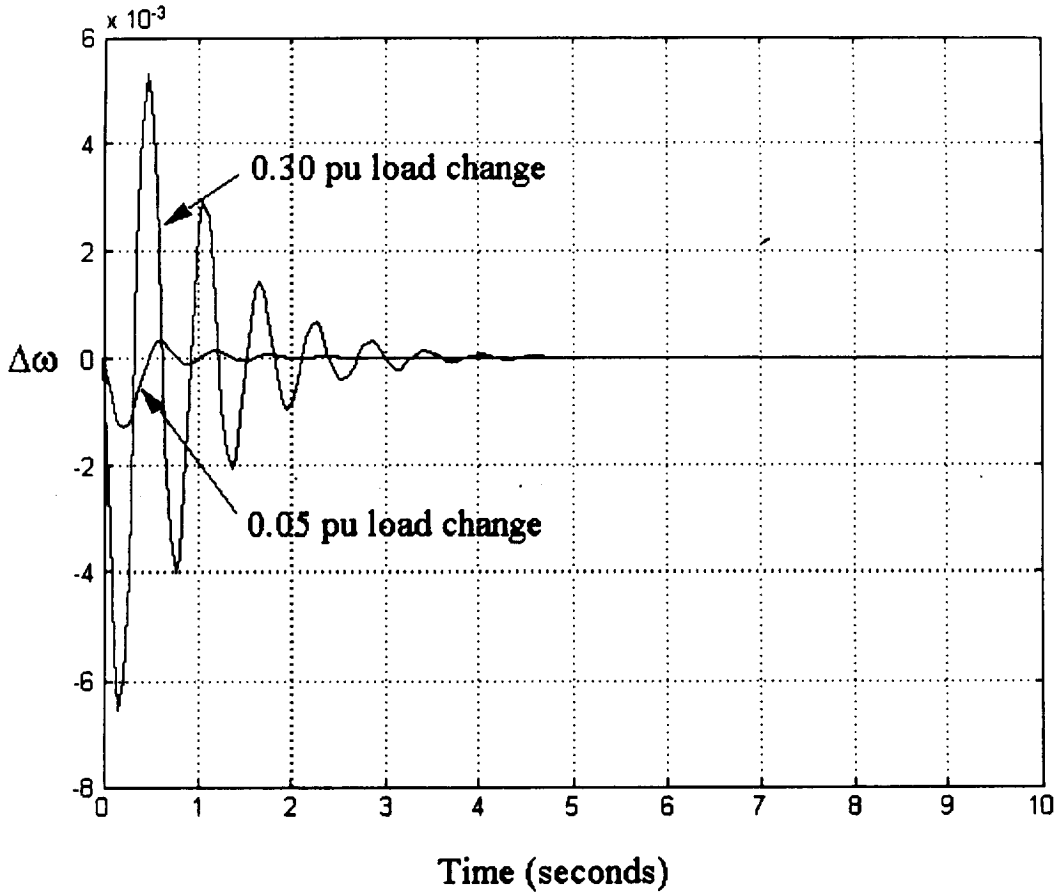

Figure 2.13: Transient Process for Selected Load Changes.

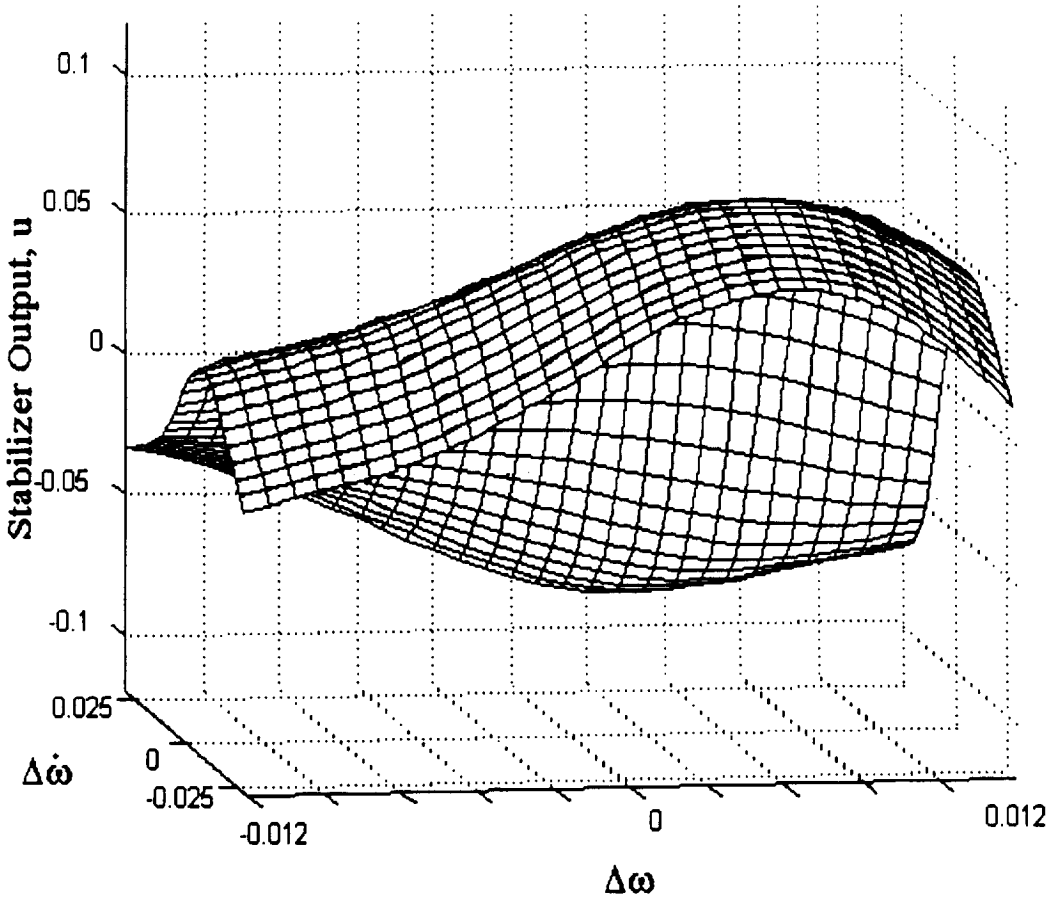

Figure 2.14: Learned Control Surface for Stabilization.

noted:

- The controller successfully learned the control law via its own experience. It did not require the analytic solution of a dynamical model, the tuning of parameters as in PID control, and it did not rely on existing expert knowledge about the control of the process.
- The learning phase of the controller took less than 5 minutes to complete. Thus, there is a huge time savings in development over the existing controllers.
- The internal controller parameters (that control the learning and other attributes) are very robust—the parameters used for the inverted pendulum problem were used for the power system stabilization problem. No tuning of these parameters was performed, although doing so might have improved the resulting control.
- The resulting control is robust—the controller can handle a more extreme range of load changes than the PID controller [34].

2.6.2 Reinforcement Learning and Optimal Control

The lack of optimality with respect to settling time is not unexpected since the controller, which learns via reinforcements, is only given one goal: *Drive the power system to its set point* $\vec{x} = [0, 0]$. The only reinforcements that are given are upon *success* or *failure* of the plant and a desired trajectory through the state space is not defined. These external reinforcements are used to update an internal prediction function for each maximally activated node n_t at time t via $P_{t+1}(n_t) = P_t(n_t) + \alpha(P_t(n_{t+1}) - P_t(n_t))$, similar to Sutton's method of temporal differences. Thus, only by trial and error does the controller learn to drive the system to the set point, and it does so without requiring (and therefore without necessarily satisfying) a performance objective function.

The two controllers that performed better [35, 34] both utilized external information about the plant or the behavior of the controller. This additional information about the plant dynamics and desired control behavior could possibly be used by our controller to develop near-optimal control automatically. For example, knowing the desired trajectory of the power system through the state space allows the controller to give itself *additional* external reinforcement about its behavior, thereby changing $P_t(n_t)$ and the learned control law.

2.7 Unique Features

This controller has several unique features mentioned earlier which we recapitulate here. It is adaptive and well suited to the control of complex processes. In particular, it is capable of learning effective

control using process data and improving its control through on-line adaption. The controller performs a fuzzy discretization of the state and control spaces and learns the fuzzy relations for these fuzzy subsets using a variation of the TD method with its dynamic programming inspirations. While it adapts both the membership functions and the control rule state-control association, the controller primarily learns the control rule associations, unlike many other methods which fix the rules and adjust the membership functions. Most important, it does so for the entire state space. Additionally, no training data sets nor any error signal derived from knowledge of the desired plant trajectory are needed. This self-learning controller has been successfully applied to the inverted pendulum problem, the DC servomotor position control problem, the switching problem in a distributed communications network, and the power system stabilization problem.

2.8 Conclusions

We have reported the development an intelligent controller with many unique features and successfully applied it, with various modifications, to an array of problems. Of note are: Esogbue and Murrell, 1993a; Esogbue and Murrell, 1993b; Murrell, 1994; Esogbue and Murrell, 1994; Esogbue and Hearnes, 1995; Esogbue, Hearnes and Song, 1995). Further investigations and extensions which are outside the scope of this project are underway. These include the use of more intelligent reinforcement strategies especially those that are DP-based algorithms useful in learning real-time control strategies. Of interest are Watkin's Q-learning algorithm and its variations and hybridization of the controller involving dynamic switching between the reinforcement controller and a stabilizing controller.

Chapter 3

The SFAL-C Controller: Theory and Development

3.1 Theoretical Results

We have reported, in the previous chapters, the essential underlying structure of the controller, as well as experiments which illustrate its performance in sample problem arenas. These led to the elicitation of its properties enunciated earlier. In this chapter, we present a detailed account of its theoretical aspects and properties. To do so, we reproduce the relevant sections of J. A. Murrell's doctoral thesis supervised by the author of this report and PI of the project under sponsorship by both the National Science Foundation(NSF) and the Electric Power Research Institute (EPRI). As indicated earlier, J. A. Murrell's work on the project led to his doctoral thesis in question. Specifically, Sections 5.4 and 5.5 are involved. Without loss of generality, throughout this chapter, the references to the equations as for example Eqs. (5.14-5.21)are exactly the same as the relevant portions of the thesis.

3.1.1 The SFDN Algorithm

The purpose of this section is to show that the distribution of the node location vectors is adapted to reflect the distribution of information in the state space. In terms of the spatial distribution of the states, information is concentrated where the spatial variance is small; that is, in the empirical distribution which counts where states have occurred, there is more information at locations in the state space where many states have occurred and thus the average distance between them is small. This is the kind of information that ordinary clustering algorithms tend to maximize. Here we would like to consider a broader definition of information. For example, in some areas of the state space, the certainty that a particular control action is correct may be greater than in other areas. The greater certainty may be due to a lesser degree

of inherent fuzziness, or less statistical noise, or both. Moreover, for a certain region of the state space, certain controls are better to use than others. This type of information should also impact on how states are grouped.

Assume without loss of generality that the state space X is continuous. A continuous distribution can be given as the local density defined over the domain. Characterize the continuous distribution of information by a density $I(\boldsymbol{v})$ at location $\boldsymbol{v}$ in state space X. The distribution of nodes $D(\{N_i\})$ is a discrete distribution. A discrete distribution can be characterized by the frequency of occurrence within determined subregions which partition the domain. We would like to be able to describe the closeness of a discrete distribution to a continuous one. In order for the distributions to be close, (1) the frequency of points within each region should be proportional in some sense to the average density in the region, and (2) the discrete points within each region should be near concentrations of density in the region. Both should be satisfied. With only the first criterion, points within a region could be distributed in any fashion even though the frequencies match the densities for each region. If only the second criterion were satisfied, then one region could use all the available points to distribute according to the local density, leaving none to cover the densities in other regions.

An objective function whose minimization balances the distribution of the node locations $\boldsymbol{m}_i$ among the regions with nearness within each region is given as

$$\min V = \sum_i \sum_{i'} \int_{\eta_{ii'}} (\boldsymbol{v} - \boldsymbol{m}_i)^2 I(\boldsymbol{v}) d\boldsymbol{v} \tag{3.1}$$

where

$$\begin{aligned} I(\boldsymbol{v}) &= \text{the information density at } \boldsymbol{v} \\ \eta_{i'} &= \text{the } i'^{th} \text{ region of points in } X. \\ \eta_{ii'} &= \{\boldsymbol{v} \mid \boldsymbol{v} \in \eta_{i'} \text{ and } \boldsymbol{m}_i \in \eta_{i'}\} \end{aligned}$$

The information density $I(\boldsymbol{v})$ used in the controller is a function also of $\boldsymbol{m}_i, \boldsymbol{g}_i, \boldsymbol{v}$, and ρ_l, as well as of the iteration, since it is actually an estimate which is updated by the algorithm over time. Eq. (5.15) gives the nearness between node N_i and node N_{i^*} as

$$v_{i^*}(k, \boldsymbol{g}_i) = v_{i^*}(k) \bullet \sigma(\boldsymbol{g}_{i^*}, \boldsymbol{g}_i) \bullet I(\boldsymbol{g}_{i^*})$$

This quantity can be viewed as a kind of information potential between the two nodes. The degree of membership of N_i in the node neighborhood of N_{i^*} is based on this nearness, but cuts off for nodes beyond a crisp radius:

$$\mu_{i^*}(\boldsymbol{m}_i(k), \boldsymbol{g}_i(k), \boldsymbol{v}(k), \rho_l(k)) = \begin{cases} v_{i^*}(k, \boldsymbol{g}_i) & \text{if } d_{i^*}(\boldsymbol{m}_i, v) \leq \rho_l(k) \\ 0 & \text{otherwise} \end{cases}$$

is the information available to node N_i from node N_{i^*}. Hence, the total information available to N_i could be constructed as a weighted average that interpolates the information learned at each of the nodes:

$$I(v, m_i, g_i, \rho_l) = \sum_{i=1}^{S} w_i(v) \bullet \mu_i(m_i, g_i, v, \rho_l))$$

$$I(v, m_i, g_i, \rho_l) = \sum_{i=1}^{S} w_i(v) \bullet \mu_i(m_i, g_i, v, \rho_l))$$

where, for example, $s_i(v)$ could be $\mu_{X_i}(v)$ and $\sum$ and $\bullet$ indicate any appropriate disjunction and conjunction operators. If we let $w_i(v)$ be an indicator function for the index of the node in which v has the largest membership, so that

$$I(v, m_i, g_i, \rho_l) = \mu_{i^*}(m_i, g_i, v, \rho_l) \tag{3.2}$$

then the resulting definition for $I(v)$ is sufficient; in practice, the crisp cut-off limits the distance over which information from other nodes affects node N_i. In this definition, the information density is actually piecewise constant with respect to v, other arguments fixed.

For the theorems which follow, the following are needed. Let

$\triangle_k m_i = m_i(k+1) - m_i(k)$.

$v(k)$ be distributed as a uniform distribution F_v over X

$h(v) = h = 1/ \int_X^{\bullet} dv$ = probability density function of $v(k)$

$i^* = i(v) = i$ such that $\mu_{X_{i^*}}(v) > \mu_{X_i}(v) \forall i \neq i^*$

$\delta(d_{i^*}(m_i, v) \leq \rho_l(k)) = \left\{ \begin{array}{ll} 1 & \text{if } d_{i^*}(m_i, v) \leq \rho_l(k) \\ 0 & \text{otherwise} \end{array} \right\}$

$\eta_v(k, i, i^*) = \{v | d_{i^*}(m_i, v) \leq \rho_l(k)\}$ = crisp neighborhood of states about node N_i.

$A(k, i, i^*) = \int_{\eta_v(k, i, i^*)} dv$

$E_v\{\bullet\}$ represent expectation operator with respect to distribution of $v(k)$

The first theorem says that nodes of the SFDN map tend to move towards concentration of information.

Theorem 1 *Using the SFDN algorithm of Eqs. (5.14-5.21), the expected change in node location parameter vector at a given step k is weighted according to the average magnitude and location of information density.*

Proof

According to the SFDN algorithm Eq. (5.14),

$$\begin{aligned}
\triangle_k \boldsymbol{m}_i &= \boldsymbol{m}_i(k) + \alpha(k)\mu i^*(\boldsymbol{m}_i, \boldsymbol{g}_i, \boldsymbol{v}, \rho_l)(\boldsymbol{v}(k) - \boldsymbol{m}_i(k)) \\
E\boldsymbol{v}\{\triangle_k \boldsymbol{m}_i\} &= E\boldsymbol{v}\{\alpha(k)\mu_{i^*}(\boldsymbol{m}_i, \boldsymbol{g}_i, \boldsymbol{v}, \rho_l)(\boldsymbol{v}(k) - \boldsymbol{m}_i(k))\} \\
&= \alpha(k) E\boldsymbol{v}\{\mu_{i^*}(\boldsymbol{m}_i, \boldsymbol{g}_i, \boldsymbol{v}, \rho_l)(\boldsymbol{v}(k) - \boldsymbol{m}_i(k))\} \\
&= \alpha(k) \int_{\boldsymbol{v} \in X} \mu_{i(\boldsymbol{v})}(\boldsymbol{m}_i, \boldsymbol{g}_i, \boldsymbol{v}, \rho_l)(\boldsymbol{v}(k) - \boldsymbol{m}_i(k)) dF_v(\boldsymbol{v})
\end{aligned}$$

Let $D(N_i)$ be the direction of the information for node N_i due to $\boldsymbol{v}$. Then the direction is

$$D(N_i) = \frac{\boldsymbol{v} - \boldsymbol{m}_i}{\|\boldsymbol{v} - \boldsymbol{m}_i\|} \propto \boldsymbol{v} - \boldsymbol{m}_i$$

The information density at node N_i is $I(\boldsymbol{v}, \boldsymbol{m}_i, \boldsymbol{g}_i, \rho_l) = \mu_{i^*}(\boldsymbol{m}_i, \boldsymbol{g}_i, \boldsymbol{v}, \rho_l)$. Therefore, since $\alpha(k) > 0$,

$$E\{\triangle_k \boldsymbol{m}_i\} \propto \int_{\boldsymbol{v} \in X} I(\boldsymbol{v}, \boldsymbol{m}_i, \boldsymbol{g}_i, \rho_l) D(N_i) dF_{\boldsymbol{v}}(\boldsymbol{v}) \tag{3.3}$$

and the theorem is proved.

Q.E.D.

Lemma 1

Using the SFDN of Eqs. (5.14-5.21), the following equation holds:

$$E\boldsymbol{v}\{\triangle_k \boldsymbol{m}_i\} = \alpha(k) \sum^{S_{i'=1}} v_{i^*}(k, \boldsymbol{m}_i, \boldsymbol{g}_i) hA(k, i, i^*)[\boldsymbol{m}_{i'}(k) - \boldsymbol{m}_i(k)] \tag{3.4}$$

Proof

$$\triangle_k \boldsymbol{m}_i = \boldsymbol{m}_i(k) + \alpha(k)\mu_{i^*}(\boldsymbol{m}_i, \boldsymbol{g}_i, \boldsymbol{v}, \rho_l)(\boldsymbol{v}(k) - \boldsymbol{m}_i(k))$$

$$\begin{aligned}
E\boldsymbol{v}\{\triangle_k \boldsymbol{m}_i\} &= E\{\alpha(k)\mu_{i^*}(\boldsymbol{m}_i, \boldsymbol{g}_i, \boldsymbol{v}, \rho_l)(\boldsymbol{v}(k) - \boldsymbol{m}_i(k))\} \\
&= \alpha(k) E\{\mu_{i^*}(\boldsymbol{m}_i, \boldsymbol{g}_i, \boldsymbol{v}, \rho_l)(\boldsymbol{v}(k) - \boldsymbol{m}_i(k))\} \\
&= \alpha(k) \int_{\boldsymbol{v} \in X} \mu_{i(\boldsymbol{v})}(\boldsymbol{m}_i, \boldsymbol{g}_i, \boldsymbol{v}, \rho_l)\boldsymbol{v}(k) - \boldsymbol{m}_i(k)) dF_{\boldsymbol{v}}(\boldsymbol{v}) \\
&= \alpha(k) \sum_{i'=1}^{S} \int_{\boldsymbol{v} \in \eta_l(k,i,i')} v_{i'}(k, \boldsymbol{g}_i)(\boldsymbol{v}(k) - \boldsymbol{m}_i(k)) dF_{\boldsymbol{v}}(\boldsymbol{v})
\end{aligned}$$

$$
\begin{aligned}
&= \alpha(k) \sum_{i'=1}^{S} v_{i'}(k, g_i) \int_{v \in \eta_v(k,i,i')} (v(k) - m_i(k)) dF_v(v) \\
&= \alpha(k) \sum_{i'=1}^{S} v_{i'}(k, g_i) \left[\int_{\eta_v(k,i,i')} v(k) dF_v(v) - \int_{\eta_v(k,i,i')} m_i(k) dF_v(v) \right] \\
&= \alpha(k) \sum_{i'=1}^{S} vi'(k, g_i) \left[\int_{\eta_v(k,i,i')} v(k) dF_v(v) - m_i(k) \int_{\eta_v(k,i,i')} dF_v(v) \right] \\
&= \alpha(k) \sum_{i'=1}^{S} v_{i'}(k, g_i) hA(k, i, i')[m_{i'}(\mathbf{k}) - m_i(\mathbf{k})]
\end{aligned}
$$

Q.E.D.

Comment

Note that $hA(k, i, i^*) < 1$, and that it tends to decrease with k, since $\rho_l(k)$ decreases with k, though its actual value depends on the values of a_i, a_{i^*}, and $a_{i\_}$. Also, if $\eta_v(k, i, i^*)$ is empty, that is, if m_i is not in the neighborhood of m_{i^*} for any choice of v, then $A(k, i, i^*) = 0$.

Comment Note that if $M = v_{i^*}(k, g_i)$ were a constant M with respect to i^*, and if the distribution of nodes around N_i were fairly uniform, then the average change in location would be close to zero:

$$\alpha(k) M_{i' \in S}[m_{i'}(\mathbf{k}) - m_i(\mathbf{k})] \approx \mathbf{0}$$

Hence, the distribution of information among the nodes N_{i^*} surrounding N_i, indicated by the $v_{i^*}(k, g_i)$ values, weights the average direction to surrounding nodes. The expected change in the location parameter of node N_i is weighted in the direction of the node concentration of information $v_{i^*}(k, g_i)$ in the distribution of nodes N_{i^*} around node N_i.

The next theorem says that the node distribution $D\{N_i\}$ tends towards the information distribution $I(v, m_i, g_i, \rho_l)$.

Theorem 2 *The expected one-step change in a node location parameter for one step is along the negative gradient of the objective function V:*

$$E_v\{\triangle_k m_i\} \propto -\frac{dV}{dm_i} \tag{3.5}$$

Proof

Taking the derivative of Eq. (5.44), we obtain

$$\frac{dV}{dm_i} = -2 \sum_{i'} \int_{\eta_{i'i}} (v - m_i) I(v) dv$$

Define the crisp neighborhood of states about each node N_{i^*} as $\eta_{i^*} = \eta(i, i^*, k)$. Since exactly one maximally activated node is selected,

$\{\eta(i,i',k)|i' = 1,\ldots,s\}$ is a set of mutually exclusive and exhaustive subsets of X to serve as the distribution bins.

Since $I(\boldsymbol{v},\boldsymbol{m_i},\boldsymbol{g_i},\rho_l) = \mu_{i^*}(\boldsymbol{m_i},\boldsymbol{g_i},\boldsymbol{v},\rho_l)$ is a function of $\boldsymbol{m_i}$, the complete derivative of Vrequires the I be differentiated. It is not a continuous function of $\boldsymbol{m_i}$, but $\boldsymbol{m_i}$ is only used to determine the size of the neighborhood. When $\boldsymbol{m_i}$ changes at step k as a result of the update algorithm, the change in the size of the neighborhood does not affect $I(\boldsymbol{v},\boldsymbol{m_i},\boldsymbol{g_i},\rho_l)$ until the next iteration of the algorithm at $k+1$. Since the result to be established pertains only to what happens at step k, the rate of change of $I(\boldsymbol{v},\boldsymbol{m_i},\boldsymbol{g_i},\rho_l)$ with respect to $\boldsymbol{m_i}$ does not enter into the computation of the derivative of V. Thus we have

$$\begin{aligned}
\frac{dV}{d\boldsymbol{m_i}} &= -2\sum_{i'=1}^{S} \int_{\eta_{\boldsymbol{v}}(i,i',k)} (\boldsymbol{v}-\boldsymbol{m_i})I(\boldsymbol{v},\boldsymbol{m_i},\boldsymbol{g_i},\rho_l)d\boldsymbol{v} \\
&= -2\sum_{i'=1}^{S} \int_{\eta_{\boldsymbol{v}}(i,i',k)} (\boldsymbol{v}-\boldsymbol{m_i})\mu_{i'}(\boldsymbol{m_i},\boldsymbol{g_i},\boldsymbol{v},\rho_l))d\boldsymbol{v} \\
&= -2\sum_{i'=1}^{S} \int_{\eta_{\boldsymbol{v}}(i,i',k)} (\boldsymbol{v}-\boldsymbol{m_i})v_{i'}(k,\boldsymbol{g_i}))d\boldsymbol{v} \\
&= -2\sum_{i'=1}^{S} v_{i'}(k,\boldsymbol{g_i}) \int_{\eta_{\boldsymbol{v}}(i,i',k)} (\boldsymbol{v}-\boldsymbol{m_i})d\boldsymbol{v} \\
&= -2\sum_{i'=1}^{S} v_i'(k,\boldsymbol{g_i}) \left[\int_{\eta_{\boldsymbol{v}}(k,i,i')} \boldsymbol{v}\,d\boldsymbol{v} - \int_{\eta_{\boldsymbol{v}}(k,i,i')} \boldsymbol{m_i}d\boldsymbol{v} \right] \\
&= -2\sum_{i'=1}^{S} v_{i'}(k,\boldsymbol{g_i}) \left[\int_{\eta_{\boldsymbol{v}}(k,i,i')} \boldsymbol{v}\,d\boldsymbol{v} - \boldsymbol{m_i}\int_{\eta_{\boldsymbol{v}}(k,i,i')} d\boldsymbol{v} \right] \\
&= -2\sum_{i'=1}^{S} vi'(k,\boldsymbol{g_i})A(k,i,i')[\boldsymbol{m_{i'}} - \boldsymbol{m_i}] \\
\propto\ & -\alpha\ (k)\sum_{i'=1}^{S} v_{i'}(k,\boldsymbol{g_i})hA(k,i,i')[\boldsymbol{m_{i'}} - \boldsymbol{m_i}]
\end{aligned}$$

This theorem says that the expected one step change in every node location is in the direction of minimizing V, which means that the distribution of nodes is brought closer to the distribution of information. The location parameter update algorithm is thus a (stochastic) steepest descent, i.e., gradient algorithm on the objective of closeness of the distributions. algorithm asymptotically approaches minimizing V, which would indicate the node distribution were as close as possible to the information distribution. It has only been shown that the expected tendency is towards minimizing V.

It is clear that if a function $I(\boldsymbol{v})$ were available independent of the node locations $\boldsymbol{m_i}$, then V could be optimized when

$$\sum_{i'} \int_{\eta_{i'i}} (\boldsymbol{v}-\boldsymbol{m_i})I(\boldsymbol{v})d\boldsymbol{v} = 0$$

which would occur if every $\boldsymbol{m_i}$ were positioned right at the centroid of $\boldsymbol{v}$ in each region. In our replacement of $I(\boldsymbol{v})$ by the estimate $I(\boldsymbol{v},\boldsymbol{m_i},\boldsymbol{g_i}\rho_l)$, the minimization would occur when every $\boldsymbol{m_i}$ within a region equals

the $\boldsymbol{m}_{i^*}$ of the region. This would mean that either each $\boldsymbol{m}_i$ is the only node in its region or every other node in each region has collapsed into $\boldsymbol{m}_*$. This latter is not desirable, yet it is almost sure to eventually occur to any node which is within the crisp neighborhood boundary of only one other node for all values of $\boldsymbol{v}$. Nodes with locations $\boldsymbol{m}_i$ which belong to the crisp neighborhoods $\eta_{\boldsymbol{v}}(i, i', k)$ of several other nodes $N_{i'}$ for some value of $\boldsymbol{v}$ still feel the pull of each of those nodes and so, on the average, do not collapse into another node.

The SFDN algorithm is a self-organizing algorithm; the sense that we mean that here is that it attempts to organize the location vectors according to a function of the location vectors themselves, i.e., it is self-referencing. It tries to "bootstrap" itself into a meaningful order with no teacher assistance. The inherent danger in such an algorithm manifests itself here in the possibility of nodes collapsing into each other, thus destroying its own information-carrying capacity. To some extent, this may be alright, if the process is begun with more nodes than are really necessary. However, if the algorithm were left to run indefinitely, as we would like to do at least conceptually to prove an asymptotic convergence result, reaching an absolute minimum of V might be an undesirable result.

The heuristic which avoids this is to decrease the size of the crisp neighborhoods in time at the right speed. If the size is decreased too quickly, then the nodes have too little time to distribute themselves according to I and perhaps too little time for an accurate estimate of I (which depends on other algorithms in the controller, as well as on the plant) to develop in the first place. If the rate of decrease is too slow, then there may be too little net movement of nodes, on the average, since a node may belong to too many neighborhoods for too long; or, if the initial neighborhoods are very small, too much time may be spent in the situation where nodes belong to only one other node's crisp neighborhood, and too much collapsing then occurs.

The rate of decay of the neighborhoods also must be balanced with the size and rate of decay of the rate parameter $\alpha(k)$, which is the step size of the increment to each $\boldsymbol{m}_i$. The increment to $\boldsymbol{m}_i$ also cannot be so large that the information function $I(\boldsymbol{v}, \boldsymbol{m}_i, \boldsymbol{g}_i, \rho_l)$ itself shifts too rapidly to be followed by subsequent location updates. The dilemma here is similar to that found in the analysis of Kohonen self-organizing feature maps, for which few analytical results have been proven. Also as with SOS, there are edge effects, which manifests here in the situation of nodes near the edge of the proscribed state space which are not surrounded by other nodes on all sides. Whenever there is not a fairly even distribution of other nodes around a given node, the efficacy of the algorithm suffers.

Ideally, we would like for the process to develop as follows. First, build up a good estimate of the information $I(\boldsymbol{v}, \boldsymbol{m}_i, \boldsymbol{g}_i, \rho_l)$; then, give the nodes sufficient time in each others' large neighborhoods (using each other as estimates of centers of density) with large step sizes for the node to make the gross movements to arrange themselves according to the global pattern of the density; then spend time with neighborhoods small enough to include only a very few close neighbors and have a small step size in

order to adjust each node's position within in its own neighborhood to move it towards the centroid; then finally come to the point where each node is the only node in its neighborhood. At present, no results are known regarding the balance of $\alpha(k)$ and $\rho_l(k)$ which guarantee that this is achieved. As with SOMs, these working values for these quantities have been found through extensive simulation, since precise theoretical specification is lacking (see [75], [131], [132], [133], [27] for discussion of the analogous situation for SOMs).

Several very important things are accomplished when the algorithm does work, however. The estimated density $I(\boldsymbol{v}, \boldsymbol{m}_i, \boldsymbol{g}_i, \rho_l)$ can be defined to reflect not only the distribution of the states, but the distribution in the state space of information about the correctness of controls. Hence, nodes tend to move to areas in the state space where knowledge about the correctness of the control for that locality is most certain. Therefore, nodes tend to move away from areas in the state space where the correct control is ambiguous, i.e., the control switching curve where the transition from one discrete control to another should take place. The information can also include how closely matched the controls of two nodes are. Hence, nodes with similar tendency to choose a certain control gravitate together more than nodes with unlike control choice tendencies, so that the nodes tend to cluster according to their associated control value as well as according to the distribution of the states. The SFDN algorithm of the SFAL controller tends to cluster nodes together that have a similar control in a manner that reflects the distribution of appropriate controls assigned to states in the state space.

The idea of the spreads algorithm is similar to that of the location update algorithm. It is a gradient algorithm that brings the error between s_i and $\|^{\boldsymbol{v}} - \boldsymbol{m}_i\|^2$ towards zero, but the errors are weighted to count more when $\boldsymbol{v}$ is close to $\boldsymbol{m}_i$ and when node N_i has not been frequently visited. The former weight is equivalent to computing the average error only over a local fuzzy set centered at $\boldsymbol{m}_i$ with spread equal to the fuzziness parameter ϕ. the latter weighting makes the spreads tend to be larger for nodes that have not been frequently visited. A node that has not been frequently visited has less information and so this uncertainty is reflected as a larger dispersion in the membership function there. A proof for the gradient descent property of the spreads algorithm could proceed along the lines of Lemma 1 and Theorem 2, but it would be more difficult due to the necessity to take expectations of a more complicated, non-separable function of the random variables $\boldsymbol{v}$.

3.1.2 The SLFCN and IFCN Algorithms

In this section, it is shown that the correlation parameters reflect the distribution of reinforcements among the control actions, and that the c_{ij} and hence the g_{ij} estimate the fuzzy control relation which is correct according to the reinforcements.

In the last section, we were concerned with a distribution which is a continuous-valued function over a continuous space of points, and a distribution which is a discrete-valued frequency count for intervals

or regions in a continuous space of a finite number of points drawn from that continuous space. In this section, we are concerned with continuous-valued functions over a discrete space of points. Each point in the discrete space corresponds to one of the r different fuzzy control actions. Hence, to describe a distribution, there is no need to arbitrarily construct intervals or regions in a continuous space. The following definitions make precise what is meant by the term distribution as it is used in this section. Let $\mathcal{G} = \{1, 2, \ldots, r\}$ be the index set for an ordered set of objects $U_F = \{U_1, \ldots, U_r\}$.

Def. 1: A set $F = \{f_i \in (0, 1] \mid i \in \mathcal{G}\}$ is called a *distribution* over U_F.

Def. 2: A vector $\boldsymbol{f} = [f_1, \ldots, f_r]^T$ corresponding to the distribution F is called a distribution vector.

We may sometimes write $F\{\boldsymbol{f}\}$ to indicate the distribution corresponding to distribution vector $\boldsymbol{f}$.

Def. 3: A distribution is called a probability distribution if $\sum_{i=1}^{r} f_i = 1$.

Recall that z_{ij} is the fuzzy reinforcement that should result when the state is classified as X_i and the control action taken is classified as U_j($\boldsymbol{Z} = [z_{ij}]$ wsa defined in Eq. (5.8)). Then $z_i = [z_{i1}, z_{i2}, \ldots, z_{ir}]^T$ is the fuzzy reinforcement vector for action i, and $\boldsymbol{Z} = [z_{ij}]$ is the fuzzy reinforcement matrix. For a deterministic plant, z_i is a fixed, not random quantity. For stochastic plants where the reinforcement for a fixed state and action is random, then we would like z_{ij} to represent the expected value with respect to any random components of the process, assuming that the distribution is stationary. If the process under consideration is time-varying so that the reinforcement distribution is not stationary, then, as with all adaptive control situations, it would be necessary to assume that the rate of variation were small compared to the adaptation rates of the controller algorithms.

Let $\zeta_i(k) = [\zeta_{i1}(k), \zeta_{i2}(k), \ldots, \zeta_{ir}(k)]^T$ be the sample reinforcement vector received at step k for action i. Even for deterministic plants, this vector is random due to the random nature of the controller itself. Recall that the control action index j^* is randomly selected according to the correlation parameter vector $\boldsymbol{c}_i(k) = [c_{i1}(k), c_{i2}(k), \ldots, c_{ir}(k)]^T$, which gives the finite probability distribution at time k for the control actions for an input state vector classified as X_i. We have

$$\begin{aligned} \zeta_i(k) = [\overline{a}_1(k)b_{1r}(k), \ldots, \overline{a}_r(k)b_r r(k)]^T &= [0, 0, \ldots, \overline{a}_i(k)b_{j^*} r(k), \ldots, 0]^T \\ &= [0, 0, \ldots, z_{ij^*}, \ldots, 0]^T \end{aligned}$$

All of the components of $\zeta_i(k)$ are zero except for the j^{*th} component, where j^* is the index of the control action taken at step k. Thus, $\zeta_{ij^*}(k)$ is the reinforcement $r(k)$ (weighted according to a_i, the degree of membership of the state $x(k)$ in X_i) actually received by the controller at step k when the state

is classified as X_i and the control action taken is classified as U_j. In practice, the quantity $r(k)$ has a random component due to the fact that it is a statistical estimate or prediction provided by the PES using past data.

The uncertainty in the reinforcement is not fundamentally random, but rather it is fuzzy. First, we allow that the correctness of an action as indicated by the reinforcement $r(k)$ may be a matter of degree, even given crisp x and u. The correctness of an action (as indicated by the reinforcement $z_{ij}(t) = a_i(x(t))b_j(u(t))r(t))$ within each fuzzy neighborhood Xi is further fuzzified, since the performance resulting from applying any single control action varies over the states in X_i. The fuzzy neighborhoods of the SFDN map induce a fuzziness on the reinforcement vectors z_i. The randomness of the sample reinforcement vectors ζ_i is due to any randomness in the plant affecting the determination of which fuzzy cell the state most belongs to, due to the controller's random selection of a control to be evaluated, and due to accumulation of randomness from these two causes which builds up in the statistical estimates of $r(k)$.

Since the reinforcement is inherently fuzzy, the correlation parameter vector which gives the action probabilities cannot be expected to converge to a unit vector, as is the case with crisp environments usually considered with learning controllers. It is the objective of this section to show that the learning algorithm of the proposed controller distributes the action probability distribution vector according to the reinforcements, i.e., the measure of the correctness of the controls.

We are interested in the distribution of the net reinforcement over the set of discrete control actions. Since reinforcement lie in the interval (-1,1], the vector z_i does not fit our definition of a distribution. Let $T(z_i) = [T(z_{i1}), \ldots, T((z_{ir})^T$ where $T : (-1, 1] \rightarrow (0, 1]$ is transformation that takes any reinforcement into the unit interval. For example, $T_1(z) = (z + 1)/2$. Then $T(z_i)$ is a distribution vector with corresponding distribution. It can be considered the membership distribution over fuzzy controls U_j giving the degree of membership for each control in the set of correct controls $\mathcal{U}$, given the fuzzy state X_i.

The (empirical) distribution $Z_k\{T(\zeta_i(k))\}$ is the distribution of vectors $T(\zeta_i(k))$ through time k. Since each random vector $T(\zeta_i(k))$ is a random sample of the true value of the distribution vector $T(z_i)$, an estimator of the expected value is needed. Let $\boldsymbol{f}_i(k) = [f_{i1}(k), f_{i2}(k), \ldots, f_{ir}(k)]^T$ be the finite distribution vector for $Z_k\{T(\zeta_i(k))\}$. The classical estimator for this (empirical) distribution is the simple average over the number of times $k_j(k)$ that action j is chosen when at state i after k steps:

$$f_{ij}(k) = \sum_{m=1}^{k_j} T(\zeta_{ij}(m))/k_j \tag{3.6}$$

This estimator is the maximum likelihood estimator, and it is known that $E\{\boldsymbol{f}_i(k)\} = z_i$. The estimator is also known to be consistent, that is, $\boldsymbol{f}_j(k)$ approaches the true distribution z_i as k increases [31].

A recursive formula for an estimator of this type can be found as follows. First consider the simple

case where k indexes the occurrence of a sample quantity $x(k)$. Then

$$\begin{aligned} f(k+1) &= \sum_{m=1}^{k+1} x(m)/(k+1) \\ &= \frac{k}{k+1}\sum_{m=1}^{k} x(m))/k_j + \frac{1}{k_j+1}(k+1) \\ &= \frac{k}{k+1}f(k) + \frac{1}{k+1}x(k+1) \\ &= \frac{1}{a+\frac{1}{k}}f(k) + \frac{\frac{1}{k}}{1+\frac{1}{k}}x(k+1) \end{aligned}$$

Note that the weighting factors $k/(1+k)$ and $1/(1+k)$ sum to 1. When these are replaced by factors α and $(1-\alpha)$ for some positive $\alpha < 1$, then we have

$$f(k+1) = \alpha f(k) + (1-\alpha)x(k+1)$$

which gives the exponential smoothing estimate, or "forgetting factor" estimate, and has the basic form of the delta rule. A more general form of estimate can be given by replacing $1/k$ with a general quantity $\omega(k)$.

$$f(k+1) = \frac{1}{1+\omega(k)}f(k) + \frac{\omega(k)}{1+\omega(k)}x(k+1)$$

For a recursive estimation of the empirical distribution of the quantities $T(\zeta_i(k))$, account must be taken of the fact that only one component is updated at each step k.

$$f_{ij}(k+1) = \begin{cases} \frac{1}{1+\frac{1}{k_j}}f_{ij}(k) + \frac{\frac{1}{k_j}}{1+\frac{1}{k_j}}T(\zeta_{ij}(k+1)) & \text{if } j = j* \\ f_{ij}(k) & \text{otherwise} \end{cases}$$

Let

$$\begin{aligned} \omega_j(k) &= \delta_{jj^*}/k_j \\ w_j(k) &= \frac{\omega_j(k)}{1+\omega_j(k)} \\ \delta_{jj^*} &= \begin{cases} 1 & \text{if } j = j^* \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Then

$$\begin{aligned} \boldsymbol{f}_j(k+1) &= (1-w_j(k))[f_{i1}(k),\ldots,f_{ir}(k)]^T + [w_1(k)T(\zeta_{i1}(k)),\ldots,w_r(k)T(\zeta_{ir}(k))]^T \\ &= (1-w_j(k))[f_{i1}(k),\ldots,f_{ir}(k)]^T + [0,\ldots,w_{j^*}(k)T(\zeta_{ij^*}(k)),\ldots,0]^T \\ &= (1-w_j(k))\boldsymbol{f}_i(k) + w_{j^*}(k)T(\zeta(k+1) \end{aligned}$$

Def. 4 Given a sequence $\{\zeta(m) \in (-1,1]; m = 1,2,\ldots,k\}$ of distribution vectors, if a distribution vector $\boldsymbol{f}(k)$ can be written in the form

$$\boldsymbol{f}(k) = (1 - w_j(k))\boldsymbol{f}_i(k) + w_{j^*}(k)T(\zeta(k+1)$$

for some transformation $T : (-1,1] \rightarrow (0,1]$ and some functions $\omega_j(k), w_j(k)$ such that

$$0 < \omega_j(k) < 1, 0, < w(k) < 1 \text{ and } w(k) = \frac{\omega(k)}{1+\omega(k)}$$

then $\boldsymbol{f}(k)$ is a normalized frequency count of the vectors $\{T(\zeta(m)); m = 1,2,\ldots,k\}$ at time step k, and is thus an estimate of $E\left\{\sum_{m=1}^{k_j} T(\zeta_{ij}(m))/k_j\right\} = E\{T(z_i)\}$.

When T and $w_i(k)$ have certain properties, the distribution estimator yields a distribution which is similar, in some sense, to that obtained with the classical estimator, and in fact, similar to the true distribution. In the next theorem, we will show that action probability distribution given by the correlation parameter vector $\boldsymbol{c}_i(k)$ has the above form and that it is expected to be close or similar to the distribution of the reinforcements $T(z_i(k))$, for each i. Therefore, some concept is needed for what it means for two discrete distributions to be similar to each other when they are not necessarily the same(e.g., when one is an estimate or approximation of the other).

Def. 5 A distribution vector $\boldsymbol{c}$ is said to be *similar* to a distribution vector $\boldsymbol{f}$, written $\boldsymbol{c} \cong \boldsymbol{f}$, if the descending order ranking of the components of $\boldsymbol{c}$ equals that of $\boldsymbol{f}$, that is

$$\text{order } \{c_j; j = 1,\ldots,r\} = (j_1, j_2, \ldots, j_r) = \text{ order } \{f_j; j = 1,\ldots,r\}$$

where $c_{j1} \geq c_{j2} \geq \ldots \geq c_{jr}$

and if whenever $c_{j_a} > c_{j_b}$ then $f_{j_a} > f_{j_b}$ and whenever $f_{j_a} > f_{j_b}$ then $c_{j_a} > c_{j_b}$.

Clearly, this relation of similarity is reflexive, symmetric and transitive, since it is based on the equality of their orders. It is also obvious that multiplying by a positive constant or adding a constant does not affect the similarity of one distribution vector to another.

Def. 6 Two distributions C and F are similar if the corresponding distribution vectors are similar.

Before presenting the main theorem, a few preliminary results are established.

Lemma 2

If conditions (a) through (d) hold:

(a) $\gamma \in (0,1)$

(b) $\zeta_{ij}(k) \in (-1,1), \forall i$ at each k

(c) $c_{ij}(0) \in (0,1), \quad \forall i,j$ at each k

(d) $\sum_{j=1}^{r} c_{ij}(0) = 1, \quad \forall i$

then, using the correlation update algorithm of Eqs. (5.22-5.32), the following hold

(i) $c_{ij}(k) \in (0,1), \quad \forall i,j$ at each k

(ii) $\sum_{j=1}^{r} c_{ij}(k) = 1, \quad \forall i$ at each k

That is, if $\boldsymbol{c}_i$ begins in the unit simplex, it always remains in the unit simplex.

Proof

Conditions (c) and (d) imply that (i) and (ii) hold at $k = 0$. Now, for any arbitrary k, it is shown that the one step of the update algorithm preserves them. We have that, for all i such that c_{ij} is updated (that is, such that N_i is in the crisp neighborhoods of nodes about $N_{i\bullet}$),

$$c_{ij}(k+1) = \frac{c_{ij}(k) + \gamma\delta_{ij}(k)\zeta_{ij}(k+1)}{1 + \gamma\delta_{ij\bullet}(k)z_{ij\bullet}}$$

Let

$$c'_{ij}(k) = \begin{cases} c_{ij}(k) + \gamma c_{ij}(k)\zeta_{ij}(k+1) & \text{if } z_j < 0 \\ c_{ij}(k) + \gamma(1 - c_{ij}(k))\zeta_{ij}(k+1) & \text{if } z_j \geq 0 \end{cases}$$

$$\triangle c'_{ij}(k) = c'_{ij}(k) - c_{ij}(k)$$

If $\zeta_{ij}(k+1) = 0$, then $c'_{ij}(k) = c_{ij}(k)$, which by assumption satisfies (i) and (ii). If $\zeta_{ij}(k+1) < 0$, then

$$\gamma > 0 \text{ and } c_{ij}(k) > 0 \Rightarrow \gamma c_{ij}(k)\zeta_{ij}(k+1) < 0$$

$$\Rightarrow c_{ij}(k) + \gamma c_{ij}(k)\zeta_{ij}(k+1) = c'_{ij}(k) < c_{ij}(k) < 1$$

$$\text{Hence } c'_{ij}(k) < 1.$$

$$\zeta_{ij}(k+1) \in (-1,1) \Rightarrow |\zeta_{ij}(k+1)| < 1.$$

$$\gamma < 1 \text{ and } |\zeta_{ij}(k+1)| < 1$$

$$\Rightarrow c_{ij}(k) > |\gamma c_{ij}(k)\zeta_{ij}(k+1)| \Rightarrow c_{ij}(k) - |\gamma c_{ij}(k)\zeta_{ij}(k+1)| > 0.$$

$$c_{ij}(k) - |\gamma c_{ij}(k)\zeta_{ij}(k+1)| = c_{ij}(k) + \gamma c_{ij}(k)\zeta_{ij}(k+1) = c'_{ij}(k)$$

$$\Rightarrow c_{ij}(k) - |\gamma c_{ij}(k)\zeta_{ij}(k+1)| > 0$$

$$\Rightarrow c'_{ij}(k) > 0.$$

If $\zeta_j(k+1) > 0$, then

$$c_{ij}(k) \in (0,1) \Rightarrow c_{ij}(k) < 1 \text{ and } c_{ij}(k) > 0.$$

$$c_{ij}(k) < 1 \Rightarrow (1 - c_{ij}(k)) > 0.$$

$$(1 - c_{ij}(k)) > 0 \text{ and } \gamma > 0 \text{ and } \zeta_{ij}(k+1) > 0 \Rightarrow \gamma(1 - c_j(k))\zeta_{ij}(k+1) > 0.$$

$$\begin{aligned} c_{ij}(k) > \quad & 0 \quad \text{and } \gamma(1 - c_{ij}(k))\zeta_{ij}(k+1) > 0 \\ \Rightarrow \quad & c_{ij}(k) + \gamma(1 - c_{ij}(k))\zeta_{ij}(k+1) = c'_{ij}(k) > 0 \end{aligned}$$

$$c_{ij}(k) + (1 - c_{ij}(k)) = 1 \text{ and } \gamma < 1 \text{ and } \zeta_j(k+1) < 1$$

$$\Rightarrow c_{ij}(k) + \gamma(1 - c_{ij}(k))\zeta_j(k+1) < 1 \Rightarrow c'_{ij}(k) < 1.$$

Now observe that

$$c_{ij}(k+1) = \begin{cases} c'_{ij}(k)/(1 + \gamma c_{ij^*}(k) z_{j^*} & \text{if } z_j < 0 \\ c'_{ij}(k)/(1 + \gamma(1 - c_{ij^*}(k)) z_{j^*} & \text{if } z_j \geq 0 \end{cases}$$

and that $c'_{ij}(k) = c_{ij}(k)$ for $j \neq j^*$ and $c'_{ij^*}(k) = c_{ij^*}(k) + \gamma c_{ij^*}(k) z_{j^*}$.

Therefore,

$$c_{ij}(k+1) = c'_{ij}(k)/(1 + \Delta c'_{ij^*}(k)).$$

If $\sum_{j=1}^{r} c_{ij}(k) = 1$, then

$$\begin{aligned} (1 + \Delta c'_{ij}(k)) &= \sum_{j=1}^{r} c_{ij}(k) + \Delta c'_{ij^*}(k) \\ &= \sum_{j=1}^{r-1} c'_{ij}(k) + c'_{ij}(k) \\ &= \sum_{j=1}^{r} c'_{ij}(k) \end{aligned}$$

Hence

$$c_{ij}(k+1) = \frac{c'_{ij}(k)}{\sum_{j=1}^{r} c'_{ij}(k)} \Rightarrow \sum_{j=1}^{r} c_{ij}(k+1) = 1$$

which satisfies (ii) and also implies (i).

Proposition 1

Let

$\theta_{ij}(k)$ = Pr { visit control j at step k | visit state i at step k}

$\theta_i(k)$ = Pr {visit state i at step k}

If the correlation update algorithm of Eqs. (5.22-5.32) and the conditions of Lemma 2 hold, then if $\theta_i(k) > \forall i$ at each k then $\theta_{ij}(k) > 0 \forall i, j$ at each k. This says if each state i is visited infinitely often, then each control action j associated with state i is visited infinitely often by the SLFCN algorithm.

Proof

With $\theta_{ij}(k) = c_{ij}(k)$, the result follows from Lemma 2, part (i).

Comment

The controller depends on the process being recurrent, so that every state is visited infinitely often. It may be naturally recurrent, as with the case of a stochastic recurrent Markov process, or artificially made recurrent. For example, if the controller is applied to learning a robotic motion or machining process motion, a series of learning trials would be conducted, with the initial state of the system for each trial set randomly.

Proposition 2

The correlation parameter vector $\mathbf{c}(k)$ is a probability distribution vector for all $k > 1$, given that $c(0)$ is a distribution vector.

Proof

The result follows from Lemma 2.

Proposition 3

$E\{\mathbf{f}(k)\}$ given by Eq. (5.49) is similar to $T_1(z)$.

Proof $E\{\boldsymbol{f}_i(k)\} = T_1(z)$, and therefore the distribution vectors are similar.

Lemma 3

Suppose we have a set $X = \{x | x \in (0,1)\}$, an arbitrary function $a : X \times X \rightarrow (-1, 1]$, and random variables $X_1, X_2 \in X$ with means z_1 and z_2 and distributions F_1 and F_2 which differ only in their mean value (i.e., the distributions have the same shape).

If $z_1 > z_2$ and $a(z,x)$ is an increasing function in $z \forall x \in X$, then

$$E\{a(z_1, X_1)X_1\} > E\{a(z_2, X_2)X_2\}.$$

Proof

For each $i = 1, 2$, the expected value is simply a sum (integral)

$$E\{a(z_i, X_i)X_i\} = \int_X a(z_i, x)x dF_i(x).$$

For each $x, a(z_1, x) > a(z_2, x)$, since $z_1 > z_2$, and $a(z,x)$ is an increasing function in z. Since the distributions have the same shape, their shifted probability density functions are equal. Therefore, we have on the left side a weighted sum with the same weights as the right side, but in which each summand is larger than the corresponding summand on the right. Hence

$$\int_X a(z_1, x)x dF_1(x) > \int_X a(z_2, x)x dF_2(x)$$, and the result follows.

A condition for the next theorem is that the probability distribution for any component of the distribution $\mathbf{c}_i$ is the same as any other, except for their means. The SLFCN algorithm selects each j^* at random according to $\mathbf{c}_i$. If $\mathbf{c}_i$ were always uniform, it is easy to see that each component would be sampled in the same way, and thus would have the same sampling distribution. In actuality, the control corresponding to a larger component would be selected more often, which one might think would result in a smaller variance. However, the larger the component, the greater are the algorithm increments away from the mean (recall the δ factor). Therefore, the assumption is not unreasonable. In any case, the more concentrated the distribution (i.e., the closer the density approaches a spike at the mean value), the closer to true is the conclusion of Lemma 3, regardless of whether the distribution shapes are the same or not. All of the sampling distributions become more concentrated as the sample size k increases.

Theorem 3 *Let $\mathbf{c}$ be the correlation vector computed by SLFCN algorithm of Eqs. (5.22-5.32), and z be the (fixed) reinforcement vector, respectively for any arbitrary index i. Assume that the distribution of a component $\mathbf{c}_j$ differs from any other component's distribution only in it mean value. Then, the distribution $E\{\mathbf{c}(k)\}$ is similar to the distribution $T_1(z)$ at each step k.*

Proof

Let

$$\delta_j(k) = \begin{cases} c_j(k) & \text{if } z_j < 0 \\ 1 - c_j(k) & \text{if } z_j \geq 0 \end{cases}$$

Then, the correlation parameter update given in Eq. (5.26) can be re-written as

$$
\begin{aligned}
c_j(k+1) &= \frac{c_j(k)+\gamma\delta_j(k)\zeta_j(k+1)}{1+\gamma\delta_{j^*}(k)z_{j^*}} \\
&= \frac{c_j(k)}{1+\gamma\delta_{j^*}(k)z_{j^*}} + \frac{\gamma\delta_j(k)\zeta_j(k+1)}{1+\gamma\delta_{j^*}(k)z_{j^*}} \\
c(k+1) &= \left[\frac{c_1(k)}{1+\gamma\delta_{j^*}(k)z_{j^*}},\ldots,\frac{c_r(k)}{1+\gamma\delta_{j^*}(k)z_{j^*}}\right]^T \\
&\qquad + \left[\frac{\gamma\delta_1(k)\zeta_1(k+1)}{1+\gamma\delta_{j^*}(k)z_{j^*}},\ldots,\frac{\gamma\delta_r(k)\zeta_r(k+1)}{1+\gamma\delta_{j^*}(k)z_{j^*}}\right]^T \\
&= \left[\frac{c_1(k)}{1+\gamma\delta_{j^*}(k)z_{j^*}},\ldots,\frac{c_r(k)}{1+\gamma\delta_{j^*}(k)z_{j^*}}\right]^T \\
&\qquad + \left[0,\ldots,\frac{\gamma\delta_{j^*}(k)z_{j^*}}{1+\gamma\delta_{j^*}(k)z_{j^*}},\ldots,0\right]^T \\
&= \frac{1}{1+\gamma\delta_{j^*}(k)z_{j^*}}[c_1(k),\ldots,c_r(k)]^T \\
&\qquad + \frac{\gamma\delta_{j^*}(k)z_{j^*}}{1+\gamma\delta_{j^*}(k)z_{j^*}}[0,\ldots,1,\ldots,0]^T \\
&= \frac{1}{1+\omega_{j^*}(k)}[c_1(k),\ldots,c_r(k)]^T + \frac{\omega_{j^*}(k)}{1+\omega_{j^*}(k)}[0,\ldots,1,\ldots,0]^T \\
&= (1-w_{j^*}(k))[c_1(k),\ldots,c_r(k)]^T + w_{j^*}(k)[0,\ldots,1,\ldots,0]^T
\end{aligned}
$$

Finally,

$$c(k+1) = (1-w_{j^*}(k))\mathbf{c}(k) + w_{j^*}(k)T(\zeta(k+1))$$

where

$$w_j(k) = \frac{\omega_j(k)}{1+\omega_j(k)} = \frac{\gamma\delta_j(k)z_j}{1+\gamma\delta_j(k)z_j}$$
$$T(\zeta(k)) = \zeta(k)/\zeta_{j^*}(k)$$

Thus, $\mathbf{c}(k)$ is a normalized frequency count of, and hence an estimator of $T(\zeta(k))$.

Next, the expected value is computed. The random variables are $j^* = j^*(k)$ and $\mathbf{c}(k)$. The expectation is the total expectation over both $\mathbf{c}(k)$ and j^*. Note, however, that the stochastic event at step $k+1$ which is used to choose j^* is independent of all previous events used to determine the value of $\mathbf{c}(k)$ through step k, therefore, $\mathbf{c}(k)$ and $j^*(k+1)$ as random variables are independent (even though the value of $c(k)$ is the distribution vector used to help choose j^*). Thus, the expectation operators $E_{\mathbf{c}}$ and E_{j^*} of the two can be separated. Since the $j^*(k)^{th}$ entry of the vector $\zeta(k)$ is nonzero with probability $c_{j^*}(k)$, then the expectation with respect to the action probabilities is simply z_j weighted by $c_j(k)$ for each component j. Hence, we have

$$E\{\mathbf{c}(k+1)\} = E\left\{(1-w_{j^*}(k))[c_1(k),\ldots,c_r(k)]^T + w_{j^*}(k)[0,\ldots,1,\ldots,0]^T\right\}$$

$$= E_{\mathbf{c}}\left\{E\{(1-w_{j^*}(k))[c_1(k),\ldots,c_r(k)]^T + w_{j^*}(k)[0,\ldots,1,\ldots,0]^T|\mathbf{c}(k)\}\right\}$$

$$= E_{\boldsymbol{c}} \left\{ \sum_{j=1}^{r} c_j(k)(1 - w_j(k))[c_1(k), \ldots, c_r(k)]^T \right\}$$

$$+E_{\boldsymbol{c}}\{[w_1(k)c_1(k), \ldots, w_r(k)c_r(k)]^T\}$$

Thus

$$E\{\boldsymbol{c}(k+1)\} = E_{\boldsymbol{c}}\{(1 - \overline{w}(k))[c_1(k), \ldots, c_r(k)]^T\}$$

$$+E_{\boldsymbol{c}}\{[w_1(k)c_1(k), \ldots, w_r(k)c_r(k)]^T\} \tag{3.7}$$

where

$$w(k) = E_{j^*}\{w_j(k)\}$$

$$1 - \overline{w}(k) = E_{j^*}\{1 - w(k)\} = E_{j^*}\left\{\frac{1}{1 + \gamma\delta_j(k)z_j}\right\} = \sum_{j=1}^{r} \frac{c_j(k)}{1 + \gamma\delta_j(k)z_j}$$

The rest of the proof proceeds by induction, using Eq. (5.50).

Base Step Claim: At $k = 1, E\{\boldsymbol{c}(1)\}$ is similar to $T_1(z)$.

Proof

Initially, $\boldsymbol{c}(0)$ is fixed at a uniform distribution, so $E\{\boldsymbol{c}(0)\} = \boldsymbol{c}(0) = [\frac{1}{r}, \ldots, \frac{1}{r}]^T$. Thus, from Eq. (5.50) we have

$$E\{\boldsymbol{c}(1)\} = \frac{1}{r}(1 - \overline{w}(0)) \left[\frac{1}{r}, \ldots, \frac{1}{r}\right]^T + \frac{1}{r}[w_1(0), \ldots, w_r(0)]^T$$

Let (i) be the indices according to the descending order of z. Then test the relations:

$$E\{c_{(i-1)}\} \overset{?}{>} E\{c_{(i)}\} \overset{?}{>} E\{c_{(i+1)}\}$$

Let $K = \frac{1}{r}(1 - \overline{w}(0))\frac{1}{r}$, which is constant with respect to the control action index (i). Then we have

$$K + \frac{1}{r}w_{(i-1)} \overset{?}{>} K + \frac{1}{r}w_{(i)} \overset{?}{>} K + \frac{1}{r}w_{(i-1)}$$

Subtract K from each part of the inequality and expand each w_i to obtain

$$\frac{1}{r}\frac{\gamma\delta_{(i-1)}z_{(i-1)}}{1 + \gamma\delta_{(i-1)}z_{(i-1)}}) \overset{?}{>} \frac{1}{r}\frac{\gamma\delta_{(i)}z_{(i)}}{1 + \gamma\delta_{(i)}z_{(i)}}) \overset{?}{>} \frac{1}{r}\frac{\gamma\delta_{(i+1)}z_{(i+1)}}{1 + \gamma\delta_{(i+1)}z_{(i+1)}}$$

Observe that $\delta_j = \frac{1}{r}$ when $z_j < 0$ and $\delta_j = 1 - \frac{1}{r}$ when $z \geq 0$. It is easy to verify that the function

$$w(z) = \begin{cases} \frac{\gamma\frac{1}{r}z}{1+\gamma\frac{1}{r}z} & \text{if } z < 0 \\ \\ \frac{\gamma(1-\frac{1}{r})z}{1+\gamma(1-\frac{1}{r})z} & \text{if } z \geq 0 \end{cases}$$

is an increasing function of z on the interval (-1,0], for fixed γ and r. Hence, since $z_{(i-1)} > z_{(9i)} > z_{(i+1)}$, it is clear that the relations are true. Therefore $E\{c(1)\}$ is similar to $T_1(z)$.

Induction step:

If $E\{c(k)\}$ is similar to z at an arbitrary step k, then $E\{c(k+1)\}$ is similar to z at step $k+1$.

Proof

We have

$$E\{c(k+1)\} = E_c(1-\overline{w}(k))[c_1(k),\ldots,c_r(k)]^T\} + E_c\{[w_1(k)c_1(k),\ldots,w_r(k)c_r(k)]^T\}$$

Let (i) be the indices according to the descending order of z. Then test the relations:

$$E_c\{(1-\overline{w}(k))c_{(i-1}\} + E_c\{w_{(i-1)}(k)c_{(i-1)}(k)\} \overset{?}{>} E_c\{(1-\overline{w}(k))c_{(i)}\}E_c\{w_{(i)}(k)c_{(I)}(k)\}$$

$$\overset{?}{>} E_c\{(1-\overline{w}(k))c_{(i+1)}\} + E_c\{w_{(i+1)}(k)c_{(I+1)}(k)\} \tag{3.8}$$

Consider the first term in each part. The quantity $(1-\overline{w}(k))$ is not a function of the index and multiplies every c_j, and so does not affect the ordering (invoking Lemma 3). Since by assumption we know that $E\{c(k)\}$ has the same ordering as z, then we have that

$$E_c\{(1-\overline{w}(k))c_{(i-1)}\} > E_c\{(1-\overline{w}(k))c_{(i)}\} > E_c\{(1-\overline{w}(k))c_{(i+1)}\} \tag{3.9}$$

Next we examine the second term in each part of the inequality relations. Consider the function

$$w(cz) = \begin{cases} \frac{\gamma cz}{1+\gamma cz} & \text{if } z < 0 \\ \\ \frac{\gamma(1-c)z}{1+\gamma(1-c)z} & \text{if } z \geq 0 \end{cases}$$

Since this is an increasing function of z on (-1,1] for every fixed c, then by Lemma 3,

$$E_c\{w_{(i-1)}(k)c_{(i-1)}(k)\} > E_c\{w_{9i)}(k)c_{(I)}(k)\} > E_c\{w_{(i+1)}(k)c_{(I+1)}(k)\} \tag{3.10}$$

Summing the inequalities in (5.52) and (5.53), we see that the relations in (5.51) hold and the induction step is proved. Thus, the theorem is proved.

Q.E.D.

Comment

The result of this theorem is essentially equivalent to showing that the learning algorithm is expedient (see[107]).

Comment

In practice, statistical estimates of the z_{ij} provided by the PES are used, and so are subject to additional random uncertainty, and inaccuracy early in the learning procedure. Also, since it is the expected value of $c(k)$ which is considered, there is always the possibility in practice of "unfortunate" sample paths.

Comment

No theoretical restriction on γ has been established, other than $0 < \gamma < 1$. In practice, values as large as 0.75 have been used with no ill-effects on the performance of this particular algorithm. However, the speed with which this algorithm learns impacts on the other algorithms of the controller, namely, the location parameter update and the performance estimates. A balance must be achieved with the parameters of those algorithms.

We have shown that $E\{\boldsymbol{c}_i\}$ is similar to z_i for any i, which means that the matrix $E\{\boldsymbol{C}\}$ is similar to $\boldsymbol{Z}$, by which we mean that the row vectors are similar. The SLFCN node for each i does not just learn individually, but shares what it has learned with its neighbors.

Def. 7 Node N_i is said to learn at step k if $| \Delta c_{ij}(k)| > 0$ for any j, and is said to have learned (to some degree) by the time step K if $| \Delta c_{ij}(k)| > 0$ for any $k \leq K$, for any j.

Def. 8 The neighbors of a node N_{i^*} are all N_i such that $N_i \in \eta_c(k, i^*)$.

Def. 9 A node N_{i^*} is said to share its learning at time step k if $\eta_c(k, i^*) \neq \oslash$ and for all $N_i \in \eta_c(k, i^*)$ and all j, the following hold:

i) $| \Delta c_{ij}(k)| > 0$

ii) $| \Delta c_{i^*j}(k) - \Delta c_{ij}(k)| < \epsilon$ for some small $\epsilon s.t. 0 < \epsilon << 1$.

Proposition 4

At each step k of the SLFCN algorithm for which $\eta_c(k, i^*) \neq \oslash$ and $\mu_{X_i}(\boldsymbol{x}(k)) > 0$, node N_{i^*} learns, shares its learning, and all of its neighbors learn.

Proof

Follows immediately from the update algorithm for $c_{ij}(k)$ and the above definitions. (It is shown later that the condition $\mu_{X_i}(\boldsymbol{x}(k)) > 0$ for all $\boldsymbol{x}(k)$ holds.)

Next it is shown that the SLFCN and IFCN estimate a fuzzy relation. First, we present some more definitions. Let G be a fuzzy relation, which is defined as the membership function $G : X \times U \rightarrow [0,1]$ such that $G(x,u) =$ degree { state x relates to u}. The relation G represents the control law of the fuzzy controller.

Def. 10 We say that $\boldsymbol{x}$ relates to $\boldsymbol{u}$ if $\boldsymbol{u} \in \mathcal{U}(x) = \{\boldsymbol{u} \mid u$ is the correct control given state $\boldsymbol{x}\}$.

Let $\mu_{\mathcal{U}(\boldsymbol{x})}(\boldsymbol{u}) =$ degree $\{\boldsymbol{u} \in \mathcal{U}(\boldsymbol{x}) | \boldsymbol{x}(k) = \boldsymbol{x}\}$. Then $G(\boldsymbol{x},\boldsymbol{u}) = \mu_{\mathcal{U}(\boldsymbol{x})}(\boldsymbol{u})$ is the correct control law by some criterion used to define $\mathcal{U}(\boldsymbol{x})$. We suppose that the fuzzy control law can be approximated by a finite set of fuzzy rules of the form

$$\text{if } \boldsymbol{x} \text{ is } X_i, \text{ then } \boldsymbol{u} \text{ is } U_j$$

which can be written compactly as $X_i \Rightarrow U_j$, according to the fuzzy discretization given by the referential fuzzy sets $\{X_1, X_2, \ldots, X_s\}$ for state space X and $\{U_1, U_2, \ldots, U_r\}$ for the control space U. It is assumed that every point in X belongs to at least one of the X_i's, and similarly for U. Hence, we approximate G by

$$G = \bigcup_{ij} (X_i \Rightarrow U_j)$$

Then $\tilde{G}$ can be represented by a fuzzy relation matrix $\boldsymbol{G_0} = [g_{ij}]$ where

$$g_{ij} = \mu_{X_i}(\boldsymbol{x}(k)) \bullet \mu_{U_j}(\boldsymbol{u}(k)) \bullet \mu_{\mathcal{U}(\boldsymbol{x})}(\boldsymbol{u}) \approx \text{ degree } \{(\boldsymbol{x}(k), \boldsymbol{u}(k)) \in G\}$$

and where $\bullet$ is a t-norm implication operator (here the t-norm type of implication is used). $\tilde{G}$ is known as a referential fuzzy relation [122].

Def. 11 A matrix $\boldsymbol{C}$ is a proper estimator of a matrix $\boldsymbol{Z}$ if $\boldsymbol{C}$ is an estimator of $\boldsymbol{Z}$ and $E\{\boldsymbol{C}\} \cong \boldsymbol{Z}$.

Theorem 4 *The SLFCN algorithm of Eqs. (5.22-5.32) generates a proper estimate of the fuzzy relation matrix* $\boldsymbol{G_0}$.

Proof

¿From the definition of reinforcement,

$$T(r(t)) = \mu_{\mathcal{U}(\boldsymbol{x})}(\boldsymbol{u}(t)) = \text{ degree } \{\boldsymbol{u}(t) \in \mathcal{U}(\boldsymbol{x}) | \boldsymbol{x}\ (t-1)\ = \boldsymbol{x}, \boldsymbol{u}\ (t)\ = \boldsymbol{u}\}$$

that is, $T(r(k))$ is the degree to which control $\boldsymbol{u}(k)$ is a member of the set of correct controls given state $\boldsymbol{x}(k-1)$. We have

$$\zeta_i(k) = [0, 0, \ldots, a_i(k) b_j \cdot r(k), \ldots, 0]^T$$

$$E\{\zeta_i(k)\} = [z_1, \ldots, z_r]^T = [a_i(k)b_1 y_1, \ldots, a_i(k)b_r y_r]^T$$

$$\begin{aligned} a_i(\boldsymbol{x}(k)) &= \text{degree } \{\boldsymbol{x}(k) \in X_i\} = \mu_{X_i}(\boldsymbol{x}(k)) \\ b_j(\boldsymbol{u}(k)) &= \text{degree } \{\boldsymbol{u}(k) \in U_j\} = \mu_{U_j}(\boldsymbol{u}(k)) \end{aligned}$$

Thus

$$\begin{aligned} E\{T(\zeta_{ij}(k))\} &= T(a_i(k)b_j(k)y_j) = T(z_{ij}) \\ &= \mu_{X_i}(\boldsymbol{x}(k)) \bullet \mu_{U_j}(\boldsymbol{u}(k)) \bullet \mu_{\mathcal{U}(\boldsymbol{x})}(\boldsymbol{u}) \\ &= g_{ij} \approx \text{ degree } \{(\boldsymbol{x}(k), \boldsymbol{u}(k)) \in G\} \end{aligned}$$

¿From Theorem 3, we have that $c_{ij}(k)$ is a normalized frequency count (at step k) of z_{ij} and that $E\{\boldsymbol{C}(k)\} \cong T(\boldsymbol{Z}) = T(\boldsymbol{a}_i^T \boldsymbol{Y})$. Hence, since $T(z_{ij}) = g_{ij} \approx$ degree $\{(\boldsymbol{x}(k), \boldsymbol{u}(k)) \in G\}, c_{ij}(k)$ is the normalized frequency count of the fuzzy event $(\boldsymbol{x}(k), \boldsymbol{u}(k)) \in G$, and so is an estimator of g_{ij}, furthermore, $E\{\boldsymbol{C}(k)\} \cong \boldsymbol{G}_0$. Therefore, $E\{\boldsymbol{C}(k)\}$ is a proper estimator of relation matrix $\boldsymbol{G}_0$. Q.E.D.

Comment

The rows of $\boldsymbol{C}$ are probability distributions. Rather than use probability distributions c_i for the fuzzy membership distributions $\boldsymbol{g}_i$, any of a number of different methods for deriving a membership distribution from a probability distribution may be used to obtain an estimator $\boldsymbol{G}$ for $\boldsymbol{G}_0$ from $\boldsymbol{C}$. In this case, we have used a simple low pass filter given in Eq. (5.34), which emphasizes the most certain information and discounts or discards the most uncertain.

Comment

It has not been very important here that estimates be anything more than similar (as opposed to being equal or exactly proportional) for several reasons. First, what is estimated is a normalized frequency distribution, i.e., an empirical probability, which is typically transformed into a fuzzy distribution by non-standard, heuristically-guided means anyway. Second, the precise scaling of the distribution is not critical, since the defuzzification process such as COA normalizes by the total sum of the components of the distribution the distribution-weighted sum of the controls. Whether the estimate $\boldsymbol{G}$ provides optimal control by the definition given in Sct. 5.2.3 depends on the choice of operators. If the operators were ordinary sum and product (linear but not fuzzy operators), then the optimal choice of $\boldsymbol{G}$ is $\boldsymbol{G} = \boldsymbol{Y}$, which is easy to show using linear algebra and our definition of optimality (Sct. 5.2.3). For other choices, the optimal $\boldsymbol{G}$ would be close to $\boldsymbol{Y}$ in some sense.

Comment

Note that in the learning mode, when $\bar{a}_i$ is used in place of a_i, then

$$b_j(\boldsymbol{u}(k)) = \begin{cases} 1 & \text{if } j = j^* \\ 0 & \text{otherwise} \end{cases}$$

Therefore, in the learning phase, it can easily be shown that the result of the implication $a_i(k)b_j(k)r(k)$ (whose frequency count through k is measured by c_{ij}) is the same for most common t-norm implication operators, including min and product.

The parameters c_{ij} can be viewed as crisp random variables which give the crisp probability measure on the fuzzy events $\{(\boldsymbol{x}(k), \boldsymbol{u}(k)) \in G\}$. The random variable is the reinforcement quantity $r(k)$, which is a crisp measure of the fuzzy event $\{\boldsymbol{u}(t) \in \mathcal{U}\ (x\ \}$. This use of crisp measures of fuzzy statistics is addressed in [65], and [33]; see also [111].

Next, we demonstrate some of the properties of the set of rules embodied by the estimated relation matrix $\boldsymbol{G}$.

Def. 12 (from [123]). If $\mu_{X_i}(\boldsymbol{x}(k)) > \varepsilon$ for every $\boldsymbol{x} \in X$, for some $\varepsilon \in (0,1]$ and for some fuzzy rule $X_i \Rightarrow U_j$, then the set of rules represented by $\tilde{G}$ is said to be a *complete* set of rules, or to have the *completeness* property. Thus, a set of rules is complete if the controller can generate a control for any fuzzy process state.

Lemma 4

If the SFDN is implemented with gaussian membership functions, then the set of rules generated by the SLFCN/IFCN represented as the relation matrix $\boldsymbol{G}$ is complete.

Proof

We have that

$$\mu_{X_i}(x(k)) = a_i(x(k)) = \exp(-\|\boldsymbol{x}(k) - \boldsymbol{m}_i\|/s_{i^*}) > 0 \quad \forall i, \forall \boldsymbol{x} \in X$$

and

$$g_{ij}(k) = \mu_{X_i}(x(k)) \cdot \mu_{U_j}(u(k)) \cdot \mu_{\mathcal{U}(x)}(u)$$

Hence, all the rules represented by the g_{ij} in $\boldsymbol{G}$ satisfy the definition of completeness.

Comment

With asymptotically sloping functions like a gaussian, the entire input space is covered, even if the rule base is sparse, although the coverage may not be strong. The fuzziness parameter ϕ can be set so that, on the average, ε is at a certain minimum level, as follows. Let $d_{max} = \max_{i,i'}\{\|\boldsymbol{x}_{i'} - \boldsymbol{x}_i\|\}$. Then

$$\mu_{X_i}(\boldsymbol{x}(k)) = a_i(\boldsymbol{x}(k)) \quad = \quad \exp(-\|\boldsymbol{x}(k) - \boldsymbol{m}_i\|/s_i)$$

$$\geq \quad \exp(-d^2_{\max}/s_i)$$

$$\exp(-d^2_{\max}/s_i) > \varepsilon \quad \Rightarrow \quad s_i < 1n(1/\varepsilon)d^2_{\max}$$

Since ϕ (roughly) governs the average value of the s_i's, choose $\phi < 1n(1/\varepsilon)d^2_{\max}$ for the desired value of ε.

The usual meaning of rule consistency is that any two rules with different antecedents should not have the same consequent. When the antecedents of fuzzy rules are associated with each fuzzy set corresponding to each node in the state space map, the resulting rule base may not be consistent, since it is quite reasonable for different parts of the state space to require the same control, particularly in the case of neighboring regions. However, clustering similar terms together into one term allows the rules to be consistent. The SFDN algorithm of the SFAL controller tends to cluster nodes together that have a similar control in a manner that reflects the distribution of appropriate controls assigned to states in the state space (by Theorem 2 when the information includes the factor $\sigma(g_i, g_{i\bullet})$). By associating the fuzzy control rules with these clusters of nodes rather than simply the fuzzy sets of states that these nodes represent, the rule base so generated is automatically consistent. This can be seen as follows.

In general, a fuzzy set A in a discrete space $X_F = \{x_1, x_2, \ldots, x_n\}$ is defined by a discrete membership function which is a relation on X_F x [0,1] which assigns to each element in a subset S of X (called the support of A), a real number from the interval [0,1]. For example, the fuzzy set A can be described by the membership function $\{(x_2, 0.3), (x_5, 0.8), (x_8, 0.24)\}$. If the original space X is continuous, then a discretization X_F of X must first be constructed, so that any element x of X can first be assigned to an element x_i in X_F.

Define a collection of fuzzy subsets $\{\tilde{X}_i : i = 1, \ldots, r\}$ of the discrete space $X_F = \{X_1, \ldots, X_s\}$ as follows. The support for $\tilde{X}_i$ is

$$S_i = \{X_i | c_{ii} \geq c_{ij} \forall j\}$$

Then the membership function for $\tilde{X}_i$ is given by the relation

$$\{(X_i, \mu_{X_i}(\boldsymbol{x})) | X_i \in S_i\}$$

Hence, $\{\tilde{X}_i : i = 1, \ldots, r\}$ is really a family of collections of subsets $\{\tilde{X}_i(\boldsymbol{x}) : i = 1, \ldots, r\}$ of the discrete space X_F parameterized by the state $\boldsymbol{x} \in X$. Alternatively, $\{\tilde{X}_i(\boldsymbol{x}) : i = 1, \ldots, r\}$ is a collection of subsets of the continuous space X, where the membership functions are

$$\mu_{\tilde{X}_i}(\boldsymbol{x}) = \begin{cases} \mu_{X_i}(\boldsymbol{x}) & \text{if } X_i \in S_i \\ 0 & \text{otherwise} \end{cases}$$

The corresponding fuzzy control rules are then

$$\text{If } \boldsymbol{x} \text{ is } \tilde{X}_i, \text{ then } \boldsymbol{u} \text{ is } U_i$$

What we have done is define membership functions on the *fuzzy* discretized space X_F of the original space X, but these are piecewise continuous rather than discrete membership functions. The support S_i

for fuzzy subset $\tilde{X}_i$ corresponds to all those nodes N_i of the SFDN for which the strongest association c_{ij} is for the i^{th} control action U_i. The rules could be rephrased as

If the state is a state for which control i should be used, then use control i.

Hence, the corresponding fuzzy control rules are automatically consistent, since the fuzzy sets are defined by whether or not X_i is most strongly associated with the i^{th} control action U_i.

Def. 13 The set of rules represented by $\tilde{G}$ interact if whenever $(X_i, U_j) \in G$ then

$$X_i \circ \tilde{G} \neq U_j.$$

By this definition, most practical sets of rules interact. What is usually given is some type of measure of the degree of interaction (see [123, p. 112-126]); too much interaction is considered undesirable. In the case here, a gross determination of rule interaction is given as follows:

$$\text{if } \max_j\{b_j\} \neq \max_j\{g_{i*j}\} \text{ where } \boldsymbol{b}^T = \boldsymbol{a}_i^T(x(k))\boldsymbol{G}, \text{ then the rules of } \boldsymbol{G} \text{ interact.}$$

For the algorithms of this proposed controller, it cannot be guaranteed that the rules do not interact, even by this latter criterion. In the simulation experiments reported in Chapter VI and VII, it has been observed that rule interaction is rare except near a control switching curve. For example, a node near an equilibrium that is the set-point of dynamical process may favor most strongly a nonzero control value, but the most favored control value of the fuzzy composition is near zero, as it should be near an equilibrium set-point.

3.2 Comparisons to other Methods

3.2.1 Adaptation of Rules versus Adaptation of Membership Functions

A common approach in many "learning" or automatic fuzzy control derivation methods is as follows. First, a fixed or variable number of possible fuzzy subsets of the state space and of the control space are set up; then a fixed or variable number of links between state subsets and control subsets are made; then a training algorithm is run which determines the parameters that define the fuzzy subsets. In some approaches, the initial membership function parameters are set arbitrarily (e.g., small random values) and an arbitrary connection is made between each state fuzzy set and some control fuzzy set; since the initial sets are arbitrary, it does not matter which connects to which. A connection defines a rule. The training algorithm then adjusts the membership function parameters of the state fuzzy set and/or control fuzzy set which are paired together so that the pairing is appropriate for providing good control. Often it is only the control sets which are adjusted, and sometimes only the location parameter, not the spreads of

the membership function (e.g., [85]). Others, (e.g., [64] and [88]) set up all possible connections between every state-control pair and allow the adaptation algorithm to establish the appropriate links; Jang [64] allows the connections to be continuously variable weights, while Lin and Lee [88] use a mechanism which establishes a connection strength of 1 for the rule with the maximum weight after some training, and sets to 0 the weights to all other control subsets. In either case, most of the work in establishing the rules is in the determination of membership functions with possibly some consideration given after some training to modifying the rule association *perse*.

There is some debate about the most appropriate way to adapt a fuzzy rule base. Some authors argue that the way to adapt a system to new or unknown environment is to keep the control rules in the knowledge-base unchanged and modify the definitions of the membership functions instead [40, p. 402]. Whereas Altrock, et al. argue

> State of the art is now to tune the membership functions, so that the behavior desired is established. This clearly violates the concept of fuzzy control. The membership functions shall represent the engineer's linguistic concept of physical figures to be self-explanatory. To fumble around with the membership functions for the fine-tuning of a control strategy ... is therefore counterproductive, especially the more complex a problem gets. ... The solution is simple: If one wants to fine-tune the control strategy, one should not touch the membership functions, but tune the rules [4, p. 836-7].

The approach taken with the SFAL controller is that of representing the rules in the form of a (fuzzy) relation matrix, and then modifying the rules by modifying the entries in the relation matrix. In this way, the membership functions may be separately defined and either fixed or themselves adapted; in this case, the state membership functions are adapted, but not those of the controls. It should be noted that the objective here is not really fine tuning, but to establish a basic set of rules in the first place. Any of a number of fine tuning approaches could be added on to the basic mechanism here. A significant challenge, though, has been to find an effective way to generate a set of rules automatically in situations where very little is known *apriori* about the plant.

In back propagation methods, the membership functions are adjusted according to the change in the error function or other measure of performance [64], [88]. This is in general an advantage over methods which use competitive self-organizing to adjust the locations of fuzzy set prototype or central members based only on the distribution of the inputs, for reasons already discussed (autocorrelation of sequence of states in dynamic process, spatial distribution does not contain information directly about performance). For methods based on generating good rules by adjusting the membership functions, this is critical. However, a gradient-type method applied to the membership functions can lead to strange and counter-intuitive fuzzy sets, which often end up not really overlapping, in which case the fuzzy inference does no interpolation [64], [13].

The SFAL controller adapts both the rules in the form of a relation and the membership functions. The relation performs the weighting and balancing and interpolation among the prototype members of the fuzzy sets, so that their actual values are important but not critical. Hence, less of the burden is

on the adaptation of the locations and spreads of the membership functions for the state space, and really none is needed for the control space; a uniform grid suffices. The adaptation of the membership functions is done with a competitive self-organization process, rather than applying a gradient method to minimize directly a performance function. Furthermore, the basis of the location updates at each step is not simply the distribution of the states, but also includes information about what controls have been found to be effective in each region of the state space. Thus, the membership functions are adjusted to cover according to where performance information is available.

The way in which the relation is adapted is comparable to the identification methods used in [175], [55], [122], and others. While the task in the SFAL controller is not identification, and the problem is not posed in terms of fitting a fuzzy model to a set of input/output data points, some ideas are shared in common. In [175] and [55], the relation adaptation algorithms include a forgetting factor for discounting old associations in favor of new ones. This is essentially a delta rule approach. The proposed method accomplishes the weighting of new versus old information differently. The parameters c_{ij} in the sector c_i are simply normalized after each increment to one of the parameters, and the increment size is scaled according to how close the current parameter is to 0 or 1 in the direction of the change. This tends to make information that is more certain (indicated by a parameter's closeness to 1 or 0) to be discounted *less* than more ambiguous information (indicated by values near $1/r$). This has been found in practice to work much better than arbitrary exponential forgetting as occurs with the delta rule.

3.2.2 Use of Generalization in Associative Reinforcement Learning Methods

The learning system should perform an approximation of the best control policy locally based on the sample of events $e(t)$ drawn from the probability space and then perform a generalization from that sample by applying what it has learned to parts of the space not directly experienced but similar to what it has seen. A crisp discrete context-based learning system, such as the one described in [101] does not perform such a generalization. As pointed out by Narendra, the method of using a finite number of crisp regions to partition the state space may be effective when he number of distinct context vectors is small, but it becomes inefficient when the number is very large and impractical when the context space is continuous [108, p. 20]. In [108], a system of SLAs is augmented by a back propagation neural network to perform the generalization and to help address the dimensionality issue. CMAC [1] uses overlapping crisp regions to perform a kind of generalization; radial basis function methods [23] and Franklin's method [43] use what may be described as overlapping gaussian membership functions; however, these use a linear recombination, which is inference with ordinary sum and product, and it has been shown that nonlinearity is necessary to obtain universal function approximation capability [28], [77]. Jang [64] and Lin and Lee [88] use membership functions and fuzzy inference, but their methods, like the other just mentioned, do no employ the mathematical learning theory involving experimental hypotheses and trials. Instead,

these methods, like Franklin's method, employ gradient back propagation. Throughout this work, we have made a case concerning the relative advantages and disadvantages of back propagation methods and learning trial methods.

The generalization addressed above is in terms of the inference process to obtain an output using what has been learned. None of the discrete learning trial methods found in the literature use generalization in the incremental updating algorithms themselves. However, the concept has been used in other types of adaptive algorithms. For example, the approach of adapting a neighborhood of cells around the active one is proposed by Gardner [48]. The main differences are that Gardener's controller is completely deterministic, uses a fixed performance measure table and is set up for PD control of a one dimensional input state. The SFAL controller is stochastic to provide good sampling for estimates, does not rely on a pre-determined performance table, and accommodates multi-dimensional input states. Also, the control provided by the SOC is tuned to the initial state used in the training, whereas the SFAL controller learns a control law over the state space.

3.2.3 Comparison of Methods to Perform Fuzzy Discretization

The computational complexity of the SFAL controller, the size of the controller need not increase exponentially with the size of the problem in terms of state and control dimensions. The SFAL controller employs a vector fuzzy quantization to obtain fuzzy sets in a feature or state space, hence the objects are actually fuzzy patterns, rather than an assembly of fuzzy intervals in a multidimensional space. As discussed earlier, this permits redundancy of information in simultaneous directions to be exploited in the fuzzy inference process, so that the density of coverage need not be as great. As observed by Horikawa, et al., "The characteristics of a fuzzy model depend heavily on the structures rather than the parameters of the membership functions in the premises [60]." As discussed in Sct. 5.4.2, the actual fuzzy sets of states, characterized by which control is used for them, is a composition of the fuzzy discretized neighborhoods for each state reference. These neighborhoods are elliptical (from gaussian membership functions), and the fuzzy set shapes are whatever irregular shape can be composed of these ellipses.

Other methods have been proposed which fuzzify vectors in a state space or feature space. A similar approach is taken by Sultan and Janabi with their so-called "Clearness Transformation Fuzzy Logic Controller" [149]. Hayashi, et al., in their work treat the inputs as a vector with membership in vector fuzzy sets [56]. The use of fuzzy sets of vectors in the state space is also considered in Berenji and Khedar [12] who use a gradient method on a training data set to adapt gaussian-shaped fuzzy clusters in product space.

3.2.4 Comparison to SOC Fuzzy Controller

The Procyk and Mamdani controller [127] needs no precise, explicit model of the process and need no explicit derivatives with respect to controller parameters, and so shares these desirable properties with the SFAL controller. Their method does require, however, at least a crude inverse plant model and the specification of an index for the desired performance as a function of quantities computed from the system state. Though the procedures of Procyk and Mamdani have worked reasonably well in various situations, they may be difficult to apply to more complex systems and multi-input multi-output processes, as the authors themselves have pointed out. The SFAL controller does not rely on a predetermined performance index table which may be difficult to generate for many types of problems, such as the combinatorial-type of control problems found in manufacturing and large operations, in contrast to ordinary position-velocity regulator problems for which performance index tables have been developed.

Chapter 4

The SFAL-C Controller: Extensions and Enhancements

A number of improvements and modifications have been made to the Statistical Fuzzy Associative Learning Controller (SFAL-C) previously reported by Esogbue and Murrell and presented in the foregoing chapters. As stated there, the SFAL-C integrates mathematical learning theory, neural networks, fuzzy sets, and statistical modeling as a framework for intelligent control. Using the SFAL-C as a point of departure, two related algorithms, the Continuous Action Space (CAS) and Generalized Continuous Space (GCS-Δ) reinforcement learning controllers, have been proposed and investigated. The following compares the primary components of each reinforcement learning controller and the respective advantages and disadvantages of each approach.

4.1 Modifications to the Representation of the State Space

The Statistical Fuzzy Discretization Network (SFDN) in the SFAL-C performs a fuzzy discretization, inducing a fuzzy partition of the state space $\mathbf{S}$ into S reference fuzzy subsets. This subsystem consists of a network of automata nodes $\mathbf{N}_i, i = 1, 2, \ldots, S$ with corresponding prototype location vector $l_i \in \mathbf{S}$. The nodes are arranged in a uniform grid throughout the m-dimensional state space $\mathbf{S}$. The degree of membership $\mu_{\mathbf{N}_i}$ of the current state vector of the process, $\mathbf{s}(k)$, is calculated for each of the S nodes. An S-dimensional activation vector

$$\gamma = [\mu_{\mathbf{N}_1}(\mathbf{s}(k)), \ldots, \mu_{\mathbf{N}_S}(\mathbf{s}(k))] \tag{4.1}$$

is a measure of the degree of similarity of the input vector to the ideal, or prototype, member of each of the respective fuzzy sets. The CAS algorithm uses a crisp (non-fuzzy) discrete approximation to the

state space $\mathbf{S}$ in order to learn the optimal continuous action for each region. The GCS-Δ algorithm utilizes a fuzzy discretization scheme similar to the SFAL-C.

4.1.1 Shape and Size of Fuzzy Set Membership Functions

Two important differences between the GCS-Δ fuzzy partition and the SFAL-C fuzzy partition are:

1. The actual shape of the gaussian membership functions in multiple dimensions.
2. The manner in which the size, defined by a spread parameter σ, is determined.

In the SFAL-C, the spread parameter σ is adapted online via a fairly complex update procedure. Let the spread parameter for $\mathbf{N}_i$ at time step $k+1$ be

$$\sigma_i(k+1) = \sigma_i(k) + \beta(k)\phi_i(k)\left[\|\mathbf{s}(k) - l_i(k)\|_2^2 - \sigma_i(k)\right] \quad i = 1, \ldots, S \tag{4.2}$$

where $\phi_i(k)$ is a function related to the membership degree of $\mathbf{s}(k)$ in $\mathbf{N}_i$ and the reciprocal of the normalized frequency of state visitation at node $\mathbf{N}_i$ and $\beta(k) \in (0,1)$ is an adaption parameter. Generally, the more times that a node i has been visited, the lower the value of ϕ_i. Similarly, if the current state of the process has a low membership value in fuzzy subset $\mathbf{N}_i$, then ϕ_i will be also be low and very little change in the spread parameter is allowed. The SFAL-C approach provides some distinct advantages:

1. The variation in σ_i creates a dynamic neighborhood function that is designed to adapt itself to the proper size.
2. The locations of the fuzzy sets in the SFAL-C are themselves modified to reflect the inputs during the learning phase and modification of σ_i can prevent unnecessary overlap of fuzzy subsets.

Empirically, the SFAL-C approach has been successful on several problems []. However, the generality of the method, as well as the number of parameters that it requires to operate effectively, does not lend itself well to theoretical convergence analysis. The approach taken in the GCS-Δ algorithm is much less complicated but retains many of the basic features.

The fuzzy additive system used to approximate the continuous state space in the GCS-Δ controller is based on the learned actions for each reference fuzzy state in $\mathbf{S}$. The CAS algorithm learns this optimal action for each $\mathbf{N}_i$ and provides this information to the GCS-Δ algorithm. Unlike the SFAL-C, the fuzzy subsets remain at fixed locations in $\mathbf{S}$ in the GCS-Δ algorithm. This allows for the calculation of an initial spread value σ_i based on some desired properties. Let the estimated policy $\hat{\pi}_\sigma$ be the output of the fuzzy additive model:

$$\hat{\pi}_\sigma(\mathbf{s}) = \frac{\sum_{i=1}^{S} \mu_{\mathbf{N}_i}(\mathbf{s})\mathbf{a}_{j_i^*}}{\sum_{i=1}^{S} \mu_{\mathbf{N}_i}(\mathbf{s})} \tag{4.3}$$

where $\mathbf{a}_{j_i^*}$ is the learned optimal action for $\mathbf{N}_i$ and

$$\mu_{\mathbf{N}_i}(x) = \exp(-\|x - l_i\|^2/2\sigma^2) \quad i = 1, \ldots, S. \tag{4.4}$$

An important property of the GCS-Δ algorithm is that it preserves the learned control law via the discrete approximation in the CAS algorithm. Consider the state l_i. By definition, $\mu_{\mathbf{N}_i}(l_i) = 1$. If the CAS algorithm determines that $\pi^*(l_i) = a^*$, then it is desirable property that $\hat{\pi}_\sigma(l_i) = a^*$ also. With standard center-of-area defuzzification, if the value of σ is chosen to be too large, it is likely that this is not true.

4.1.2 Allocation of Resources

One of the major concerns in reinforcement learning is the curse of dimensionality. Methods that attempt to use discrete approximations to continuous spaces must use other methods in an attempt to alleviate this problem. Generalizing the state space via fuzzy subsets is one successful method. In the SFAL-C approach, a prespecified number, S, of fuzzy subsets are spread uniformly throughout the state space. The locations and spread parameters of these fuzzy subsets are then modified as the learning phase progresses. The procedure is similar to that of a Kohonen self-organizing map. The intent is to concentrate the fuzzy subsets in the regions of the state space where there is a larger number of learning trials. There is no procedure in the SFAL-C that can add or remove nodes dynamically when and where they are needed. This shortcoming is addressed in the GCS-Δ algorithm.

In the GCS-Δ algorithm, the locations of the fuzzy subsets $\mathbf{N}_i$ are fixed but the number is not. Initially, a fairly low number of fuzzy subsets S is used. As the learning phase progresses, additional resolution is added in certain regions where it is needed. Adding states to the CAS algorithm provides a finer discrete approximation to the continuous state space and, therefore, improves the controller's approximation of the continuous space. However, increasing the number of states S directly increases the computational requirements for practical convergence of the Q-learning algorithm in each iteration of the CAS algorithm. Beneficial regions of the state space include:

1. Regions throughout **S** that are highly visited.

2. Region of **S** near the set-point $\mathbf{s}^*$.

In experiments with the SFAL-C controller, the inverted pendulum problem has been successfully controlled using as few as 4 fuzzy states and 5 discrete reference actions. However, the probability of learning a successful control law over a large number of runs is not high when very few fuzzy states and actions are used. As more nodes are added, the probability of learning a successful control law increases. Generally, in the SFAL-C experiments in [57], 25 nodes are spread uniformly throughout the state space which represent the prototype members of 25 fuzzy states. During a typical learning phase of 100,000

time steps, the distribution of states visited in Figure 4.1 shows how little of the time is spent in the regions far from the setpoint of [0,0]. Figure 4.2 maps this to the corresponding nodes in the state space.

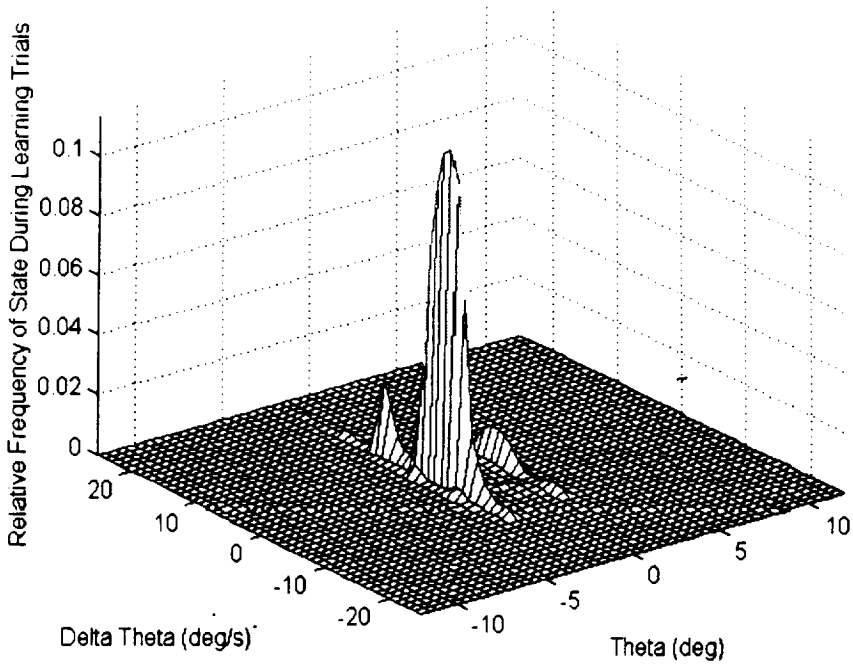

Figure 4.1: Distribution of states visited during typical inverted pendulum learning phase.

Using the SFAL-C controller with 25 nodes spread uniformly throughout the state space, the controller learned to successfully balance the pendulum 90% of the time in 20 experiments. The average angle that the pendulum was balanced at was 1.3177° with a standard deviation of 1.7926. Allocating more nodes around the set-point for a finer resolution in the region that is most visited reduces the average angle of balance to 0.0955° with a standard deviation of 0.3801. This is a significant improvement and illustrates the advantage of placing nodes non-uniformly in the state space. A procedure that systematically adds nodes to regions that are visited more often and removes or merges nodes in other regions is an integral part of the new CAS and GCS-Δ algorithms. Its application greatly enhances both the learning rate and the success rate of the controllers.

We examine adding states in the region of $\mathbf{S}$ near the set-point $\mathbf{s}^*$. For the CAS algorithm, the discrete state regions must be defined. The j-th dimension of the state space $\mathbf{S}$ is divided into S_j regions in the CAS algorithm. The locations are uniformly spread throughout the region as in Figure 4.3. This allowed the GCS-Δ algorithm to determine a single σ value for all states since there was a uniform distance between them. If additional nodes are added in the region surrounding the set-point, this no longer

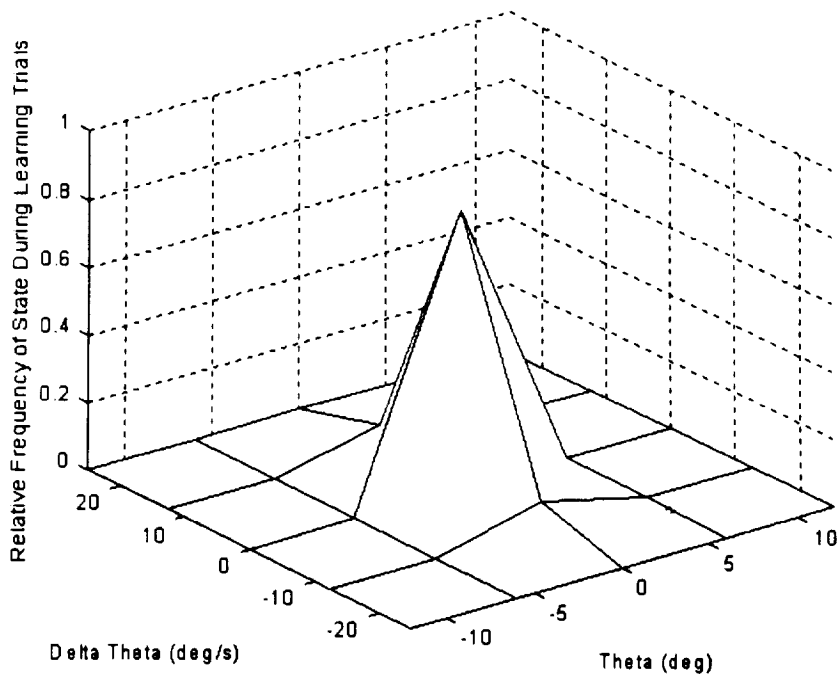

Figure 4.2: Distribution of nodes visited during typical inverted pendulum learning phase with 25 nodes.

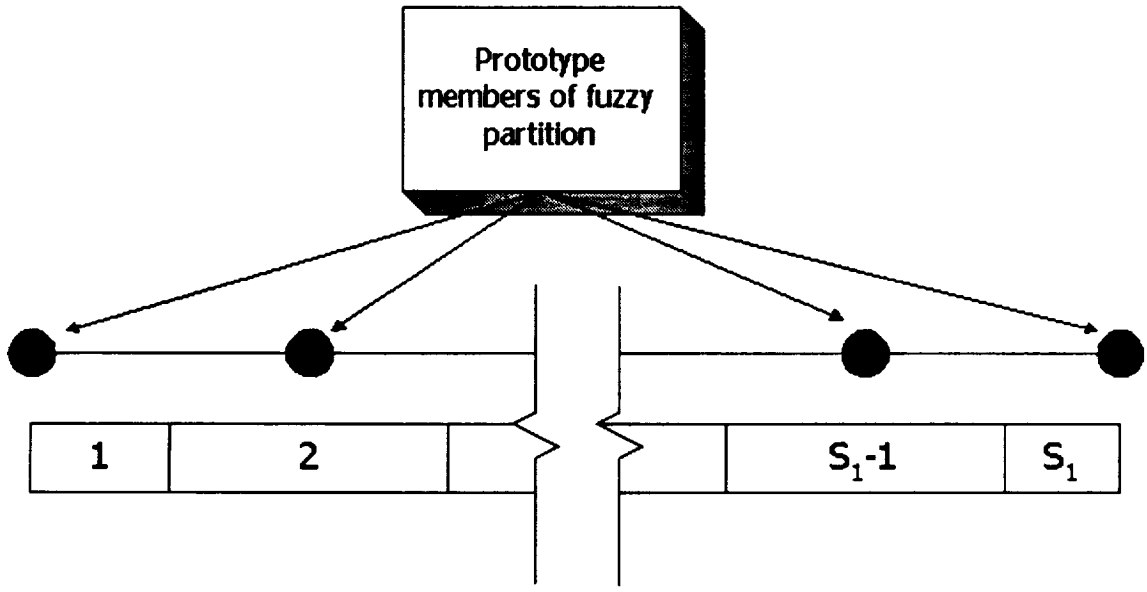

Figure 4.3: Discrete regions in CAS and their respective fuzzy prototypes in GCS-Δ in one dimension.

applies. Therefore, care must be taken to ensure that the value of σ chosen in the GCS-Δ algorithm does not overrepresent this region. An appropriate procedure for dealing with this scenario is being developed.

4.2 Modifications to the Representation of the Action Space

In the SFAL-C, the action space is discretized in the same manner as the fuzzy state space. Each fuzzy state $\mathbf{N}_i$ is mapped to a set of A reference control actions via scalar weights $c_{ij} \in (0,1), j = 1, \ldots, A$. The weights determine the degree that the reference control action is the correct action to take in that fuzzy region of the state space. Depending on the method of defuzzification used, a single weighting vector can generate different crisp control actions to be applied to the plant.

The CAS and GCS-Δ algorithms represent the continuous action space as a discrete approximation. However, unlike many approaches, the reference action set that represents this discrete approximation is continually updated during the learning phase. This approach converges on a single crisp action that is the learned optimal action a^* for that particular fuzzy state. This approach gives the distinct advantage of allowing a coarse (low cardinality) approximation to the continuous space be refined to any ϵ-accuracy. This is contrasted with the approach taken by the SFAL-C, Q-learning, and other methods where the ϵ-accuracy predetermines the number of reference control actions A.

An additional improvement of the CAS and GCS-Δ algorithms over the SFAL-C is mulitple output variable capability. The SFAL-C theoretically had this capability but, due to the complexity of the learning algorithm and its lack of convergence properties, it was not successfully implemented in our laboratory.

4.3 Modifications to the Learning Algorithm

Approximate methods based on dynamic programming (DP) are a fertile and ongoing area of research for model-free intelligent control. Among the major classes of algorithms are Sutton's temporal differences (TD) [67], Watkin's Q-learning [70, 71], and Baird's advantage updating [1]. These are online versions of the policy improvement and successive approximation algorithms in classical dynamic programming [21]. Considering reinforcement learning can be formulated as a dynamic program, a number of reinforcement learning controllers reported in the literature, including the SFAL-C [24, 25], have approximate DP-based learning algorithms.

Dynamic programming methods, including many of the model-free algorithms, are generally based on a finite action space. Continuous state or action spaces bring on the curse of dimensionality unless some type of approximation is used. The proposed CAS and GCS-Δ algorithms approximate the optimal control policy for set-point regulation problems with continuous state and action spaces. This differs from the SFAL-C controller and others in the literature in that the CAS and GCS-Δ algorithms possess

theoretical convergence properties while the others do not. Both the CAS and GCS-Δ algorithms take advantage of the property that the optimal policy is usually found considerably quicker than the optimal functional values in the Q-learning algorithm. A derivative-free search algorithm determines an improved policy during each policy improvement phase.

4.4 Experimental Application

The CAS algorithm has been applied to two set-point regulation applications of interest to this research First, as a testbed control problem, the inverted pendulum balancing problem is examined. Second, the power system stabilization problem, a problem of interest to the electric power industry such as EPRI, represents an underdetermined system for which optimal control is sought. We had also used it to study the performance and properties of our SFAL-C controller. We note that both of these problems are multiple-input/single-output nonlinear control problems that are approximated by a fine discrete state space.

Inverted Pendulum Balancing:

This control problem has four inputs: [θ, $\Delta\theta$, x, Δx] and one output F. The objective is to balance the pendulum at $\theta = 0°$ and keep the cart at $x = 0$. With the functional equation defined as in Equation 2.12 and the state space discretized into $51 \times 7 \times 51 \times 7$ states, the CAS algorithm was run.

For a standard to compare the CAS algorithm, the Q-learning algorithm was run with 101 discrete actions to give a fine approximation to a continuous action space. This requires 12,872,349 Q-values. After 105 full-backup iterations, the Q-learning algorithm converged to within $\epsilon = 0.00001$ of the optimal functional values for each state-action pair. This required over 1,351,596,645 Q-value updates. With the CAS algorithm, only $A = 7$ actions were used with a reduction factor $\beta = 0.8$. The CAS required 437 full-backup iterations but, with the reduced action space, it only required 389,866,491 Q-value updates. This is a reduction of over 70% in the computational effort.

The differences in the optimal trajectories between the ideal and the CAS approximation are shown in Figures 4.4 and 4.5. Since the strictly quasi-convex assumption may not hold, we are not guaranteed convergence. However, a choice of a higher β may increase our probability of finding the optimal action but at the price of increased computational effort.

Power System Stabilization:

The power system stabilization problem has six state variables, only two of which can be measured. Therefore the system is only partially observable. In this experiment, the control system consists of two inputs: [ω, $\Delta\omega$] and one output u. The stabilization system is composed of a synchronous machine with

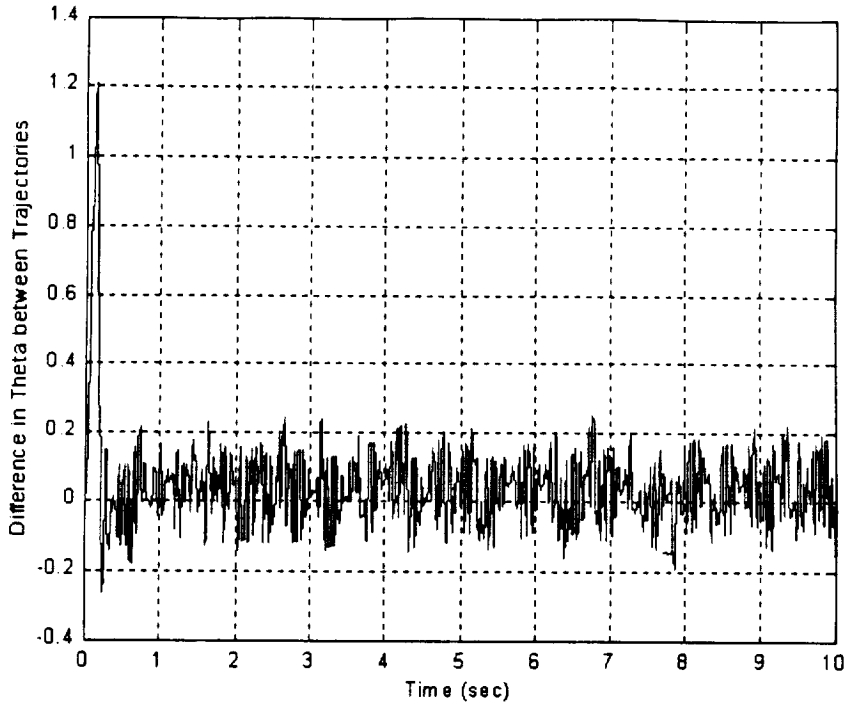

Figure 4.4: Error in optimal trajectory of θ over time for initial state [10,0].

an exciter and a stabilizer connected to an infinite bus. The dynamics of the synchronous machine (used for simulation only) are represented by a linearized incremental model [25]. The objective is to dampen the oscillations [ω, $\Delta\omega$] under various load changes. With the functional equation defined

$$V(i) = \min_{\mathbf{a}_j} \left[\omega^2 + \gamma \sum_{l \in \mathbf{S}} p_{il}(\mathbf{a}_j) V(l) \right]. \tag{4.5}$$

and the state space discretized into 101 $\times$ 51 states, the CAS algorithm was run.

The inverted pendulum example above was run using full backups, which required knowledge of the state transitions. This was done for demonstration purposes. In the power system stabilization problem, sample backups are used to learn a model-free control law. Because the system is only partially observable with the four unobservable state variables taking on independent values, comparisons to an optimal policy cannot be made. The number of reference control actions were $A = 7$ with $\beta = 0.8$ and 16 reductions of the IoU were performed. The stabilization of the system after the completion of learning when presented with a load change of $0.05pu$ is shown in Figure 4.6. The control sequence is shown in Figure 4.7.

Additional research into the sample backup approach and its effect on the number of Q-value updates, β, and A is underway in our laboratory.

The CAS algorithm combines the derivative-free line search methods of nonlinear optimization with online dynamic programming to create an approximate policy iteration procedure to estimate an optimal

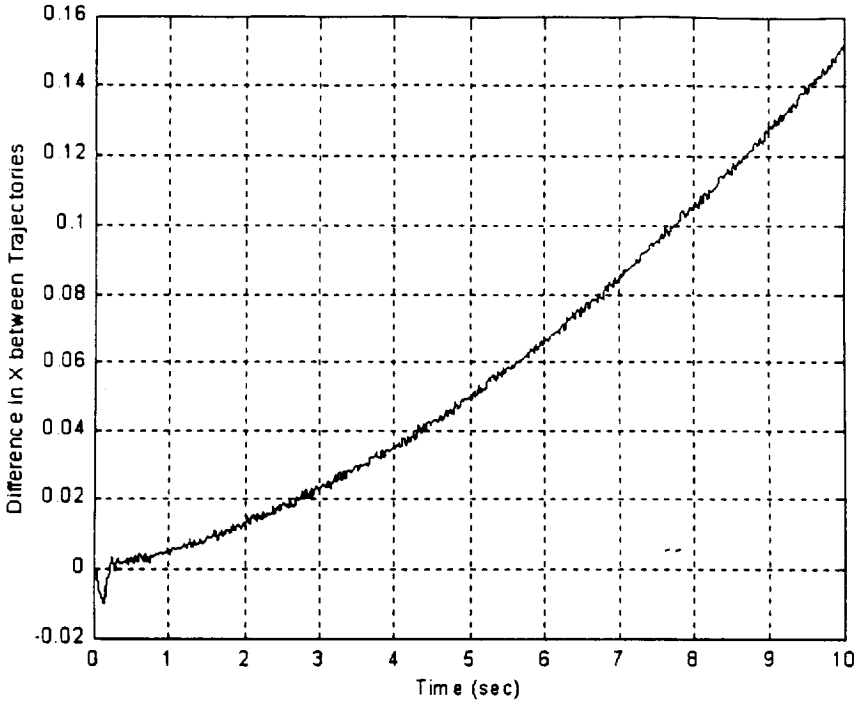

Figure 4.5: Error in optimal trajectory of x over time for initial state [10,0].

control law. The convergence properties of the Q-learning algorithm are used to search a continuous action space using a discrete subset of reference actions. The reduction in computational effort over a fine discrete approximation is evident. Example applications are shown for both full and sample backups. Under the assumption of strict quasi-convexity, the policy is indeed improved in each successive reduction step. If the reductions in the intervals of uncertainty are applied systematically, convergence to the optimal control action is assured. In practical applications, where the assumption may not hold, an adroit choice of β and A can increase the probability of finding the optimal action.

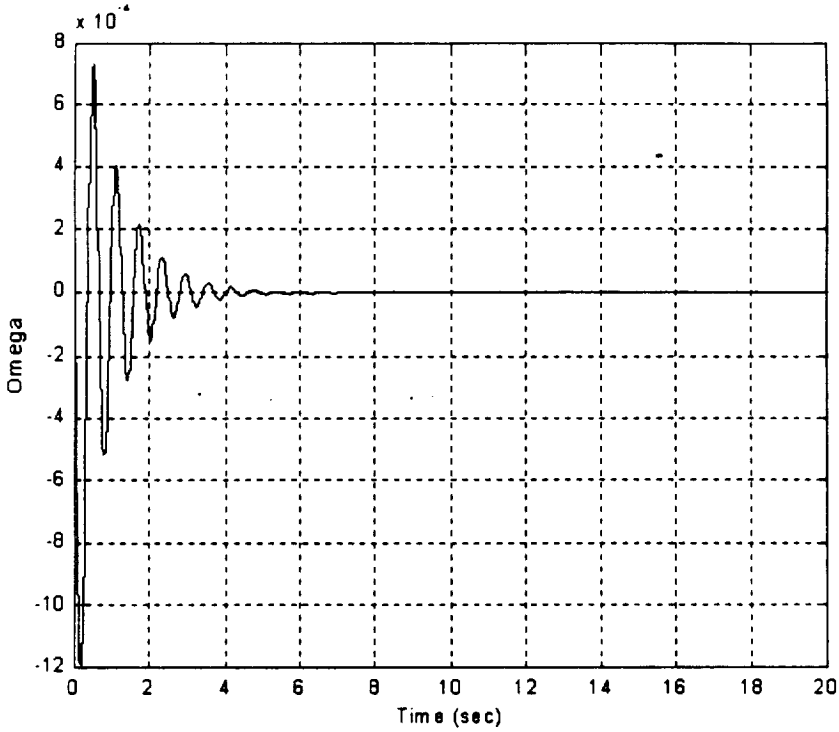

Figure 4.6: Trajectory of ω over time for initial state [0.01,0].

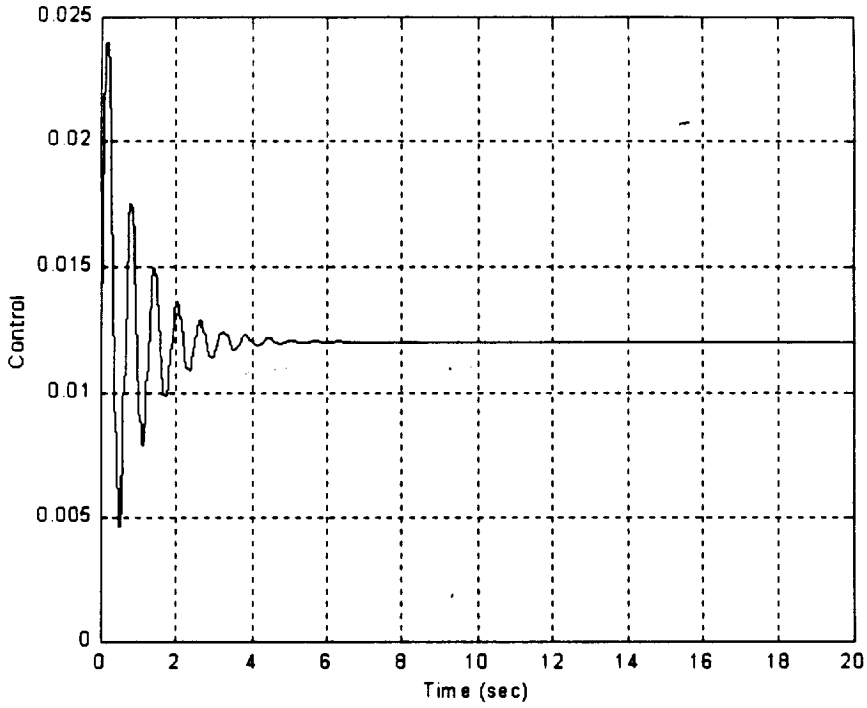

Figure 4.7: Learned control policy of u over time for initial state [0.01,0].

Chapter 5

Theoretical Foundations for Application to Space Systems

Two important theoretical developments which contribute to an improved controller system that is suitable for application to complex space problem analysis involve efficient clustering methods and effective optimization algorithms or procedures. In the sequel, we address both of them.

5.1 Efficient Clustering Algorithms

As pointed out in the Introduction chapter of this report, many studies centering on efficient information processing and retrieval systems for space systems have been embarked by NASA at the JSC facillity. Many of these involve the subject of clustering. Clustering is also very critical to the success of the design of an effective neuro-fuzzy controller similar to the one that we developed in our NSF study. Therefore, it is useful to pursue these studies as proposed in our work mission.

The importance of cluster analysis as a tool in pattern recognition is well recognized. Basically, we may view the task in hand as that of dividing a set of K data points into N clusters in an optimal fashion where the number N may be a preassigned integer. Clustering can be done both classically and via fuzzy set theory. The latter generally recognizes the following two problem classes of interest: The first one is to group fuzzy data points into some fuzzy sets. The other is to divide the crisp data points into a specified number of subsets which need not be fuzzy but utilizing fuzzy set theoretic methods in developing the clusters.

Our focus is on fuzzy clustering whose literature in recent years has grown increasingly vast. Perhaps, the earliest reference to fuzzy cluster analysis may be traced to Bellman *et al.*[4] and Ruspini[62]. According to Yang[73], the studies of cluster analysis employing fuzzy set theory can be divided into three

categories: fuzzy clustering based on fuzzy relation, fuzzy clustering based on objective function, and the fuzzy generalized k-nearest neighbor rule. The first one, fuzzy clustering based on fuzzy relation, was first proposed by Tamura *et al.*[68]. They presented a multi-step procedure by using the composition of fuzzy relations beginning with a reflexive and symmetric relation. The second and more interesting to us is fuzzy clustering based on objective function. This approach is best illustrated via the method proposed by Dunn[20] and generalized by Bezdek[7].

A variety of generalizations of this method has been developed. The fuzzy general k-nearest neighbor rule, for example, is a type of nonparametric classifiers. Let a finite data point set $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_K\}$ be given. Furthermore, let a set of n correctly classified samples be $(\mathbf{x}_1, \theta_1)$, $(\mathbf{x}_2, \theta_2)$,$\cdots$, $(\mathbf{x}_n, \theta_n)$ where θ_i's represent the labeling variables of N clusters and tThe tethered satellite system retrieval problem was revisited as a problem of central interest to NASA for which inadequate and inefficient control methodologies had been used due to the complexities of the system. The literature showed that fuzzy logic controllers can be effective for these types of problems. Our neuro-fuzzy adaptive controller offers significant advantages over both classical and fuzzy controllers for this problem.ake values in the set $\{1, 2, \cdots, N\}$. A new pair $(\mathbf{x}, \theta)$ is given, where only the measurement $\mathbf{x}$ is observable by the statistician, and it is desired to estimate θ by utilizing the information contained in the set of correctly classified points. We shall call $\mathbf{x}' \in \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ a nearest neighbor to $\mathbf{x}$ if $||\mathbf{x}' - \mathbf{x}|| = \min_{1 \le i \le n} ||\mathbf{x}_i - \mathbf{x}||$. Then the point $\mathbf{x}$ is assigned to the cluster θ' of its nearest neighbor $\mathbf{x}'$. In addition, Esogbue[22] introduced fuzzy dynamic programming to the area of optimal fuzzy clustering. Application to the evaluation of fuzzy data generated in connection with non-point source water pollution control strategies was also reported.

This chapter presents a concept of fuzzy prototype as opposed to crisp prototype and introduces a new type of fuzzy clustering which we call fuzzy criterion clustering. We have proposed two forms of fuzzy criteria for clustering analysis. The first one is fuzzy average criterion in which we maximize the weighted sum of all degrees of membership of given data points. In the second, we maximize the minimum degree of membership of all given data points. Different from the approaches of traditional clustering methods, fuzzy criterion clustering preassigns the membership functions for all possible clusters to form a collection of fuzzy prototypes and then selects a number of clusters from all fuzzy prototypes by fuzzy criterion as the optimal fuzzy partition.

5.1.1 Fuzzy Prototypes and Fuzzy Criteria

Initially, our given data points are the samples of all possible point set $\mathcal{X}$. For example, let $X = \{1, 5, 6, 30, 70, 72\}$ represent 6 people by their ages. We call the set X the given data set. This can be regarded as a sample set of all people set $\mathcal{X}$ with ages from 0 to 100. Thus, $X \subset \mathcal{X} = [0, 100]$. We consider the given data point set X in the total point set $\mathcal{X}$.

Recall that in traditional clustering process the given data point set is grouped into a specified number of (crisp or fuzzy) clusters in such a way that some generalized variance is minimized. In other words, after determining the generalized variance as the objective function, the clustering process produces a number of clusters minimizing the objective. Roughly speaking, it will design the *position* and *size* for each cluster. The term position usually means the cluster center which hopefully is not in conflict with the the perception of the decision maker. However, the situation about size shall be changed. In fact, the purpose of clustering is such that we can perform one policy for all the elements in one cluster or what prototype the points in one cluster like. If the size is too large, can we still use one policy to handle one cluster? Clearly, we can not. It is noteworthy that the objective function of existing clustering methods does not directly provide the decision maker any information about the cluster size.

We feel that a desirable property of a clustering algorithm which will enhance its practical utility is its ability to provide the decision maker with some guidance in advance on the cluster size. This question is the basic motive of our approach. Its utility in complex on line applications is especially important.

Since we have set all given data points on the *background* set $\mathcal{X}$, we will regard the cluster as a box which includes not only some given data points but also the points on the background $\mathcal{X}$. If we go back to our given data point set, it is clear that the points in one cluster should be assigned to one class.

To illustrate, consider the example of 6 people. Suppose that according to some purpose, the decision makers wish that the cluster size should be 3, i.e., in one cluster the maximum difference of ages should not exceed 3. Thus, the collection of all possible clusters should be

$$U = \{[x, x+3] | 0 \leq x \leq 97\}$$

which includes infinite intervals with length 3. If we want to group the set X into 4 clusters, then we can define them as

$$\mathbf{u}_1 = [0,3], \quad \mathbf{u}_2 = [5,8], \quad \mathbf{u}_3 = [28,31], \quad \mathbf{u}_4 = [69,72].$$

Thus, the given data points are all contained in the above 4 clusters.

As an extension of crisp cluster, we introduce fuzzy set theoretic concepts. We will represent the length of interval, radius of disc, width of rectangle or a general zone by a fuzzy number. Let each possible cluster $\mathbf{u}$ be a fuzzy set with membership function μ. Then μ describes both the cluster *position* and *size*. Furthermore, the point x^*, satisfying

$$\mu(x^*) \geq \mu(x), \quad \forall x \in \mathcal{X},$$

is the cluster center representing the position while the size is determined by the form of the membership function $\mu(x)$. A sharp one represents small size while a plain one may represent a large size. A data point is regarded to be closer to the cluster center if the degree of membership of this point is closer to the best value 1. We call the possible fuzzy clusters *fuzzy prototypes*. and the set of elements in a fuzzy prototype with degree of membership 1 is called a *center* of the fuzzy prototype.

Suppose that we have determined some fuzzy clusters, for example, $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_N$ with membership functions $\mu_1, \mu_2, \cdots, \mu_N$, respectively. We are interested in the degree of membership of a data point x. It is more reasonable to regard x as being in $\mathbf{u}_i$ rather than $\mathbf{u}_j$ if $\mu_i(x) > \mu_j(x)$. Thus, we define the degree of membership of x is

$$\mu_1(x) \vee \mu_2(x) \vee \cdots \vee \mu_N(x)$$

where the symbol $\vee$ represents the binary maximum operator. That is , we take the degree of membership at this point in the fuzzy cluster it most likely belongs to.

We note that we do not employ a distance function because it can be represented to some degree by the membership function of our approach. Thus, our clustering criterion is to let the degrees of membership of all given data points be as high as possible. We term such type of clustering criterion *fuzzy criterion*. For this problem, the clustering criteria can take various forms. Let us however specify the following two fuzzy criteria for our fuzzy criterion clustering algorithm as being quite reasonable.

Fuzzy Average Criterion: Here, we maximize the weighted sum of all degrees of membership of given data points.

Fuzzy Maximin Criterion: As is implied, we maximize the minimum degree of membership of all given data points.

It is instructive at this stage to pause and compare the concepts of fuzzy prototype and traditional prototype, fuzzy criterion clustering and traditional fuzzy clustering. Clearly, fuzzy prototypes are very different from prototypes employed in traditional fuzzy clustering, including the circular and elliptical shapes reported by Coray[13], Dave[16], Dave and Bhaswan[18], etc.. In fact, all of the traditional prototypes are essentially of size 0. For example, a crisp circle in a plane is of size 0 because its geometric measure is 0, and the sum of distances from the data points to the prototypes is used as the objective function. However, fuzzy prototypes allow nonzero size represented by a fuzzy number but abandon the concept of distance. Additionally, fuzzy criterion employs fuzzy measures such as the above fuzzy average criterion or fuzzy maximin criterion proposed in the foregoing to produce a predetermined number of fuzzy clusters.

5.1.2 Fuzzy Criterion Clustering

Suppose that we have K data points, $\mathbf{x}_k$, $k = 1, 2, \cdots, K$, each $\mathbf{x}_k$ is a m-dimensional vector, i.e., $\mathbf{x}_k = (x_{k1}, x_{k2}, \cdots, x_{km})$. These data points may be crisp or fuzzy.

Let $U = \{\mathbf{u} | \mathbf{u} \subset R^m\}$ be a collection of all fuzzy prototypes given by the decision maker(s). This collection may be finite or infinite(countable or not). Each prototype $\mathbf{u}$ is a fuzzy subset with membership function μ such as fuzzy interval, fuzzy disc, fuzzy circle, fuzzy parabola, fuzzy ellipse, fuzzy line, etc.. In fact, all the fuzzy prototypes in U accord with the demand of cluster *size*.

Let N be the predetermined positive integer representing the number of clusters.

Our problem then is to group the set of K (crisp or fuzzy) data points into N clusters selected from the collection U. In other words, we want to choose N fuzzy prototypes from U such that they suit the given data point set X very well based on some criterion.

Let $\mathbf{u}_n$, $n = 1, 2, \cdots, N$ be a sequence of N fuzzy prototypes selected from the collection U. Since our fuzzy average criterion for the clustering problem is to maximize the weighted sum of all degrees of membership of given data points, we may view our problem as a fuzzy optimization problem with the following associated mathematical programming model,

$$\max J(\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_N) = \sum_{k=1}^{K} \lambda_k [\max_{n=1}^{N} \{\mu_n(\mathbf{x}_k)\}] \tag{5.1}$$

where $\mathbf{u}_n \in U$ are fuzzy prototypes with membership functions μ_n, $n = 1, 2, \cdots, N$, respectively, λ_k are weighted factors, and typically, we can define $\lambda_k = 1/K$. If we employ the fuzzy maximin criterion which is to maximize the minimum degree of membership of all data points, then the following model results,

$$\max J(\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_N) = \min_{k=1}^{K} \left\{ \max_{n=1}^{N} \{\mu_n(\mathbf{x}_k)\} \right\}. \tag{5.2}$$

Let $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_N$ be the optimal N fuzzy prototypes optimizing the objective function $J(\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_N)$ in model (5.1) or (5.2). Then $\{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_N\}$ is an optimal fuzzy criterion partition of the given data point set. Each $\mathbf{u}_n$ is a fuzzy cluster with membership function μ_n.

Remark 1: If our data points are crisp and the decision maker wants crisp clusters, i.e., hard partition of the given data point set, then we can define them as follows,

$$\mathbf{u}'_n = \{\mathbf{x} | \mu_n(\mathbf{x}) \geq \mu_i(\mathbf{x}), i = 1, 2, \cdots, N\} \tag{5.3}$$

for $n = 1, 2, \cdots, N$. Thus, a partition of the data points into N (hard) clusters can be represented by mutually disjoint crisp sets $\mathbf{u}'_1, \mathbf{u}'_2, \cdots, \mathbf{u}'_N$ such that all the given data points are in $\mathbf{u}'_1 \cup \mathbf{u}'_2 \cup \cdots \cup \mathbf{u}'_N$ if we ignore their boundary points.

Remark 2: Let $\mathbf{x}$ be some given data point in X. If

$$\mu(\mathbf{x}) = \mu_1(\mathbf{x}) \vee \mu_2(\mathbf{x}) \vee \cdots \vee \mu_N(\mathbf{x})$$

is too low for our optimal fuzzy clusters, then the fuzzy criterion clustering process has to be redone by enlarging the integer N. In fact, we can employ some interactive method to handle this process for balancing the fuzzy criterion value and the number of clusters.

Remark 3: If the value of objective function in model (5.1) or (5.2) is too low, we also need to redo it just as in Remark 2. Certainly, the situation of (5.2) is identical to that of Remark 2.

Remark 4: If some noise data points are included,then Remarks 2 and 3 may be ignored.

5.1.3 Illustrative Examples

In this, section we wish to illustrate the effectiveness of the fuzzy criterion clustering based on fuzzy prototypes sketched above by solving some sample test problems. We have coded fuzzy criterion clustering algorithm on a workstation using C language. A genetic algorithm, a type of stochastic search method, is employed to obtain the solution to the resultant global optimization problem. Genetic algorithms have become popular for solving intractable large complex optimization problems. A number of good expository references including Pal and Wang [61] discuss its application to problems arising in clustering. Using a population size of 30 and implementing our algorithm with no more than 2000 generations in each of the following examples, we obtained the desired results.

The general form of the fuzzy criterion clustering algorithm can be stated as follows.

Procedure Fuzzy Criterion Clustering

Input number of clusters and other parameters;

Initialize the fuzzy prototypes (chromosomes);

REPEAT

Update the fuzzy prototypes by genetic operators;

Select the fuzzy prototypes by sampling mechanism;

UNTIL(*termination_condition*)

Example 1

Using an extreme example, let us consider the data points as $1, 2, \cdots, 100$. Each possible fuzzy prototype u given by the decision makers is a fuzzy subset with membership function μ characterized by a parameter y, i.e.,

$$\mu(x, y) = \exp\left[-\frac{|x - y|}{10}\right] \tag{5.4}$$

where x is some data point. Meanwhile, each fuzzy prototype is a fuzzy interval. Suppose that we are asked to group the 100 data points into 5 clusters such as that described by (5.4). Which 5 clusters are the best choice under the fuzzy criterion? Our problem is to find 5 parameters y_n, $n = 1, 2, \cdots, 5$, each y_n represents a fuzzy cluster.

If we employ the fuzzy average criterion, according to equation (5.1), we can translate this problem into the following mathematical programming model

$$\max_{y_1, \cdots, y_5} \frac{1}{100} \sum_{k=1}^{100} \mu(k, y_1) \vee \mu(k, y_2) \vee \cdots \vee \mu(k, y_5) \tag{5.5}$$

where 1/100 is the weighted factor. If on the other hand, we employ the fuzzy maximin criterion, then the model becomes

$$\max_{y_1, \cdots, y_5} \min_{1 \le k \le K} \mu(k, y_1) \vee \mu(k, y_2) \vee \cdots \vee \mu(k, y_5). \tag{5.6}$$

It is reasonable to group these 100 data points into the following 5 clusters: $\{1, 2, \cdots, 20\}$, $\{21, 22, \cdots, 40\}$, $\{41, 42, \cdots, 60\}$, $\{61, 62, \cdots, 80\}$ and $\{81, 82, \cdots, 100\}$. A run of our computer program shows that the optimal objective value of (5.5) is 0.6326 with 5 parameters

$$(y_1, y_2, y_3, y_4, y_5) = (11, 31, 51, 71, 91),$$

and the optimal objective value of (5.6) is 0.3867 with 5 parameters

$$(y_1, y_2, y_3, y_4, y_5) = (10.5, 30.5, 50.5, 70.5, 90.5).$$

Both of the results of fuzzy criterion clustering are coincident with the above reasonable partition. For example, we consider the second fuzzy cluster $C_2 = \{21, 22, \cdots, 40\}$. We mention that, for the fuzzy average criterion,

$$\mu(i, 31) \geq \mu(j, 31), \ \forall i \in C_2, j \notin C_2$$

with equalities hold at only $i = 21$ and $j = 41$, for the fuzzy maximin criterion,

$$\mu(i, 30.5) \geq \mu(j, 30.5), \ \forall i \in C_2, j \notin C_2.$$

If we want to group these data points into crisp as opposed to fuzzy clusters, we can use equation (5.3) in Remark 1 to produce them which are shown to be identical to the above reasonable crisp clustering. Thus, the fuzzy criterion clustering is successful for this example.

Example 2

This one is a variation of an example from Ruspini[63]. We take 100 points which are shown in Figure 5.1 by dots. Each possible fuzzy prototype **u** given by the decision maker(s) is a fuzzy subset with membership function μ characterized by a parameter vector , i.e.,

$$\mu(\mathbf{x},) = \exp(-||\mathbf{x} - ||) \tag{5.7}$$

where $\mathbf{x}$ is some data point and $||\cdot||$ represents the Euclidean distance. The fuzzy prototype is clearly a fuzzy disc.

Now we want to group these data points into 4 fuzzy clusters with membership function characterized by (5.7). Our fuzzy average criterion based approach yields the 4 cluster centers as follows,

$$_1 = (0.73, 3.42), \quad _2 = (2.65, 1.50),$$

$$_3 = (4.93, 5.06), \quad _4 = (6.32, 2.56)$$

with fuzzy average value 0.785. Figure 5.1 shows that data points enclosed in one curved boundary are grouped into one cluster.

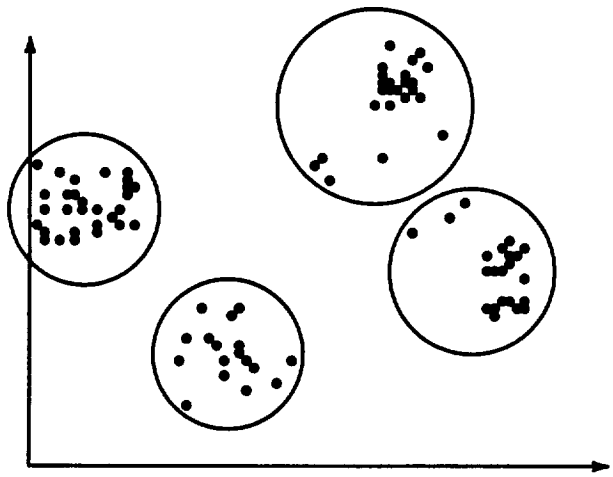

Figure 5.1: An Example of Ruspini

Example 3

This example considers the 10×10 region in which 60 points are uniformly distributed on the circle $(x-4)^2+(y-6)^2=16$, 60 points on $(x-6)^2+(y-4)^2=9$ and 30 noise points on the whole region.

We suppose that our fuzzy prototypes are fuzzy circles, i.e, if a center (a,b) and a radius r are given, the fuzzy circle is characterized by membership function

$$\mu(x,y,a,b,r)=\exp\left(-||(x-a)^2+(y-b)^2-r^2||\right) \tag{5.8}$$

in which if (x,y) is on the circle $(x-a)^2+(y-b)^2=r^2$, then $\mu=1$.

Let the number of clusters be 2. A run of our computer program obtains two fuzzy circles which are shown in the right column in Table 5.1 with fuzzy criterion value 0.705.

Original Circles
$(x-4)^2+(y-6)^2=16$
$(x-6)^2+(y-4)^2=3^2$
Centers of Fuzzy Circles
$(x-3.978)^2+(y-5.996)^2=4.004^2$
$(x-6.005)^2+(y-3.999)^2=3.005^2$

Table 5.1: Comparison of Original Circles and Fuzzy Circles

We find in Table 5.1 that the centers of fuzzy circles generated by our algorithm is very close to the original ones (It is important that we do not confuse fuzzy center with geometric center of a crisp circle!). The little difference may result from the noise data points or computation error. The data points and two centers of fuzzy circles are shown in Figure 5.2.

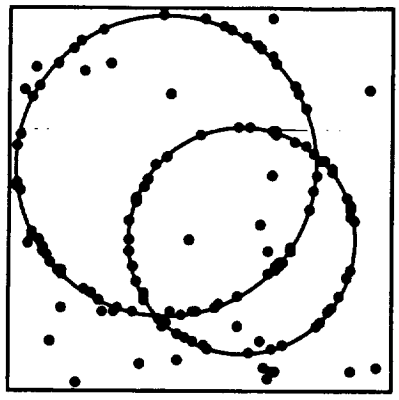

Figure 5.2: Two Circles and Thirty Noise Points

Example 4

Here, we consider 60 data points for each of the following three circles defined by: $(x-3)^2+(y-7)^2=4$, $(x-7)^2+(y-7)^2=4$, and $(x-5)^2+(y-4)^2=9$, but there is a uniformly distributed noise with an interval 0.5 added to the x and y locations of the data points so that the points do not lie on the ideal curves, on a region 10×10. The total number of data points is 180.

We suppose also that our fuzzy prototypes are fuzzy circles with membership function defined by (5.8). The results obtained by our algorithm are shown in Table 5.2. The fuzzy criterion value is 0.601.

Original Circles
$(x-3)^2+(y-7)^2=2^2$
$(x-7)^2+(y-7)^2=2^2$
$(x-5)^2+(y-4)^2=3^2$
Centers of Fuzzy Circles
$(x-3.081)^2+(y-6.976)^2=2.024^2$
$(x-6.891)^2+(y-6.954)^2=2.072^2$
$(x-5.035)^2+(y-4.083)^2=3.086^2$

Table 5.2: Comparison of Original Circles and Fuzzy Circles

The data points and two centers of fuzzy circles are shown in Figure 5.3.

Example 5

This example considers a case consisting of a mixture of multiple types of fuzzy prototypes. We produce 60 noise data points on a parabola $y=(x-5)^2$, 60 noise data points on a circle $(x-5)^2+(y-4)^2=9$, 20 noise data points on a straight line $y=4$, and 20 noise data points on a straight line $x=5$ on the region 10×10, where the noise is uniformly distributed on an interval 0.5. The total number of data

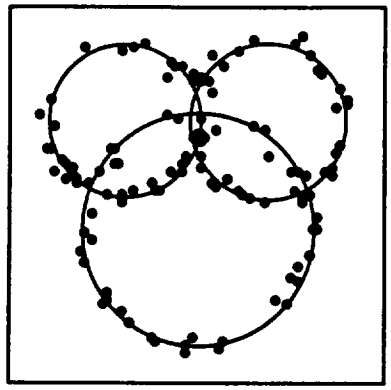

Figure 5.3: Three Noise Circles

points is 160.

Our fuzzy prototypes are composed of fuzzy parabola, fuzzy circles and fuzzy lines. Generally, let $g(\mathbf{x}) = 0$ be a center of a fuzzy prototype. It can be of any geometric sharp. A fuzzy prototype with center $g(\mathbf{x}) = 0$ will be defined by a fuzzy set with membership function

$$\mu(\mathbf{x}) = \exp(-c \cdot |g(\mathbf{x})|) \tag{5.9}$$

where c is a positive coefficient, for example, $c = 1$, in which if $\mathbf{x}$ is on $g(\mathbf{x}) = 0$, then $\mu(\mathbf{x}) = 1$.

Let us show the results by Table 5.3 when the number of clusters is 4.

Original Prototypes	
$y = (x-5)^2$	(parabola)
$(x-5)^2 + (y-4)^2 = 3^2$	(circle)
$y = 4$	(line)
$x = 5$	(line)
Centers of Fuzzy Prototypes	
$y = 0.906(x-4.946)^2$	(parabola)
$(x-4.834)^2 + (y-3.947)^2 = 3.132^2$	(circle)
$y = 3.900$	(line)
$x = 4.900$	(line)

Table 5.3: Comparison of Original Prototypes and Fuzzy Prototypes

We also show the results by Figure 5.4.

5.1.4 Cluster Validity

Cluster validity is clearly an issue in cluster analysis especially when the methods used are heuristics and algorithmically derived. For example, how does a given algorithm thus obtained guarantee that

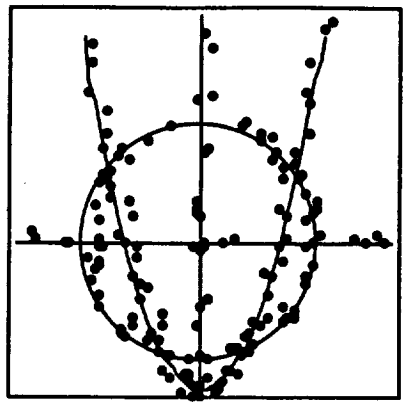

Figure 5.4: Noise Circle, Noise Parabola and Two Noise Lines

the number of clusters used is the one and only one (optimal) for a given study? Additionally, for a given experiment, how does the algorithm ensure that a particular classification is efficient or optimal? This problem is particularly relevant for the popular fuzzy algorithms such as the fuzzy c-means type algorithms of Bezdek[7] and others,the fuzzy c-shells algorithms of Dave and others[16], etc.

The classical validity techniques employed in each of the foregoing include the use of the partition coefficient as proposed by Bezdek[6], the classification entropy of classical statistics and extended to the fuzzy c- partition, the fuzzy hypervolume and fuzzy partition density, and the array of validity measures for fuzzy c-shells proposed by Dave[17].

The issue of validity measures, on the other hand,is rather mute for the fuzzy dynamic programming clustering algorithm of Esogbue[22] which inherits the optimality properties of dynamic programming but is limited by the computational demands of that algorithm. This desirable feature is apparently also transferable to the fuzzy clustering algorithm described here. However, because solution of the model was eventually realized by a genetic algorithm, we have addressed the validity issue by proposing a cluster validity measure which we outline in the sequel.

5.1.5 Validity Measures

In Esogbue and Liu[30], we presented some analytic validity measures for this algorithm. Basically, given a number of clusters, N, we show that we can obtain N optimal clusters represented by parameter vector by employing fuzzy criterion clustering. We denote the N clusters by $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_N$. Then the K data points can be classified into one and only one cluster of $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_N$.

If we let the length of $\mathbf{u}_n$ be L_n, the number of data points belonging to $\mathbf{u}_n$ be K_n, $n = 1, 2, \cdots, N$,

respectively, then we have $K_1 + K_2 + \cdots + K_N = K$. The validity measure $g(,N)$ is defined by

$$g(,N) = \min\left\{\frac{K_n}{L_n},\ n = 1,2,\cdots,N\right\}. \tag{5.10}$$

On the other hand, we know that cluster validity should eliminate spurious clusters and merge compatible clusters. According to the partition rule of fuzzy criterion clustering, any point can belong to one and only one cluster, so a cluster is considered spurious if the number of its data points is *too small*, meanwhile, the validity measure $g(,N)$ should also be *too small*. Furthermore, compatibility of two clusters implies that there is at least one cluster such that it contains only a few number of data points, i.e., the validity measure $g(,N)$ is very small. Hence $g(,N)$ can be regarded as a validity measure.

We note that this validity measure presumes the existence of an approximate estimation on the number of potential points of certain prototypes. Meanwhile, the length L_n will be represented by that number. The inspiration for this assumption is from the theory and operation of the digital process of a camera.

5.1.6 Summary

In this section, the use of the concepts of fuzzy prototype as opposed to crisp prototype as well as fuzzy criterion for optimal clustering was presented. Numerical experiments show that fuzzy criterion clustering based on fuzzy prototypes can overcome the effect of noise points. Its effectiveness was demonstrated for these and other studies reported elsewhere. The validity issue was also addressed.

This method which we have termed fuzzy criterion clustering has the capability of accurately classifying and detecting typical clusters including circles, ellipses and various shapes that have posed problems for some well known algorithms reported elsewhere in the literature. Additionally, it is robust and appealing to practitioners because of its user driven fuzzy criteria clustering objective function. Applications to the clustering of real world data arising from water pollution control studies as well as MRIs in cardiac sequence detection experiments are in progress in our laboratory.

5.2 Efficient Dynamic Programming Methods

Fuzzy dynamic programming is a powerful control and analysis apparatus which seeks to extend classical dynamic programming to many real life situations characterized by uncertainty, especially of the imprecise and ambiguous variety. Penetrating reviews of the developments in the field of fuzzy dynamic programming as well as an insightful discussion of possible extensions are provided by Esogbue and Bellman [23] and recently by Esogbue and Kacprzyk [27]. A particularly interesting generalization involves decision situations in which the decision, constraints, goals, and system dynamics are all fuzzy as given is treated by Baldwin and Pilsworth [2]. The details of these concepts and proofs are provided in Liu and Esogbue [55]. We present the framework for fuzzy criterion set and fuzzy criterion dynamic programming

which is a general tool for dealing with many decision and control situations arising in many fields including the stochastic reservoir operation and stochastic inventory control models of operations research and engineering. Specifically, the objective is to maximize the expected fuzzy criterion function of the product of fuzzy criterion sets. We outline existence, uniqueness and stability theorems of the derived solutions to this model whose resultant optimal control is a bounded critical number policy under the usual regular hypothesis assumptions.

5.2.1 Fuzzy Criterion Sets and Fuzzy Criterion Functions

Let X be a collection of elements denoted generally by x with the fuzzy set A in X the set of ordered pairs, defined as

$$A = \{(x, \mu_A(x)) | x \in X\} \tag{5.11}$$

where $\mu_A(x)$ is called the membership function of x in A. We say that A is a *fuzzy criterion set* if A is the set of all *satisfactory* elements and $\mu_A(x)$ is the *satisfactory degree* of x. In this section, we call $\mu_A(x)$ the *fuzzy criterion function.*

As in the basic theory of fuzzy sets, we define an *α-level set* as one of elements that belong to the fuzzy criterion set A at least to the degree α:

$$A_\alpha = \{x \in X | \mu_A(x) \geq \alpha\}. \tag{5.12}$$

A fuzzy criterion set A is said to be *unimodal* if its fuzzy criterion function $\mu_A(x)$ is unimodal. For additional concepts employed in the exposition of fuzzy criterion dynamic programming see Liu and Esogbue[55].

5.2.2 Fuzzy Criterion Dynamic Programming

To motivate our model, consider an inventory system. If the demand must be satisfied regardless of the physical constraints of the warehouse or reservoir, then the state of the system can be described by the *imaginary* inventory level. When the imaginary inventory level is less than the dead inventory level (dead storage), then we cannot fulfill demand. In this case the difference represents the shortage quantity. When the imaginary inventory level is greater than the largest physical storage, the amount exceeds the capacity of the warehouse or reservoir, and the difference represents the degree of exceeding the level or flood. Usually, there exists a best state at which we define the value of fuzzy criterion to be 1. When the inventory level deviates from the best state, the fuzzy criterion value decreases. Thus, the set of all satisfactory states is a fuzzy criterion set whose fuzzy criterion function is the satisfactory degree of elements.

For a given N-stage decision process, inventory control process or reservoir operation problem, let $A_1, A_2, \ldots, A_N$ be fuzzy criterion sets of satisfactory states with fuzzy criterion functions $\mu_1, \mu_2, \ldots, \mu_N$

at stages $1, 2, \ldots, N$, respectively, on the real line R. Assume that $\lambda_1, \lambda_2, \ldots, \lambda_N$ are coefficients of convex combination representing the relative importance among $A_1, A_2, \ldots, A_N$. Let x_i, d_i and ξ_i be the state, decision and stochastic variable respectively at stage i, then the state transition equation has the following form:

$$x_{i+1} = x_i + d_i + \xi_i, \quad i = 1, 2, \ldots, N. \tag{5.13}$$

Our problem is then to control this system such that the states over all stages are satisfactory, i.e., at the stage n, the objective is to maximize the expected fuzzy criterion function of the product $A_n \otimes A_{n+1} \otimes \ldots \otimes A_N$.

Based on the fuzzy criterion set operations, the expected fuzzy criterion function J_n of product $A_n \otimes A_{n+1} \otimes \ldots \otimes A_N$ is

$$J_n(\mathbf{x};) = \sum_{i=n}^{N} \gamma_i^{(n)} \int_R \mu_i(x_i + d_i + \xi_i) d\Phi_i(\xi_i) \tag{5.14}$$

where $\mathbf{x}$ is a state vector, $isadecisionvector, and\gamma_i^{(n)} = \lambda_i / \sum_{j=n}^{N} \lambda_j (5.15)$For the quantities d_i and ξ_i, positivity implies that it is an input, negativity implies that it is an output. We also mention that $\gamma_n^{(n)} : \gamma_{n+1}^{(n)} : \cdots : \gamma_N^{(n)}$ are simply coefficients of convex combination and $\gamma_n^{(n)} : \gamma_{n+1}^{(n)} : \cdots : \gamma_N^{(n)} = \lambda_n : \lambda_{n+1} : \cdots : \lambda_N$.

Let us now introduce the fuzzy criterion dynamic programming model associated with problem (5.14) as follows:

$$\begin{aligned} f_N(x) &= \sup_{d \in D_N} L_N(x + d) \\ f_n(x) &= \sup_{d \in D_n} \Big\{ \theta_n L_n(x + d) \\ &\qquad + (1 - \theta_n) \int_R f_{n+1}(x + d + \xi) d\Phi_n(\xi) \Big\} \\ n &\leq N - 1 \end{aligned} \tag{5.16}$$

where

$$L_n(y) = \int_R \mu_n(y + \xi) d\Phi_n(\xi), \tag{5.17}$$

and $\theta_n = \gamma_n^{(n)}$, $D_n = [q_n, Q_n]$ is a set of feasible policies, q_n and Q_n are not necessarily finite and positive.

We also make the following assumptions:

(A_1) The quantities ξ_1, ξ_2, ..., ξ_N are independently stochastic variables with distributions Φ_1, Φ_2, ..., Φ_N, respectively, and $E|\xi_i| < +\infty$, where E denotes the expected operator.

(A_2) The fuzzy criterion functions $\mu_i(x)$ of fuzzy criterion sets A_i: $R \to [0, 1]$ are continuous almost everywhere for all i, and $\lim_{x \to \pm\infty} \mu_i(x) = 0$.

(A_3) At least one of the following conditions holds,

1. μ_i are continuous functions for all i;

2. Φ_i are continuous distributions for all i.

Additionally, we mention the following facts:

1. When $D_n = R^+$ and the support of the distribution Φ_n is R^-, the equation (5.16) is a standard inventory model. In this case, the control is to order commodities from outside and the stochastic variables are quantities of demand.

2. When D_n is a closed interval on R^- and the support of the distribution Φ_n is R^+, the equation (5.16) is a reservoir operation model. Meanwhile, the control is to release water from the reservoir and the stochastic variables are quantities of inflow.

3. $f_n(x)$ is the expected fuzzy criterion function of the product $A_n \otimes A_{n+1} \otimes \cdots \times A_N$ in Euclidean space R^{N-n+1} for any n.

5.2.3 The Basic Theorem

Consider the following convolution operator,

$$H(y) = \int_R h(y+\xi) d\Phi(\xi) \tag{5.18}$$

where h is an integrable and bounded functional. Usually, the functional $H(y)$ is not necessarily continuous, but we have the following result.

Lemma 1 *If h is a measurable bounded functional which can have at most a countable number of discontinuities, then so is H. In particular, H is a continuous functional if at least one of the following conditions holds:*

1. *h is a continuous functional;*

2. *Φ is a continuous distribution.*

Theorem 1 *Assume (A_1), (A_2) and (A_3) for all stages, then the dynamic programming equation (5.16) defines a sequence of continuous functions. Moreover, there exists a Borel function $\widehat{d}_n(x)$ such that the supremum in (5.16) is attained for any x if D_n can be restricted to a compact set for any n.*

Proof: First, we know that $L_n(y)$ is continuous by assumption (A_3) and Lemma 1. At stage N, it is easy to show that $f_N(x)$ is continuous and there exists a Borel function $\widehat{d}_n(x)$ such that $f_N(x) = L_N(x+\widehat{d}_N(x))$, from a classical selection theorem, since we can restrict the feasible set D_N on a compact set.

For stage $n+1$, we suppose that f_{n+1} is continuous and there exists a Borel function $\widehat{d}_{n+1}(x)$ such that the supremum is attained by induction.

Next, we consider stage n. Since f_{n+1} is continuous, we know that $\int_R f_{n+1}(x+d+\xi)d\Phi_n(\xi)$ is continuous, and similarly the bracketed term of the equation. A similar argument can prove that f_n posses the same properties. □

Remark: Since the supremum in dynamic programming equation (5.16) is attained for any x, the *supremum* can be replaced by *maximum* in (5.16) under assumptions (A_1), (A_2) and (A_3).

5.2.4 Infinite Horizon Problem

We consider the infinite horizon problem which is to maximize

$$J_n(\mathbf{x};) = \sum_{i=n}^{\infty} \gamma_i^{(n)} \int_R \mu_i(x_i + d_i + \xi_i) d\Phi_i(\xi_i) \tag{5.19}$$

where $\gamma_i^{(n)} = \lambda_i/(\lambda_n + \lambda_{n+1} + \cdots)$ and $\lambda_1 + \lambda_2 + \cdots = 1$. Then the dynamic programming equation associated with problem (5.19) can be written as follows:

$$\begin{aligned} f_n(x) &= \sup_{d \in D_n} \Big\{ \theta_n L_n(x+d) \\ &\quad + (1-\theta_n) \int_R f_{n+1}(x+d+\xi) d\Phi_n(\xi) \Big\} \end{aligned} \tag{5.20}$$

where $\theta_n = \gamma_n^{(n)}$. In this section, we will suppose that all $\lambda_i > 0$. This implies that

$$0 < \theta_n < 1, \quad \forall n. \tag{5.21}$$

To establish existence and uniqueness theorems for the solution to (5.20), we first propose the following lemma:

Lemma 2 *Assume (A_1), (A_2) and (A_3) for all stages, then the relations*

$$\begin{aligned} W_n(x) &= \theta_n L_n(x+d_n) \\ &\quad + (1-\theta_n) \int_R W_{n+1}(x+d_n+\xi) d\Phi_n(\xi) \end{aligned} \tag{5.22}$$

define a bounded sequence of continuous functions. Moreover, W_n is explicitly defined by

$$W_n(x) = \sum_{i=n}^{\infty} \gamma_i^{(n)} L_{n,i}(x + d_n + \cdots + d_i) \tag{5.23}$$

where

$$L_{n,i}(y) = \int_R \cdots \int_R \mu_i(y + \xi_n + \cdots + \xi_i) d\Phi_n(\xi_n) \cdots d\Phi_i(\xi_i) \tag{5.24}$$

and d_n are any given feasible policies in D_n, respectively.

We will also need the following lemma which we state without proof:

Lemma 3 *Consider*

$$t(p) = \sup_{d \in D} \left\{ \theta h(p,d) + (1-\theta) \int_R f(p,d,r) d\Phi(r) \right\} \tag{5.25}$$

and

$$T(p) = \sup_{d \in D} \left\{ \theta H(p,d) + (1-\theta) \int_R F(p,d,r) d\Phi(r) \right\} \tag{5.26}$$

where Φ is a probability measure, h, H, f and F are integrable and bounded and $0 \leq \theta \leq 1$. Then, for any given p and $\epsilon > 0$, there exists a point $d^0 \in D$ such that

$$\begin{aligned} |t(p) - T(p)| \quad \leq \quad & \theta |h(p,d^0) - H(p,d^0)| \\ & + (1-\theta) \int_R \left[|f(p,d^0,r) \right. \\ & \left. - F(p,d^0,r)| d\Phi(r) \right] + \epsilon. \end{aligned} \tag{5.27}$$

We also state, without proof, the following theorems:

Theorem 2 [Existence Theorem] *We consider*

$$\begin{aligned} W_n^0(x) \quad &= \quad W_n(x) \\ W_n^{k+1}(x) \quad &= \quad \sup_{d \in D_n} \left\{ \theta_n L_n(x+d) \right. \\ & \quad \left. + (1-\theta_n) \int_R W_{n+1}^k (x+d+\xi) d\Phi_n(\xi) \right\}. \end{aligned} \tag{5.28}$$

Then we have

$$W_n^{k+1} \geq W_n^k \quad , \qquad k = 0, 1, 2, \ldots \tag{5.29}$$

and

$$W_n^k \uparrow W_n^*. \tag{5.30}$$

Moreover, the limit $W_n^*(x)$ is a continuous bounded solution to (5.20) if we assume (A_1), (A_2) and (A_3).

Theorem 3 [Uniqueness Theorem] *Assume (A_1), (A_2) and (A_3) for all stages, then there is one, and only one, bounded solution to (5.20).*

5.2.5 Stability Theorem

In the theory of functional equations, an interesting problem is the stability of solutions.

We set

$$\|\mu\|_\Phi = \sup_p \left\{ \int_R |\mu(p+\xi)| d\Phi(\xi) \right\}.$$

It is obvious that

$$\|\mu\|_\Phi \leq \sup_p |\mu(p)|.$$

Now, let $\{\widetilde{\mu}_n\}$ be another sequence of fuzzy criterion functions and $\{F_n\}$ be the corresponding solutions to (5.16). Then we have the following stability theorem.

Theorem 4 [Stability Theorem] *We consider model (5.16) and assume (A_1), (A_2) and (A_3) for all stages, then*

$$|f_n(x) - F_n(x)| \leq \sum_{i=n}^{N} \|\mu_i - \tilde{\mu}_i\|_{\Phi_i}, \quad \forall x. \tag{5.31}$$

5.2.6 Optimal Control Policy

In this section, we will give the operating characteristics of the resultant control policy.

Convolution transformation is an important operator. One important problem is to know whether the property of unimodality is preserved under the operation of convolution.

Now let f and g be two functions, then the convolution $f * g$ of f and g is given by

$$f * g(x) = \int_R f(x+y)g(y)dy.$$

Since the convolution of unimodal functions is, in general, not unimodal, Ibragimov[37] called a function g *strongly unimodal* if the convolution $f * g$ is unimodal for every unimodal f and proved that a nondegenerate density is strongly unimodal if and only if it is logconcave[1].

The Ibragimov's characterization of strong unimodality enables us to identify several standard distributions that are strongly unimodal. Some of these are:

(i) The normal distribution;

(ii) The uniform distribution on interval (a, b);

(iii) The gamma distribution with shape parameters $p \geq 0$;

(iv) The beta distribution with parameters (p, q) with $p \geq 1$ and $q \geq 1$;

(v) The Pearson-III distribution.

Lemma 4 *Let $f(x)$ be a unimodal function about a mode v on R, then we have*

$$h(x) = \sup_{d \in D} f(x+d) = \begin{cases} f(x+q), & v - q < x \\ f(v), & v - Q \leq x \leq v - q \\ f(x+Q), & x < v - Q \end{cases}$$

and $h(x)$ is also unimodal. Moreover,

$$[v - Q, v - q] \subset O$$

where O is the set of modes of h, $D = [q, Q]$ and q and Q are not necessarily finite and positive.

Proof. The proof is immediate. □

[1] A nonnegative function g is called logconcave if $\log g$ is concave.

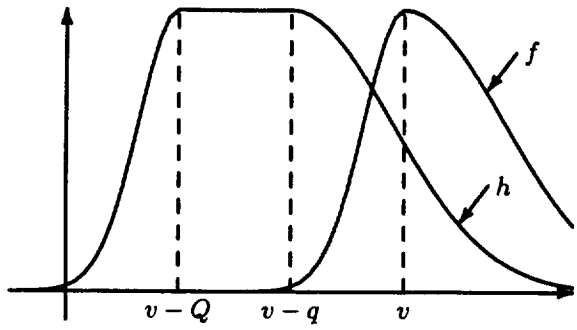

Figure 5.5: Comparison of h and f

Next, we consider the fuzzy criterion function μ_n which is unimodal about a mode x_n^*. When the return $f_{n+1}(y)$ is unimodal, we propose the following hypothesis.

Regular Hypothesis. *For any n, the mode*

$$x_n^*$$

is constrained as follows:

$$x_n^* \in O_{n+1}$$

where O_{n+1} is the set of modes of f_{n+1}.

The regular hypothesis means that the most satisfactory point at the current stage should lie on the set of modes of the return at the successive stage; in other words, it should be a beneficial initial state of the next stage.

Theorem 5 *Assume that (a) all fuzzy criterion functions are unimodal; (b) all distributions are strongly unimodal; and (c)the regular hypothesis holds, then, for any n, the optimal control is a bounded critical number policy, i.e.,*

$$d_n(x) = \begin{cases} q_n, & \overline{x}_n - q_n < x \\ \overline{x}_n - x, & \overline{x}_n - Q_n \le x \le \overline{x}_n - q_n \\ Q_n, & x < \overline{x}_n - Q_n \end{cases} \tag{5.32}$$

where the critical number $\overline{x}_n$ is a mode of $F_n(y)$.

5.2.7 Summary

In this section, we presented the concept of fuzzy criterion set and the resultant fuzzy criterion dynamic programming established on the fuzzy criterion sets. Some key properties of this novel model which is especially useful in multi criteria decision making were obtained.

Chapter 6

Applications to Space Systems

6.1 Application to Tethered Satellite System Retrieval

The tethered satellite system problem represents a highly nonlinear control problem with a five state variables, which is considerably more than the usual test problem found in the literature. A tethered system is any two or more bodies connected by a long thin structure. The system focused on in this example is the deployment, station-keeping, and retrieval of a target satellite from the Space Shuttle. With a fixed-length tether for systems in the 'station-keeping' phase, the equations of motion are still complex. Also, with a variable length tether—i.e., for systems in the deployment or retrieval phase—the equations of motion are further complicated by time-varying coefficients. The system is described as follows:

Inputs: State vector—$\mathbf{s} = \left[\theta, \dot{\theta}, \phi, \dot{\phi}, l\right]$.

Outputs: Tether length rate—$\mathbf{a} = \left[\dot{l}\right]$.

Equations of Motion: The model used in our simulation is a simplification of the actual dynamics [?]. There are two coordinate systems in the model—the orbital axes and the tether axes. See Figure 6.1. The orbital axes, XYZ, are such that the positive Z direction points to the center of the Earth, the positive X axis points in the direction of the trajectory, and thus the Y axis is perpendicular to the XZ plane. The tether axes, xyz, are such that the z axis is aligned with the tether and have the same origin as XYZ. The system attitude is described by

- In-plane motion or pitch: Rotation θ about the Y axis.
- Out-of-plane motion or roll: Rotation ϕ about the instantaneous X axis.

These equations serve only to simulate the system and are not used in the derivation of the control law:

$$Q_\theta(t) = \left(M_s + \frac{m_c}{3}\right) l^2 \left(\ddot{\theta} - \ddot{\theta_p}\right) c_\phi^2 - 2\left(M_s + \frac{m_c}{3}\right) l^2 \left(\dot{\theta} - \dot{\theta_p}\right) \dot{\phi} s_\phi c_\phi$$

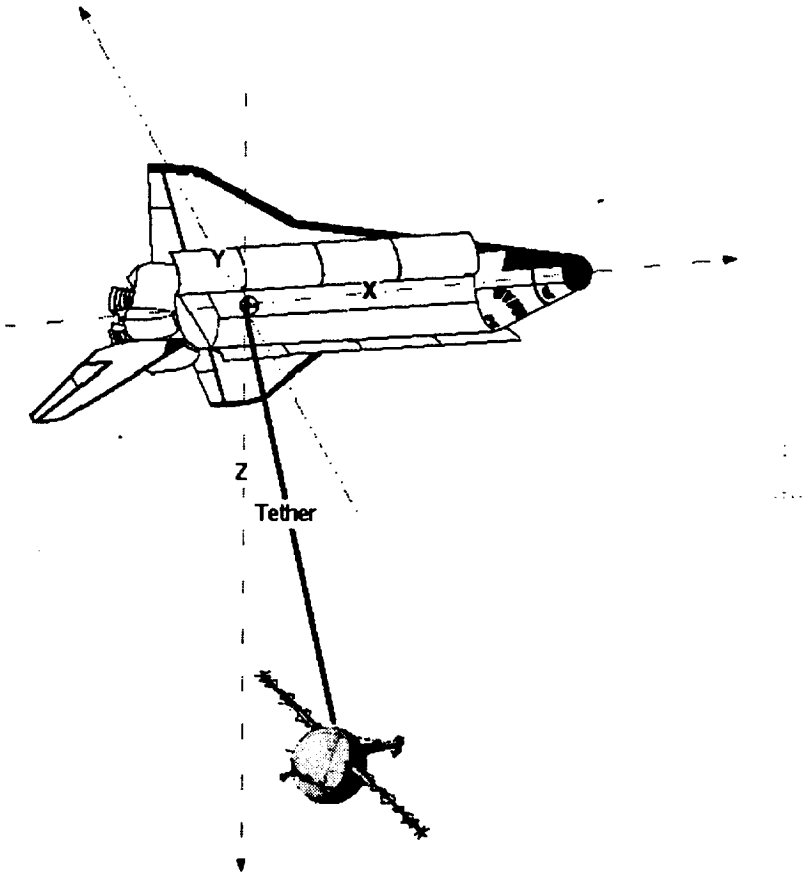

Figure 6.1: Tethered satellite system on the space shuttle.

$$+3\left(M_s+\frac{m_c}{3}\right)l^2\left(\frac{\mu_E}{R_o^2}\right)c_\phi^2 s_\phi c_\theta+2\left(M_s+\frac{m_c}{2}\right)\dot{l}l\left(\dot{\theta}-\dot{\theta}_p\right)c_\phi^2 \tag{6.1}$$

$$\begin{aligned} Q_\phi(t) &= \left(M_s+\frac{m_c}{3}\right)l^2\left(\dot{\theta}-\dot{\theta}_p\right)^2 s_\phi c_\phi+\left(M_s+\frac{m_c}{3}\right)l^2\ddot{\phi} \\ &\quad +3\left(M_s+\frac{m_c}{3}\right)l^2\left(\frac{\mu_E}{R_o^2}\right)c_\theta^2 s_\phi c_\phi+2\left(M_s+\frac{m_c}{2}\right)l\dot{l}\dot{\phi} \end{aligned} \tag{6.2}$$

where s_ϕ, s_θ, c_ϕ and c_θ are the sin and cosine functions of the respective state variables, M_s and m_c are the subsatellite and instantaneous tether mass, Q_ϕ and Q_θ are generalized external forces, R_o is the orbit radius, θ_p is the *true anomaly* and μ_E is the Earth gravitational constant. The values for the parameters used in the simulation are

$M_s = 150\text{kg}$	$\rho A = 0.0015\text{kg/m}$

Table 6.1: Parameters of Tethered Satellite System Simulation.

The control variable is the deployed tether length rate, $\dot{l}$. The state vector is defined as

$$\vec{X} = \{x_1, x_2, x_3, x_4, x_5\}^T \equiv \left\{\theta, \dot{\theta}, \phi, \dot{\phi}, l\right\}^T. \tag{6.3}$$

Controller Performance

The tethered satellite system experiments were replicated using various initial random number seeds. The learning algorithm was allowed to continue online for 500,000 time steps of 1 second each. The controller performed satisfactorily in these preliminary experiments in that it learned to retrieve the satellite from 100km to almost 10km before failure. The physical characteristics of the system make the retrieval phase near the Shuttle very nonlinear and dangerous to the crew. The controller learned the characteristics of the optimal control, namely the 'fishing' motion of sending the satellite out and then reeling it back in. The control variable during retrieval is shown in Figure 6.2 and the resulting tether length starting at 100km is shown in Figure 6.3. Using the TD(0) method for the learning algorithm, the average control surface using product-probability sum (P-PS) inference and center-of-area (COA) defuzzification method for all 20 replications along with the standard deviation are given in Figures 6.4 and 6.5. From the set of 20 replications, 6 resulted in a control policy that did not lead to failure. Of these 6 successful policies, the mean SSE, $\overline{SSE}$, was calculated as

$$\overline{SSE} = 0.00002\sum_{i=1}^{50000}\left[\Theta^2+\dot{\Theta}^2+\Phi^2+\dot{\Phi}^2+l^2\right] \tag{6.4}$$

with the results given in Table 6.2.

Using the Q-learning method for the learning algorithm, the objective is to minimize SSE via Q-learning. The average control surface using P-PS inference and COA defuzzification method for all 20 replications along with its standard deviation are given in Figures 6.6 and 6.7. From the set of 20

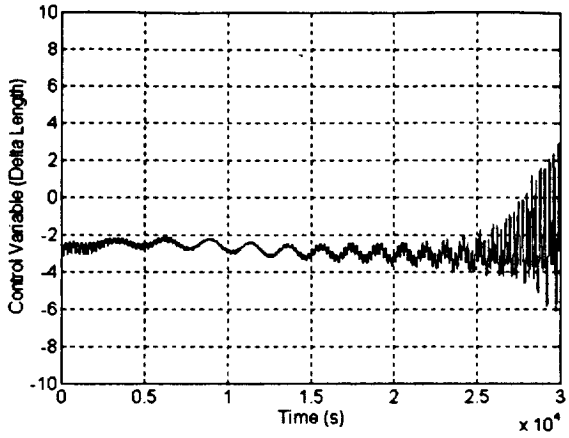

Figure 6.2: Control variable (length) during the retrieval of the satellite.

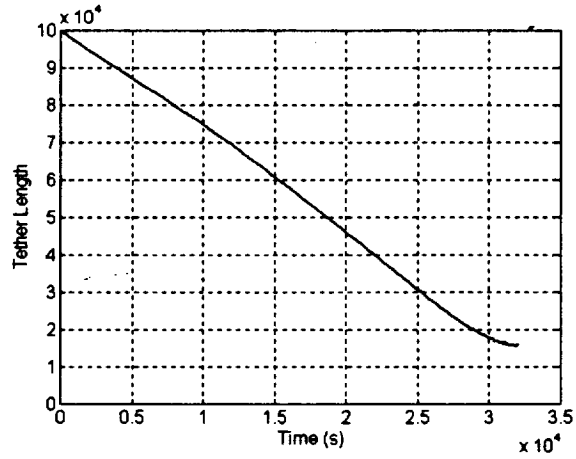

Figure 6.3: Tether length during the retrieval of the satellite.

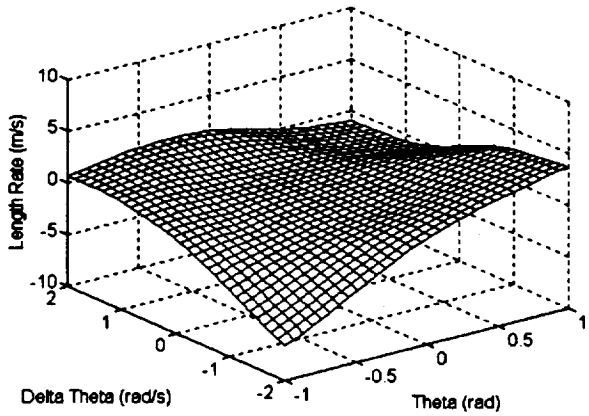

Figure 6.4: Average of control surfaces using TD(0)-based controller on the tethered satellite system.

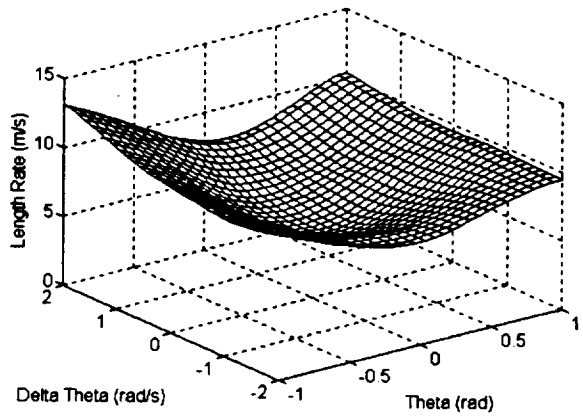

Figure 6.5: Standard deviation of control surfaces using TD(0)-based controller on the tethered satellite system.

Number of trials	20
Number of successful trials	6
Average SSE (success)	$5.3462 \cdot 10^9$
Standard Deviation SSE (success)	$3.4759 \cdot 10^9$

Table 6.2: Tethered Satellite System Results, TD(0) Algorithm.

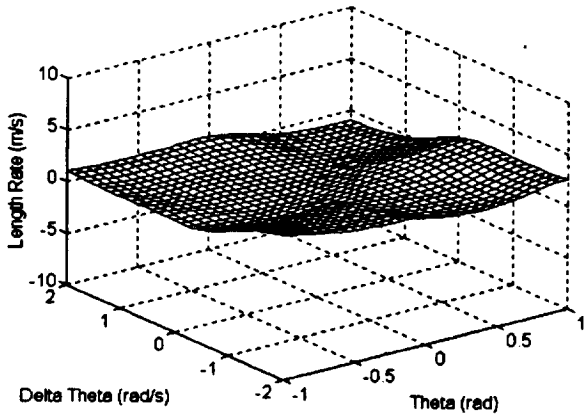

Figure 6.6: Average of control surfaces using Q-learning-based controller on the tethered satellite system.

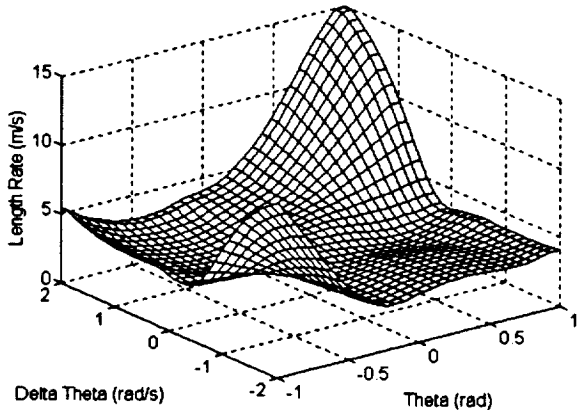

Figure 6.7: Standard deviation of control surfaces using Q-learning-based controller on the tethered satellite system.

replications, 7 resulted in a control policy that did not lead to failure. Of these 7 successful policies, the mean SSE, $\overline{SSE}$, was calculated with the results given in Table 6.3.

Number of trials	20
Number of successful trials	7
Average SSE (success)	$6.9573 \cdot 10^9$
Standard Deviation SSE (success)	$3.7682 \cdot 10^9$

Table 6.3: Inverted Pendulum System Results, Q-Learning Algorithm.

With $\alpha = 0.01$, we cannot reject the hypothesis that $\overline{SSE}$ for Q-learning is equal to $\overline{SSE}$ for TD(0). Therefore, the TD(0) algorithm and the Q-learning algorithm perform approximately the same under the above conditions on the tethered satellite system retrieval problem.

6.2 Application to Power System Stabilization Problems

One of the most intriguing and frequently investigated problem areas for deploying potent and novel tools of control engineering is the power system stabilization problem. Many different control strategies as well as controllers have been tested on this problem. Part of this interest is engendered by both the challenge and intractability of power systems which are characterized by the existence of power inherently complex, nonlinear, time-varying and indeterminable elements, simple controllers which work well in one situation may not perform equally well in another. As part of the experimental investigations with the SFAL controller, we explored its ability to learn a robust control law to stabilize the power system under various operating conditions.

The earliest stabilizers consisted of a lead-lag analog circuit with the speed as the input. Such a simple

controller cannot satisfy the high standards of the power system. PID controllers [34] perform better than the lead-lag circuit. Yet, unless their parameters are tuned automatically as operating conditions change, PID controllers in general do not work satisfactorily within a wide range of conditions. The self-tuning controller [12, 31, 53, 52] and the adaptive controller [9, 10, 36] are designed specifically for this purpose. By continuously identifying the model of the plant, the self-tuning controller adjusts its parameters to achieve optimal performance, while the adaptive controller adjusts its parameters based on the knowledge of the plant model. These two controller paradigms are time-consuming in design but can perform well under different operating conditions. Most recently, fuzzy logic controllers [32, 33, 35, 52, 64] have been successfully applied to stabilize power systems. It has been found that the fuzzy logic controller performs as well as the self-tuning controller in power system stabilization [52] and shows great potential for application in power systems. When the system is of large scale and of high complexity, however, it is not easy to extract the control rules from human expert(s) [39] and, even if this can be done, the expert's experience is still limited. Therefore, it is necessary to design a controller that can "learn" the control law via its own experience.

6.2.1 Mathematical Models of the Power System

The system considered here is composed of a synchronous machine with an exciter and a stabilizer connected to an infinite bus. The dynamics of the synchronous machine can be expressed as follows using the linearized incremental model [34]. These equations serve only to simulate the system and are not used in the derivation of the control law:

$$\begin{aligned}
\Delta\dot{\omega} &= \frac{1}{M}(\Delta T_m - \Delta T_e - \\
&\quad \Delta T_L - D\Delta\omega) && (6.5)\\
\Delta\dot{\delta} &= 377\Delta\omega && (6.6)\\
\Delta T_e &= K_e\Delta\delta + K_2\Delta e_q && (6.7)\\
\Delta\dot{e}_q &= \frac{1}{K_3T_{de}}(K_3\Delta e_{fd} - \\
&\quad K_3K_4\Delta\delta - \Delta e_q) && (6.8)\\
\Delta V_t &= K_5\Delta\delta + K_6\Delta e_q && (6.9)\\
\Delta\dot{V}_f &= \frac{1}{T_F}(K_f\Delta\dot{e}_{fd} - \Delta V_F) && (6.10)\\
\Delta\dot{e}_{fd} &= \frac{1}{T_E}(\Delta V_A - K_E\Delta e_{fd}) && (6.11)\\
\Delta\dot{V}_A &= \frac{1}{T_A}(K_A\Delta V_{ref} - K_A\Delta V_F + K_Au - \\
&\quad K_AK_6\Delta e_q - K_AK_5\Delta\delta - \Delta V_A) && (6.12)
\end{aligned}$$

$$|u| \leq u_{max} \tag{6.13}$$

where

V_{ref}	constant reference input voltage
ΔV_t	terminal voltage change,
ΔV_o	infinite bus voltage change
Δe_{fd}	equivalent excitation voltage change
Δe_q	q-axis component voltage behind
	transient reactance change
ΔV_F	stabilizing transformer voltage change
u	stabilizer output
ΔT_m	mechanical input change
ΔT_e	energy conversion torque change
ΔT_L	load demand change
$\Delta\delta$	torque angle deviation,
$\Delta\omega$	angular velocity deviation
K_A, K_E	voltage regulator gains
T_A, T_E	voltage regulator time constants
K_F	stabilizing transformer gain
T_F	stabilizing transformer time constant
$K_1, \ldots, K_6$	constants of the linearized
	model of synchronous machine
T_{do}	d-axis transient open circuit
	time constant
M	inertia coefficient
D	damping coefficient
T_s	sampling period

The objective of the controller is to drive the state of the system $\vec{x}$ to $[0, 0]$ via the stabilizer output u. The values for the above parameters are given in Table 6.4 below.

6.2.2 Simulation Results

We are interested in investigating the practicality and effectiveness of our newly developed controller in stabilizing the power system whose model depicted in the foregoing section was used in the simulation phase for system mimicking only but not for control purposes. The experiments run on the power

$K_1 = 1.4479$	$K_2 = 1.3174$	$K_3 = 0.3072$
$K_4 = 1.8050$	$K_5 = 0.0294$	$K_6 = 0.5257$
$K_A = 400$	$T_F = 1.0$	$T_A = 0.05$
$D = 0$	$T_{do} = 5.9$	$K_E = -0.17$
$M = 4.74$	$T_E = 0.95$	$K_F = 0.025$
$\Delta T_m = 0$	$\Delta V_{ref} = 0$	$T_s = 0.01$

Table 6.4: Parameters of Simulation.

system stabilization problem consisted of multiple replications of the learning phase of the controller on a simulated power system written in C. The inputs to the controller are $\Delta\omega$ and $\Delta\dot{\omega}$. Thus, the controller in effect mimics a PD-like controller with unknown structure. The state space is defined as $\Delta\omega \in [-0.012, 0.012]$ and $\Delta\dot{\omega} \in [-0.025, 0.025]$. The number of nodes for the SFDN is set at $N = 25$ and there are 5 reference control fuzzy sets defined for $u \in [-0.12, 0.12]$. Once the controller has completed the learning phase, it is used as a stabilizer in the system.

Several experiments were run and an example of the resulting controller is shown in the figures below. Figure 6.8 shows the transient process of $\Delta\omega$ when the load increases 0.05 pu and 0.3 pu, respectively. It takes about 2 seconds for the speed deviation $\Delta\omega$ to vanish for the 0.05 pu load change and about 3 seconds for the 0.3 pu load change. Figure 6.9 shows the learned control surface using product-limited sum inference and center-of-area defuzzification.

Comparison to Existing Controllers

The simulation results clearly showed that the controller can learn an effective control law to stabilize the system under varying load conditions. However, the results, are not optimal with regard to settling time. The settling time obtained with our controller was slightly longer than the results obtained using an existing PID controller [34] and a fuzzy controller [35], but comparable to or shorter than the settling time for other fuzzy controllers [32, 33, 64] reported in the literature. The optimality issue (see Section 2.6.2) was subsequently investigated further. Although not optimal, the relative ease of developing an "efficient" controller via the self-learning controller with respect to the existing methods demonstrates the potential of this approach and in fact sufficiently meets the objectives of this project.

Despite the foregoing, the advantages of our controller over other controllers reported in the literature that have been applied to the power systems stabilization problem are very significant and should be noted:

- The controller successfully learned the control law via its own experience. It did not require the analytic solution of a dynamical model, the tuning of parameters as in PID control, and it did not rely on existing expert knowledge about the control of the process.
- The learning phase of the controller took less than 5 minutes to complete. Thus, there is a huge

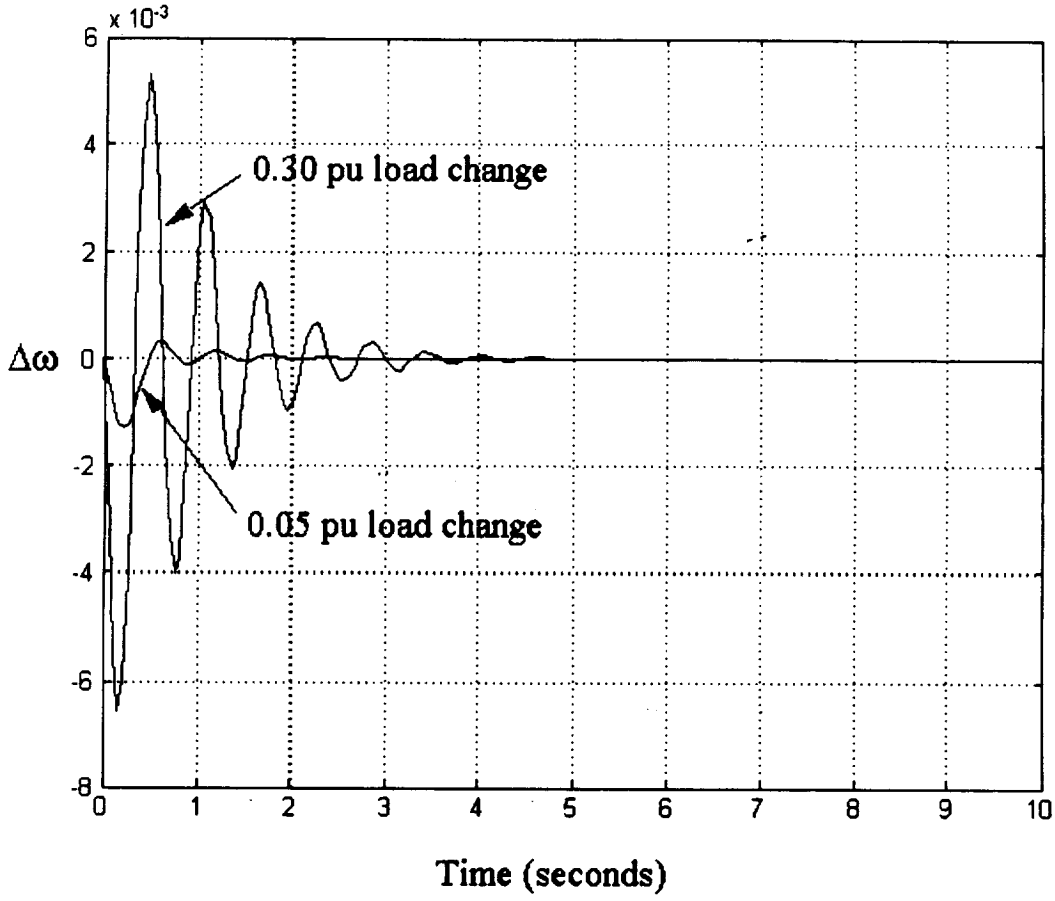

Figure 6.8: Transient Process for Selected Load Changes.

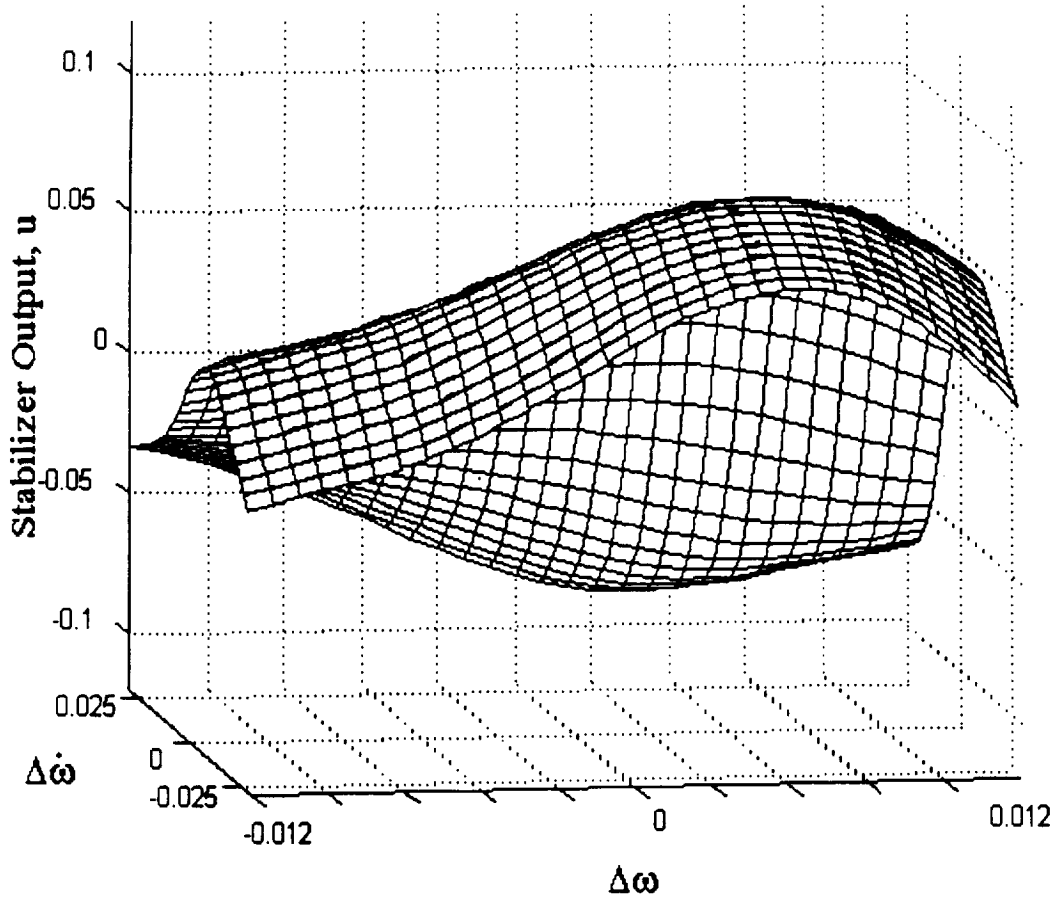

Figure 6.9: Learned Control Surface for Stabilization.

time savings in development over the existing controllers.

- The internal controller parameters (that control the learning and other attributes) are very robust—the parameters used for the inverted pendulum problem were used for the power system stabilization problem. No tuning of these parameters was performed, although doing so might have improved the resulting control.
- The resulting control is robust—the controller can handle a more extreme range of load changes than the PID controller [34].

Experimental Results with Defuzzification Filter

With equations from [29], we have

$z_1 = \Delta\Omega,$

$z_2 = -.2779\Delta e_q - .3055\Delta\delta - T_l/M,$

$z_3 = .1533\Delta e_q - .0471\Delta e_{fd} - 115.1735\Delta\Omega + .085\Delta\delta,$

$z_4 = 31.9259\Delta e_q + .0344\Delta e_{fd} - .0496\Delta V_a +$
$32.045\Delta\Omega + 35.1345\Delta\delta + 115.1735T_l/M,$

$z_5 = \Delta\delta,$

$z_6 = \Delta e_q + \Delta V_f.$

Since the purpose of the stabilization problem is to control the system under disturbance, the desired output can be chosen as $y_r = 0$. Thus, with Equation (??), we have the desired defuzzified value as where $c_0 = 10, c_1 = 20, c_2 = 30$ and $c_3 = 10$ which can be chosen by a proper pole placement method. Parameters of the power system can be found in Table 6.5. The desired defuzzified control signal is given by Equation 6.14.

Table 6.5: Parameters of power system used in simulation

$K_1 = 1.4479$	$K_2 = 1.3174$	$K_3 = 0.3072$
$K_4 = 1.8050$	$K_5 = 0.0294$	$K_6 = 0.5257$
$K_A = 400$	$T_F = 1.0$	$T_A = 0.05$
$D = 0$	$T_{do} = 5.9$	$K_E = -0.17$
$M = 2H = 4.74$	$T_E = 0.95$	$K_F = 0.025$

$$u = \frac{1}{396.8}(182.077\Delta e_q + 396.8\Delta V_F + 5.405\Delta e_{fd}$$
$$+1.0282\Delta V_A + 13246\Delta\omega - 7.8899\Delta\delta + \sum_{i=1}^{4} c_{i-1}z_i) \quad (6.14)$$

Once the desired defuzzification control signal is obtained, the recursive formula in Section 3 is used to estimate the parameters of the filter. The actual defuzzification control signal is sent to the generator as stabilization signal. In this application, control signals which were obtained by a adaptive fuzzy-neuro

controller are filtered by the proposed defuzzification filter, and then the filtered control signals are sent to the generator (See Fig. 6.10). Since the fuzzy-neuro controller could not always produce control signals which stabilize the generator, the introduction of the defuzzification filter has a remarkable stabilization effect: For control signals which were able to stabilize the generator, the defuzzification filter will filter the control signal such that the settling time of the system is shorter (See Fig. 6.11); for control signals where were not able to stabilize the generator, the defuzzification filter can filter the control signal such that the new control signal can stabilize the generator (See Fig. 6.12); when the neuro-fuzzy self-learning controller was able to stabilize the system but the settling time was satisfactory, the defuzzification filter can revise the control signal and yield a better settling time (See Fig. 6.13). Thus, those experiments are all supportive of the application of defuzzification filter.

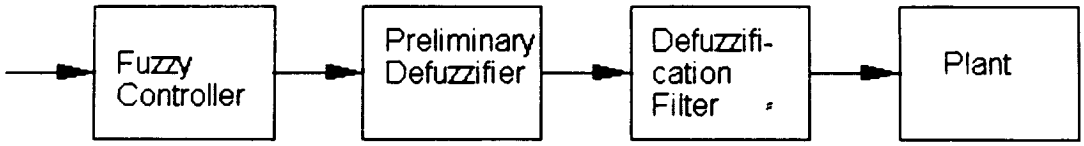

Figure 6.10: Structure of the control system with the defuzzification filter.

Comparison of the neuro-fuzzy self-learning controller and the filter

Two different measures of performance can be utilized here. One performance measure is the settling time of the transient process, and the other the success rate of stabilization. Therefore, we will compare the neuro-fuzzy self-learning controller and the defuzzification filter with respect to these two measures. In addition, control signals from the neuro-fuzzy self-learning controller and the defuzzification filter will also be compared. Comparison of the control signals and settling times It will be interesting to compare, in each case, the control signals from the filter and the neuro-fuzzy self-learning controller, and to find out why the neuro-fuzzy self-learning controller sometimes fails to stabilize the system while the filter does not.

Comparisons were made when the load was increased by $.05pu$ and $.30pu$ respectively. Fig. 6.14 shows the control signals when the load change was $.05pu$ and in this case, the neuro-fuzzy self-learning controller could yield a satisfactory settling time and the filter gives a better settling time. Fig. 6.15 shows the control signals when the neuro-fuzzy self-learning controller failed to stabilize the system while the introduction of the filter can stabilize the system. Obviously, the control signal from the filter is much smaller than that from the neuro-fuzzy self-learning controller. Fig. 6.16 shows a different case where the self-learning controller can stabilize the system but the settling time was not satisfactory. Again, it has been found that the control signal from the neuro-fuzzy self-learning controller is greater than that of the filter. When the load disturbance is $.3pu$, experiments were also carried out with different simulation

results obtained. Figures 6.20–6.22 show the control signals when the load disturbance is $.3pu$. Obviously, it can be seen that

- On the average, the control signal from the neuro-fuzzy self-learning controller is quite different from the ideal one while the control signal from the filter is quite close to the ideal one;
- In most cases, the control signal from the filter is about 180 degrees lag of the ideal signal while the control signal from the neuro-fuzzy self-learning controller could be in step with that from the filter;
- When the control signal from the neuro-fuzzy self-learning controller is quite close to the ideal one, the self-learning controller could stabilize the system;
- The signal from the filter can stabilize the system whether or not the neuro-fuzzy self-learning controller can.

This, indirectly, indicates that the functionality of the defuzzification filter is to rectify the control signal from the preliminary defuzzifier to make it be close to the ideal one. Another implication from this observation is that how can one make the neuro-fuzzy self-learning controller learn the ideal control signal in a larger probability.

Comparison of the success rates To indicate the effectiveness of the defuzzification filter, let us have an interesting comparison of the success rates of both the neuro-fuzzy self-learning controller and defuzzification filter. Assume whether or not the system can be stabilized is totally unknown in advance. Thus, if we assume $x = 1$ when the system is stabilized and $x = 0$ if not, then, x will be a random variable (or indicator). Denote x_s and x_f as the indicators for the neuro-fuzzy self-learning controller and the filter respectively, then $E(x_s)$ and $E(x_f)$ will be the mean value of the probability that the system is stabilized by the neuro-fuzzy self-learning controller and the filter respectively.

We can estimate $E(x_s)$ and $E(x_f)$ with $\hat{P}_1 = \sum x_s/N$ and $\hat{P}_2 = \sum x_f/N$ where N is the number of runs. To do so, twenty runs of the simulation were carried out with both $E(x_s)$ and $E(x_f)$ recorded. It was found that $\sum x_s/20 = 0.45$ and $\sum x_f/20 = 1$. Thus, the difference in success rates of the neuro-fuzzy self-learning controller and the defuzzification filter is 0.55. To investigate if this difference is due to chance or due to the improvement of the filter, we applied the following procedure [59]:

Let P_1 and P_2 be the success rates of the neuro-fuzzy self-learning controller and the defuzzification filter, respectively. We will use the normal approximation to test the hypothesis that these two success rates are equal; that is,

$$H_0 : \ P_1 = P_2$$

$$H_1 : \ P_1 \neq P_2$$

Then, compute $\hat{P}$ with the following formula:

$$\hat{P} = \frac{\hat{P}_1 + \hat{P}_2}{2}. \tag{6.15}$$

The statistic to test H_0 is

$$Z_0 = \frac{\hat{P}_1 - \hat{P}_2}{\sqrt{\hat{P}(1-\hat{P})(\frac{2}{N})}}. \tag{6.16}$$

We should reject H_0 when $|Z_0| > Z_{\alpha/2}$ where α is a preselected significance level (= .05 in this study). A simple computation yielded that $Z_0 = -3.8952$ and it is known that $Z_{.025} = 1.95$. Thus, the hypothesis H_0 should be rejected. That is, the difference between the success rates of the neuro-fuzzy self-learning controller and the defuzzification filter in this study is not due to chance, but due to the significant contribution of the filter to the stability of the system.

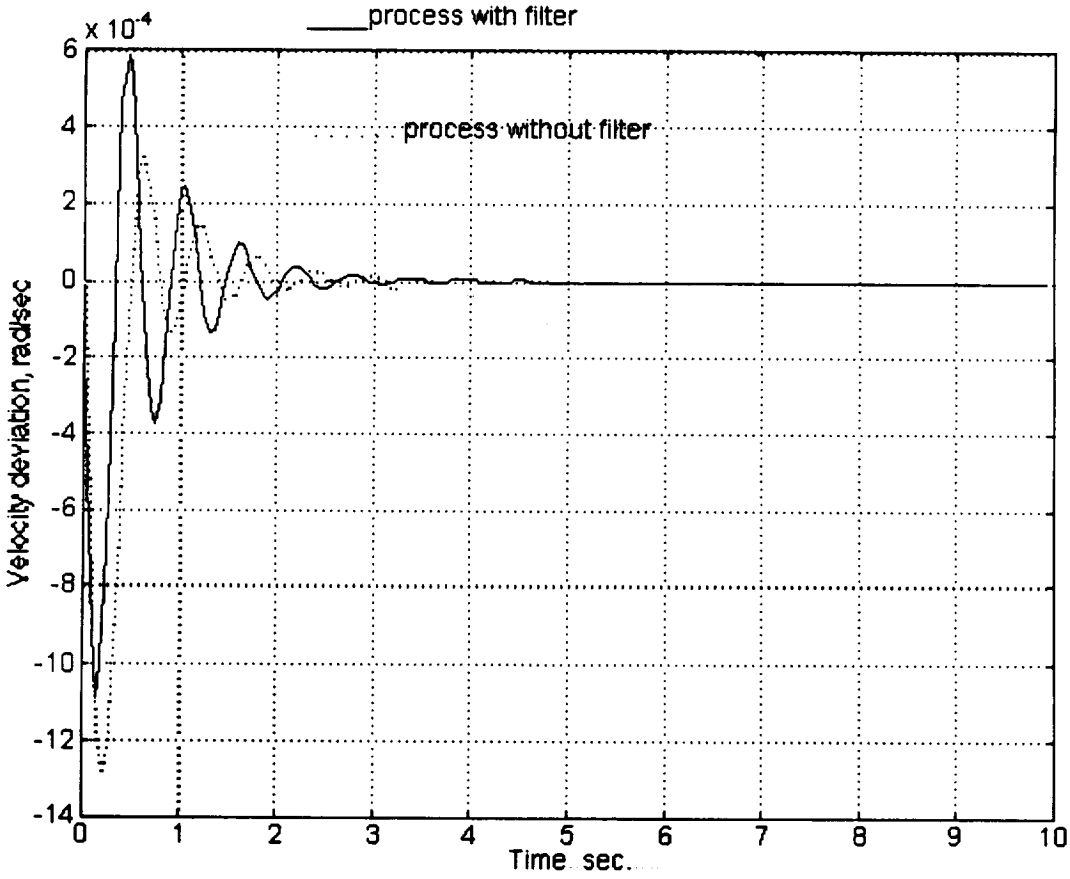

Figure 6.11: Transient Process—Case 1: Both the neuro-fuzzy self-learning controller and the filter stabilize the system.

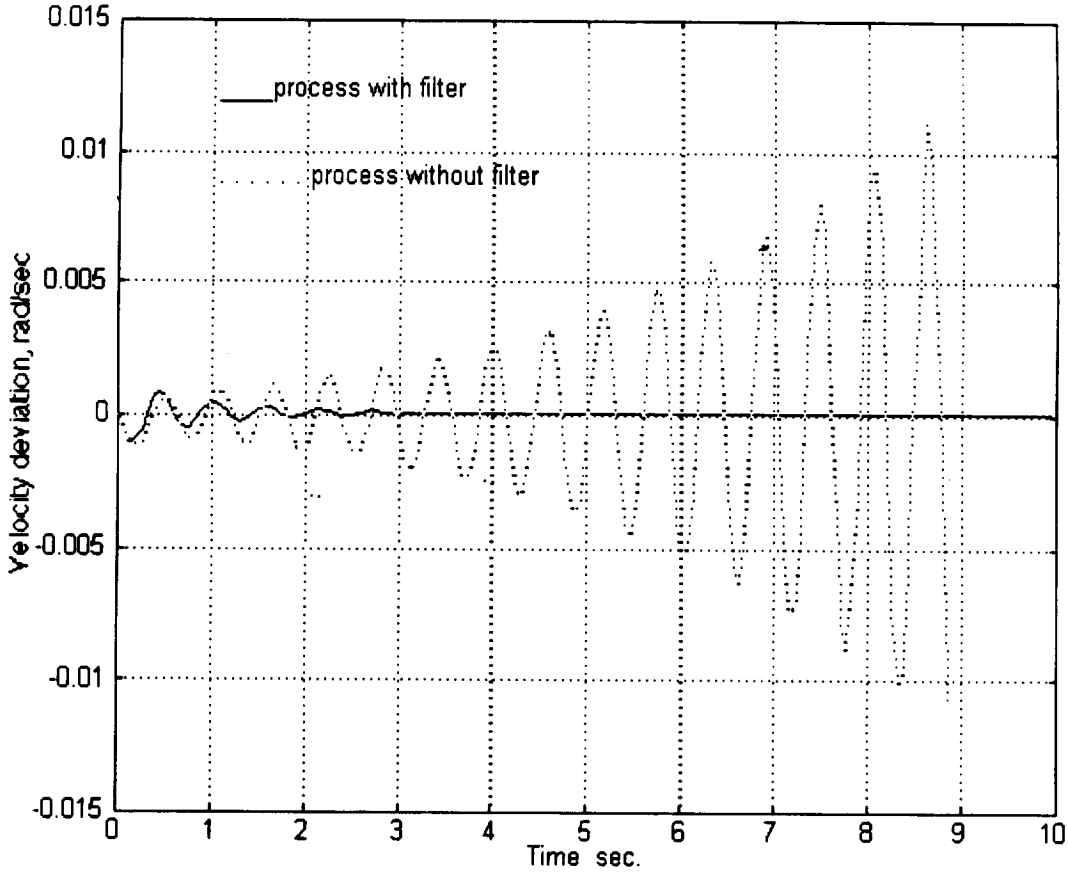

Figure 6.12: **Transient Process—Case 2:** The neuro-fuzzy self-learning controller fails but the filter does not.

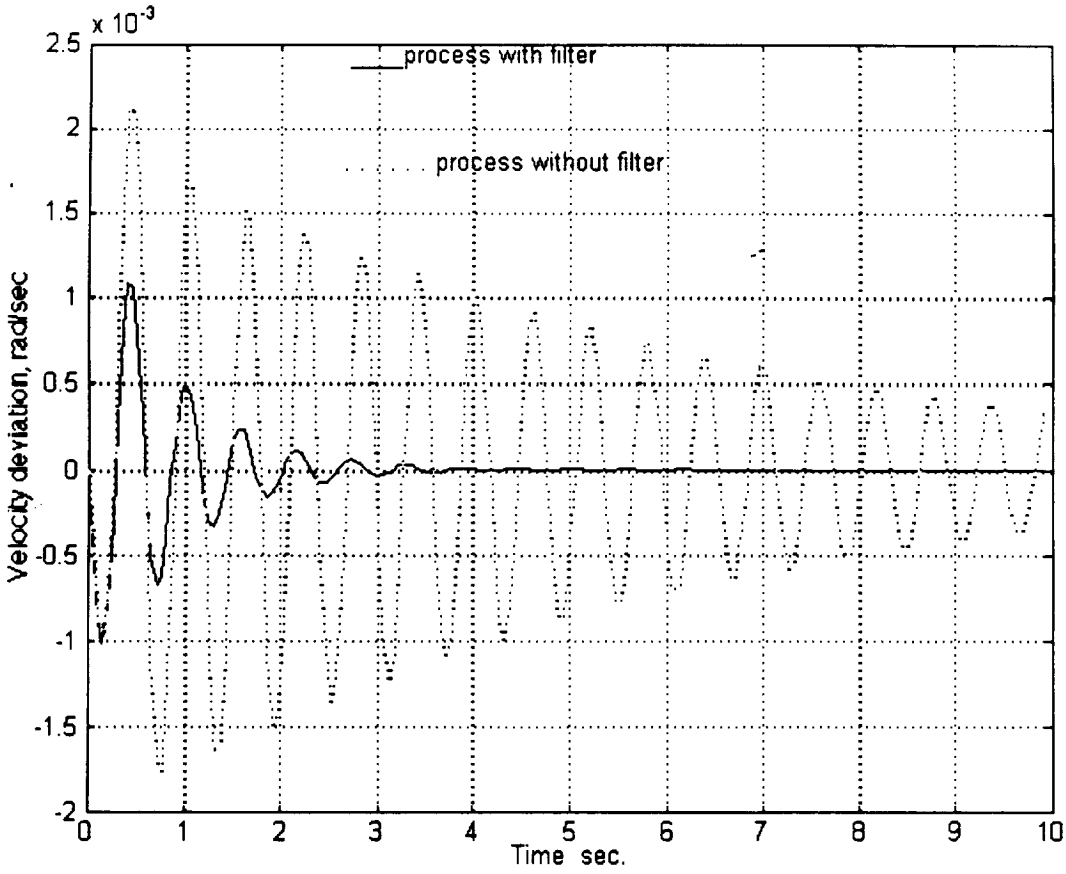

Figure 6.13: **Transient Process—Case 3:** The neuro-fuzzy self-learning controller yields a longer settling time.

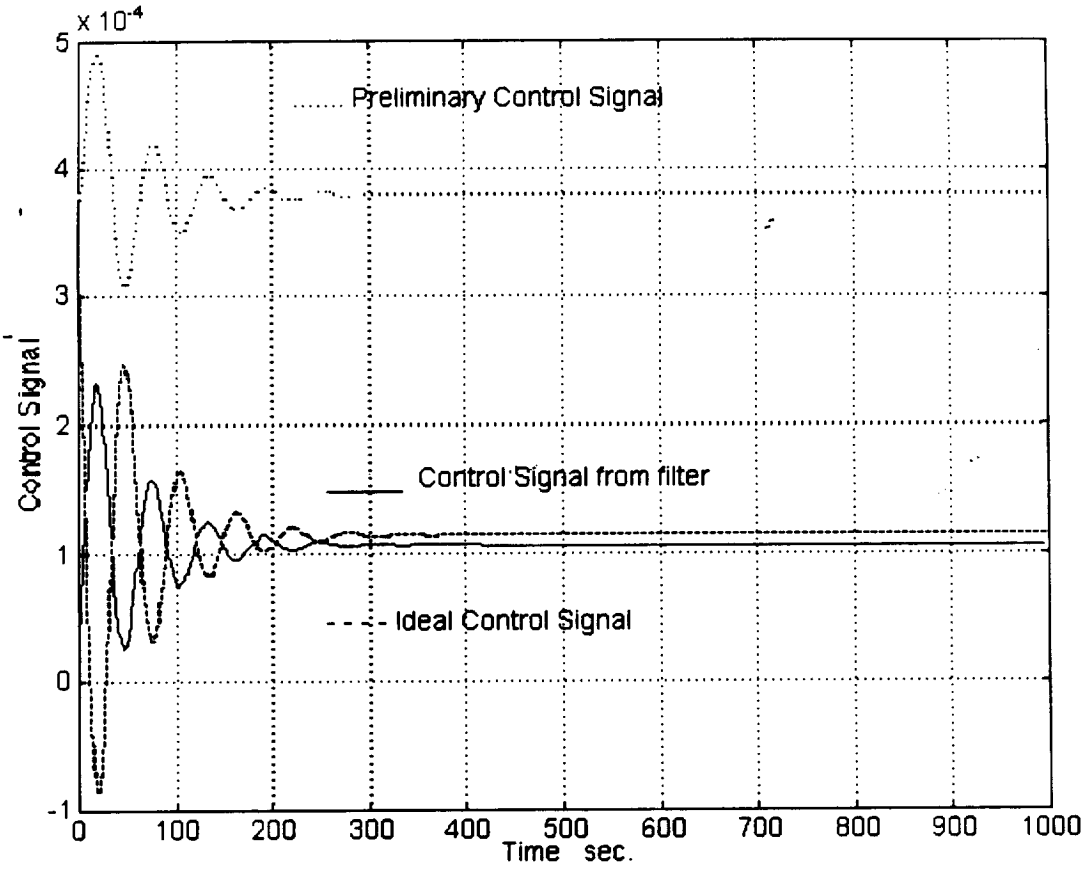

Figure 6.14: Control Signals—Case 1.

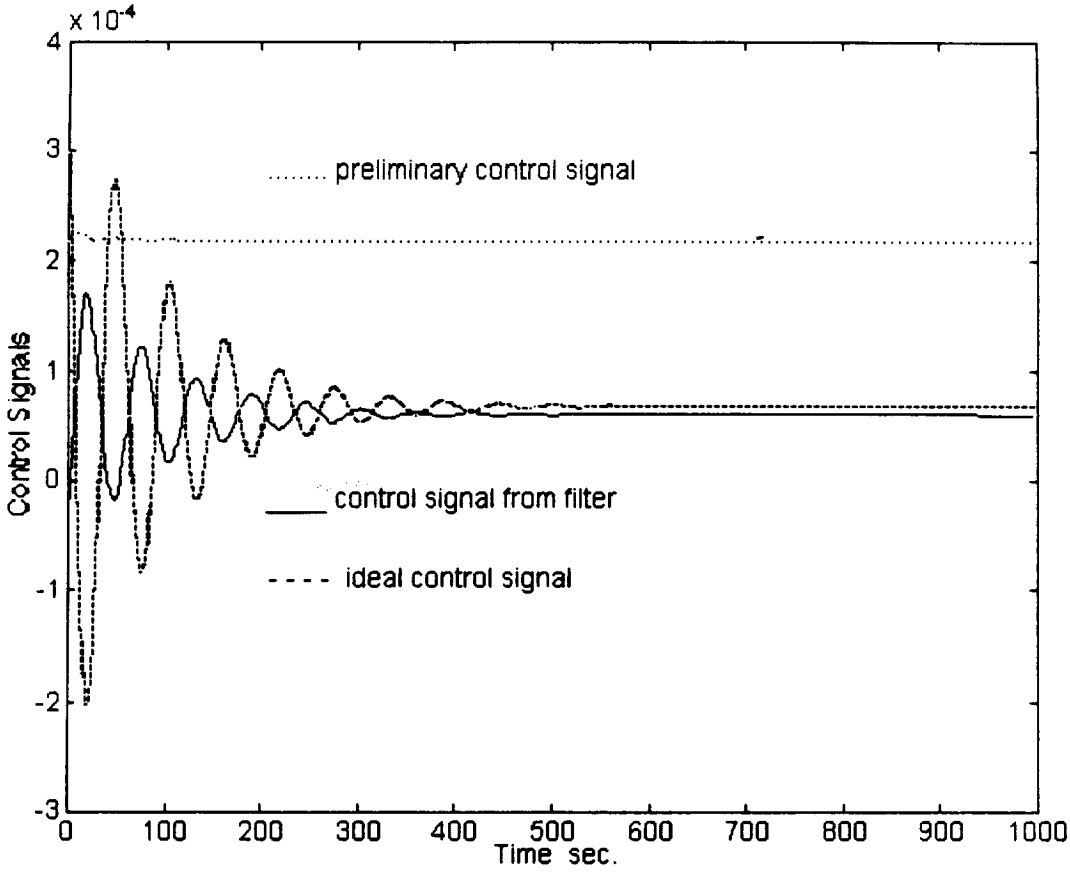

Figure 6.15: Control Signals—Case 2.

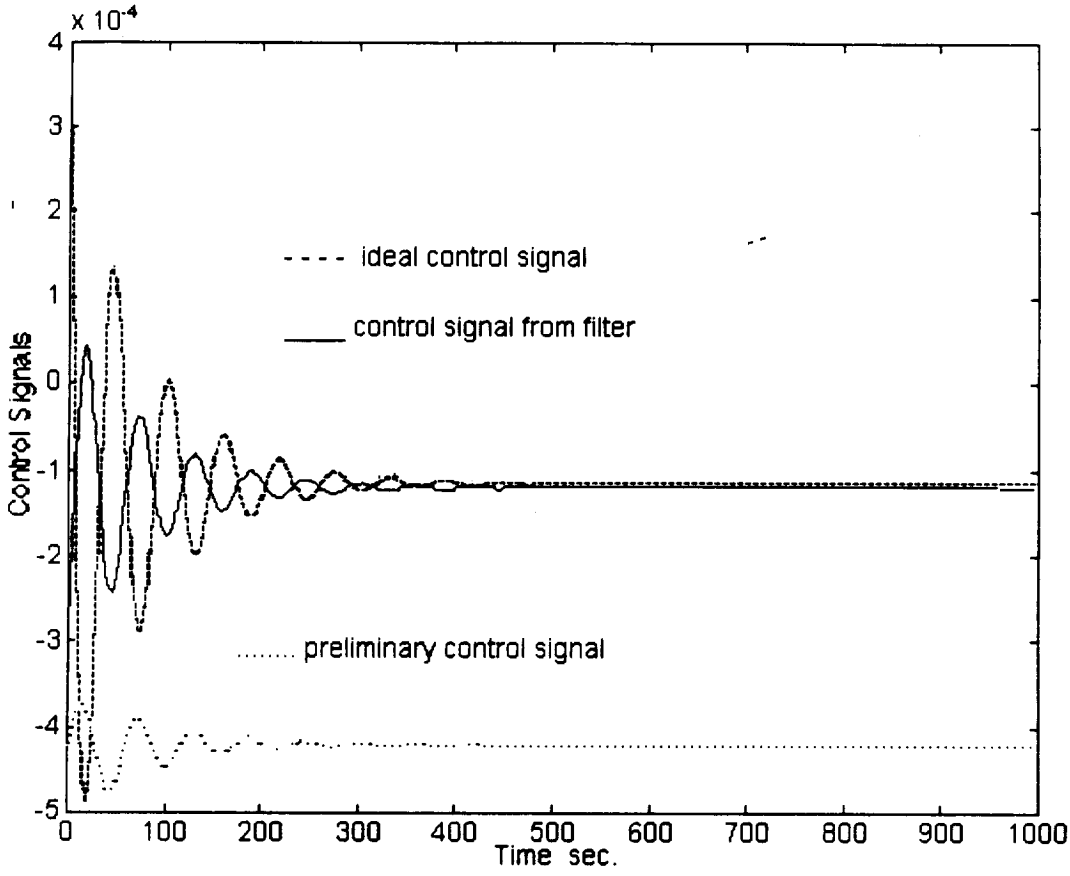

Figure 6.16: Control Signals—Case 3.

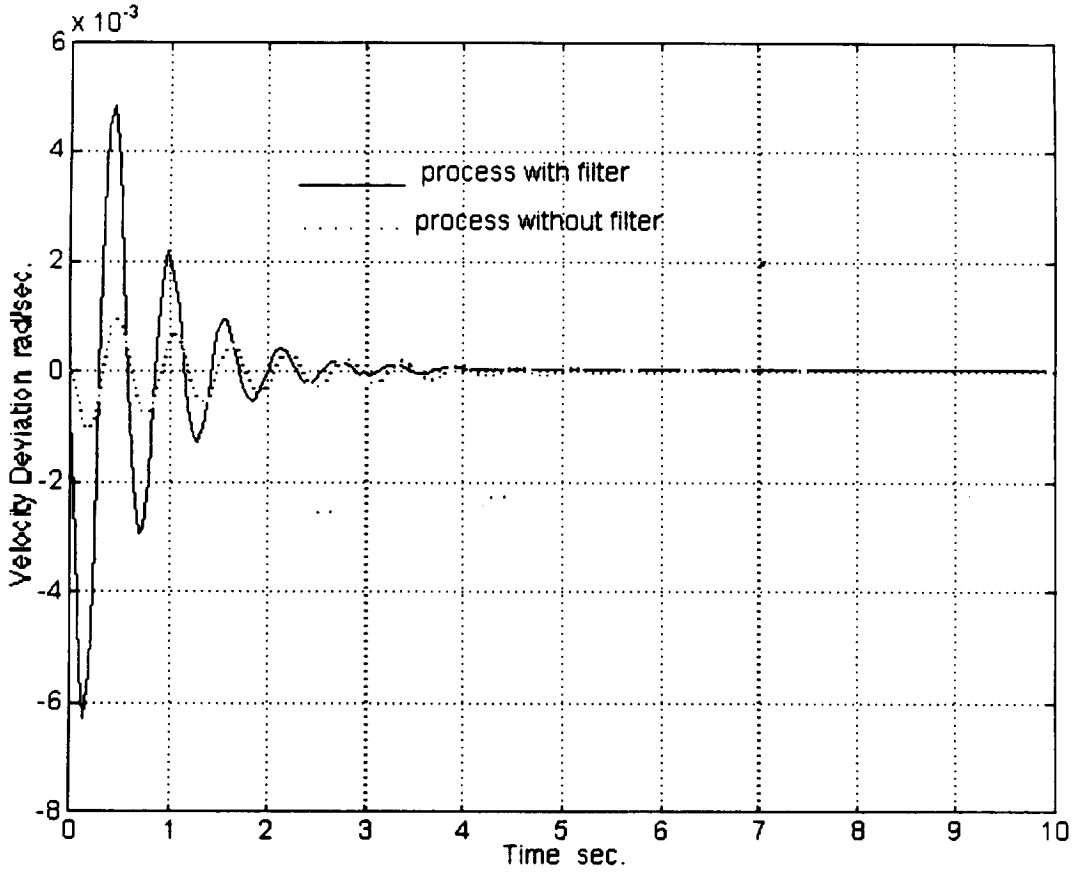

Figure 6.17: **Transient Process—Case 4:** Both the neuro-fuzzy self-learning controller and the filter stabilize the system.

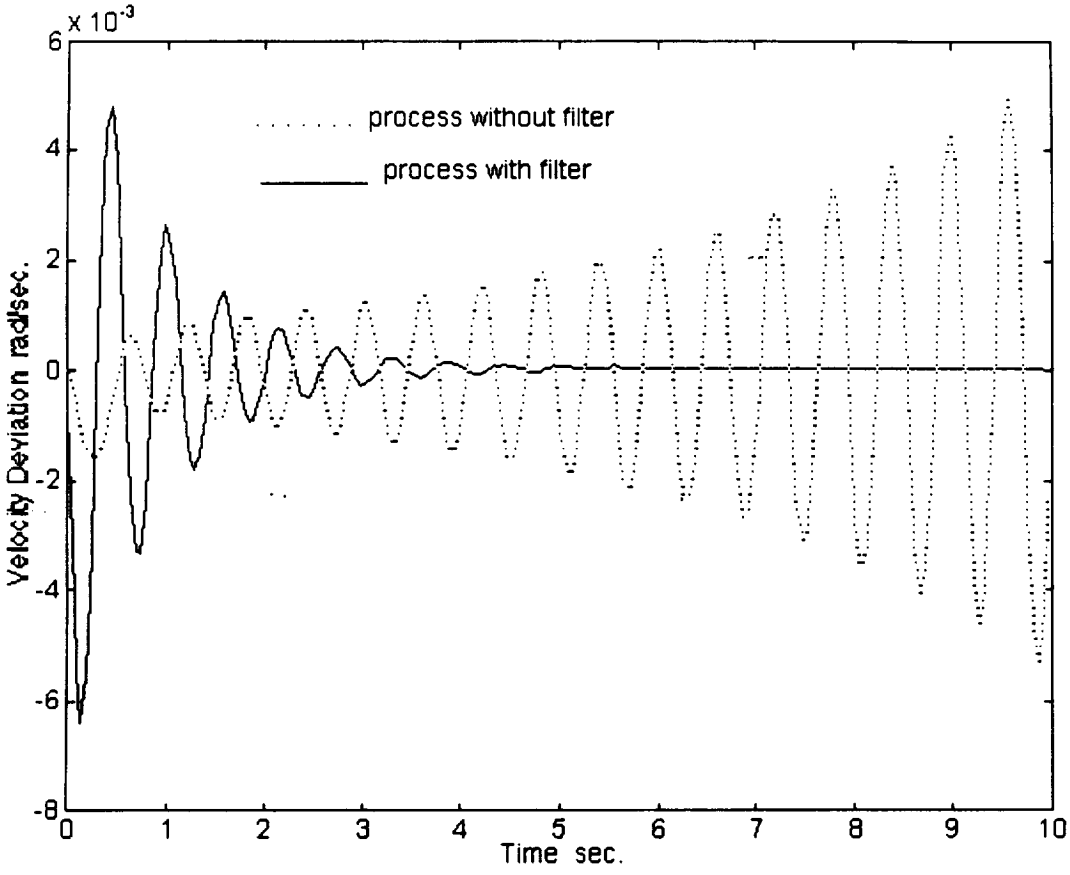

Figure 6.18: **Transient Process—Case 5**: The neuro-fuzzy self-learning controller yields a longer settling time.

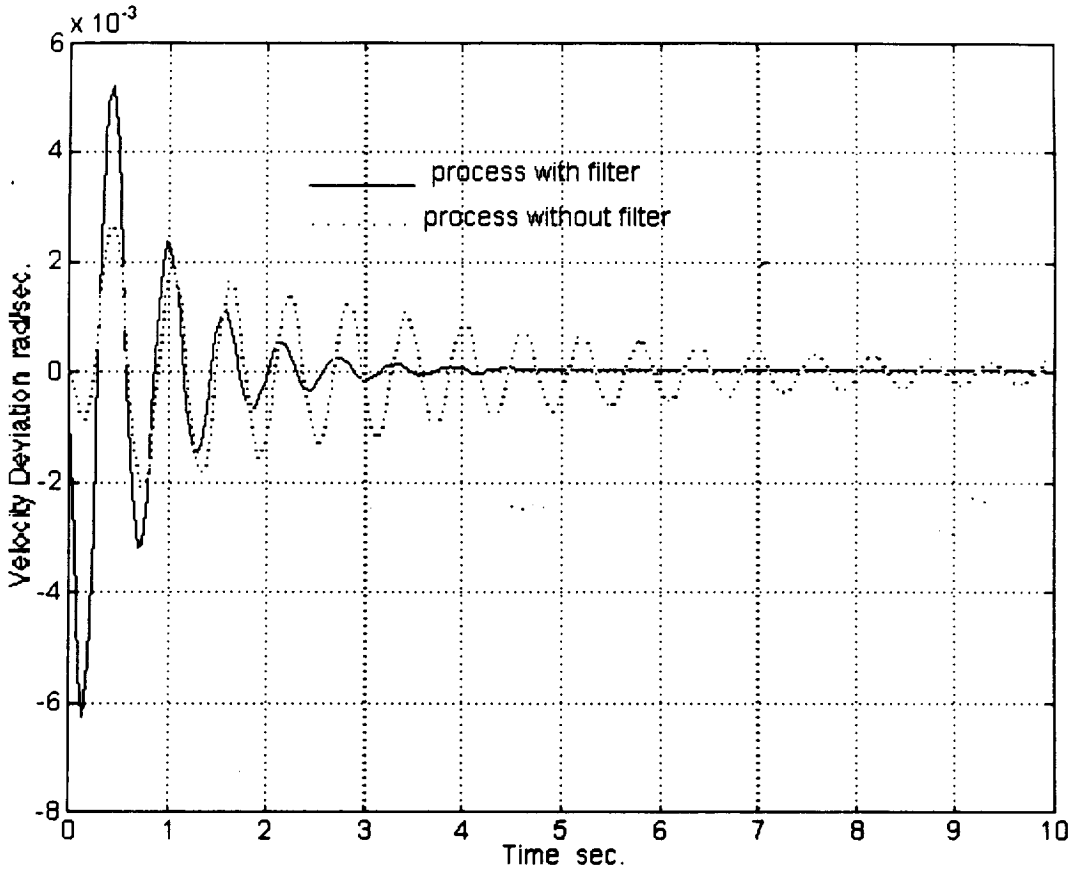

Figure 6.19: Transient Process—Case 6: Both the neuro-fuzzy self-learning controller and the filter stabilize the system.

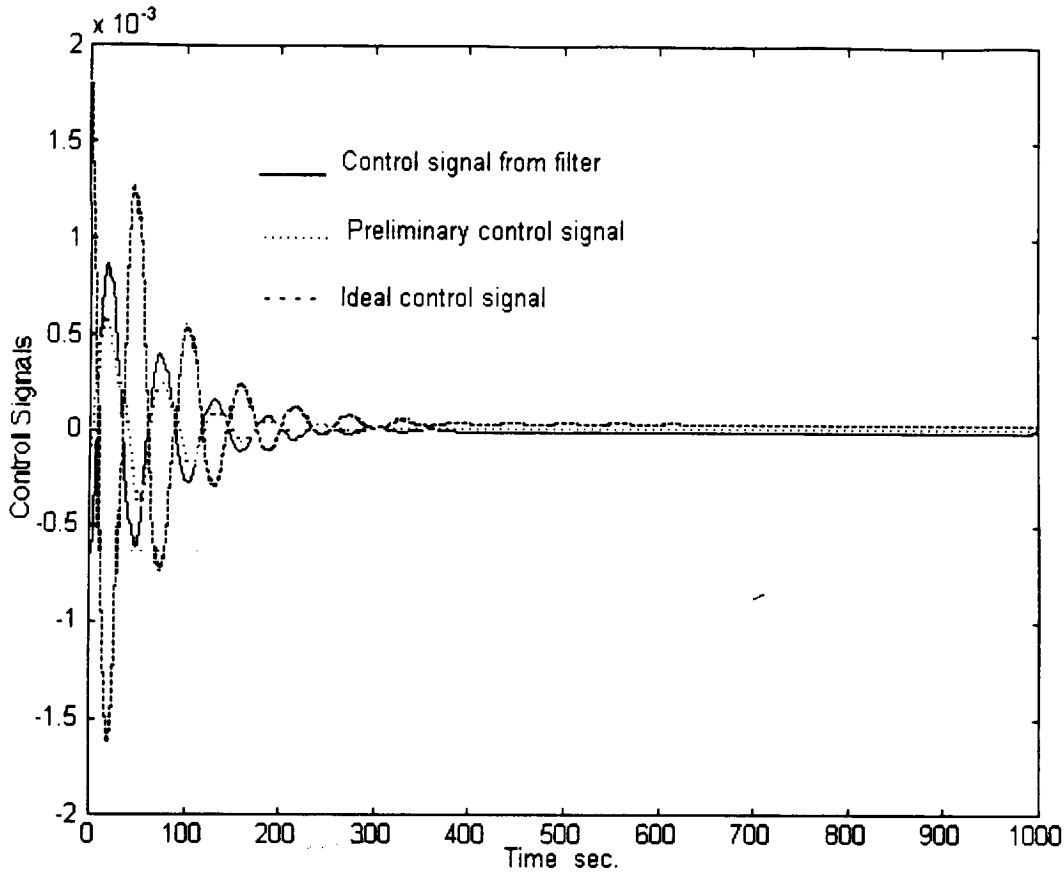

Figure 6.20: Control Signals—Case 4.

6.2.3 Summary

In this section, the design and applications of the defuzzification filters were presented. To design a defuzzification filter, the inverse of the plant must be known, and reference signal was used to generate the filter output—the defuzzified signal. The parameters of the filter were estimated by means of a recursive learning law, which is much less computational intensive. A filter was designed for the power system stabilization problem for which a neuro-fuzzy self-learning controller had been designed and applied. It was found that the defuzzification filter can yield better performance in the sense that when the neuro-fuzzy self-learning controller could stabilize the system, the application of the defuzzification filter will yield faster transient process and if the neuro-fuzzy self-learning controller failed to stabilize the system, the defuzzification filter will stabilize the system. Simulation results indicated the significant improvement of the filter applied in the power system stabilization problem. For future study, we are intending to expand the concept of defuzzification filter and search for a better way to design such filters, and explore potential applications of defuzzification filters.

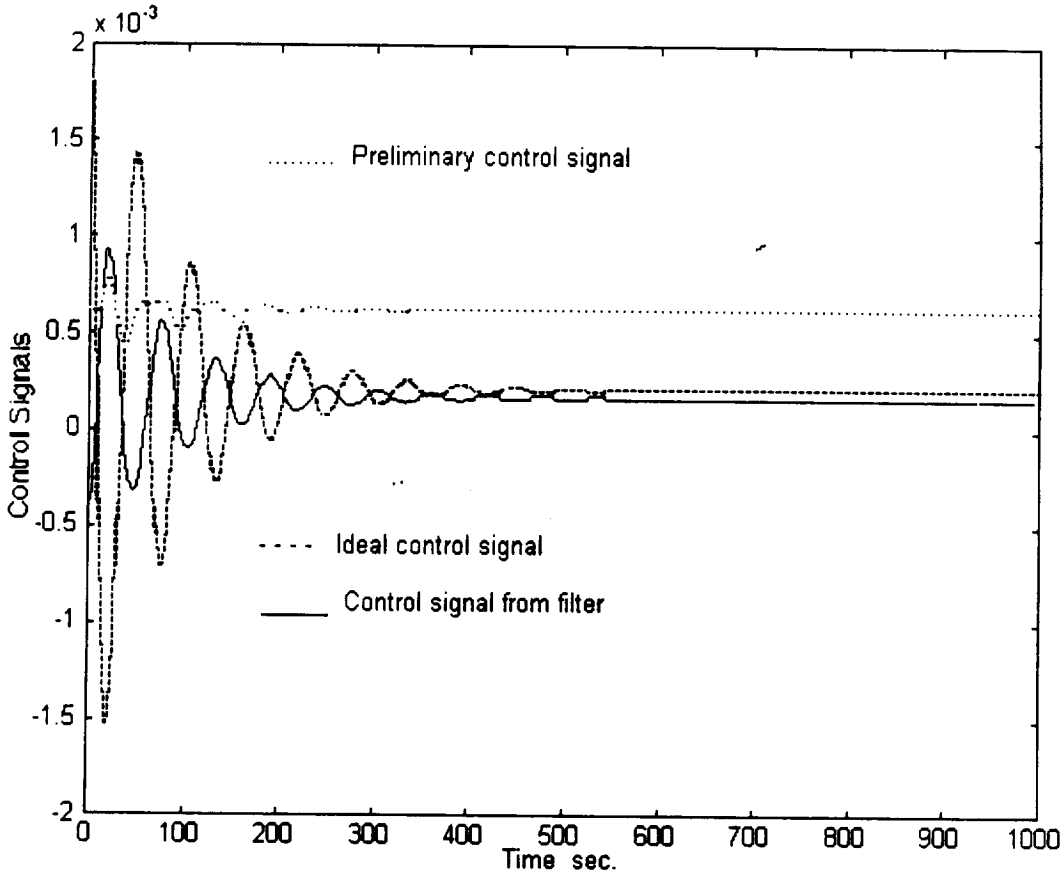

Figure 6.21: Control Signals—Case 5.

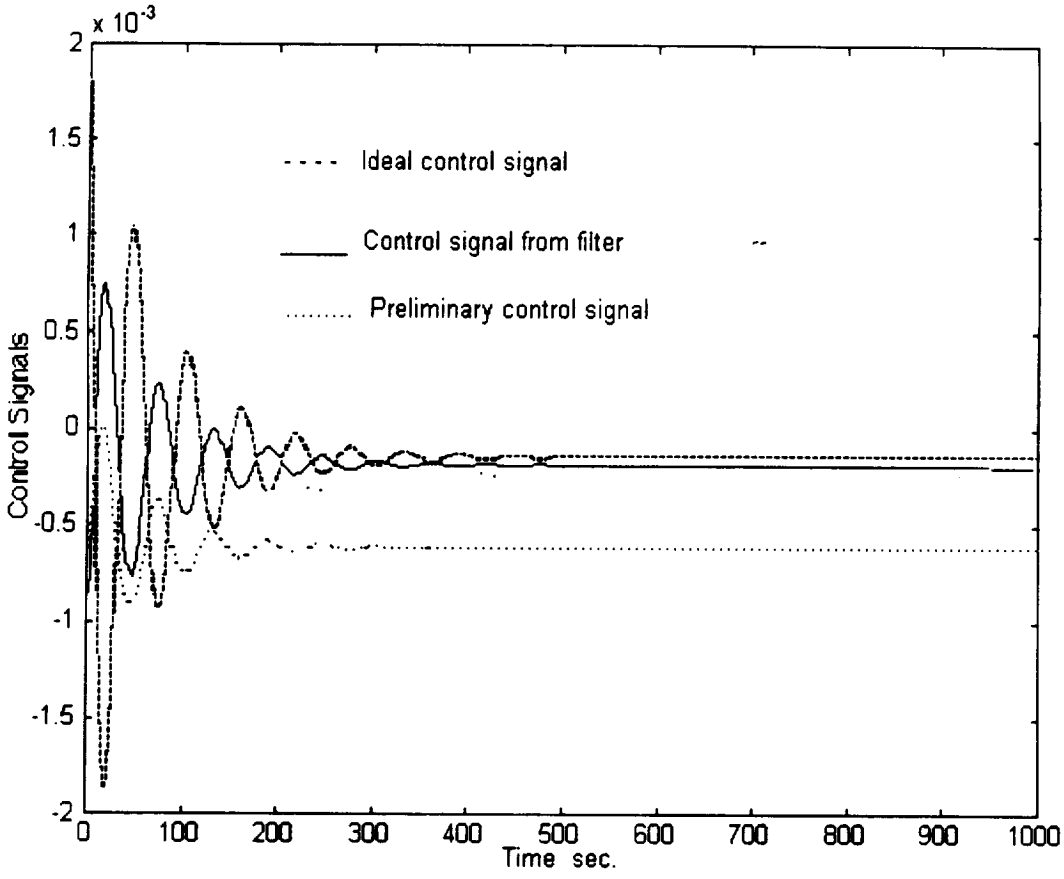

Figure 6.22: Control Signals—Case 6.

Bibliography

[1] L.C. Baird III, Reinforcement learning in continuous time: Advantage updating, in *Proceedings of the 1994 International Conference on Neural Networks*, 1994, pp. 2448-2453.

[2] Baldwin J.E. and B.W. Pilsworth, Dynamic programming for fuzzy systems with fuzzy environment, *Journal of Mathematical Analysis and Applications*, 1-23, 1982.

[3] Bellman R.E. and L.A. Zadeh, Decision-making in a fuzzy environment, *Management Science*, Vol. 17, 8141-8164, 1970.

[4] Bellman R., R.Kalaba and L.A.Zadeh, Abstraction and pattern classification, *J. Math. Anal. and Appl.*, Vol.2, 581-586, 1966.

[5] H.R. Berenji, Y. Jani, and R.N. Lea, Approximate Reasoning-Based Learning and Control for Proximity Operations and Docking in Space, *Proceedings of AIAA Guidance, Navigation, and Control Conference*, New Orleans, LA, August 12-14, 1991.

[6] Bezdek J. C., Cluster validity with fuzzy sets, *J. Cybernet.*, 58-73, 1974.

[7] Bezdek J.C., *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.

[8] A. Boschitsch O.O. Bendiksen, Nonlinear control laws for tethered satellites. *Advances in Astronautical Sciences*, **62**, 1986, 257-276.

[9] Chin-Hsing Cheng and Yuan-Hih Hsu, Damping of generator oscillations using an adaptive static VAR compensator, *IEEE Trans. on Power Systems*, vol. 7, no.2, May 1992.

[10] Shi-jie Cheng, Y. S. Chow, O. P. Malik and G. S. Hope, An adaptive synchronous machine stabilizer, IEEE Trans. on Power Systems, vol.-PWRS, no. 3, August 1986.

[11] Shi-Jie Cheng, O. P. Malik and G. S. Hope, Self-tuning stabilizer for a multi machine power system, *IEE Proceedings*, vol. 133, TP. C, no.4, May 1986.

[12] S.-J. Cheng, O.P. Malik, and G.S. Hope, Self-tuning stabilizer for a multi machine power system, in *IEE Proceedings*, Vol. 133-C:4, May 1986, pp. 176-185.

[13] Coray C., Clustering algorithms with prototype selection, in *Proc. Hawaii Intern. Conf. Syst. Sci.*, 945-955, 1981.

[14] E. Czogala and W. Pedrycz, On identification in Fuzzy Systems and Its Applications in Control Problems, *Fuzzy Sets and Systems*, **6**, 73-83.

[15] E. Czogala, Transformation of Knowledge Expressed by Means of Fuzzy Decision Tables into Neural Network Representation, *Proceedings of the IFSA Fourth World Congress*, July, 1991.

[16] Dave R.N., Fuzzy shell-clustering and application to circle detection in digital images, *Int. J. Gen. Syst.*, Vol.16, 343-355, 1990.

[17] Dave R.N., Validating fuzzy partitions through c-shells clustering, *Pattern Recognition Letters*, Vol.17, 613-623,1996.

[18] Dave R.N. and K. Bhaswan, Adaptive fuzzy c-shells clustering and detection of ellipses, *IEEE trans. Neural Networks*, Vol.3, No.5, 643-662, 1992.

[19] Dave R.N. and R. Krishnapuram, Robust clustering methods: a unified view, *IEEE Trans. Fuzzy Systems*, Vol.5, No.2, 270-293, 1997.

[20] Dunn J.C., A fuzzy relative of the ISODATA process and its use in detecting compact, well-separated clusters, *J. Cybernetics*, Vol.3, 32-57, 1974.

[21] S.E. Dreyfus and A.M. Law, *The Art and Theory of Dynamic Programming*, 1977, New York, Academic Press.

[22] Esogbue A.O., Optimal clustering of fuzzy data via fuzzy dynamic programming, *Fuzzy Sets and Systems*, Vol.18, 283-298, 1986.

[23] Esogbue A.O. and R.E. Bellman, Fuzzy dynamic programming and its extensions, *Fuzzy Sets and Decision Analysis, TIMS Studies in the Management Science*, Vol.20, 147-167, 1984.

[24] A. O. Esogbue and J. A. Murrell, A fuzzy adaptive controller using reinforcement learning neural networks, in *Proceedings of Second IEEE International Conference on Fuzzy Systems*, March 28–April 1, 1993.

[25] A.O. Esogbue, Q. Song, and W.E. Hearnes, Application of a self-learning fuzzy-neuro controller to the power system stabilization problem, in *Proceedings of the 1995 World Congress on Neural Networks*, Vol. II, Washington, DC, July 17-21, 1995, pp. 699-702.

[26] A. O. Esogbue and W. E. Hearnes II, Constructive experiments with a new fuzzy adaptive controller. *NAFIPS/IFIS/NASA '94. Proceedings of the First International Joint Conference of the North American Fuzzy Information Processing Society Biannual Conference. The Industrial Fuzzy Control and Intelligent Systems Conference, and the NASA Joint Technology Workshop on Neural Networks and Fuzzy Logic*, San Antonio, TX, December 18-21, 1994, 377–380.

[27] Esogbue, A.O. and J. Kacprzyk, Fuzzy Dynamic Programming: A Review of Its Main Developments, *Archives of Control Systems*, In Print.

[28] A. O. Esogbue and Q. Song, Optimal defuzzification and applications, in *Proceedings of Fourth Annual Conference on Fuzzy Theory and Technology*, Sept. 28–Oct.1, 1995, Wrightsville Beach, N.C.

[29] A.O. Esogbue and Q. Song, Defuzzification Filters: Design and Applications, Submitted to *IEEE Transactions on Fuzzy Sets*, 1996.

[30] Esogbue A.O. and B. Liu, Cluster validity for fuzzy criterion clustering, *Computers and Mathematics With Applications*, In Print

[31] A. A. Ghandakly and A. M. Farhoud, A parametrically optimized self-tuning regulator for power system stabilizers, *IEEE Trans. on Power Systems*, vol.7, no. 3, August 1992.

[32] M. A. M. Hassan, O. P. Malik and G. S. Hope, A fuzzy logic based stabilizer for a synchronous machine, *IEEE Trans. on Energy Conversion*, vol. 6, no.3, Sept. 1991

[33] T. Hiyama and T. Sameshima, Fuzzy logic control scheme for on-line stabilization of multi-machine power system, *Fuzzy Sets and Systems*, vol. 39, 1991.

[34] Yuan-Yih Hsu and Chung-Yu Hsu, Design of a proportional-integral power system stabilizer, *IEEE Trans. on Power Systems*, vol. PWRS-1, no. 2, May 1986.

[35] Yuan-Yih Hsu and Chin-Hsing Cheng, A fuzzy controller for generator excitation control, *IEEE Trans. on Systems, Man and Cybernetics*, vol. 23, no.2, March/April, 1993.

[36] E. Irving, J. P. Barret, C. Charcossey and J. P. Monville, Improving power network stability and unit stress with adaptive generator control, *Automatica* vol. 15, 1979.

[37] Ibragimov I.A., On the composition of unimodal distributions, *Theor. Probability Appl.*, Vol.1, 255-266, 1956.

[38] A. Isidori, Nonlinear Control Systems: An Introduction (2nd Ed.), Springer-Verlag, 1989.

[39] Jyh-Shing R. Jang, ANFIS: Adaptive-Network-Based fuzzy inference system, *IEEE Trans. on Systems, Man and Cybernetics*, vol. 23, no.3, May/June 1993.

[40] J.S.R. Jang, Self-Learning Fuzzy Controllers Based on Temporal Back Propagation, from a personal communication, 1992.

[41] C.L. Karr, L.M. Freeman, and D.L. Meredith, Improved Fuzzy Process Control of Spacecraft Autonomous Rendezvous Using a Genetic Algorithm, *Proceedings of the SPIE Intelligent Control and Adaptive Systems*, **1196**, 274-288, 1990.

[42] R.N. Lea and Y. Jani, Fuzzy Logic in Autonomous Orbital Operations, *International Journal of Approximate Reasoning*, **6:2**, 151-184, 1992.

[43] R.N. Lea, Fuzzy Sets and Autonomous Navigation, from a personal communication, 1988.

[44] R.N. Lea, Automated Orbital Rendezvous Considerations, *Proceedings of IEEE International Conference on Robotics and Automation*, Philadelphia, PA, Apr. 24-29, 1988.

[45] R.N. Lea, Automated Space Vehicle Control for Rendezvous Proximity Operations, *Telematics and Informatics*, **5:3**, 179-185, 1988.

[46] R.N. Lea, I. Chowdhury, Y. Jani, and H. Shehadeh, Design and Performance of the Fuzzy Tracking Controller in Software Simulation, *Proceedings of the IEEE International Conference on Fuzzy Systems*, San Diego, CA, Mar. 8-12, 1992.

[47] R.N. Lea, J. Hoblit, and Y. Jani, Performance Comparison of a Fuzzy Logic Based Attitude Controller with the Shuttle On-Orbit Digital Auto-Pilot, *Proceedings of the 1991 NAFIPS Workshop*, 291-295, 1991.

[48] R.N. Lea, J. Villareal, Y. Jani, and C. Copeland, Fuzzy Logic Based Tether Control, *Proceedings of the 1991 NAFIPS Workshop*, 398-402, 1991.

[49] R.N. Lea, J. Villareal, Y. Jani, and C. Copeland, Tether Operations USing Fuzzy Logic Based Length Control, *Proceedings of the IEEE International Conference on Fuzzy Systems*, San Diego, CA, Mar. 8-12, 1992.

[50] R.N. Lea, J. Villareal, Y. Jani, and C. Copeland, Space Time Neural Networks for Tether Operations in Space, *Proceedings of the 1992 NAFIPS International Conference on Fuzzy Set Theory and Applications*, Puerto Vallarta, Mexico, Dec. 15-17, 1992.

[51] R.N. Lea, J. Villareal, Y. Jani, and C. Copeland, Learning Characteristics of a Space Time Neural Network as a Tether Skiprope Observer, *Proceedings of the 1992 NAFIPS International Conference on Fuzzy Set Theory and Applications*, Puerto Vallarta, Mexico, Dec. 15-17, 1992.

[52] C. M. Lim and T. Hiyama, Comparison study between a fuzzy logic stabilizer and a self-tuning stabilizer, *Computers in Industry* 21, 1993.

[53] C. M. Lim and T. Hiyama, Self-tuning control scheme for stability enhancement of multi machine power systems, *IEE Proceedings*, vol.137, Pt. C, no. 4, July 1990.

[54] Liu B., *Studies in Inventory Processes and Reservoir Operations*, Ph.D. Dissertation, Institute of Systems Science, Chinese Academy of Sciences, 1992.

[55] Liu, B. and A.O. Esogbue, Fuzzy Criterion Set and Fuzzy Criterion Dynamic Programming, *Journal of Mathematical Analysis and Applications*, Vol. 199, 293-311, 1996.

[56] V.J. Modi, P.K. Lakshmanan, and A.K. Misra, On the Control of Tethered Satellite Systems, *Acta Astronautica*, **26:6**, 411-423, 1992.

[57] J.A. Murrell, A Statistical Fuzzy Associative Learning Approach To Intelligent Control, Ph.D. Thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, December, 1993.

[58] S. Mabuchi, A proposal for a defuzzification strategy by the concepts of sensitivity analysis, *Fuzzy Sets and Systems*, vol.55, 1993 pp1–4.

[59] D. C. Montgomery, Introduction to Statistical Quality Control (2nd ed.), John Wiley & Sons, Inc.,1991.

[60] K. Narendra and M.A.L. Thathachar, Learning Automata: An Introduction, Englewood Cliffs, NJ: Prentice-Hall, 1989.

[61] Pal S. K. and P. P. Wang, *Genetic Algorithms for Pattern Recognition*, Boca Raton, FL: CRC, 1996

[62] Ruspini E.H., A new approach to clustering, *Inform. and Control*, Vol.15, 22-32, 1969.

[63] Ruspini E.H., Numerical methods for fuzzy clustering, *Information Sciences*, Vol.2, 319-350, 1970.

[64] J. Shi, L. H. Herron and A. Kalam, A fuzzy logic controller applied to power system stabilizer for a synchronous machine power system, in *Proceedings of IEEE Region 10 Conference*, Tencon 92, Melbourne, Australia, Nov. 11–13, 1992.

[65] J. E. Slotine and W. Li, Applied Nonlinear Control, Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[66] Q. Song and R. P. Leland, Adaptive learning defuzzification techniques and applications, to appear on *Fuzzy Sets and Systems*, 1996.

[67] R.S. Sutton, Learning To Predict By The Method Of Temporal Differences, *Machine Learning*, Vol. 3, 1988, pp. 9-44.

[68] Tamura S., S.Higuchi and K.Tanaka, Pattern classification based on fuzzy relations, *IEEE Trans. Syst. Man Cybern*, Vol.1, No.1, 61-66, 1971.

[69] J.A. Villareal and R.O. Shelton, A Space-Time Neural Network (STNN), *Proceedings of the 2nd International Joint Conference on Neural Networks and Fuzzy Logic*, Houston, TX, Apr. 11-13, 1990.

[70] C.J.C.H Watkins, Learning from delayed rewards, Ph.D. Thesis, Cambridge University, Cambridge, England, 1989.

[71] C.J.C.H Watkins and P. Dayan, Q-Learning, *Machine Learning*, Vol. 8, 1992, pp. 279-292.

[72] P.J. Werbos, Neurocontrol and Related Techniques, in *Handbook of Neural Computing Applications*, A.J. Maureen, C.T. Harsten and R.M. Pap, eds., Academic Press, Inc.

[73] Yang M.-S., A survey of fuzzy clustering, *Mathl. Comput. Modeling*, Vol.18, No.11, 1-16, 1993.

[74] Yager R.R. and D.P. Filev, Approximate clustering via the mountain method, *IEEE Trans. Syst. Man, Cybern.*, Vol.24, 1279-1284, 1994.

[75] R. R. Yager and D. Filev, SLIDE: a simple adaptive defuzzification method, *IEEE Trans. on Fuzzy Systems* Vol. 1, No.1, 1993 pp 255–271.

[76] L. A. Zadeh, Fuzzy sets, *Information and Control*, vol. 8, 1965.