Evolutionary and Reinforcement Fuzzy Control

By

Mina Munir-ul Mahmood Chowdhury BEng, MSc

A thesis submitted in partial fulfilment of the requirement for the degree of

Doctor of Philosophy

April 1999

Department of Electronics and Electrical Engineering
University of Glasgow
Glasgow G12 8LT
United Kingdom

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

For

my parents, Ammu and Abbu

and

in loving memory of my grandparents

(may they rest in peace)

Abstract

Many modern and classical techniques exist for the design of control systems. However, many real world applications are inherently complex and the application of traditional design and control techniques is limited. In addition, no single design method exists which can be applied to all types of system. Due to this 'deficiency', recent years have seen an exponential increase in the use of methods loosely termed 'computational intelligent techniques' or 'soft-computing techniques'. Such techniques tend to solve problems using a population of individual elements or potential solutions or the flexibility of a network as opposed to using a rigid, single point of computing. Through use of computational redundancies, soft-computing allows unmatched tractability in practical problem solving. The intelligent paradigm most successfully applied to control engineering, is that of fuzzy logic in the form of fuzzy control. The motivation of using fuzzy control is twofold. First, it allows one to incorporate heuristics into the control strategy, such as the model operator actions. Second, it allows nonlinearities to be defined in an intuitive way using rules and interpolations.

Although it is an attractive tool, there still exist many problems to be solved in fuzzy control. To date most applications have been limited to relatively simple problems of low dimensionality. This is primarily due to the fact that the design process is very much a trial and error one and is heavily dependent on the quality of expert knowledge provided by the operator. In addition, fuzzy control design is virtually *ad hoc*, lacking a systematic design procedure. Other problems include those associated with the *curse of dimensionality* and the inability to learn and improve from experience. While much work has been carried out to alleviate most of these difficulties, there exists a lack of drive and exploration in the last of these points.

The objective of this thesis is to develop an automated, systematic procedure for optimally learning fuzzy logic controllers (FLCs), which provides for autonomous and simple implementations. In pursuit of this goal, a hybrid method is to combine the advantages artificial neural networks (ANNs), evolutionary algorithms (EA) and reinforcement learning (RL). This overcomes the deficiencies of conventional EAs that may omit representation of the region within a variable's operating range and that do not in practice achieve fine learning. This method also allows backpropagation when necessary or feasible. It is termed an *Evolutionary NeuroFuzzy Learning Intelligent Control technique* (ENFLICT) model. Unlike other hybrids, ENFLICT permits globally structural learning and local offline or online learning. The global EA and local neural learning processes should not be separated. Here, the EA learns and optimises the ENFLICT structure while ENFLICT learns the network parameters. The EA used here is an improved version of a technique known as the messy genetic algorithm (mGA),

which utilises flexible cellular chromosomes for structural optimisation. The properties of the mGA as compared with other flexible length EAs, are that it enables the addressing of issues such as the curse of dimensionality and redundant genetic information. Enhancements to the algorithm are in the coding and decoding of the genetic information to represent a growing and shrinking network; the defining of the network properties such as neuron activation type and network connectivity; and that all of this information is represented in a single gene.

Another step forward taken in this thesis on neurofuzzy learning is that of learning online. Online in this case refers to learning unsupervised and adapting to real time system parameter changes. It is much more attractive because the alternative (supervised offline learning) demands quality learning data which is often expensive to obtain, and unrepresentative of and inaccurate about the real environment. First, the learning algorithm is developed for the case of a given model of the system where the system dynamics are available or can be obtained through, for example, system identification. This naturally leads to the development of a method for learning by directly interacting with the environment. The motivation for this is that usually real world applications tend to be large and complex, and obtaining a mathematical model of the plant is not always possible. For this purpose the reinforcement learning paradigm is utilised, which is the primary learning method of biological systems, systems that can adapt to their environment and experiences. In this thesis, the reinforcement learning algorithm is based on the advantage learning method and has been extended to deal with continuous time systems and online implementations, and which does not use a lookup table. This means that large databases containing the system behaviour need not be constructed, and the procedure can work online where the information available is that of the immediate situation.

For complex systems of higher order dimensions, and where identifying the system model is difficult, a hierarchical method has been developed and is based on a hybrid of all the other methods developed. In particular, the procedure makes use of a method developed to work directly with plant step response, thus avoiding the need for mathematical model fitting which may be time-consuming and inaccurate.

All techniques developed and contributions in the thesis are illustrated by several case studies, and are validated through simulations.

Acknowledgements

I never thought writing acknowledgements would be just as difficult, if not more so, as writing the thesis itself. There are so many people to remember, thank, acknowledge and credit that I hope anyone reading this who has been influential in some way will forgive me if I have omitted them.

Two people I have to most definitely thank and express my love and appreciation are my parents for first bringing me to this world and for sticking with me through thick and thin. This work could not have been completed without their love, understanding, patience, encouragement and guidance. There were times when I would have given up on myself, during my various mood and personality changes, but they didn't

Most people leave their gratitude and thanks to their supervisors to last. I have always been taught that the teacher's position is next to the parent, and hence I feel it only apt that my supervisor, Yun Li, should be mentioned after my parents. Dr Li has been more than my supervisor. I have now known him for 7 years, almost from the time he joined the Department, and in that time he has been my teacher, my supervisor, a mentor and I hope a friend. I have been very impressed with his understanding of the academic subjects I have had the privilege of studying under him. However, my lasting impression of him will be his caring, understanding and patience and dynamism. There were times when only he and I know that his duties as a supervisor were exceeded, and I can not thank him enough for those times and at the same time apologise for the pains I have put him through. Nor can I express enough gratitude for his being a constant source of inspiration and guidance through all those years.

All my brothers and sisters (Moti, Mohi, Atia and Aliah) need mentioning for their constant encouragement, help and support, but more specifically my elder sister, Atia. Everyone seems to bug her for proof reading their work and various secretarial works. God only knows the number of times I've been a pain to her. If I've not shown it or said it, thanks for all those times and good luck to her when she, God willing, starts her own research work this time next year.

Thanks also to Prof Murray-Smith for the advice and guidance he has given me in Dr Li's absence and matters outwith my research. A special thanks to Tom O'Hara, often the unsung soldier in the Centre for Systems and Control, for being there when I needed help with any equipment problems and during those lab demo sessions.

Finally, there have been many friends and colleagues who have given me a lot of encouragement, help and support, specially all the folks of the "box" and the "dungeon": Euan, KC, Albert, Anna, Graham, Gary, Henrik, Lipton, Wenyuan, Markus, Torsten, Iain, Seelan, Muktihar and Jay. Outside the "box", people who deserve a mention for their constant support and encouragement are my good friends, Max, Kenny, Asif and Mo! Thanks guys!

List of Figures

Chapter 4:

Chapter 5:

Appendices:

List of Tables

Table of Contents

Chapter 1

Introduction

You see things and you say Why?
But I dream things that never were; and I say Why not?
- George Bernard Shaw

1.1 Motivation

Many industrial systems are inherently non-linear and time-varying. To deal with this, controllers are often designed by first linearising the system model about a given operating condition. Clearly, this can imply severe consequences when operation moves to a new region. Therefore, robustness, adaptiveness and autonomy in the algorithm and design are very important, but may not be addressed adequately by conventional control schemes. Although many control techniques such as PID, Bode-Nyquist, adaptive, H_∞, sliding mode and inverse model based schemes exist, it cannot be argued that they are equally suitable or applicable in practice.

Many practical control systems in operation still need a human operator. An example would be a vehicle cruising along a defined path, where a model and a controller would somehow have to accommodate changes and handle noise or disturbances within the system and the environment, such as an accident further down its path. In addition, the mathematical models for such applications can be ill-defined, very complex or too difficult to obtain. A conventional controller may rely too heavily on a mathematically rigid model of the system and the environment and hence may not cope with the situation, while a human controller (driver) can deal with it first by learning and then by reinforcing.

Mathematically rigid limitations have led many designers to more *'intelligent'* control schemes, which exhibit properties such as 'knowledge representation', inferencing, 'learning', and 'evolution'. Research into such schemes also arises from the insufficient flexibility and autonomy of traditional control techniques. The desire on the operator's part is not surprising considering the fact that humans are able to generalise, infer reason, and evaluate complex functions simply from knowledge and events encountered in everyday activities.

In control system design, the plant input/output measurements need to be mapped onto the controller parametric space under an optimal output requirement criterion. When this mapping is described using various pieces of uncertain knowledge, conventional control methods are faced with various limitations and difficulties such as dealing with ill-defined and

time-varying environments, potentially unknown systems, uncertainty with systems and adaptation necessary to compensate for changing operating conditions. As has been mentioned, the design of controllers typically involves reasoning, describing the system and control instructions, adapting and learning the controller to various and changing operating situations, and optimising and evolving the controller to operate optimally to local and global levels.

It is therefore desirable and appropriate to utilise methods exhibiting the above properties, such as soft computing (SC) techniques. Soft computing techniques differ from conventional computing in that, unlike conventional techniques, it is tolerant of imprecision, uncertainty and partial truth, emulating the human mind. Current research into SC or intelligent control can be divided into three strategies, namely:

- *reasoning* (encompassing knowledge based systems, classifiers and fuzzy logic)
- *learning theory*, and
- *evolution.*

Fuzzy logic control is an extension of Zadeh's fuzzy set and fuzzy logic principles (Zadeh 1965), and was pioneered by Mamdani (Mamdani 1974). Just as a human controller would define a control action in the form of a set of linguistic rules, fuzzy logic controllers (FLCs) are also defined by a set of linguistic rules in the form of a set of '***IF*** ... ***THEN***' statements.

A general FLC consists of four blocks as shown in Figure 1.1. First, measurements are taken of all variables that represent relevant conditions of the controlled process. Next, these measurements are converted (*fuzzified*) into appropriate fuzzy sets to express measurement uncertainties. The inference engine then uses the fuzzified measurements to evaluate the control rules stored in the fuzzy rule-base. The result of this evaluation is a fuzzy set (or several fuzzy sets) defined on the universe of possible actions. Finally, the fuzzy set is converted (*defuzzified*) into a single (*crisp*) value, which is the best representation of the fuzzy set. The defuzzified values represent actions taken by the FLC in individual control cycles.

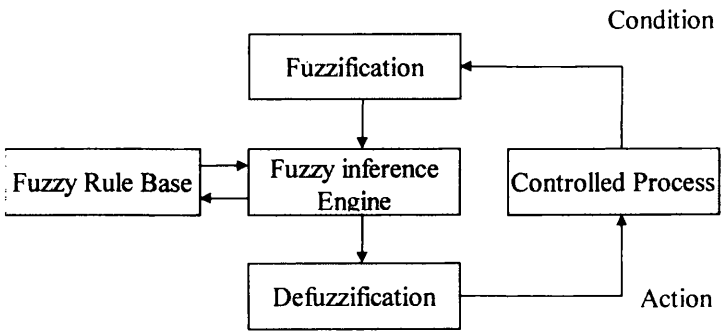


Figure 1.1 Schematic of FLC

The advantage that a controller based on fuzzy logic has over conventional controllers, is that it is easier to understand and implement because it emulates human reasoning. Control actions can be described using linguistic descriptions that even a non-control individual can understand and interpret. In addition, the generality of FLCs makes them very suitable for non-linear control. Fuzzy control is also able to operate without a clear mathematical definition of system. It can be seen as a loosely defined form of table based control method. Its development can be viewed as a type of knowledge based expert system. It essentially consists of a knowledge base expressed in terms of relevant fuzzy inference rules, and an appropriate inference engine to solve a given control problem. In contrast to conventional controllers, FLCs are capable of utilising knowledge extracted from human operators. The knowledge of an experienced human operator may be used as an alternative to a precise model of the controlled process.

Fuzzy control has been successfully applied to a wide range of industrial problems such as heating systems (Altrock *et al* 1994); steam engines (Mamdani 1974, Kiupel and Frank 1993); cement kiln control (Larsen 1980, Umbers and King 1980, Holmblad and Ostergaard 1982); water purification plants (Tang and Mulholland 1987); oil refineries (Graham and Newell 1988, Aliev *et al* 1992); traffic control (Gegov 1994, Jia and Zhang 1994, Ngo and Li 1994, Pappis and Mamdani 1977, Sasaki and Akiyama 1988); air conditioning systems (Tobi and Hanafusa 1991); warm water plants (Kickert and Van Nauta Lemke 1976); refuse incineration plants (Krause *et al* 1994); robot control (Nedungadi 1993, Uragami *et al* 1976), control of space structures (Ross *et al* 1993), hydropower plants (Djukanovic *et al* 1997) and nuclear power systems (Uhrig and Tsoukalas 1998)

While research and development on the three main areas of intelligent control have broadly progressed independently of each other, there is in fact much similarity and interconnection between them as illustrated in Figure 1.2. The overlap between fuzzy logic and each of the other SC methods is significant as well as "logical".

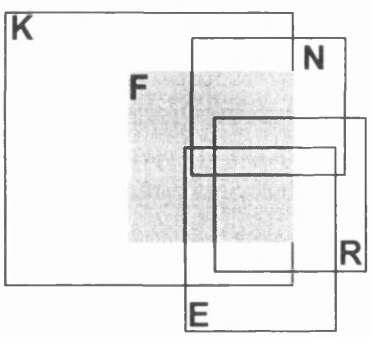
Artificial neural networks (ANNs) give rise to a particular class of parameterised controllers and models. They are essentially an interconnection of non-linear units, with local memory elements such as integrators or delay lines when dynamic behaviour is of interest. The *weights* characterising the connections play a role similar to the concentrations of neurotransmitters in biological synapses, while the non-linear elements correspond to the neurons themselves. The weights are then adjusted in learning to model a plant or act as a controller. From a theoretical perspective, the connection between fuzzy systems and neural networks is that they are both universal approximators of continuous functions. Since the early 1990s, it has also been identified that fuzzy systems can be mapped to a particular type of neural network. The input nodes and layers of the neural network would represent the fuzzification process of the fuzzy system; the output layer the defuzzification and the hidden and internal nodes and layers the inferencing mechanism of the fuzzy system. Indeed, under certain

circumstances, there is a functional equivalent between fuzzy systems and neural networks (Jang 1993).

Genetic algorithms (GAs) are loosely modelled on processes that appear to be at work in biological evolution and the immune system (Holland 1975, Goldberg 1989a). The connection between FLCs and GAs is not as incongruent as it appears. GAs have proven to be a very useful tool in dealing with various optimisation problems involving FLCs. GAs are typically utilised for optimising the fuzzy rules and linguistic variables in FLCs, resulting in near optimal controller operation.

In nature, humans follow a *nature cycle* of evolution-reasoning-learning and reinforcing what has been learned. It is therefore not a surprise that *Reinforcement learning* (RL), an approach to machine intelligence combining unsupervised learning and dynamic programming to solve problems that neither of these disciplines are able to address alone (Barto *et al* 1983), also finds itself interconnected with the intelligent methods already discussed. In fact, RLs exhibit similar properties to ANNs and GAs, and thus the learning and evolving of FLCs is maintained through the entire nature cycle.

Despite much exploration and exploitation of FLCs with many of these other intelligent techniques, research has mainly been limited to passive combination of the methods. Thus, the potential of the other methods in the nature cycle are not fully utilised or realised. For example, when integrating FLCs to neural networks, much of the undesirable properties of ANNs are also brought in. During learning, the neurofuzzy controller (NFC) is prone to getting trapped in regions of local optima. On the other hand, while fuzzy-GA combinations find near optimal controllers, such controllers are typically designed and applicable around limited operating conditions. While the underlying subject of this thesis is fuzzy logic control, the aim of the thesis is to systematically build up a paradigm based on fuzzy control, exhibiting evolutionary and learning properties, and the net objective is optimal fuzzy control.



E = Evolutionary Algorithms
F = Fuzzy Logic
K = Knowledge Based Systems
N = Neural Networks
R = Reinforcement Learning

Figure 1.2 Overlap between SC control algorithms

1.2 Methodology and Literature Review

Fuzzy control is a very effective, flexible, robust and intuitive tool for dealing with complex and non-linear systems, and despite its apparent success, it has many problems associated with it. Its major handicap is that it is not always implemented in the best way. It is very much ad hoc, lacking any systematic design procedure, and is very dependent on input and interaction from a human, and hence the quality of the controller may vary. For instance, in some complex applications such as robot control and ship auto pilot, fuzzy control is applied at the lowest level. Such controllers would have to be very fast and precise where bandwidths are high and nonlinearities are strong. Of course, it is possible to obtain fast and accurate FLCs on VLSI chips. The problem lies in the fact that in such fast systems, human experience alone is not sufficient. One person's reasoning of a certain problem may not ally with another's and controller parameter tuning becomes difficult.

Unlike conventional control, the design and implementation of fuzzy control have, on the whole, been ad hoc. There has not been any real drive or effort towards formalising or generalising fuzzy control theory. Simply looking at the basic structure of the FLC, one can see the difficulty fuzzy researchers face when attempting to formalise fuzzy control. As a result, there have been numerous fuzzification, inferencing and defuzzification methods. This further hinders practising engineers with confusing design choices. To address and alleviate some of these issues and problems, there grew a need for computer-aided design and tuning techniques highlighted in Figure 1.2, and such procedures will now be reviewed.

1.2.1 Neurofuzzy Learning: A First Step Towards Automated Tuning

The complexity of manually tuning a FLC has prevented it from better and wider applications. Further, if system parameters change or if the environment in which the system functions changes, the FLC needs to be tuned again for the new settings. There have been various attempts at automating and optimising the design of FLCs by utilising other "intelligent paradigms" shown in Figure 1.2, such as neural networks, genetic algorithms and machine learning. The primary purpose for most of these hybrid systems is to tune the parameters of the FLC.

The combination of ANNs with FLCs are generally termed *neurofuzzy* controllers and such combinations present the advantages of both while avoiding many of the drawbacks of both. While fuzzy control uses reasoning, it can not learn from path experience without some level of supervision. On the other hand, ANNs are able to function supervised or unsupervised and can learn from past experience or data. However, in order for this learning procedure to be effective, quality data representing different states or conditions have to be provided. In

addition, tuning and identifying the different nodes and elements of an ANN require some degree of expertise and knowledge or else the learning procedure may be handicapped. By mapping a FLC to an ANN, such difficulties can be overcome as the components of the fuzzy controller are intuitive and simple to implement (Berenji 1990, Brown and Harris 1991, Lin and Lee 1991, Kosko 1991, Horikawa *et al* 1992, Nomura *et al* 1992, Bruske *et al* 1993, Jang 1993, Kim 1993, Khalid *et al* 1994, Chen and Chen 1994, Linkens and Nie 1994, Nauck and Kruse 1994, Fukuda and Shibata 1994, Ichihashi *et al* 1995, Lee *et al* 1995).

An example of such a neurofuzzy hybrid is that proposed by Khalid *et al* (Khalid *et al* 1994) called NeuFuz. The neurofuzzy scheme is similar to a self-organising FLC set-up (Procyk and Mamdani 1979, Scharf and Mandic 1985) and consists of two multi-layered neural network models. The first neural network is a plant emulator and the second is used as a compensator to improve the performance of the basic fuzzy logic controller. The development of this system consists of three phases. The first phase is developing a basic FLC for the plant. The second phase involves training a neural network model in the forward dynamics of the plant to be controlled. The training of this neural plant emulator can be done off-line as well as on-line, depending on the type of plant. For fast-acting plants, such as robotics manipulators or servo-motors, it is possible to train the neural network to emulate the plant in an on-line way. However, if the plant is a slowly varying process, the neural plant emulator needs to be trained off-line as convergence is rather slow. The function of the neural network plant emulator is to provide the correct error signal at the output of the neurofuzzy compensator, without the need for any mathematical modelling of the plant. The third phase involves on-line learning of the neurofuzzy compensator. The performance error, which is the error between the desired output and the actual plant output, is backpropagated through the neural plant emulator to adapt the weights of the neurofuzzy compensator on-line. The performance of the neural plant emulator can be further improved on-line by backpropagation of the error between the neural plant emulator and the actual plant output. Variations and use of this method are found in Rao and Gupta (1994) and Spooner and Passino (1996).

Despite its potential it has not been too warmly embraced by the fuzzy logic community, the main objections being the amount of training required, the quality of the performance tables, and questions about the stability of the resulting controller.

Such early neurofuzzy methods used neural networks and fuzzy systems independently, but functioning together. After a FLC is designed, a neural network is used to track changes in the system. Therefore, the fuzzy system parameters are not tuned, and if any tuning does take place it is partial. In other words, either the fuzzy rules or the fuzzy sets are tuned, not the structure as a whole.

More recently, neurofuzzy systems are treated as single structure, i.e., they are either a neural network with fuzzy properties, or fuzzy systems in a neural network structure. While there are numerous fuzzification, inferencing and defuzzification strategies, the actual fuzzy

system topology is fixed. Hence, any representation of fuzzy systems as a neural network means that the number of layers of the neurofuzzy structure is limited. In fact, as a result, there has been very little change in the way fuzzy systems are represented as a neural network structure, and thus most new neurofuzzy structures are derivatives of early popular and established ones.

One such method is that proposed by Lin and Lee (Lin and Lee 1991). The fuzzy logic components are directly integrated in the neural network, and have a multi-layer feed-forward topology. The input and output nodes represent the input states and control signals, respectively, and in the hidden layers there are nodes that code membership functions and rules. The learning algorithm used for building rule nodes and training the membership functions is based on the backpropagation algorithm. The limitations of this system are that it only tunes the fuzzy sets, no rule reduction of formation takes place to reflect changes in the system behaviour, and only gaussian membership functions are used. The shape and type of the fuzzy sets are important as they can influence the smoothness of the control surface. The most used shapes are trapezoids and triangles because they are simple to implement and are computationally efficient. However, because of their piece-wise nature, they are not well suited for a smooth transition between fuzzy sets and instead the smoother membership functions, gaussian and generalised bell, are used. However, the advantage this system has over the ANFIS structure (see below) is that this system can represent the output variables as fuzzy sets. Most neurofuzzy controllers have structures similar to this model, with slight variations, and the Lin and Li method is in fact the work that best resembles the neurofuzzy structure developed here.

Possibly the most well known of neurofuzzy models is Jang's ANFIS (adaptive neurofuzzy inference system) (Jang 1993). It is a variation of the Lin and Lee model, but the learning algorithm is described only for the Sugeno fuzzy model, and employs Kalman filters. Again, the network only adjusts the parameters of the fuzzy sets and does not allow for rule modification. This means that in addition to the operator knowing the control surface, the network size can potentially increase in size exponentially depending on the size of the input space. Another drawback of both systems is that they are supervised, and hence quality, accurate and reliable training data has to be provided.

A third type of neurofuzzy structure can be found in the works of Harris *et al* (Harris *et al* 1993). It uses B-splines to implement fuzzy sets, and the network resembles a CMAC or RBF network. While using B-spline functions well for storing information locally, the structure does not store the membership functions directly as in the models described above. Instead, fuzzy rules and sets are learned through data clusters. Another difference between this and other models is that the fuzzy sets may be subnormal. That is, at least one element of the fuzzy set may not have value unity within the universe of discourse of the variable, and hence the set may have special constraints placed on it which it would not have if a standard fuzzy set were used.

1.2.2 Learning Through Interaction

One of the limitations of using ANNs for learning fuzzy systems is that ANNs require gradient information to guide the learning. Also, in order to define the training data used by ANNs, the condition-action of the problem has to be known or simulated. Often this is difficult or impossible to obtain. For example, a robot may be trained to navigate around a room while avoiding obstacles in its path. To train it to avoid any obstacle anywhere in its path would require a large amount of data which would be expensive, tedious and possibly difficult to obtain. In "nature", such systems would not have data to train it, instead it would learn to navigate by interacting with the environment directly. Learning through interaction is also known as *reinforcement* learning, and is the primary learning method of biological systems.

Recently, efforts to apply the RL methods to fuzzy systems have been reported (Berenji and Khedkar 1992, Whitely 1993, Lin and Lee 1994, Glorennec 1994, Buijtenen *et al* 1998, Lin and Kan 1998). The majority of these is based on Q-learning and is applied to classifier systems where patterns are matched using fuzzy linguistic type if-then rules. Usually, in an FLC, some rules trigger on the same crisply defined state, and together co-operate to produce an action. There is a one-to-one mapping between an agent (i.e. a set of rules) and the action it produces. Therefore, the performance of each agent is evaluated independently of each other. The difficulty with this method is that the rule structure contains all possible combinations of rules making it computationally inefficient.

The system closest to the work presented in this thesis is Nauck and Kruse's NEFCON (NEural Fuzzy CONtrollers) model, (Nauck and Kruse 1994). Nauck and Kruse proposed a generic three-layer neurofuzzy model with a single output. The network is trained using reinforcement learning, which uses a rule based fuzzy error measure as the reinforcement signal. In NEFCON, both the fuzzy rule base and the fuzzy sets are achieved. The drawback of the NEFCON approach is that it starts from an empty rule base and builds up the rule base. The rule antecedent is formed by finding membership functions for each variable that yields the highest membership value for the appropriate input variable. The rule consequents are formed by guessing the output value from the fuzzy error. This form of rule creation implies that the operator has sufficient information about the desired output data, hence the model is only suitable for supervised off-line learning. In contrast, the reinforcement learning model presented in this thesis is applicable to both off-line and online learning, and assumes no information on the desired output.

Another method that uses reinforcement learning was proposed by Whitely (Whitely 1993). In this method the system receives a signal of success or failure from the real world, and learns from the strength of this signal to improve its success rate. However, since the quality of such a feedback signal is generally poor, learning is inefficient. Another drawback is a noisy

value function and biased signals, which means that not all the possible state occurrences are learned.

1.2.3 Fuzzy Genetic Combination: Towards Global Tuning

Evolutionary algorithms (EA), of which genetic algorithms is a specific type, are techniques based on the Darwinian principles of evolution through survival of the fittest. Central to the evolutionary system is the idea of a population of *genotypes* or *phenotypes* that are elements of high dimensional search space. Through "natural selection" and genetic operators, *genotypes* or *phenotypes* with better fitness are learned. Thus by survival of the fittest GA over several generations, the population gradually evolves towards genotypes that correspond to high fitness phenotypes.

EAs work in a similar manner to RLs, both using an evaluation function to guide the learning and optimisation process, and neither requiring the gradient information that neural networks use. However, EAs and RLs differ in a number of ways: first, EAs search the solution space in a completely random manner and hence ignore a lot of the information between state transitions. Second, an EA discards poor solutions in favour of good ones whereas RLs take this information on board and attempts to improve on it. Finally, unlike EAs, the evaluation function defined for RLs does not indicate a performance measure, but whether the learning system is performing well or badly.

The primary purpose for most of these hybrid systems is to tune the parameters of the fuzzy sets defining the linguistic variables, while some systems also deal with rule reduction (Thrift 1991, Homaifar and McCormick 1992, Linkens and Okola 1992, Chen *et al* 1993, Lee and Takagi 1993, Surman *et al* 1993, Buckley and Hayashi 1994, Cooper and Vidal 1994, Renders and Bersini 1994, Bastian 1995, Cordon and Herrera 1995, Fukuda *et al* 1995a, Hishiyama *et al* 1995, Cotta *et al* 1996, Filipic and Juricic 1996, Gonzalez and Perez 1996, Huang and Hung 1996, Magdalena and Velasco 1996, Popovic and Xion 1996, Tarng *et al* 1996). A first attempt at optimising fuzzy control with genetic algorithms was mainly concerned with tuning the positions of the fuzzy membership functions (Karr *et al* 1989, Homaifar and McCormick 1991, Wang and Kwok 1992). The approach uses a fixed predefined number of fuzzy sets to define the input and output domains. Genetic algorithm is used only to adjust the shape of the fuzzy sets in the given rule base. The chromosome is made up of binary numbers representing the supports of the membership function. Nonetheless, this is very basic and no optimisation of the rule base is carried out, and an exclusive membership function shape is used. This requires a great deal of knowledge on behalf of the expert operator such as knowing what control actions to take for a given situation.

Herrera *et al* (Herera *et al* 1995a) encodes the entire knowledge base. The fuzzy sets are of trapezoidal form represented by a 4-tuple parameter set indicating the apex and the base points of the set. Each rule is thus represented in the following way:

$$C_{rl} = \left(a_{l1}, b_{l1}, c_{l1}, d_{l1}, \cdots, a_{ln}, b_{ln}, c_{ln}, d_{ln}, a_l', b_l', c_l', d_l'\right) \tag{1.1}$$

where $(a_{ln}, b_{ln}, c_{ln}, d_{ln})$ is the 4-tuple representing the fuzzy set in the said domain. The complete rule base representing the whole chromosome is thus a concatenation of all such C_{rl}. Real values are used to encode the genes, and the whole GA operates using *simple* and *min-max arithmetical* crossovers and non-uniform mutation. This approach is slightly different from the other approaches mentioned above in that the fitness function is a square-medium error function using an input-output training data set similar to that used in neural network training. This implies that the operator must provide quality and accurate data for the GA to converge smoothly to the desired goals. The other drawbacks of this approach are that it uses only trapezoidal fuzzy sets and that it is limited to Mamdani-type FLCs.

Another approach to optimising the entire rule base structure was proposed by Kinzel *et al* (Kinzel *et al* 1994). This method uses a $n_1 \times \ldots \times n_d$ matrix instead of a string to code the rule base. Here n_I is the number of fuzzy sets in domain *i*. Each element of the matrix consists of a fuzzy set of the output domain. The fuzzy sets are coded by a string of genes where each gene represents the membership values of the fuzzy sets of domain *d* at a certain x-value. The initial population is generated by applying the mutation operator on all genes, and the initial fuzzy partitions are homogeneous. Crossover on the rule base is carried out using a point radius operator. A two-point crossover is used on the fuzzy sets, which exchanges ranges of the partitions represented by the two chromosomes. A side effect of this type of crossover is that a repair algorithm has to be used to repair any resulting non-convex fuzzy sets to convex ones. In addition, there may also be situations where some fuzzy sets may not be present for the controller to operate in some ranges.

This naturally led to the search for systematic designs and ways to optimise the entire fuzzy control structure (Lee and Takagi 1993, Kinzel *et al* 1994, Ng and Li 1994, Herera *et al* 1995a). Lee and Takagi optimises the rule base, the number of rules and the shape of the fuzzy sets. The rule base is encoded such that three genes represent each fuzzy set in each domain, where each gene represents the distance of the support of the working fuzzy set from the support of the previous fuzzy set. The drawback of this system is that in order to keep the length of the encoded chromosome short, it is only applicable for Sugeno-type FLCs, i.e. the output domain is not fuzzified. Another drawback is that it only handles rule reduction and there is no provision for rule adding.

Within the research group at Glasgow, Ng and Li (Ng and Li 1995) developed an approach that uses base-7 coding if 7-level fuzzification and a 7×7 rule-base is predefined. The positioning and shape of the linguistic variables are determined by the set parameters (α, β, σ) representing the position, shape and scaling respectively. The remaining part of the chromosome encodes the membership function parameters. That said, the Ng and Li approach does present a novel and efficient solution to the curse of dimensionality problem by reducing an n-dimensional rule base to a one-and-half dimension or a two-dimensional rule base. As can be seen, the drawbacks of this system are that the number of rules is fixed, is limited to a two-input and single output configuration, and that there is no provision for rule modification, and once again, it is left up to the operator to define the control actions.

As can be seen, due to the amount of information needed to represent the entire structure, it is neither efficient nor practical to attempt to liberate the complete structure of the FLC by encoding it in a fixed size GA chromosome. That is, the more information is encoded in a GA chromosome, the larger it gets. Thus the performance of the GA is degraded despite using non-binary coding schemes to keep the length of the chromosome short. In addition to affecting the performance of the GA, the stability and performance of the system is also compromised. The stability of the system is governed largely by the number of rules and the combination of premise and action for each rule. Hence, to accommodate higher order problems, such methods as described above compromise by using a smaller number of rules. Therefore, fixed structure FLCs and length EA chromosomes are often inadequate.

1.2.4 Optimisation With Flexible Structures

At Glasgow, structural design problems were recognised but remained unsolved (Ng 1995, Li and Ng 1996). In order to liberate the structure, there have been a few approaches that used flexible GA coding schemes (Lee and Takagi 1993, Cooper and Vidal 1994, Hoffmann and Pfister 1995, Chowdhury and Li 1996, Carse *et al* 1996, Li and Häußler 1996). The structural optimisation method for selecting neural network architectures reported by Li and Häußler (Li and Häußler 1996) is based on network pruning and may be adopted here. In pruning however, the EA chromosome has to start from a parent architecture that has to be very large to accommodate all predicted possible architectures. However, some a priori knowledge of the controller structure is not always possible, as has already been argued. It also means that if the operating conditions change and there is need for growing the network to follow the change, it is not possible. It is also limited to small problems as for larger and more complex problems, the size would be too large and computationally inefficient. Hence, pruning is not fully adequate to achieve the objectives of the thesis.

Cooper and Vidal tried to address the complexities and issues highlighted in the previous subsection through a coding scheme based on matching fuzzy rules that are similar (Cooper and Vidal 1994). In this, each membership function representing a control variable is represented by two integers, and only triangular membership functions are used where only the centres and half widths of the functions are tuned. Each rule is a concatenation all the membership functions of the variable, and the rule base, represented by a full chromosome, is a concatenation of all such rules. Cooper and Vidal address the curse of dimensionality and redundant coding issues by using a mechanism whereby variables are ignored when their half-length values fall outside a certain range. Reproduction and combination in this approach are based on crossing over rules of similar structure. That is, before reproduction, the rules in the two mating chromosomes must be aligned so that they match as closely as possible. Any rules that are not matched are appended to the end of the chromosome.

As can be seen from the above description, the Cooper and Vidal method has many problems and limitations. First of all, using two integers to represent a single membership function lengthens the chromosome and the problem gets worse with higher order problems. The method is also not flexible with regard to the type of membership functions that can be used. Only triangular membership functions are used, and tuning of such functions is limited to the centre and the widths, but not the supports. It is also not clear how membership function overlap is maintained when the half-length of the fuzzy sets falls outside a certain range and is then ignored. This would suggest discontinuities in certain regions of operation. Before reproduction and combination, the Cooper and Vidal approach requires reordering of the rules, and hence the genes of the chromosomes. This not only introduces computational inefficiency, but also the likelihood of matching schemes is not guaranteed.

Based on the work by Cooper and Vidal, Carse *et al* proposed a flexible representation scheme to tune fuzzy sets for classifier systems (Carse *et al* 1996). A chromosome represents fuzzy parameters belonging to an input or output variable, which must be a triangular fuzzy set. The parameters of the set encoded are its centres and widths. Each rule is in the form,

$$R_k : (x_{c_{1k}}, x_{w_{1k}}); \ldots (x_{c_{nk}}, x_{w_{nk}}) \Rightarrow (y_{c_{1k}}, y_{w_{1k}}); \ldots (y_{c_{mk}}, y_{w_{mk}}) \tag{1.2}$$

where x is the input; y is the output; n is the number of inputs; m is the number of outputs; k is the rule index; $x_{c_{nk}}$ is the centre of the fuzzy set representing input n and $x_{w_{nk}}$ is the width of the fuzzy set representing input n. This representation allows rule premise and consequent variables to have their own fuzzy sets instead of sharing a global one, as found in most fuzzy rule base representations.

To accommodate a variable number of fuzzy sets, this approach uses a variable length chromosome and the crossover operator in a standard GA is replaced by a new one which works differently depending on the size of the input dimension. For the case of a single input, the

genes are sorted according to the centres of the input fuzzy sets before crossover takes place such that the resulting chromosomes of the offspring will be valid. For the case of n-dimensional input space, instead of using a single crossover point, a vector of points is created, as given by Carse *et al* (1996)

$$C_i = MIN_i + (MAX_i - MIN_i) \cdot R_c^{1/n} \tag{1.3}$$

where $[MIN_i, MAX_i]$ is the range of the input variable x. The vectors are applied such that $x_{cik} < c_i, \forall_i$, and rules from parent 2 such that $x_{cik} > c_i, \forall_i$. The remaining rules from both rule-sets then form the other child. Finally, a *random creep* is used to fine tune the fuzzy set parameters.

While the Carse *et al* approach offers an improvement on the Cooper and Vidal one, it still has a number of limitations. First, fuzzy tuning takes place only at the centres and widths, thus representing symmetrical fuzzy sets. Second, only triangular fuzzy sets can be accommodated. If, for example, trapezoidal sets are mixed in, the number of parameters and hence the chromosome length will be changed, which will complicate sorting the now irregular genes and selecting the crossover points. Third, such sorting is prone to premature convergence as it is against the proven philosophy of well mixing the genes as in *uniform crossover* scheme. Also, it is not clear what would happen after crossover occurs and one chromosome ends up with one single rule while the other child ends up with a very large rule base. Other issues also arise, such as why it is necessary to have different fuzzy sets for similar rules since each rule will have its own fuzzy sets.

Hoffmann and Pfister (Hoffmann and Pfister 1995) accomplished a structure using messy genetic algorithms (mGA) (Goldberg 1989b), similar to the design of FLCs. The coding, decoding and representation are simple and less ambiguous than the above process. The coding scheme has both input and output variables where the universe of discourse of each of the variables is covered by fixed fuzzy sets defined a priori. The coding element represents the fuzzy clause and is a pair of integers. The first refers to the variable and the second refers to the fuzzy set of this variable. Since the orders of the genes are irrelevant, a first-come-first-served precedence rule is also applied to resolve conflicts between two rules with identical conditional steps. The whole rule base is encoded in the string and each rule within the string is treated as a gene, thus representing a hierarchical structure. The drawback of this system is that although it allows for rule structure modification, the actual type and shape of the fuzzy sets can not be modified. In addition, the universe of discourse of each of the variables, as well as the number of fuzzy sets in each variable, is fixed. That said, the approach is novel because it was the first scheme to have a flexible structure through an EA other than the regular GA. Messy GA is significantly superior to the regular GA for such structural optimisation because it allows for representation of more than one type of information within each gene, thus enabling the coding

of higher order systems. Depending on what information each gene holds, it is also possible to encode all the information pertaining to a specific fuzzy set. In addition, the order of the genes is not important hence increasing computational efficiency.

Even with flexible encoding schemes, representing a complete FLC within a GA is difficult and learning of such systems is more involved. On-line implementation is not practical due to the nature of the EAs. Hence such hybrids usually function off-line and are generally restricted to simulation only. When encoding neurofuzzy systems in a GA chromosome, one can either decide only to type the fuzzy parameters, obtain the network structure or both. Naturally, the more information encoded, the greater the processing power required, and computational efficiency is compromised. This realisation of the limitations of both approaches has in recent years seen the reporting of hybrids comprising of both methods resulting in evolutionary based neurofuzzy FLCs. Either such hybrids utilises neural learning techniques on the EA chromosomes, or gradient decent algorithm is applied to an EA learned FLC, or the EA tunes an ANN representing a FLC.

The novel approach of Hoffmann and Pfister has inspired many of the thoughts presented in this work (Chowdhury and Li 1996). The underlying objective has been put into ensuring that all three desirable properties of a FLC are satisfied based on a mGA-neurofuzzy hybrid. By using a neurofuzzy structure, it was possible to tune the fuzzy sets as well as the rule structure. Halving the gene size of a pair of integers, a single integer is used to encode the fuzzy premise or consequence and the type of fuzzy set. In addition, the restrictions of the Hoffman approach on the number of fuzzy sets was eased by allowing the mGA to explore with a variable number of fuzzy sets. While the mGA is used to obtain the structure of the FLC, the tuning of the fuzzy set parameters is carried out using a neurofuzzy model. This forms the foundation of the work presented in this thesis.

1.4 Thesis Contributions

The aim of this thesis is to develop a method for optimally designing fuzzy control systems that are autonomous, flexible and globally and structurally explored. Autonomous in this case implies that the controller has to be self-supporting with learning and adjusting capabilities, and should be able to track environmental uncertainties and system parameter variations. Such design should be flexible enough to tackle problems where description of the plant is difficult to obtain or unavailable, and it should be independent of the need for training data. The main original contributions presented in this thesis are summarised below followed by more detailed descriptions of each point highlighting differences with other state-of-the-art work found in literature. Emphasis is on systematic and automatic design.

1. A systematic and novel approach to flexible and optimal design automation of FLCs

In chapter 3, fuzzy control components have been analysed and compared with a more "conventional" approach in order to identify the areas where learning and optimisation are best required. Earlier work relevant to this thesis

- uses supervised learning models only
- uses either Mamdani or Sugeno type FLCs
- is constrained to a specific fuzzification, inferencing and defuzzification strategy
- uses only one type of membership function
- deals with partial tuning of the fuzzy system and requires expensive, accurate, and reliable training data.

The model that has no such restrictions has been developed. The structure of the model is such that it can

- be in batch mode (supervised)
- be ready for operation and tuning online (unsupervised)
- switch between Mamdani or Sugeno type controllers, with a network structure accommodating both
- deal with mixed types of fuzzy sets; with no restriction on the type of fuzzification, inferencing and defuzzification schemes.

The coding and representation scheme developed enables the automatic FLC designer to obtain from the decoded mGA chromosome a neurofuzzy network structure in terms of its

- topology
- number of network building blocks
- type of fuzzy sets
- number of fuzzy sets for each input and output variable

With this representation, the network structure is completely liberated in terms of size, connectivity, operation and optimality. While modular in approach, the evolutionary learning and the neural learning of the FLC are part of the same model, which is labelled ENFLICT. Both learning methods complement each other, and separating the evolutionary part from the neurofuzzy part would therefore be inappropriate.

2. Advantage reinforcement learning technique achieving refinement and online learning

The need for and advantages of learning from interaction with the environment are analysed. On-line learning algorithms found in literature require prior knowledge of probability distributions or complex matrices. In this thesis, no such matrices are required. Instead online learning is achieved (Chapter 4) using ideas borrowed from simulated annealing.

The basic *advantage learning* algorithm has been extended for continuous and online learning of neurofuzzy control without the need for lookup tables.

3. Model free design of fuzzy control systems

Combining the model dependent ENFLICT structure with the reinforcement learning procedure developed, a new method is further developed in Chapter 5 to deal with complex systems or with systems where obtaining a mathematical model of the system is difficult or impossible. Unlike existing methods, the method developed here does not require any a-priori knowledge of or information on the path to the desired output. It is not limited to rule deletion only, nor to offline supervised learning.

4. A new method for hierarchical neurofuzzy structure to handle complex systems

For systems of higher dimensions respectively, a new method for hierarchical fuzzy control has been developed in Chapter 5. By using the response data as the model, it is possible to learn the controller off-line while still working with a true representation of the model. The advantage of this procedure is that mGA, which has so far been used off-line for structure optimisation, is now integrated to the neurofuzzy-reinforcement autonomous system.

Publications

The work presented in this thesis has been published in the following papers.

1. M. Chowdhury and Y. Li, 1996. Messy genetic algorithm based new learning method for structurally optimised neuro-fuzzy controllers. *Proceedings of the IEEE International Conference on Industrial Technology*, December 1996, pp.274-279. Shanghai, China.

2. M. Chowdhury and Y. Li, 1997. Evolutionary reinforcement learning for neurofuzzy control. *Proc. Seventh International Fuzzy Systems Association World Congress*, June 1997, vol. II, pp.434-439. Prague, Czech Republic.

3. M. Chowdhury, T. Brune and Yun Li, 1998. Direct design of evolutionary neurofuzzy controllers from plant response data. *Third Asia-Pacific Conference on Control and Measurement,* August 1998, pp.147-151. Dunhaung, China.

4. W. Feng, M. Chowdhury, T. Brune and Yun Li, 1998. Benchmarks for testing evolutionary algorithms. *Third Asia-Pacific Conference on Control and Measurement,* August 1998, pp.134-138. Dunhaung, China.

5. M. Chowdhury and Y. Li. Learning fuzzy control systems directly from the environment. *International Journal of Intelligent Systems*, 1998, vol. 13, no. 10-11, pp.949-974.

1.5 Thesis Organisation

The remainder of the thesis is structured as follows.

Chapter 2 provide background material necessary to the rest of the thesis. First, background to neurofuzzy control is presented, followed by analysis of and comparison between conventional and FLCs to identify areas of a FLC that need learning, tuning and optimisation. Finally, the *messy genetic algorithm* (mGA) is introduced. The semantics and benefits of mGA for flexible encoding and representation are discussed, and a simple benchmark is carried out to support the reasons for using mGA over the simple GA.

Chapter 3 builds the ENFLICT (Evolutionary NeuroFuzzy Learning Intelligent Control Technique) model from the bottom up. The objective is to construct a method for automatic controller design. Emphasis is placed on learning and flexibility of structure and implementation using neural, fuzzy and evolutionary methods. First, the neurofuzzy structure is constructed, and then, to enable online learning, a method based on annealing is developed. Then the messy genetic algorithm and coding scheme for the network structure optimisation is explained. Finally, the features of the ENFLICT model, which distinguish it from other models, are compared and discussed, and the effectiveness of the developed method is illustrated with some case studies.

Chapter 4 extends the model-based approach that ENFLICT uses in the previous chapter to work without a model by directly interacting with the environment. The interaction is achieved through reinforcement learning. Reasons for using RL are given and a correlation between EAs and RLs is established. To cope with continuous time systems, and online learning without a look-up table, an extension is made to the *advantage learning* RL algorithm. The modified algorithm is then used with the ENFLICT model to learn the network structure

and parameters online. Since no model description is present, the gradient information unavailable, the backpropagation learning algorithm is replaced with a new and novel way for learning the fuzzy sets online with the ability for rule reduction. Finally, the entire procedure is tested against some benchmark problems.

Chapter 5 discusses how to tackle more complex and large systems through hierarchical structures, where the EA works at the upper level and the reinforcement neurofuzzy structure works at the lower level. The difficulties of on-line and real-time RL are discussed and a method of overcoming this by using plant data directly is suggested. To illustrate, a comparison between this method and that developed in Chapter 3 is made on the same case study problems.

Chapter 6 analyses and discusses the work in the thesis and explores possibilities and scope for further work in this field.

Chapter 2

Background

"To be or not to be: that is the question"

– William Shakespeare

The purpose of this chapter is to lay the foundation for the contributions of the thesis. First the origins and role of neurofuzzy control is presented, followed by an investigation and analysis of the components that make up a FLC. The purpose of this is to identify and demonstrate the various combinations possible in the design of FLCs, and where learning and optimisation can be applied. Then the messy genetic algorithm is introduced. Messy genetic algorithm is the evolutionary algorithm technique in the thesis for the structure optimisation of the neurofuzzy network and the technique used as the teacher in hierarchical fuzzy control structure. The main characteristic of mGA is it enables flexible coding that is ideal for network topology optimisation. mGA is used instead of any other flexible coding scheme, or devising a coding scheme from scratch because it has been proven theoretically and has been applied to a number of applications. To demonstrate the effectiveness of mGA, a simple benchmark test is carried out, comparing it with other EA techniques. Following on from this, reinforcement learning as a tool for model free unsupervised learning is explained and the similarities and differences between RL and EA are highlighted.

2.1 Fuzzy Control Systems – Analysis and Comparison

The main reason for the popularity of fuzzy logic is the successful application of its principles in the design of fuzzy logic controllers. Fuzzy logic controllers are essentially a type of fuzzy logic system employing a knowledge base and an inference engine to solve a specific control problem. There is no set type of FLC. In other words, the way of expressing the rules to describe the knowledge base and the inferencing engine varies with the type of control problem.

Control systems based on fuzzy logic are popular because they are able to utilise knowledge extracted from human operators. Fuzzy logic control does not require a

conventional model of the process, whereas most conventional (model based) control techniques need either an analytical model or an experimental model. Therefore, fuzzy logic control is appropriate for complex and poorly defined processes in which analytical modelling is difficult due to various factors, for instance the process may only be partially known or experimental model identification is not practicable because the inputs and outputs may not be measurable. It is more feasible to express control rules in linguistic form based on the knowledge of an experienced human operator, which the operator understands.

In this section fuzzy logic controllers are explored, looking in detail at the components making up a fuzzy logic controller. When designing fuzzy logic controllers, a number of assumptions are usually made (Ross 1995).

1. The plant is observable and controllable: state, input and output variables are available for observation and measurement
2. There is a knowledge base consisting of input and output measurement data that can be fuzzified for rule extraction
3. There is always a solution
4. An optimum solution is not necessary as long as it is "good enough".
5. The controller can only be designed with the knowledge available and within an acceptable range of precision
6. Optimality and stability problems are still persistent in FLC design.

With these assumptions in mind, the architecture of a FLC will now be studied through a comparative study with PD controller. First the various components are identified and then the effect of different membership functions, numbers of rules, different fuzzy reasoning and the positioning of the membership functions are examined. The tests are carried out using computer simulation. A nominal plant (2.1) and two perturbed plants, (2.2) and (2.3), based on the nominal are defined below. The design condition is that the controller must satisfactorily control all the plants under closed loop for a step response following.

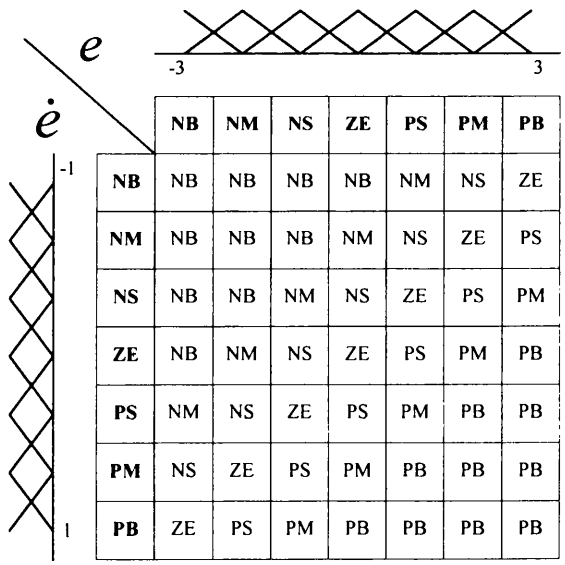
$$G_n(s)=\frac{5}{s^2+3s+2} \tag{2.1}$$

$$G_1(s)=\frac{5}{s^2+3s} \tag{2.2}$$

$$G_2(s)=\frac{1}{s^2+2} \tag{2.3}$$

The specifications are that the settling time should be about 20 seconds, with fastest rise time and smallest overshoot possible. Using trial and error manual tuning K_p = 0.2 (proportional action) and K_d = 0.1 (derivative action) are obtained. The FLC is of Takagi-Sugeno type with seven equally spaced triangular memberships for each of the input variables (error and rate of change of error). The output variable is defined using seven spikes and the *weighted average* defuzzification strategy is employed. Only a PD type controller is considered because the inputs to the FLC resemble a PD controller. Table 2.1 shows the rule base for the FLC, with the universe of discourse and membership positions shown above and to the side of it. Each input and output are described by the linguistic variables 'NB' (Negative Big), 'NM' (Negative Medium), 'NS' (Negative Small), 'ZE' (Zero), 'PS' (Positive Small), 'PM' (Positive Medium), and 'PB' (Positive Big).

Figure 2.1 shows the response of the nominal plant with the PD and FLCs to a unit step input. From these it is apparent that both controllers perform well but with the FLC there is no overshoot and has a faster rise time. However, the responses of the perturbed systems with the same controllers are quite different. Figure 2.2 and Figure 2.3 shows the response of the perturbed systems with each of the controllers. As can be seen the FLC performs significantly better than the PD in terms of robustness. Notice the case for perturbed plant 2. The PD controller is extremely erratic, while the FLC, although not reaching the desired response, is nonetheless much more stable. Note that none of the controllers are tuned after the initial design.



ė \ e	NB	NM	NS	ZE	PS	PM	PB
NB	NB	NB	NB	NB	NM	NS	ZE
NM	NB	NB	NB	NM	NS	ZE	PS
NS	NB	NB	NM	NS	ZE	PS	PM
ZE	NB	NM	NS	ZE	PS	PM	PB
PS	NM	NS	ZE	PS	PM	PB	PB
PM	NS	ZE	PS	PM	PB	PB	PB
PB	ZE	PS	PM	PB	PB	PB	PB

Table 2.1 Fuzzy rule base for nominal plant

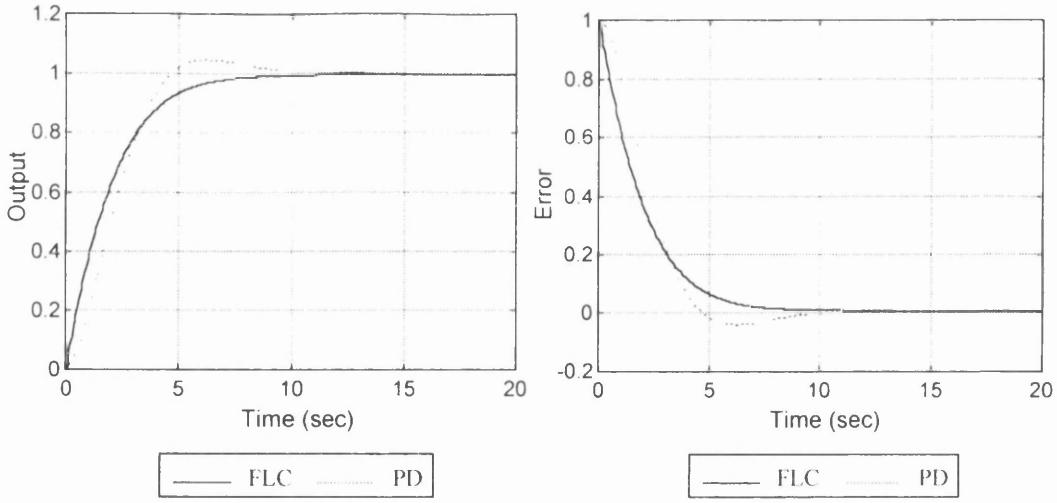


Figure 2.1 Nominal plant response: Fuzzy v PD

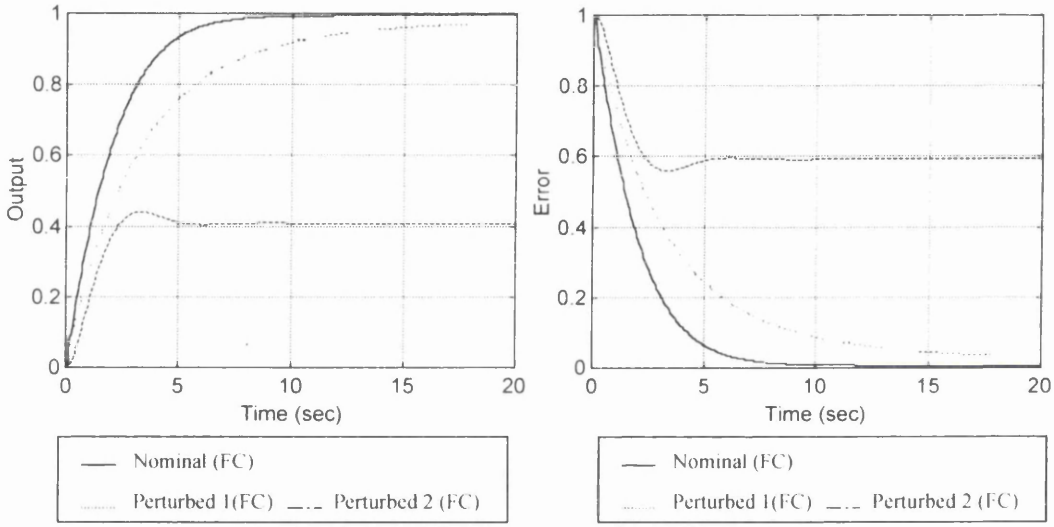


Figure 2.2 Comparison of plant response with FLC

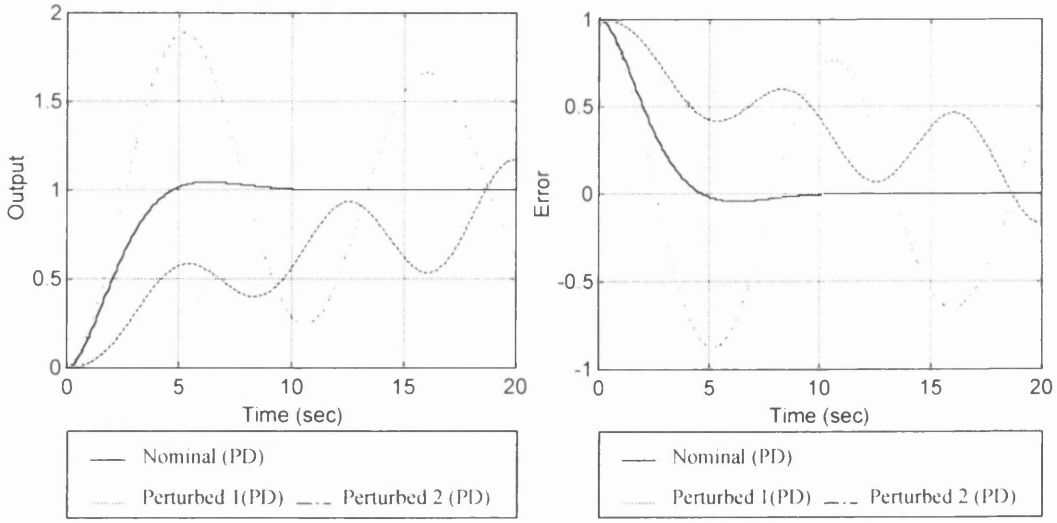


Figure 2.3 Comparison of plant response with PD controller

Next the various stages of the FLC are investigated. The first stage is the knowledge base. As opposed to the simple PID design where there are only three parameters to optimise, in fuzzy control the number of parameters to optimise are numerous. For instance, in the knowledge base the parameters to consider are choice of universe of discourse for each variable and the structure of the rule base. Figure 2.4 shows the response of the system to the non-symmetrical (and completely randomly generated rules) rule base of Table 2.2. The difference in response is obvious. The random generation of the rules was to demonstrate that the structure of the rule base is dependent on the knowledge of the operator. That said, the robustness of fuzzy control as a method for controller design is well demonstrated. The response due to the non-symmetrical rule base, while rather oscillatory and erratic, does not fail in the long run. Although not shown in the time scale for Figure 2.4, over a longer period, the non-symmetrical rule base does reach the desired states. In addition to the positions of the membership functions, the shape and number of the membership functions are also important. Figure 2.5 shows the response of the system to various shapes of membership function at the same centre points, and Figure 2.6 shows the response to a varying number of membership functions. For complex systems it is generally better to use more membership functions to define areas of specific sensitivity. The choice of the membership function shapes depends on the application. For systems where smoother transitions between regions are required gaussian shapes are better suited, but where piece-wise implementation is sufficient, triangular and trapezoidal are preferred because they are simpler to implement and computationally efficient.

Another two stages in the fuzzy control design process are the inferencing mechanism and the defuzzification process. Figure 2.7 shows the response to various implication and defuzzification processes. The mean of maximum has a faster rise time but also a greater steady state error. The advantages and disadvantages of the various defuzzification methods are found Klir and Yuan (1995). As can be seen the sensitivity of the system depends on the information the operator provides to the controller.

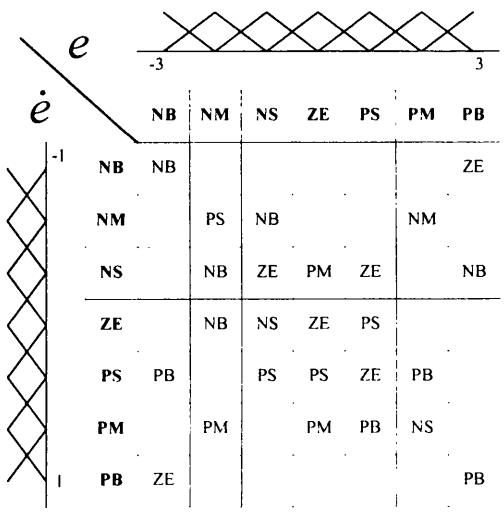


Table 2.2 Non-symmetrical rule base for nominal plant

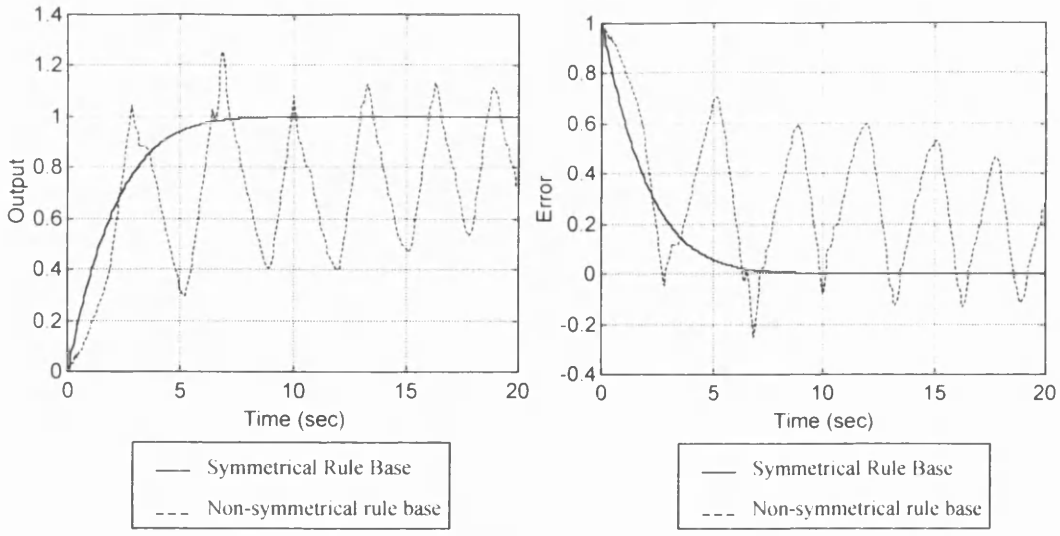


Figure 2.4 System response due to non-symmetrical rule base

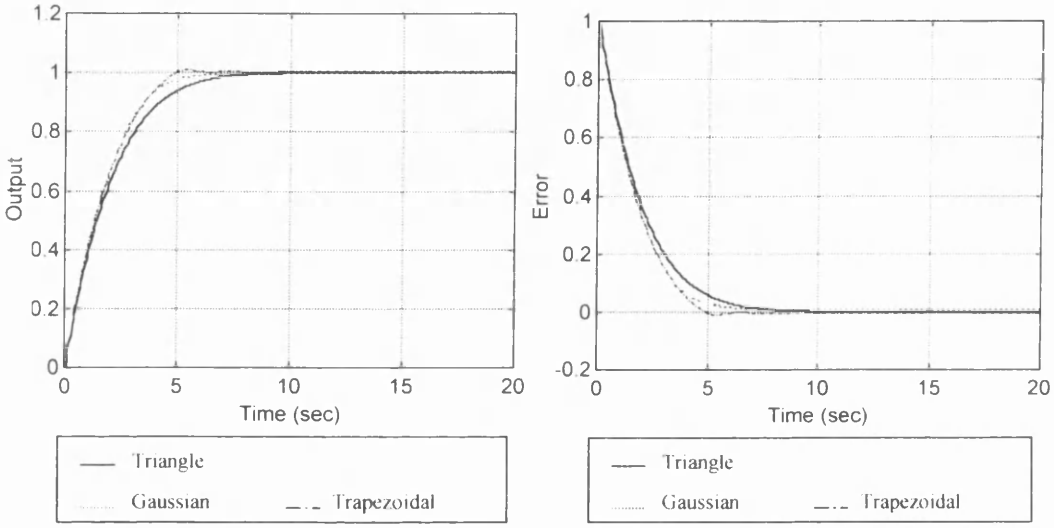


Figure 2.5 Comparison of membership function shapes

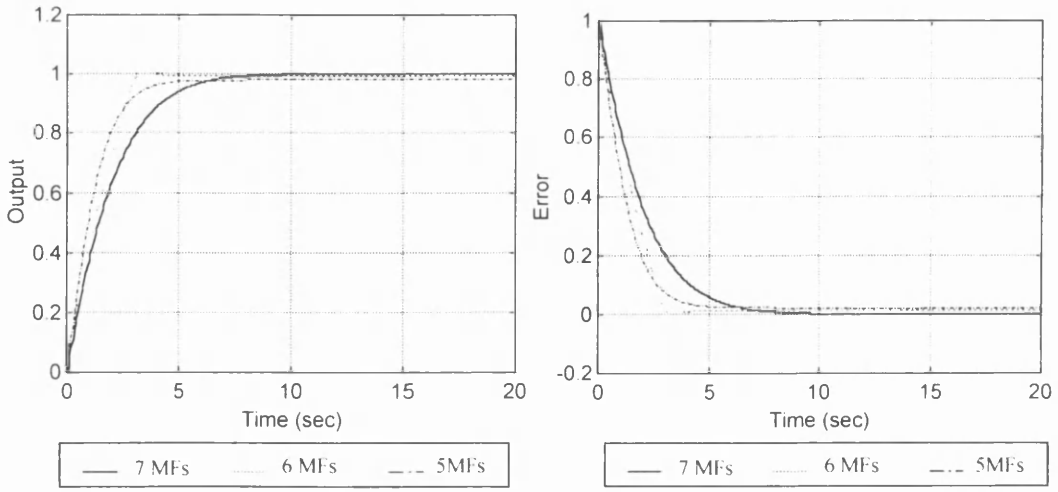


Figure 2.6 Comparison of different number of MFs

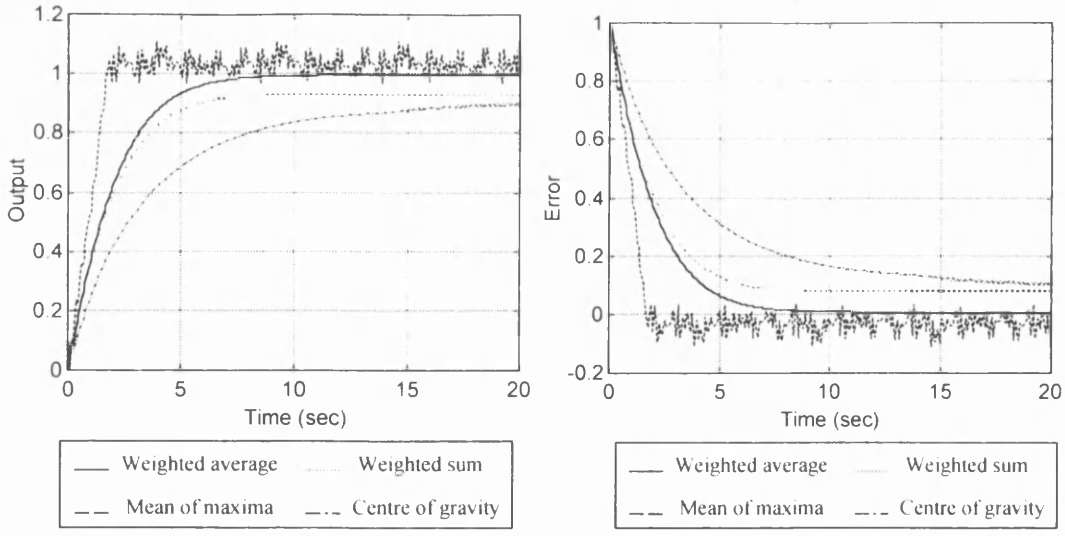


Figure 2.7 Comparison of defuzzification process

From this simple test one might conclude that fuzzy control offers as good, if not better than, an alternative to conventional methods. However the design process is not as simple. There are far more variables to consider and unless the operator is confident of the knowledge provided to the controller, the FLC will not give satisfactory results. It is therefore necessary to find ways to automate the design process and optimise the controller. It is possible, through trial and error, to work through all the possible solutions but that could often be a painful, tiring and very time consuming process. From literature, it can be seen that numerous attempts have been made at reducing a lot of the guesswork from the design process through self-organising and neural network based systems. However, while this seems to have solved some problems, it has introduced many others. How to decide what type of activation functions, thresholds, size of network, number of neurons and number of links and connections to use? In the next section, a messy genetic algorithm as a means for addressing some of these issues is introduced, and a benchmark test on the algorithm is carried out to demonstrate its effectiveness.

2.2 Neurofuzzy Control

Fuzzy logic is based on the theory of approximate reasoning that enables certain classes of linguistic statements to be treated mathematically. This is akin to decision-making by humans who tend to work with vague or imprecise concepts that can be expressed linguistically. To put it simply, fuzzy logic is an extension of conventional logic theory, and is essentially a way of mapping an input space to an output space. These input and output spaces contain objects, or parameters, with defined boundaries. From here on, fuzzy set theory moves away from conventional set theory. In conventional set theory, an object must either belong to a specified set or not, hence partial values are not possible. Therefore the two possible states are

one (existence) and *zero* (non-existence). Fuzzy set theory on the other hand assigns all the objects (*variables*) of a particular class of membership (*linguistic variable*) in the form of a membership function (degree of membership). This membership is usually defined in the range zero *(non-membership)* and one (*full membership*).

It is known that fuzzy control systems may be used as an alternative to some conventional control schemes where significant improved system behaviour can be obtained when fuzzy reasoning is applied. However, there are no optimal guidelines for designing FLCs. If they are designed manually, then a long period of trial and error and much input from experts are required. Because the design process is *ad hoc* it is difficult to defend the choice of, for example, any particular type of membership functions or the reasoning structure. What may be adequate for one set of conditions, may not be appropriate under similar but different conditions. In addition, fuzzy control systems lack the learning ability of other intelligent techniques such as neural networks.

Neural network and fuzzy logic theories were developed about the same period of time. ANNs are massive parallel structures with high non-linear processing elements whose weights and characteristics may be "trained". Fuzzy systems are also of parallel structures but are more suitable for knowledge extraction and representation. However, both knowledge extraction and knowledge representation in an ANN are difficult. On the other hand, the weak points of fuzzy systems are the difficulty of defining accurate membership functions and of applying the learning method. One of the most obvious similarities between a fuzzy system and an ANN is that they can both handle extreme nonlinearities in the system collectively by a network of "local" elements such as memberships or neurons. The functionality of the shape of the membership function in the fuzzy system and that of the threshold function in the ANN, are similar. Multiply-add operation of artificial neurons is very close to ***MAX-MIN*** operation of approximate reasoning. The ***MIN*** operation of input fuzzy variables conducted at each proposition of ***IF*** parts of fuzzy inference rules correspond to a product of input to the neuron and synaptic weights. The ***MAX*** operation to obtain the final inference value from ***THEN*** parts of these plural inference rules corresponds to the input sum with a neuron. These reasons lead to the idea of merging these two approaches.

The following is a summary of the main results, drawn from literature, regarding computational equivalence between neural networks and fuzzy systems (Kosko 1991, Lin and Lee 1991, Horikawa *et al* 1992, Nomura *et al* 1992, Jang 1993, Nauck and Kruse 1993, Spooner and Passino 1996)

1. Feedforward neural networks with n inputs, m outputs ($n \geq 1, m \geq 1$), one or more hidden layers, and a continuous activation function (e.g., the sigmoid function) in each neuron are universal approximators.

2. Fuzzy systems based on multiconditional approximate reasoning can approximate feedforward neural networks with n inputs, m outputs, one or more hidden layers, and a continuous activation function in each neuron, provided that the range of the input variable is discretised into n values and the range of the output variable is discretised into m values.
3. It follows from (1) and (2) that fuzzy expert systems of the type described in (2) are also universal approximators.
4. Fuzzy input-output controllers based on multiconditional approximate reasoning and a defuzzification of obtained conclusion, are universal approximators.

Neurofuzzy networks vary in size from 3-layer to 6-layer networks. In the three-layer format, the first layer represents fuzzy input variables, with the middle layer representing the fuzzy rule base and the final layer representing the fuzzy outputs as in the pure fuzzy logic system. Early attempts at combining neural networks and fuzzy control were limited to just tuning of the shape of the membership functions. A first work in this was by Nomura *et al*, (Nomura *et al* 1992) where the membership functions are assumed to be symmetrical triangular functions depending on two parameters, the peak and the width. Fuzzy cognitive maps proposed by Kosko (Kosko 1991) are another scheme to integrate neural networks and fuzzy logic. Here, the membership function or fuzzy rules are chosen subjectively. Below is a review of some of the more popular fuzzynet structures and models. An overview of other models can be found in (Brown and Harris 1995), (Gomide *et al* 1992) and (Nauck 1997).

2.3 Messy Genetic Algorithm

Messy Genetic Algorithms (mGAs) were developed to eliminate the major problems with the simple GA (Goldberg 1989b). The simple GA is considered too rigid where the length of the string is fixed and a good string arrangement is only possible when information on the tight coding scheme is available. So the only way to bring important alleles together is to use ordering schemes such as inversion. Messy GAs use a relaxed and flexible coding representation to solve linkage problems. In addition, Goldberg showed that mGAs were able to tackle complex higher order problems, which the simple GA was unable to do.

A similar linkage problem has also been existent in nature. In nature, evolution can also be considered rigid because members of a certain species tend only to mate with their own kind, and the evolution operators and operation thus ensure that the full gene complement for that gene is maintained. However, when one considers evolution over a wider time scale, this apparent rule of evolution was not always so. In fact evolution began with simple life forms which used and reused good building blocks through time to form more complex life forms. Therefore, putting GA theory aside, nature itself has shown that structures need not be rigid and

complex from the start, but rather begin by building simple structures and then evolve it to more complex structures through time. In the following sections this alternative form of GA is explored.

2.3.1 An Overview of mGA

A Messy GA is a two-phase iterative optimisation method with a local search template and adaptive representation. The two phases are *primordial* and *juxtapositional.* In the primordial phase, using the selection operator alone, near global solutions are built up. In the juxtapositional phase, the solutions are subjected to the mGA operators to obtain the optimal solution. An mGA is different from the simple GA in that the mGA gene contains its *locus* and *alphabet*, and also uses a variable length chromosome. Due to these features, the operators used in the juxtapositional phase are different from the simple GA.

2.3.2 Flexible Coding

Messy GAs liberates the fixed allele position of the simple GAs by allowing the construction of chromosomes whose genes are ordered independent of its position. A messy gene is an ordered pair identifying its locus (or index) and its alphabet (or value). Since the string is variable and can potentially increase in size, the size of the locus has to be limited to size. Not only is the chromosome of variable length, a certain locus can appear more than once in the string or many may altogether be missing. Consider a problem of length 3, then all of the following strings are valid:

$$S_1 = ((2,1)\,(1,0))$$
$$S_2 = ((3,0)\,(2,0)\,(3,1)\,(1,1))$$
$$S_3 = ((1,0)\,(2,0)\,(2,1))$$

The first string is an example of an *underspecified* string because reference to locus 3 is missing. The second string is a classic case of *overspecification* because reference to locus 3 occurs twice, and is typically handled by some sort or precedence rule. The third example is both underspecified and overspecified. It is underspecified because reference to locus 3 is missing, and overspecified because reference to locus 2 appears twice. Notice also the order of the genes. This is one of the characteristic properties of the mGA, i.e. the order of the genes are irrelevant unlike in the simple GA where it is critical for the decoding process.

2.3.3 mGA Decoding

As has been seen, a messy chromosome can take various forms, and methods of decoding it have to reflect the chromosome representation. More often than not, a full gene complement is required in order to evaluate the objective function, and in such cases, of the two types of representation, overspecification is the easier of the two to handle. Consider a three parameter problem represented by the string ((3,1) (1,1) (1,0)), there is a choice of selecting a value of 0 or 1 for parameter 1. Goldberg and Smith suggested that the most reliable precedence rule is the *first-come-first-served* rule (Goldberg and Smith 1987) where the first gene is taken from a left-to-right scan, hence the gene (1,1) was chosen.

Underspecification need only be dealt with when all decision variables have to be available in order to evaluate the objective function. In the above example, in order to evaluate the objective function with the three values, the missing value needs to be filled in somehow. This is done by comparing the string with a predefined template and by borrowing the missing value from the template. How the template is defined is important. The missing parameter value can be filled by randomly generating it, but this would have the effect of introducing noise that could lead to sub-optimal solutions. Alternatively, the template can be randomly set, but fixed after generating the first time. Again some of the problems with randomly generating every time persist. Another method is to initialise the template with 0s and build it up with locally optimal ones obtained from the completed run. This way, it is possible to be certain to some extent of obtaining a partial sub-optimal performance. This is the method adopted in this thesis. Supposing the locally optimal template were set to (1 1 0), then the missing gene in the above example would take the value (2,1), giving the full gene complement as ((1,1) (2,1) (3,1)).

2.3.4 mGA Operators

Due to the nature of the string representation, the standard genetic operator crossover has to be modified. Consider the fixed length coding of Figure 2.8 and the standard crossover operator on it.

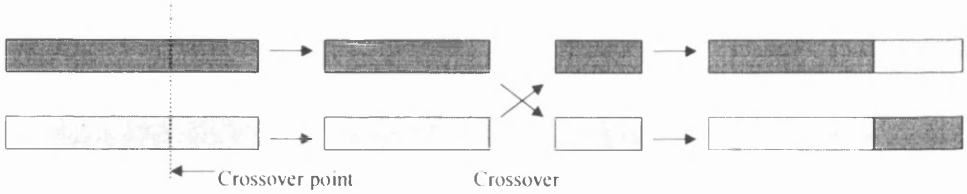


Figure 2.8 Standard crossover on fixed length string

In messy GA such operation would not work because of the variable length string because a single cut point may not be applicable or available on both chromosomes. Instead, the standard crossover is replaced by two operators: *cut* and *splice*.

The cut operator is based on a cut probability calculated as $p_c = p_\kappa(\lambda - 1)$, where λ is the length of the string and $p_\kappa \leq 1$ is a gene-wise cut probability. If the cut is called for, the chromosome is cut at a position chosen uniformly at random. For example, if cut is called for in the string ((2,1) (3,0) (1,1) (3,1) (4,0) (7,1)) and the cut occurs at position 2, the two resulting sub-strings resulting from the cut operation would be ((2,1) (3,0)) and ((1,1) (3,1) (4,0) (7,1)).

The splice operator simply joins two sub-strings, resulting from the cut operator, to form a single chromosome with splice probability P_s. Applying the splice operator on the two sub-strings above would result in the chromosome from which they were obtained through cut. Figure 2.9 illustrates the cut and splice operators graphically.

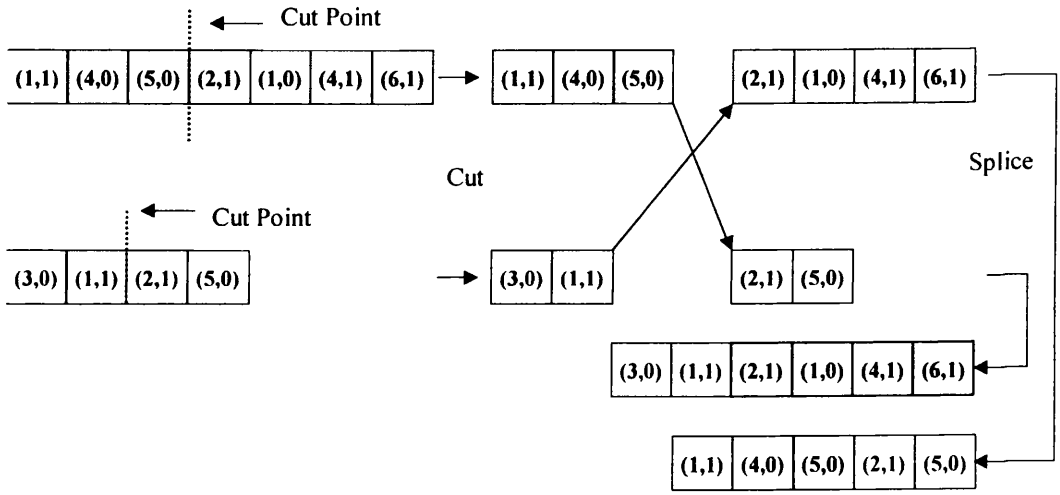


Figure 2.9 Cut and splice operation

If cut occurs on both strings, and splice is not called for in one or both cut parent parts, the non-spliced parts are reinserted back in to the population as new individual strings. Similarly, if cut occurs only on one parent string or no cut occurs at all, then the splice operator is not called for. Note also that in mGA, a single-point crossover is applied.

Mutation in an mGA chromosome can take the form of any type described above for the simple GA, as is also the case with the selection method. In this thesis, hyper-mutation (Grefenstette 1992) and tournament selection (Goldberg *et al* 1991) is used for mGA implementation.

2.3.5 mGA Operation

A messy GA works in two phases: a primordial and a juxtapositional phase. Figure 2.10 shows the schematic of the mGA algorithm.

Before the primordial phase, the population is initialised randomly such that all possible combinations of the currently considered relations are represented. Depending on how the template is to be constructed, the initial population may be evaluated to obtain the objective value for each string, and the locally optimal solution from the population can be used as the template.

During the primordial phase, through the selection operator alone, the population is enriched, over a number of generations, with locally optimal strings. In this phase, no function evaluation is performed. Since no other operators are used, the strings in this phase retain their original lengths. At this stage, if a complete *era* has passed, then the population with the fittest half is kept for future generation and the other half of the population is randomly generated to introduce diversity into the population. Depending on the length of the primordial phase, it may be that several eras will have passed before the next phase starts. After obtaining an enriched population of strings, the population is processed in the next phase where all the mGA operators are used (i.e. cut and splice) as described in §2.3.4. At the start of the juxtapositional phase, larger strings are obtained through splicing. The rest of the operation in this phase follows the standard GA process.

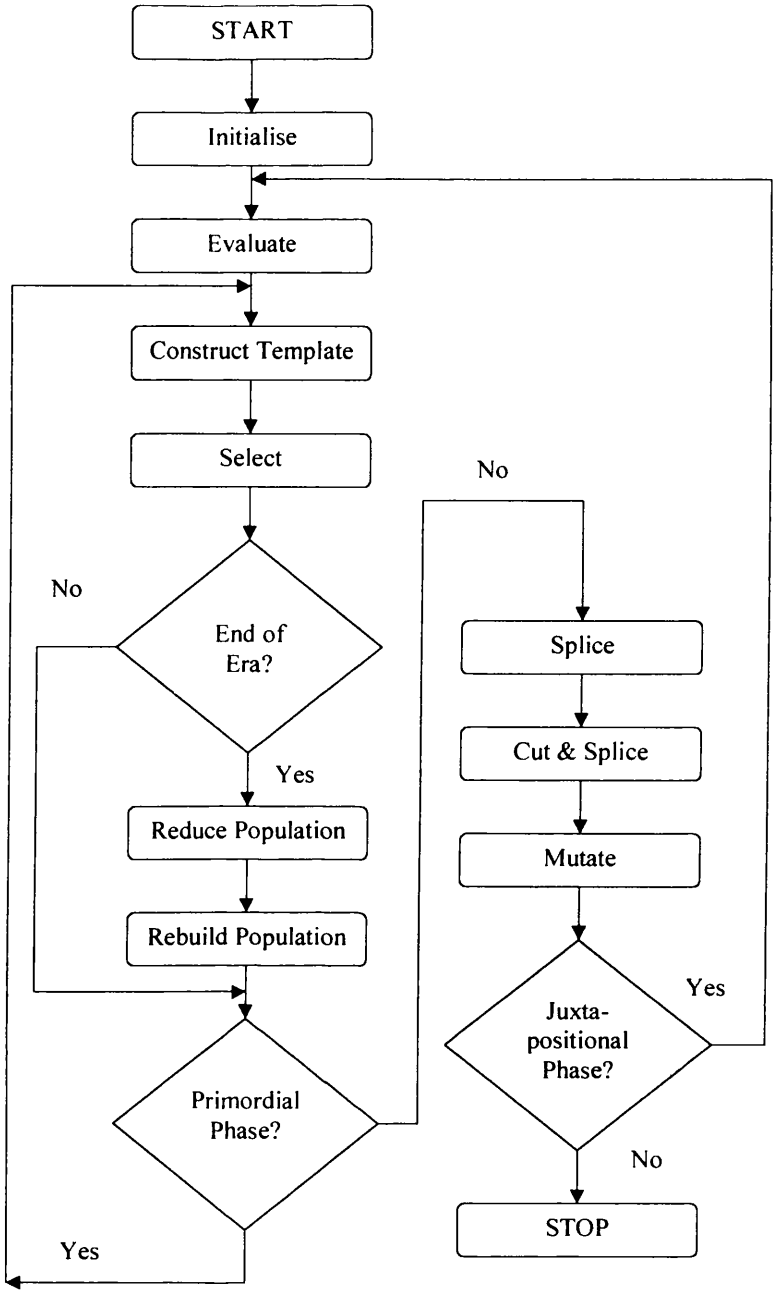


Figure 2.10 Messy GA flow diagram

2.3.6 A Benchmark Test

The objective function of an n-dimensional maximisation problem that was introduced by Michalewicz (Michalewicz 1992) and further studied by Renders and Bersini (Renders and Bersini 1994) is given by:

$$f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i) = \sum_{i=1}^{n} \sin(x_i) \sin^{2k}\left(\frac{ix_i^2}{\pi}\right) \quad \text{for } x \in [0, \pi]^n \tag{2.4}$$

It is composed of a family of amplitude-modulated sine waves whose frequencies are linearly modulated. This objective function is, in effect, de-coupled in every dimension represented by $f_i(x_i)$. Every such member function is independent and is shown in Figure 2.11 for $k = 1$ and $n = 20$. This characteristic yields the following properties:

The theoretical benchmark solution to this n-dimensional optimisation problem may be obtained by maximising n independent uni-dimensional functions, f_i, the fact of which is however unknown to an optimisation algorithm being tested. The results for $k = 1$ and $n = 10$ are shown in Table 2.3. Note that the lower boundary of the objective is $f_{\min} = 0$ within the given search space. The optimality, accuracy and sensitivity are, as shown in (Feng 1998):

$$\text{Optimality} = \frac{\hat{f}_0 - f_{\min}}{f_{\max} - f_{\min}} = \frac{\hat{f}_0}{9.6547} \tag{2.5}$$

$$\text{Accuracy}\big|_2 = 1 - \frac{\left\|\mathbf{x}_0 - \hat{\mathbf{x}}_0\right\|_2}{\left\|\bar{\mathbf{x}} - \underline{\mathbf{x}}\right\|_2} = 1 - \frac{\sqrt{\sum_{i=1}^{10} \left|x_{0i} - \hat{x}_{0i}\right|^2}}{\sqrt{10}\pi} \tag{2.6}$$

$$\text{Sensitivity} \approx \frac{1 - \text{Optimality}}{1 - \text{Accuracy}} \tag{2.7}$$

- The larger the product $k\,n$ is, the sharper the landscape becomes.
- There are $n! = 2.4329 \times 10^{18}$ local maxima within the search space $[0, \pi]^n$.
- The ease of obtaining theoretical benchmarks regardless of n makes it ideal for studying NP characteristics of the algorithms being tested.

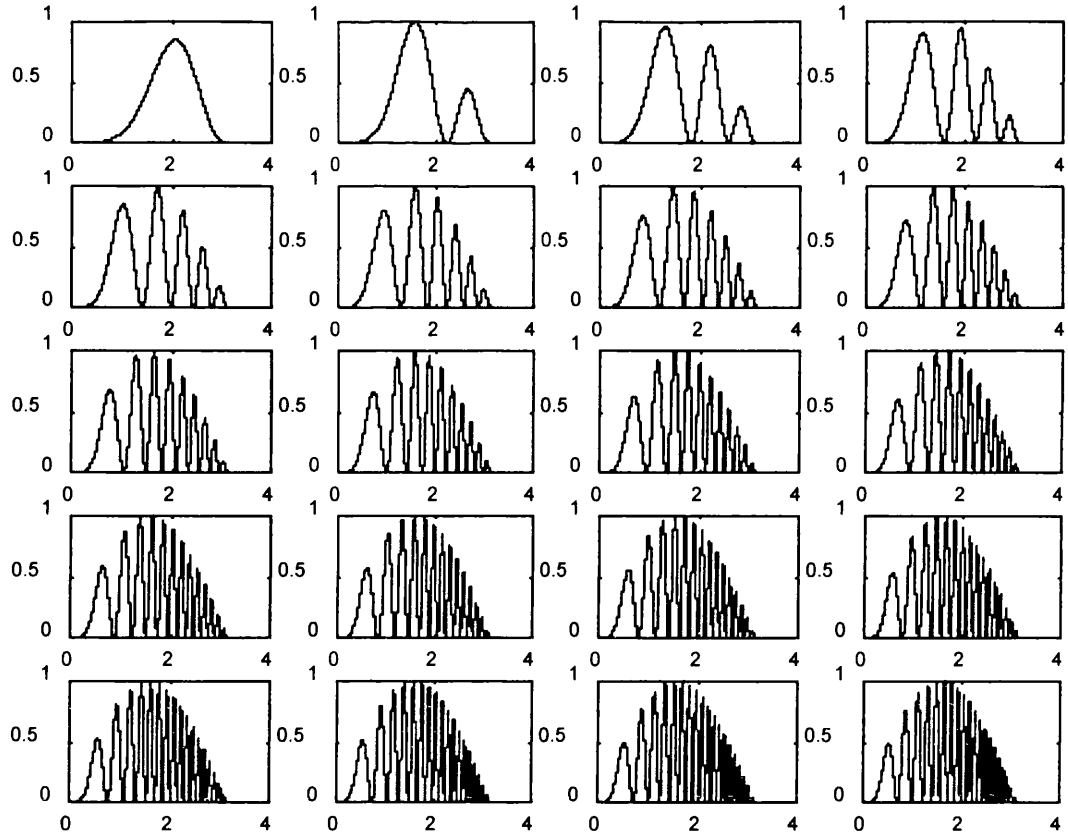


Figure 2.11 The *n* independent uni-dimensional functions that form the 20-D objective function

Using the above benchmarks, the performances of some EAs are tested. For each method, 10 repeated experiments are carried out with randomly generated initial populations. The results of optimality, accuracy, sensitivity, reach-time and optimiser overhead are shown in Table 2.4.

i	1	2	3	4	5
x_{i0}	2.072	1.571	1.305	1.916	1.718
f_{i0}	.8409	1.000	.9619	.9396	.9890
i	6	7	8	9	10
x_{i0}	1.571	1.458	1.755	1.655	1.571
f_{i0}	1.000	.9933	.9830	.9964	1.000

Table 2.3 Theoretical solutions and objectives of Benchmark Problem (2.4)

Algorithm	Supremum	Optimality	Accuracy
Random search	3.331613	**34.51%**	25.7369%
Simplex (Press *et al* 1994)	1.789972	**18.54%**	25.7807%
Hill-climbing (a posteriori)	9.6363	**99.81%**	99.21%
Simulated annealing	9.6402	**99.85%**	99.20%
Simple GA (Goldberg 1989)	5.876064	**60.86%**	21.7845%
FlexTool(GA) Toolbox	9.2081	**95.37%**	89.0542%
Messy GA	9.3743	**97.10%**	96.437%
Theoretical objective	9.6547	**100.00%**	100.00%
Algorithm	**Sensitivity**	***N* or Reach-Time**	**Overhead**
Random search	88.3786%	40,000	2.6437%
Simplex (Press *et al* 1994)	109.978%	40,000	5.5914%
Hill-climbing (a posteriori)	24.05%	39,038	25.07%
Simulated annealing	18.75%	38,721	34.23%
Simple GA (Goldberg 1989)	0.502699	40,000	1111.78%
FlexTool(GA) Toolbox	42.3732%	40,000	1170.36%
Messy GA	81.39%	40,000	1058.66%

Table 2.4 Benchmark test results on the 10-D problem

2.4 Reinforcement Learning

2.4.1 Reinforcement and Advantage Learning

In many control problems, the most appropriate control actions are unknown and thus learning techniques are employed. For learning, three types of mainstream learning methods are explored: supervised, unsupervised and reinforcement learning. Most learning fuzzy control techniques fall into the category of supervised learning systems, and their biggest drawback is the need for the desired output of the controller. These desired outputs are generally considered to be provided by a supervisor. This requirement is difficult to satisfy for a control system since the most appropriate control actions may not be known.

A learning paradigm, known as *reinforcement learning* (RL), has a more appropriate feature in that instead of requiring a supervisor to provide the correct control actions, it can accept feedback of scalar performance measured by a *critic*. The *critic* defines good and bad performance. A motivation for RL is that it is the primary learning method of biological systems. Animals learn and adapt daily with only reinforcement type error signals. Reinforcement learning studies therefore seek to capture similar capabilities in artificial systems. Just as artificial neural networks are patterned after biological neural networks, RL systems strive to emulate animal learning and are investigated in this thesis.

Reinforcement learning is an approach to machine intelligence that combines unsupervised learning and dynamic programming to solve problems that neither of these disciplines are able to address alone (Barto *et al* 1983). Dynamic programming is a field of mathematics that has traditionally been used to solve problems of optimisation and control. It works by generating a utility function (J) which is optimised in the short term for an

environment (m) with utility function (U), and U is optimised in the long run. However, traditional programming is limited in the size and complexity of the problem it can address.

In reinforcement learning, a signal is received that does not say anything about the desired response as in supervised and unsupervised learning, but it does say whether a system is performing well or badly. Usually, a system is said to be learning when it improves its performance based on a certain performance measure. Suppose that the performance measure is calculated as a function of the parameters of the learning system, which represent its current state. For instance, in a water tank system these parameters could be e the error between the current and the destination height, h the destination height, u the inflow signal sent to the pump and maybe some other sensory information. If the performance measure can be visualised as a surface, then each state of the system (e; h_d; u...) can be assigned to a point on that surface $f(e$; h_d; u...) where f is the performance-measure function. Now, if the system is to improve its performance, the point corresponding to its state on the performance surface should move towards higher points (Barto 1992).

Figure 2.12 illustrates the reinforcement learning procedure. An agent is connected to its environment via perception and action. On each step of interaction the agent receives as input some indication of the current state of the environment; the agent then chooses an action to generate as output. The action changes the state of the environment and the value of this state transition is communicated to the agent through a scalar reinforcement signal. The agent's behavior should choose actions that tend to increase the long-run sum of values of the reinforcement signal. It can learn to do this over time by systematic trial and error, guided by a wide variety of algorithms.

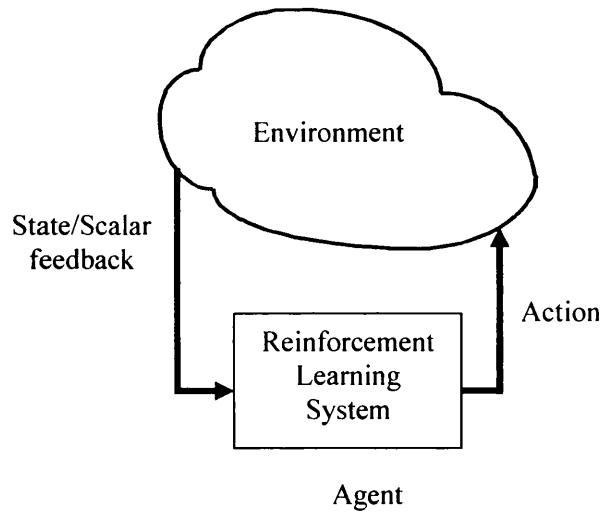


Figure 2.12 Standard reinforcement learning schematic

In addition to the environment and the agent, a reinforcement learning system has three main sub elements:

- A *policy*
- A *reinforcement function*
- A *value function.*

A policy is the central part of the agent, and on its own can define the behaviour of the agent. It decides which actions to take in a given state. All the other components work around the policy and works to improve the policy. The reinforcement function determines the objective of the reinforcement learning agent, and the aim of the agent is to maximise the reward it receives over the long run. It is fixed and indicates what is good or bad in the immediate situation. The value function is a mapping from states to state value and can be approximated using any type of function approximator (e.g. multi-layered perceptron, memory-based system, radial basis functions, look-up table, etc.) (Anderson 1989, Watkins 1989, Kaelbling 1991, Lin 1992, Millan and Torras 1992, Singh 1992, Thrun 1993, Anderson 1993, Glorennec 1994, Gullapalli *et al* 1994, Lin and Lee 1994, Lee *et al* 1995, Shijojima *et al* 1995). It is a reward predictor and indicates what is good or bad in the long run.

In application to the optimal control problems RL could be formulated in a way that the long-term consequences of actions are taken into account since in most of the cases the goal is to design a controller with an optimal long-term performance. The RL-controller is then designed to receive a reinforcement signal from the controlled process based on its performed action and the state of the process. The objective of the learning system is then to minimise or maximise the amount of reinforcement signals accumulated in the future, depending on what the signal represents, cost or benefit. This performance measure is often calculated as a discounted sum of the future reinforcement signals in which the earlier ones are weighted more.

2.4.2 Reinforcement Learning and Evolutionary Algorithms

The standard RL problem is based on dynamic programming approach, and the difficulty with that is it uses an estimate of the value and value function hence does not actually produce global solutions as evolutionary algorithms. However, on closer examination, evolutionary algorithms can be identified to be a special type of reinforcement learning system except that they differ from dynamic programming RLs in two important ways. First, EAs search in a completely random fashion and hence ignore a lot of the information between state transitions; and secondly, EAs discards poor solutions in favour of good ones whereas RLs use this information in its decision making process. To illustrate this, consider Figure 2.13.

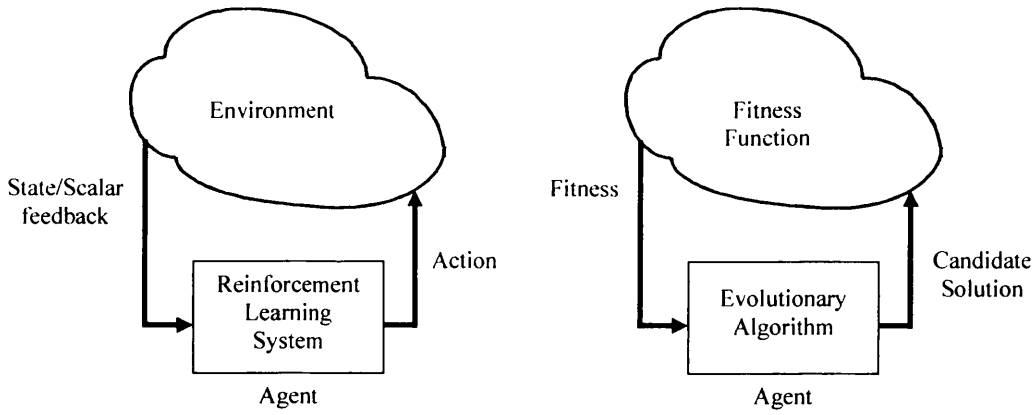


Figure 2.13 Comparison of reinforcement learning and evolutionary algorithms

Comparing the way an RL system and an EA is structured, it is evident that EAs are in fact RL systems. Evolutionary algorithms tackle the same kind of problems as the dynamic programming RLs, and have similar properties. Neither requires derivative information, and both work around a performance measure, and yet they work differently. EAs do not learn the value function, but instead learn the candidate solution (or policy) directly. Instead of working with a single policy (i.e. population member) at a time, It generates population of policies and can evaluate each one sequentially or in parallel. Through genetic operators, crossover and mutation, new pool of policies is constructed. Each policy is then evaluated against a reinforcement function, and based on the goodness of that policy, credit assignment is carried out. Unlike RLs though, the algorithm iterates until a sufficiently good policy is found. Since EAs explore the search space in a random fashion, EAs ignore much of the useful structure of the standard RL problems, in that they do not make use of the fact that the policy they are searching for is a function from state to action. Also they do not notice which states an individual passes through during its lifetime or which actions it selects.

2.5 Summary

This chapter serves as background and foundation for the contributions made in the thesis. The theme of the thesis is fuzzy learning through neural network representation, and the relationship between fuzzy systems and neural networks was highlighted. Through a simple analysis and experiment, the difficulties and confusion that a designer faces when design FLCs were identified. The conclusion of the experiment is that a method or set of methods should exists that allows for designing FLCs such that it is completely flexible and have self learning properties so that the operator or designer does not have to design in ad hoc. For this purpose, it has been decided that Goldberg's messy genetic algorithm and reinforcement learning would be used. Messy genetic algorithm for optimising and tuning neurofuzzy controllers was described, and the original binary coding of Goldberg's mGA is replaced by integer coding to enable

flexible representation of neurofuzzy systems. This was followed by a simple benchmark test to show the potential of mGA.

To demonstrate the need for flexibility in the structure, consider the general format for representing fuzzy sets within a fixed length chromosome such that each gene represents a parameter of the fuzzy set (i.e. the spread and centre for gaussian sets). Then there would be a need for 18 genes to represent a variable with 9 fuzzy sets. This is obviously not desirable if there is a large number of such variables to optimise. In addition to not knowing how many sets to represent each variable with, one also has to consider the type of the sets and the form of the rules defining the (state, action) pair. In addition, any redundant information identified as a result of the evolution process is not removed from the process as it still remains in the coded structure.

Finally, reinforcement learning is described, and the relationship between RLs and EAs has been highlighted. Just as artificial neural networks are patterned after biological neural networks, reinforcement learning systems strive to emulate animal learning. Reinforcement learning combines elements of both supervised and unsupervised learning. Like supervised learning there is some training information available. However, an external teacher does not provide this. Instead, as in unsupervised learning, there is a built-in critic that provides the training information. In addition as in evolutionary algorithms, it works around an evaluation function. In fact the correlation between evolutionary algorithms and reinforcement learning systems, will be studied. However, unlike EAs, the evaluation function does not tell the agent how it should change its behaviour. The agent simply tries to maximise or minimise the performance measure of the evaluation function.

The study has also highlighted the lack of a learning algorithm for the fuzzy systems that is not handicapped by issues such as quality training data and learning online while system parameters change. These are the objectives of the next chapter.

Chapter 3

Systematic Approach to FLC Design Automation

Imagination is more important than knowledge. Knowledge is limited. Imagination encircles the world.

- Albert Einstein

The success of a neurofuzzy control system solving any given problem critically depends on the architecture of the network. Various attempts have been made in optimising its structure using genetic algorithm automated designs. In a regular genetic algorithm, however, a difficulty exists which lies in the encoding of the problem by highly fit gene combinations of a fixed-length. For the structure of the controller to be coded, the required linkage format is not exactly known and the chance of obtaining such a linkage in a random generation of coded chromosomes is slim. This chapter presents a new approach to structurally optimised designs of neurofuzzy controllers. Here, a messy genetic algorithm, whose main characteristic is variable length chromosomes, is used to obtain structurally optimised neurofuzzy controllers. Structural optimisation is regarded important before neural network based learning is switched into. Upon structure optimisation, a new neurofuzzy learning algorithm, based on the backpropagation algorithm, is developed for online or off-line learning. Both the evolutionary structure optimisation stage and the learning stage belong to the same model, complimenting each other, and is known as ENFLICT (Evolutionary NeuroFuzzy Learning Intelligent Control Technique). The resulting model is a new method for neurofuzzy control that is completely liberated in structure and feature.

3.1 The Design of FLC

In the previous chapters it was seen that the usefulness of fuzzy control is offset by the difficulties in tuning the controller, and also that there have been numerous attempts at optimising, automating and tuning FLCs through neural network and genetic algorithm hybrids.

It is evident from the literature review that there is no systematic procedure for achieving these tasks. There are presumptions, restrictions and complications associated with every model and approach discussed. Therefore, in this thesis, an autonomous fuzzy control paradigm is developed that is systematic, flexible and autonomous. In this chapter the process of achieving an autonomous controller is worked through systematically, and since a single solution is not available, several intelligent control components are required. The phases involved in designing the controller are be broken down to:

1) Definition of the system behaviour such as system constraints and desired response.
2) Initialising the controller with a priori information. Here the input and output variable operating regions are defined as are guestimated membership types, numbers and positions.
3) Off-line learning of fuzzy parameters and global learning of structure
4) Local or online learning and fine tuning of fuzzy parameters
5) Validation of the controller with conditions outwith the optimisation process

3.2 Self-Evolving Neurofuzzy Control

From the above experiment of §2.2, it is possible to conclude that learning can be applied to at least five different aspects of a fuzzy system

1) The definition of the fuzzy sets covering the universe of discourse of the variable, i.e. the shape and position of their membership functions. This point is addressed by many proposals within the fuzzy community. Most of the proposals consider a given rule base, either covering all the possible combinations of antecedent and consequent values, or given by some expert.
2) Which values of the variables in the rule antecedents are relevant for a given application? Many of the approaches so far proposed address this problem considering all the possible antecedent configurations, but this may lead to a non-minimal, large number of rules, some of which might not match any relevant world state, thus leading to problems of unreliable evaluation.
3) Which values of the variables in the rule consequences are relevant? Generally, this is obtained as a by-product of the learning activity, centred on the next aspect.
4) What is the best combination of antecedent and consequent values in a rule? In other words, what is the most appropriate action, given a situation? Notice that, with fuzzy rules, the action sent to the actuators does not depend only on one rule, but on many different rules triggering with different degrees.
5) What combination of rules best covers all the situations occurring?

Since the description of the system behaviour is system dependent, and the FLC is desired to learn from experience, the choice of the initial FLC variables becomes less important as long as they are reasonable. Therefore, in pursuit of the main aim of the thesis, which is to develop a method for constructing self-learning controllers based on fuzzy control, the construction of the model is begun by developing a neurofuzzy model that will be responsible for the learning and respond to system and environmental changes. The neurofuzzy model will henceforth be known as ENFLICT (Evolutionary NeuroFuzzy Learning Intelligent Control Technique). Thereafter, the process of structure optimisation by the messy genetic algorithm of this model will be described.

3.2.1 ENFLICT Network Architecture

The ENFLICT structure developed in this thesis also exhibits properties similar to those highlighted in §2.1. That is, it is an universal approximator based on a feedforward neural network with i inputs and j outputs. The threshold functions of the neurons are represented by fuzzy membership functions, and the ***MIN*** operation of input fuzzy variables carried out at each ***IF*** parts of fuzzy inference rules correspond to a product of input to the neuron and synaptic weights. The ***MAX*** operation is then used to obtain the inference value from ***THEN*** parts of these rules, corresponding to the input sum with a neuron. The model is depicted in Figure 3.1. Layers $\mathbf{L}_1$ and $\mathbf{L}_2$ represent the fuzzification process while layers $\mathbf{L}_3$ and $\mathbf{L}_4$ represent the inferencing mechanism with layer $\mathbf{L}_5$ equivalent to the defuzzification process. Since the network essentially represents a fuzzy logic controller mapping there are restrictions on how much the network can be adjusted in order to achieve the desired actions from the systems, e.g. the number of layers cannot be altered since this has direct relation to the inferencing mechanism. This limits the structural optimisation to the type of activation function of the neurons, the number of neurons per layer and the necessary links between adjacent layers. Therefore, the relevant parts of the network requiring optimisation are the shapes of the antecedent membership functions, the number of rules, the network connectivity and the consequent part, indicated by layers $\mathbf{L}_3$ and $\mathbf{L}_4$ as only these influence the action of the controller. The other parts on the network are kept constant.

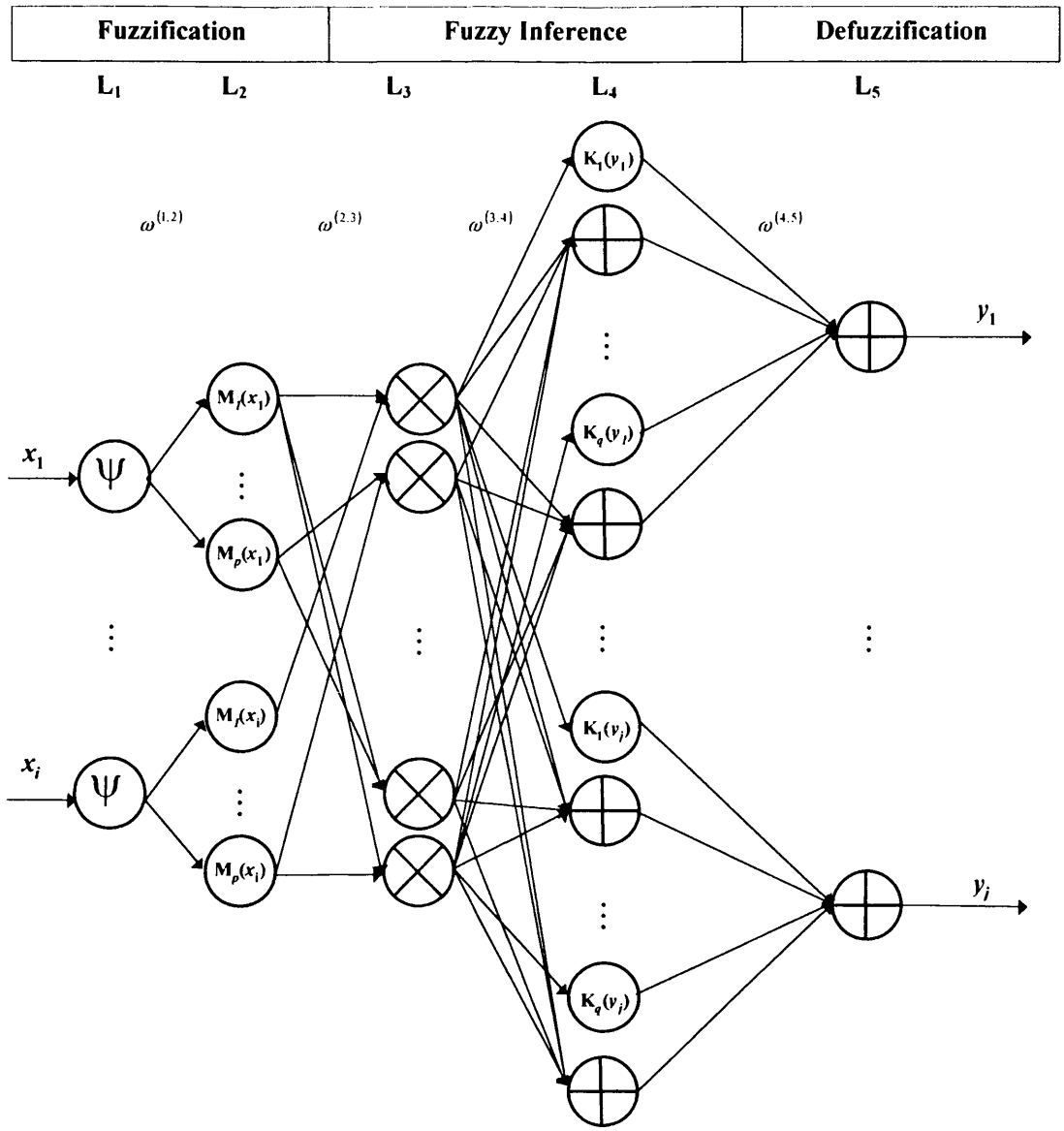


Figure 3.1 ENFLICT structure

Layer L_1: The function of this layer is simply to scale and map the input x_i to the corresponding fuzzy subspace represented by the neuron in layer L_2. The scaling is carried out to make the input lie in the interval [0,1]. Consequently, the universe of discourse for all inputs lies in the interval [0,1]. Therefore, the output of neurons of this layer does not connect to all the neurons of its adjacent layer, i.e.,

$$O_i^{(1)} = \psi(x_i) \quad i = 1,2,\ldots,n \tag{3.1}$$

where ψ is the scaling factor given by

$$\psi(x_i) = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)} \tag{3.2}$$

There are no link weights to adjust in this layer, hence all weights are unity, and the number of nodes is the same as the number of inputs.

$$\omega^{(1,2)} = 1 \tag{3.3}$$

Layer L_2: The nodes of this layer are labelled M-nodes. The output of each node depends on the type of activation (membership) function used to define each linguistic variable. The noticeable feature of each node is that each node represents a sub-network with one hidden layer whose nodes represent the parameters of the membership function. For example, Figure 3.2 shows a typical sub-network for an *M*-node representing a trapezoidal type membership function. The nodes of layer $L_{2\text{-}1}$ and $L_{2\text{-}2}$ have linearly saturated transfer functions, and the link weights of layer $L_{2\text{-}1}$ are unity. The link weights of layer $L_{2\text{-}2}$ represent the membership function parameter set. For the trapezoidal example, $\omega^{(2-2,2-3)} = \{a, b, c, d\}$, Figure 3.3 shows how these parameters map to the trapezoidal shape.

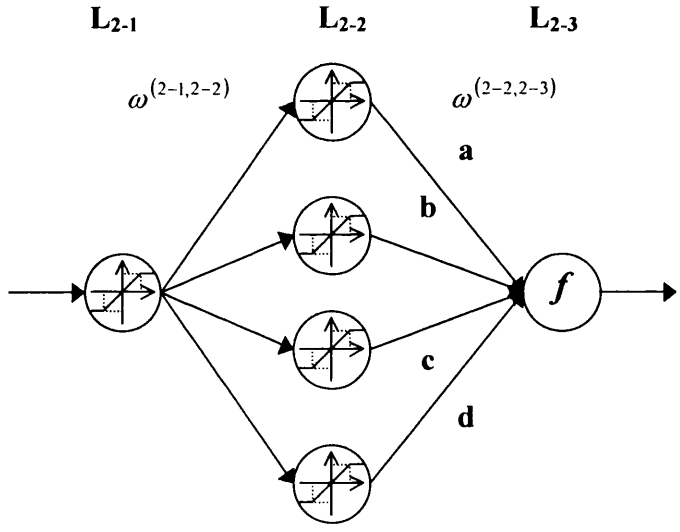


Figure 3.2 Layer 2 *M*-node sub-network

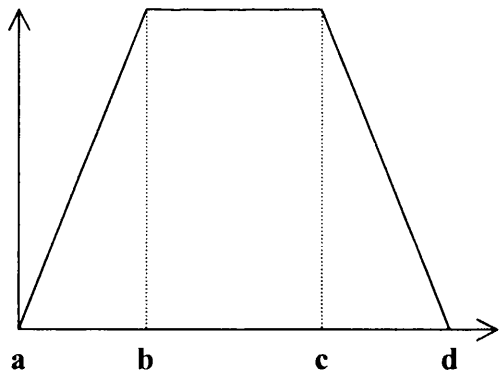


Figure 3.3 Trapezoidal membership function

The output of layer $\mathbf{L}_{2\text{-}3}$ and hence of layer $\mathbf{L}_2$ is

$$O^{(2)}\left(M_p(x_i)\right) = f\left(\omega^{(2\text{-}2,2\text{-}3)}\left(M_p(x_i)\right)\right) \quad p = 1,2, \ldots ,m \text{ and } i = 1,2, \ldots ,n \tag{3.4}$$

where f is the function defining the shape of the fuzzy set. This in essence gives the grade of membership of the p^{th} membership function of the i^{th} input variable. The link weights of layer $\mathbf{L}_2$ do not change and are of unit value.

$$\omega^{(2,3)} = 1 \tag{3.5}$$

Layers $\mathbf{L}_3$: This layer is part of the inferencing mechanism. For rules of the form

$\boldsymbol{R_l}$: **IF** (x_1 is $M_1(x_1)$) **AND** ... ... **AND** (x_i is $M_p(x_i)$) **THEN** (y_1 is $K_1(y_1)$) ... (y_j is $K_q(y_j)$)

where l is the rule number, M and K are fuzzy variables characterised by the activation functions. Every neuron in this layer then essentially performs the **AND** operation of the fuzzy inferencing mechanism using the product operator. There are as many neurons in this layer as there are rules. Each neuron output represents the firing strength of a rule.

$$O_l^{(3)} = \prod_i \mu_i(x_i) \quad i = 1,2, \ldots , n \tag{3.6}$$

where μ is the grade of the membership function activated by the fuzzy rule premise. Once again, the link weights of this layer do not change and are of unity.

$$\omega^{(3,4)} = 1 \tag{3.7}$$

Layer 4: The nodes of this layer are of two types. The ⊕ nodes resolve rules having the same consequence through the OR operation. The SUM operator is used for fuzzy OR, and there are as many ⊕ nodes as there are rules. The output of each ⊕ node is given by:

$$O_l^{(4,\oplus)} = \sum_l \left[\prod_i \mu_i(x_i) \right] \quad l = 1,2, \ldots, r \text{ and } i = 1,2, \ldots, n \tag{3.8}$$

where l is the rule number. The other type of node in this layer is the K-node. K-nodes represent a sub-network similar to the nodes of layer 2. Once again, there is one hidden layer. However, the hidden layer nodes represent the parameters used to define the output membership functions. For example, Figure 3.4 shows the sub-network for a K-node representing a trapezoidal membership function like Figure 3.3.

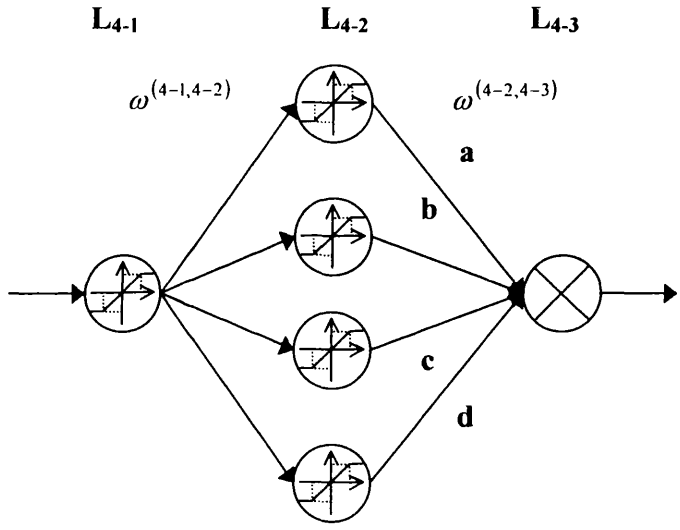


Figure 3.4 Layer 4 *K*-node sub-network

As with the M-node sub-networks, layers $\mathbf{L}_{4\text{-}1}$ and $\mathbf{L}_{4\text{-}2}$ have saturated linear transfer functions, and the link weights connecting $\mathbf{L}_{4\text{-}1}$ and $\mathbf{L}_{4\text{-}2}$ are unity. The link weights connecting $\mathbf{L}_{4\text{-}2}$ and $\mathbf{L}_{4\text{-}3}$ reflect the output membership function parameter sets. The output of $\mathbf{L}_{4\text{-}3}$, and hence $\mathbf{L}_4$ K-nodes is given by

$$O_l^{(4,K)} = \omega^{(4-2,4-3)}\left(K_q\left(y_j\right)\right) \cdot \prod_i \mu_i(x_i) \tag{3.9}$$

where l = 1,2, ... , r; i = 1,2, ..., n; j = 1,2, ..., m and $\omega^{(2-2,2-3)}$ represent the membership function parameter set, for example, $\omega^{(2-2,2-3)} = \{a,b,c,d\}$ for the trapezoidal case.

The advantage of using K-type nodes for each rule consequence, instead of using a single node to represent an output membership function, is that both Mamdani and Sugeno type FLCs may be designed. That is, it can represent continuous or discrete fuzzy sets. If desired, since there are as many M and K nodes as there are rules, each neuron can represent a different fuzzy set or fuzzy sets can be shared between rules trough grouping.

Since all the parameters relating to the shapes of membership are dealt with in the sub-network, there are no link weights of the main network that need adjusting. Hence the link weights of layer $\mathbf{L}_4$ are also unity.

$$\omega^{(4,5)} = 1 \tag{3.10}$$

Layer $\mathbf{L}_5$: This is the output layer and acts as the defuzzification process. This is dependent on the defuzzification type. Here the centroid defuzzifier is used because this method almost always displays smooth control behaviour. If a specific rule is predominant in a certain process, it may not be so dominant the next time. The centroid defuzzifier however will ensure that it will still have some influence regardless of how drastic the change in the environment.

$$O_j^{(5)} = \frac{\sum_l \left[\omega^{(4-2,4-3)}\left(K_q\left(y_j\right)\right) \cdot \prod_i \mu_i(x_i) \right]}{\sum_l \left[\prod_i \mu_i(x_i) \right]} \tag{3.11}$$

3.2.2 The Learning Algorithm

The learning algorithm resembles a backpropagation algorithm. First one needs to define an energy function which indicates how well the neurofuzzy controller is performing at meeting certain desired response or environment or condition. The error function employed is on the relative entropy function (Solla 1988) defined by.

$$E = \tfrac{1}{2}(1+\bar{y})\log\left(\frac{1+\bar{y}}{1+y}\right) + \tfrac{1}{2}(1+\bar{y})\log\left(\frac{1+\bar{y}}{1+y}\right) \tag{3.12}$$

where $\bar{y}$ is the desired response due to input x and y the actual. The function has an advantage over the standard quadratic error function in that it accelerates convergence on plains in the error landscape where the standard function could stick, and decelerates progress on sharp bends

of the cost surface. The function is defined for a single input pattern at any instance of time, but can also be adapted and used for training data in off-line batch mode.

The task is to minimise (3.12) for the given network structure. The energy function defined in (3.12) is the global energy function for the network indicating how well the whole network is performing. However, the difficulty lies with the fact that controllers based on such networks reflect only specific operating conditions. The other allied problem is that it also implies, particularly in batch learning, that one needs to know the input-output relations which the network tries to match through the learning rule. Obtaining accurate and quality training data for engineering systems is probably the primary difficulty because often it is not possible to do so or the algorithms producing the training data are not in general predictive and cannot truly represent the varying environment. Ideally, all one would like to provide is a reference input signal and one wants the system to follow this signal regardless of the states the system inputs go through. Therefore, one should not have to be concerned about producing input-output test patterns for the network to match, but simply some reference signal for the network to follow. In addition, one would also wish to take this further and want the network to be able to operate on-line where only a single input set is available at any instance of time and thereafter eliminate the input set after the network has learnt.

In order to achieve this and for the network to adapt to changing operating conditions, it is necessary first to ensure that the input pattern drawn at time t is independent of the previous input pattern, i.e. the input pattern to the network is provided at random. Second, one would need to ensure that changes in the operating conditions take place by small amounts. Third, a local energy function which takes account of such fluctuations in the operating conditions, needs to be introduced. Finally, static condition based networks have a learning algorithm with an asymptotically vanishing learning rate. Therefore, the algorithm has to be adjusted to take account of a non-vanishing learning rate.

Let all the weights and thresholds of the network be represented by vector w, then the networks weights are updated according to the rule

$$w(t) = w(t-1) + \eta\delta(t-1)\frac{\partial f}{\partial w} \tag{3.13}$$

where η is the learning rate, and

$$\delta = -\frac{\partial E}{\partial f} \tag{3.14}$$

As already stated, the operation of the learning rate is a major factor in the convergence and learning of the network. The larger this rate, the faster the response of the network to

changes in the environment. However, the negative side to this is that it causes large fluctuations around the local optimal of the vector $\boldsymbol{w}$, thus affecting the network's accuracy. On the other hand accuracy has to be compromised for generality if the network is to be truly adaptive and reflect any change in the environment. To address this issue, in the learning algorithm the learning rate is replaced by an annealing rate α which is defined by.

$$\alpha(t) = \frac{1}{t}\left[\alpha(t-1)\varepsilon(t-1) - sign(\varepsilon(t-1))\beta(t)\right] \tag{3.15}$$

where β is a variable containing the boundary information of the local weight vector, $\boldsymbol{w}$. Both β and the $\frac{1}{t}$ terms are used to prevent the annealing rate tending to zero as global energy decreases. As the network learns more and more, around a specific operating region of change in system parameters, so generality of the network will be affected as it fluctuates around the optimal value of $\boldsymbol{w}$. To take account of this, a local energy function, ε, is used to compensate for the loss of generality. The local energy function is defined as .

$$\varepsilon = \left\| w - \langle w \rangle \right\|^2 \tag{3.16}$$

where $\langle w \rangle$ indicates the expected value of $\boldsymbol{w}$ over all inputs, and in (3.15), the direction of change of the annealing rate is controlled through

$$sign(\varepsilon(t-1)) = \begin{cases} -1 & \text{if } \Delta E > 0 \\ +1 & \text{if } \Delta E < 0 \end{cases} \tag{3.17}$$

The learning rule (3.13) is now updated as follows

$$w(t) = w(t-1) + \alpha\delta(t-1)\frac{\partial f}{\partial \omega} \tag{3.18}$$

The simplicity of this learning method is that it requires no prior knowledge of any probability distributions or complex matrices such as the Hessian or Jacobian as proposed in other on-line learning algorithms (Berenji 92, Rattray and Saad 97). The other advantage of this method is that fuzzy parameters can be dealt with directly as is needed to at the fuzzification stage. As mentioned before, this rule need only be applied to weights of the membership functions of layers $\mathbf{L}_{2\text{-}2}$ and $\mathbf{L}_{4\text{-}2}$. The weights of other layers are of unit strength.

Now that the necessary components of the backpropagation algorithm for on–line and unsupervised learning have been identified and modified, it is possible to apply the algorithm and work backwards through the network.

Layer L_5: From the equations describing the behaviour of the network, it is clear that in the main network, there are actually no weights that need adjusting. Only when a sub-network node is encountered does the error need to be taken at the output of that network and the learning algorithm applied with the annealing rate. Other than that, the error only needs to be propagated backwards to the preceding layer.

$$\delta_j^{(5)} = -\frac{\partial E}{\partial \bar{y}_j} = -\frac{\partial}{\partial \bar{y}_j}\left[\frac{1+\bar{y}_j}{2}\log\left(\frac{1+\bar{y}_j}{1+y}\right)+\frac{1-\bar{y}_j}{2}\log\left(\frac{1-\bar{y}_j}{1-y}\right)\right] \tag{3.19}$$

since $\bar{y}_j = O_j^{(5)}$. Therefore, evaluating (3.19),

$$\delta_j^{(5)} = -\frac{1}{2}\left(\log\left(\frac{1+\bar{y}_j}{1+y}\right)-\log\left(\frac{1-\bar{y}_j}{1-y}\right)\right) = -\frac{1}{2}\log\left(\frac{1+\bar{y}_j}{1-\bar{y}_j}\cdot\frac{1-y}{1+y}\right) \tag{3.20}$$

Layer L_4: the layer does not have any weights that need updating. Therefore, the error is simply propagated backwards. However, there is still some computation involved. The weights of the K-node sub-network layer $L_{4\text{-}2}$ still need to be updated. Using the chain rule,

$$\frac{\partial E}{\partial \omega^{(4-2,4-3)}\left(K_q\left(\bar{y}_j\right)\right)} = \frac{\partial E}{\partial O_l^{(4,K)}}\frac{\partial O_l^{(4,K)}}{\partial \omega^{(4-2,4-3)}\left(K_q\left(\bar{y}_j\right)\right)} \tag{3.21}$$

$$\frac{\partial E}{\partial O_l^{(4,K)}} = \frac{\partial E}{\partial \bar{y}_j}\frac{\partial \bar{y}_j}{\partial O_l^{(4.K)}} \tag{3.22}$$

From (3.11),

$$\frac{\partial \bar{y}_j}{\partial O_l^{(4,K)}} = \frac{\partial}{\partial O_l^{(4.K)}}\left[\frac{\sum_l\left[\omega^{(4-2,4-3)}\left(K_q\left(y_j\right)\right)\cdot\prod_i \mu_i(x_i)\right]}{\sum_l\left[\prod_i \mu_i(x_i)\right]}\right] \tag{3.23}$$

$$\frac{\partial \bar{y}_j}{\partial O_l^{(4,K)}} = \frac{1}{\sum_l \left[\prod_i \mu_i(x_i) \right]} \tag{3.24}$$

hence,

$$\frac{\partial E}{\partial O_l^{(4,K)}} = -\delta_j^{(5)} \frac{1}{\sum_l \left[\prod_i \mu_i(x_i) \right]} \tag{3.25}$$

and then from (3.18), the weights of the hidden layer $\mathbf{L_{4\text{-}2}}$ are obtained from

$$\omega_{(t)}^{(4,K)} = \omega_{(t-1)}^{(4,K)} - \alpha \delta_j^{(5)} \frac{1}{\sum_l \left[\prod_i \mu_i(x_i) \right]} \frac{\partial O_l^{(4,K)}}{\partial \omega_{(t-1)}^{(4,K)}} \tag{3.26}$$

where

$$\omega^{(4,K)} = \omega^{(4-2,4-3)}\left(K_q\left(y_j\right)\right) \tag{3.27}$$

Since the error propagates through types of nodes at layer $\mathbf{L_4}$ the sum of the errors passed through each type of node is taken.

$$\delta_l^{(4)} = -\sum_j \frac{\partial E}{\partial O_l^{(4)}} = -\sum_j \left(\frac{\partial E}{\partial O_l^{(4,K)}} + \frac{\partial E}{\partial O_l^{(4,\oplus)}} \right) \tag{3.28}$$

$$\frac{\partial E}{\partial O_l^{(4,\oplus)}} = \frac{\partial E}{\partial \bar{y}_j} \frac{\partial \bar{y}_j}{\partial O_l^{(4,\oplus)}} \tag{3.29}$$

From (3.8), (3.9) and (3.11), it is clear that

$$\frac{\partial \bar{y}_j}{\partial O_l^{(4,\oplus)}} = \frac{\sum_l \left[\omega_l^{(4,K)} \cdot \prod_i \mu_i(x_i) \right]}{\left(\sum_l \left[\prod_i \mu_i(x_i) \right] \right)^2} \tag{3.30}$$

Using (3.25), (3.28) and (3.30), the error propagated to the preceding layer is

$$\delta_l^{(4)} = -\sum_j \left(\frac{-\delta_j^{(5)} \left(\sum_l \left[\prod_i \mu_i(x_i) \right] + \sum_l \left[\omega_l^{(4.K)} \cdot \prod_i \mu_i(x_i) \right] \right)}{\left(\sum_l \left[\prod_i \mu_i(x_i) \right] \right)^2} \right) \tag{3.31}$$

Layer L_3: Returning to the single type of nodes in this layer, the error to propagate to the preceding layer is derived as in the following:

$$\delta_l^{(3)} = -\sum_l \left[\frac{\partial E}{\partial O_l^{(3)}} \right] = -\sum_l \left[\frac{\partial E}{\partial \bar{y}_j} \frac{\partial E}{\partial O_l^{(4)}} \frac{\partial O_l^{(4)}}{\partial O_l^{(3)}} \right] = -\sum_l \left[\delta_l^{(4)} \frac{\partial O_l^{(4)}}{\partial O_l^{(3)}} \right] \tag{3.32}$$

From (3.6), (3.8) and (3.9)

$$\frac{\partial O_l^{(4)}}{\partial O_l^{(3)}} = \frac{\partial}{\partial O_l^{(3)}} \left(O_l^{(4,K)} + O_l^{(4,\oplus)} \right) = \omega_l^{(4,K)} + 1 \tag{3.33}$$

$$\delta_l^{(3)} = -\sum_l \left[\delta_l^{(4)} \left(\omega_l^{(4,K)} + 1 \right) \right] \tag{3.34}$$

Layer L_2: As with the *K*-nodes, the *M*-nodes are treated separately to adjust the parameters of its sub-network. Denote:

$$\omega^{(2,M)} = \omega^{(2-2,2-3)} \left(M_p(x_i) \right) \tag{3.35}$$

then by the chain rule,

$$\frac{\partial E}{\partial \omega_i^{(2,M)}} = -\sum_i \frac{\partial E}{\partial O_l^{(3)}} \frac{\partial O_l^{(3)}}{\partial O_i^{(2)}} \frac{\partial O_i^{(2)}}{\partial \omega_i^{(2,M)}} \tag{3.36}$$

then (3.4) and (3.6),

$$\frac{\partial O_l^{(3)}}{\partial O_i^{(2)}} = \prod_{i-1} \mu_i(x_i) \tag{3.37}$$

and,

$$\frac{\partial O_i^{(2)}}{\partial \omega_i^{(2,M)}} = \frac{\partial f_i^M}{\partial \omega_i^{(2,M)}} \tag{3.38}$$

where f_i^M is the function used to define the M^{th} membership function of the I^{th} input. Therefore,

$$\omega_{(t)}^{(2,M)} = \omega_{(t-1)}^{(2,M)} - \alpha \delta_l^{(3)} \left[\prod_{i-1} \mu_i(x_i) \right] \left[\frac{\partial f_i^M}{\partial \omega_{(t-1)}^{(2,M)}} \right] \tag{3.39}$$

There is no need to propagate the error back to layer $\mathbf{L_1}$ since there are no weights to adjust at that layer and there are no further sub-networks. Thus far the learning ability of the ENFLICT architecture has been discussed. In the next section a consideration is made as to how to obtain the structure necessary before any learning algorithm is applied.

3.2.3 Evolutionary Learning of Structure

The network topology optimisation is concerned with finding out the number of subspaces for each input variable and output variable and also the network connectivity or the rule structure. The regions of fuzzy subspaces are defined according to the information available about the plant to the operator. Where the operating regions are known, a fixed universe of discourse with varying size membership functions is used. When the operating region range is not so clear, fixed size membership functions with varying universe of discourse is used. The resulting network is one where the entire operating region is well covered with equally spaced overlapping membership functions, enabling smooth transition between states. However, such a network will only give coarse network performance without any fuzzy set tuning.

When optimising using mGA, each gene is a set of numbers that indicates the input/output (I/O) index, the neuron of the adjacent layer it connects to and the type of activation of the neuron (fuzzy set shape). Using the mechanism of the mGA, a candidate neurofuzzy controller system may be initialised, coded and decoded as described in example 3.1.

Example 3.1: Consider a 2-input and 1-output system. Before encoding the controller, it is first necessary to decide the maximum and minimum number of fuzzy subspaces desired to work with for each I/O domain for the controller to operate satisfactorily. These can be defined as 5 and 2 subspaces respectively for each domain. Then, the initial template would consist of 25 (5·5) rows where each row corresponds to a fuzzy rule. The initial population would have

strings of length 75 (number of domains·∏(maximum of each input domain)) maximum, and length 12 (number of domains·∏(minimum of each input domain)). Deciding that one wants to work only with gaussian and triangular membership shapes, these are assigned as:

Type 1: Gaussian
Type 2: Triangular
Type 3: Trapezoidal

Then, a typical gene may be encoded as follows:

$$g^{(i)} = S^{(1)} + S^{(2)} + S^{(3)} \tag{3.40}$$

where,

$$S^{(1)} = \text{ip}(r \cdot (d-1) + 1) \cdot 100 \tag{3.41}$$

$$S^{(2)} = \text{ip}(r \cdot (S^{(1)}_{D_{\max}} - S^{(1)}_{D_{\min}}) + S^{(1)}_{D_{\min}}) \cdot 10 \tag{3.42}$$

$$S^{(3)} = \text{ip}(r \cdot (S^{(1)}_{type} - 1) + 1) \tag{3.43}$$

where **ip**(·) indicates the *integer part* of a number, r is a uniformly distributed random number and d is the number of domains. $S_{\max}$ and $S_{\min}$ are the minimum and maximum subspaces of domain S. A typical chromosome is shown in Figure. 3.5.

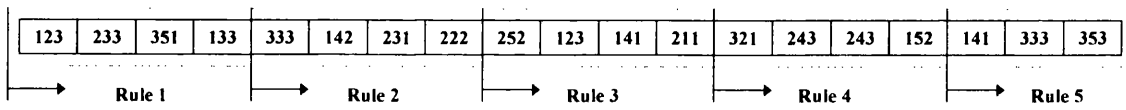


Figure 3.5 encoded ENFLICT structure

The noticeable feature is how the chromosome is divided up to form the rules. The division is made every d+1 gene from a left-to-right scan. Each gene decodes to a 3-tuple vector $[v^{(1)}\, v^{(2)}\, v^{(3)}]$ as follows:

$$v^{(1)} = \mathbf{ip}\left(\frac{g^i}{100}\right) \tag{3.44}$$

$$v^{(2)} = \mathbf{ip}\left(\frac{g^i - 100v^{(1)}}{10}\right) \tag{3.45}$$

$$v^{(3)} = \mathbf{ip}\left(g^i - 100v^{(1)} - 10v^{(2)}\right) \tag{3.46}$$

The first column indicates the input-output domains. The second column refers to the *M*-node or *K*-node of the adjacent layer to which the I/O variable connects to and the last column refers to the activation type of that node. For instance, the first gene of rule string 3, **(252)**, decodes to **[2 5 2]**. This interprets as

[2 5 2]: Input **2** connects to the **5**th *M*-node belonging to this domain, and this node has shape type **2**. Decoding the other genes,

(123) → **[1 2 3]**: Input **1** connects to the **2**nd *M*-node belonging to this domain, and this node has shape type **3**.

(141) → **[1 4 1]**: Input **1** connects to the **4**th *M*-node belonging to this domain, and this node has shape type **1**.

(211) → **[2 1 1]**: Input **2** connects to the **1**st *M*-node belonging to this domain, and this node has shape type **1**.

Notice there is no rule consequence, i.e. there is no reference to the output domain. In such a situation one would refer to the template and extract the appropriate consequent part. However, first it is necessary to resolve the premise. There is reference to both inputs more than once in this rule string. In such a case, the gene to appear first is used and the other rejected. The same precedence rule is also applied to resolve rules having the same premise but different consequent. The same rule also hold for resolving shape conflicts at both the *M*-nodes and *K*-nodes.

The parameter that introduces diversity into the GA so that it adapts to changes in the environment, is mutation. Although the standard mutation with a uniformly distributed probability of mutation every generation, performs well at a continuously changing environment, it fails even with a high mutation rate to track an environment which changes unexpectedly. Instead hyper-mutation can be used, which has the advantage of being adaptive (Grefenstette 92). The drawback of this type of mutation is that it does not perform well for large changes in the environment. Provided the changes are not large, the hyper-mutation will ensure the diversity needed in the GA even for discontinuous changes in the environment. The way that hyper-mutation works is that when the performance of the GA is poor or tends towards

poor, the mutation rate is set high with a non-uniform distribution. In all other cases, the mutation rate is set to a very low value with a uniform distribution.

3.3 Comparison of ENFLICT with other Neurofuzzy and Evolutionary-Neurofuzzy Approaches.

It is perhaps worthwhile comparing the functions and properties with other similar hybrids from literature aiming to achieve similar objectives. As was seen from the literature review in §1.2, the paths to obtaining optimal fuzzy control are many and fragmented. Since ENFLICT is a complete model, the only true comparisons can only be made against evolutionary-neurofuzzy hybrids. Although, a true comparison is not appropriate with simply neurofuzzy or evolutionary-fuzzy hybrids, there are some such hybrids that resemble individual components of ENFLICT. Therefore, comparisons between the complete ENFLICT model are made between other evolutionary-fuzzy and neurofuzzy models. The whole ENFLICT model is used for comparing both approaches because as the name suggests,, ENFLICT is a single unite and separating it inappropriate. These are summarised in Tables 3.1 and 3.2 respectively. The comparison is made on two areas: optimisation and learning. Here, in optimisation, one is concerned with using evolutionary techniques for representation of fuzzy systems in terms of structure, flexibility of rule base representation, tuning of the fuzzy sets, gene representation and ability to continuously learn and adapt to system and environmental changes.

3.3.1 Evolutionary Algorithm Optimisation

Table 3.1 compares several optimisation techniques. Early methods were concerned with simply tuning certain parts of the fuzzy system while keeping other parts fixed. One of the main reasons for doing this was that such schemes were represented by binary encoded chromosomes, as a result, the more information that is encoded, the longer the chromosomes would get (Karr 1991a, Cooper and Vidal 1993).

	ENFLICT	Hoffman & Pfister (1995)	Ng (1995)	Carse et al. (1996)	Takagi & Lee (1993)	Karr (1991)	Kinzel et al. (1994)	Herrera et al. (1995)	Thrift (1991)	Cooper & Vidal (1993)
Fuzzy System Optimisation										
Rule base construction	✓	✓	✓	✓	✓	×	✓	✓	✓	✓
Fuzzy set construction	✓	×	×	?	×	×	✓	×	×	×
Fuzzy set tuning	✓	×	✓	✓	✓	✓	✓	✓	✓	✓
Variable universe of discourse	✓	×	✓	×	×	×	×	×	×	×
[G]lobal or [L]ocal fuzzy sets	G,L	G	G	L	G	G	G	G	G	G
Fuzzy set type definition	✓	×	×	×	×	×	×	×	×	×
EA Representation										
Integer encoding	✓	✓	✓	✓	✓	×	✓	✓	✓	×
Variable length representation	✓	✓	×	✓	×	×	matrix	×	×	✓
Reproduction operator	cut and splice	cut and splice	crossover	user defined	crossover	crossover	crossover	crossover	crossover	user defined
Entropy cost function	✓	×	×	×	×	×	×	×	×	×
Single gene representation	✓	×	×	?	×	×	×	×	×	×

Table 3.1 Comparison of fuzzy system optimisation using evolutionary algorithms

Crucial to the success of EAs is the way the fitness function is represented or coded in genetic form and how the genes are represented. Generally, the parameters are encoded in two forms: *binary* and non-binary, more specifically *float*. To float or not to float has always been a subject of debate in the GA community. Traditional GA theory is based around binary coding because of its ease of manipulation and it is easier to prove theorem about them (Davis 1991). However, using floating point numbers to represent the genes has several advantages. They are more useful for higher order problem and problems requiring greater numerical precision. Although the drawbacks of binary coding can be overcome with the aid of various coding schemes such as gray coding (Srinivas 1994), the deciding and rearranging process is cumbersome and complex. Perhaps the most compelling reason for floating is that floating point representation allows a gene-variable direct mapping without the need for a complex decoding process. Michalewicz also showed through various tests other benefits of using floating point representation such as avoiding hamming cliffs, increased speed and less generation to population conformation (Michalewicz 1996).

ENFLICT presents a number of advantages over other methods. First, it uses a single chromosome to represent all information pertaining to a certain variable. The advantage of this is that the same kind of information can be presented in any gene, thus making the order of the genes irrelevant, making the reproduction procedure simple. It also means that systems of higher order can be represented without the length of the chromosome getting out of control. Of the other methods being compared, the only one that uses a similar method is that of Carse *et al* (Carse *et al* 1996), although it is difficult to judge because this has not been clearly specified in the literature. However, even if it does use a single gene representation, the order of the gene is important because the prior to crossover for reproduction, the genes have to be sorted according to the centres of the fuzzy sets. Also, if a single gene is used, then the process of encoding and decoding the parameter information is not specified. The process in ENFLICT is much simpler because every gene represents the same kind of information, that is, the index of the input or output variable; the number of the fuzzy sets that the input or output maps to and the type of that

fuzzy set. While ENFLICT does not tune the fuzzy set during the evolutionary learning stage, as most of the other methods being compared with, the feeling is that there is no need to fine tune it at this stage. Coarse tuning takes place by adding and removing the number of fuzzy subspaces for each variable. Fine-tuning is not carried out at this stage because first, fuzzy control is robust enough to sustain reasonable performance, and in engineering applications, often "reasonable" performance is sufficient. In addition, the problem with fine tuning fuzzy sets with evolutionary algorithms is that it is a one-time process only. Due to the amount of time that EAs take for a single cycle, it is almost impossible and impractical to continuously learn the sets due to changing environment. Therefore, local learning is preferred to global learning.

Another advantage that ENFLICT provides over any of the other methods is that it allows for identifying the type of fuzzy set to represent any specific subspace in the variable's universe of discourse. All the other techniques employ a single type of fuzzy sets, be it gaussian (Ng 1995), triangular (Karr 1991, Hoffman and Pfister 1995, Carse *et al* 1996) or trapezoidal (Herrera *et al* 1995a). While this may not be as important a factor as, for example, the structure of the rule base in the optimality of the fuzzy system, it nonetheless provides the versatility to work with any type or functions, and is not restricted to these three types of fuzzy sets. In addition, sometimes, mixed fuzzy sets may be desirable where the control surface varies between continuous and less continuous.

The other major factor influencing the convergence of the EA is the definition of the cost function. Most EAs have cost functions based on the standard quadric error function. ENFLICT uses a relative entropy function. Although, due to the randomness, EAs are resistant to getting trapped at local optima, they are not renowned for their speed. The purpose of the entropy function is to speed this learning process up and has been found to have a faster convergence than the standard quadratic error function.

More important to the stability of the fuzzy system is the number of rules and the rule base, that is the combination of premise and action for each rule. Most of the methods being compared with use a fixed length chromosome. The number of rules is found by introducing "don't care" entries in the chromosome (Thrift 1991, Takagi and Lee 1993, Ng 1995). This has a number of disadvantages. For example, first it implies some knowledge, on the part of the operator, of the system, and this is always not the case, specially for complex systems. The other problem is the *curse of dimensionality*. As the number of input and output variable increase, so does the size of the rule base. Do demonstrate this, a two-input-one-output system with 3 fuzzy subspaces per variable would yield a maximum of 9 rules. Now increase this to a three-input-one-output system with 3 fuzzy subspaces per variable. This would yield a maxim of 27 (3·3·3) rules. Thus the problem increases exponentially. To alleviate this, a number of works was carried out using variable length chromosome that grows and shrinks (Cooper and Vidal 1993, Hoffman and Pfister (1995), Carse *et al* 1996). However, ENFLICT is much more powerful and versatile than these methods. While Hoffman and Pfister's work, and was the

initial inspiration for the messy genetic algorithm representation developed here is innovative, they didn't go far enough and the method has a number of restrictions. First, the number of fuzzy subspaces is predefined, implying prior knowledge of the rule base structure, and second, because, the number of subspaces is fixed, there is no fuzzy set tuning by addition and subtraction of subspaces. The other differences are highlighted in Table 3.1 and have already been discussed. Both Carse *et al* (Carse *et al* 1996) and Cooper and Vidal (Cooper and Vidal 1993) also have variable chromosome representation and have to perform special reordering or sorting prior to reproduction, hence the order of the genes are important. This implies knowledge of the linkage format of the genes, where as with ENFLICT, there is no such presumption.

ENFLICT and the works of Hoffman and Pfister (Hoffman and Pfister) use the *cut* and *splice* operators to deal with reproduction. This was preferred to developing a new operator, as was the case in (Cooper and Vidal 1993) and (Carse *et al* 1996), because it's convergence properties has already been theoretically proven elsewhere (Goldberg 1989b, Goldberg 1990, Goldberg 1991) and is not the place in this thesis to prove GA properties. In (Cooper and Vidal 1993) and (Carse *et al* 1996), the operation of the new reproduction operator was explained, but no theoretical proof was provided. In (Carse *et al* 1996), the process is a little complicated because the operation of the modified crossover operator is different based on the number of inputs and outputs. This is not the case with the *cut* and *splice* operators.

Another difference between ENFLICT and all the other methods is that it allows for both global and local fuzzy set representation. Global fuzzy sets imply that rules and inputs and output variables share fuzzy sets, where as, local fuzzy sets are more appropriate for local learning and tuning. However, local fuzzy sets can present conflict problems of some dominating fuzzy sets engulfing other sets (Carse *et al* 1996) and some sort of precedence rule has to be applied. In ENFLICT, although, provision exists for both local and global fuzzy sets exist, global representation is preferred, as local learning with the neurofuzzy structure takes care of the local structure. The other time local fuzzy sets are useful is when designing a Sugeno-type controller.

As has already been stated, a main property of ENFLICT is its flexibility. With the exception Ng's work (Ng 1995), none of the methods allow for variable universe of discourse. Just as fixing the size of the rule base implies knowledge of the search space and control surface area, so does fixing the universe of discourse. Therefore, as can be seen, even simply comparing the optimisation process with existing methods, ENFLICT is much more powerful and versatile.

3.3.2 Neurofuzzy Learning

It is often claimed that during optimisation with EAs, fuzzy sets are learning (Case *et al* 1996). In the strictest sense this is incorrect. Learning implies a continuous process, with adaptation and tuning, and modification properties. At the extreme case, in the case of evolutionary-fuzzy system hybrids, it can be claimed that the fuzzy sets are learned for a specific set of environmental conditions. To reflect these learning properties, ENFLICT is a two-phase process, where the first phase involves global optimisation and the second phase is local learning using a neurofuzzy structure. To reiterate again, the two phases are inseparable because the first phase in essential to the success of the second phase. Table 3.2 compares only the learning properties of ENFLICT with other neurofuzzy "learning" methods.

	ENFLICT	Jang (1993)	Lin et al. (1991)	Kaur & Lin (1998)	Harris et al. (1996)	Bruske at al. (1993)	Khan (1993)	Spooner & Passino (1996)	Kim et al. (1993)
Fuzzy set tuning	✓	✓	✓	✓	✓	✓	✓	✓	✓
Rule modification	✓	×	×	✓	✓	✓	✓	✓	×
Mamdani controller	✓	×	✓	✓	×	✓	×	×	✓
Sugeno controller	✓	✓	×	×	×	×	×	✓	×
Non-symmetrical fuzzy sets	✓	✓	✓	✓	×	×	×	×	✓
Different inferencing mechanism	✓	×	×	×	×	×	×	×	×
Different defuzzification process	✓	×	×	×	×	×	×	×	×
Supervised learning	✓	✓	✓	✓	✓	✓	✓	✓	✓
Unsupervised learning	✓	×	×	×	✓	×	×	×	×
Online learning	✓	×	×	×	✓	✓	×	×	×
Local learning	✓	×	×	×	×	×	×	✓	✓
Model dependent	✓	×	✓	×	✓	✓	✓	✓	✓
Model independent	✓	✓	✓	✓	×	×	×	×	×

Table 3.2 Comparison of neurofuzzy networks

Neurofuzzy methods, as the name suggests, are combinations of fuzzy systems and neural networks, and the reason for such combinations arises from the properties of both. While fuzzy systems can be described using heuristics, easy to implement and interpret, it possesses no learning ability and requires detailed knowledge of the problem to be solved, although not necessarily a mathematical description of the system itself. In contrast, neural networks require very little *a priori* knowledge of the system, specifically developed for learning from patterns, work in parallel and function unsupervised. However, it is difficult to interpret the information that a neural network learns, and a priori knowledge is required of the derivative information of the functions that guide the learning process, before learning can be applied.

Neurofuzzy learning systems come in various forms. They vary from part neurofuzzy and part conventional control (Spooner and Passino 1996); through pure neural networks used to fine tune the performance of FLCs (Kim *et al* 1993, Khan and Venkatapuram 1993); to neural networks based on the functional equivalence of fuzzy systems and neural network using radial basis functions (Harris *et al* 1996); and fuzzy systems taking the form of neural networks, and

functioning as neural networks (Lin and Lee 1991, Jang 1993, Bruske *et al* 1993, Kaur and Lin 1998). ENFLICT is based on the last of these types, and is considered the best set-up because it allows directly representing, modifying, tuning and interpreting the structure.

One of the major objections to using neural networks, and subsequently neurofuzzy networks is the reliance of training data for learning the system. In order for the network to succeed in properly learning to control the system, such data has to be of sufficient quality and accuracy. Obtaining the accurate and quality training data for engineering systems is probably the primary difficulty because often it is not possible to do so or the algorithms producing the training data are not in general predictive and can not truly represent the real world. Despite this, most neurofuzzy hybrids are dependent on training data, and hence supervised. The advantage of this is that a model is not required because the network can learn directly from the data. Other than the difficulties of obtaining the quality data, the disadvantage of the supervised process is that it is off-line, and is not truly generalised to accurately control a real system. This is not necessarily the case with ENFLICT because, ENFLICT can use both training data and operate online.

One of the major reasons for people avoiding online unsupervised learning is the learning algorithm itself. Most neurofuzzy algorithms (including ENFLICT) is based on the backpropagation (BP) algorithm (Appendix A). In the BP algorithm, the major factor affecting the convergence of the network is the learning rate. The larger this rate, the faster the response of the network to changes in the environment, but the side effect being that it causes large fluctuations around the local optimal thus affecting the network's accuracy. The neurofuzzy methods that ENFLICT is being compared with are selected because they are representative of neurofuzzy structures in general, and ENFLICT is the only one of these methods that addresses this issue of learning rate so that online learning possible.

Looking at table 3.2, it can be seen that neurofuzzy networks are generally Sugeno-type (Jang 1993, Spooner and Passino 1996) or Mamdani Type (Lin and Lee 1991, Bruske *et al* 1993, Kaur and Lin 1998). Sugeno controllers do not have output variables represented by fuzzy sets, and are computationally easier to implement, and use the weighted average method to obtain the final output process (Takagi and Sugeno 1983). This is fine if enough information is available on the partitioning of the output variable and how the inputs relate to the output. It is therefore no surprise that most neurofuzzy networks are based on the Mamdani controller where the output is represented by fuzzy sets. ENFLICT does not chose either camp, instead allows for any type of controller. This is possible because the underlying network structure does not change, but instead, the fuzzy sets are tuned through sub-networks. Such fuzzy sets can either be local or global, and it is this format that allows one to design both type of controller.

By choosing to design a Mamdani type of controller, one has to then choose a type of inferencing mechanism and a type of defuzzification process. The most popular of all the

defuzzification processes in fuzzy control because it uses the whole of the output membership. It gives much smoother control behaviour than any other method and is also less sensitive to small perturbations (Sugeno 1985). However, other inferencing and defuzzification mechanisms are also available, and ENFLICT allows for any of these to be used without any modification to the structure of the network.

Neurofuzzy networks learn by adjusting the threshold function (or fuzzy sets), and generally a sigmoidal or gaussian type function is used. This means that the tuned fuzzy sets are symmetrical in shape since only two parameters can be tuned, namely the width and centres. As has been mentioned already, mixed fuzzy sets are desirable sometimes, and in contrast to the other methods, ENFLICT allows one to do this, and tune non-symmetrical fuzzy sets. For instance, the supports *a* and *d* of Figure 3.3 need not necessarily be equidistant from its centre position.

To summarise, from this comparative study, it can be seen that ENFLICT is very novel and original in its approach. It's flexibility liberates the design environment completely, and provides the operator and control designer every possible option to suit the environment under which it is to be operated. It combines two separate approaches of optimisation and learning and functions as a single model, complementing both phases.

3.4 Application to Coupled Non-linear Process Control

As an example, Figure B.1 shows the general construction of a non-linear coupled liquid level regulation system. The system consists of a container divided at the centre partition into two areas which represent the two tanks. A variable speed pump that supplies water to the first tank provides the fluid input. The actual flow rate is measured by a flow meter. The water of the second tank drains out via an adjustable tap into a tray, which provides the supply reservoir for the pump. The objective is to control the liquid level of tank 2 by means of the pump flow into the system. This sort of control problem typically occurs in the dairy, chemical or heat-balancing process industries where the fluid level in a storage tank or reaction vessel has to be controlled. Any variations in the upstream supply flow (Q_0) are filtered out by tank 1. The system dynamics are described by the equations

$$A\frac{dH_1}{dt} = Q_0 - C_{d1}a_1\sqrt{2g(H_1 - H_2)} \tag{3.47}$$

$$A\frac{dH_2}{dt} = C_{d1}a_1\sqrt{2g(H_1 - H_2)} - C_{d2}a_2\sqrt{2g(H_2 - d)} \quad . \tag{3.48}$$

where, $H_1(t)$ and $H_2(t)$ are the liquid levels of Tank 1 and Tank 2, respectively; $d(t)$ is also a pumped input but is used to test the rejection of disturbances when necessary; and $g = 9.81$ m s^{-2} is the gravitational constant. $A = 0.01$ m^2 is the cross-sectional area of both tanks; $h_0 = 0.03$ m is the minimum liquid level bounded by the height of the orifices. The derivations of the equations of motion for this system are given in Appendix B. The other parameters are variable and used to test the variations in the environment.

Flow rate $0 < Q_o < 5 \times 10^{-5}$m^3/s

Discharge constants: $0.3 < C_{d1}, C_{d2} < 0.6$

Cross sectional area of orifice 1: $30\text{x}10^{-6}$ m^2 $< a_1 < 50\text{x}10^{-6}$ m^2

Cross sectional area of orifice 2: $30\text{x}10^{-6}$ m^2 $< a_2 < 50\text{x}10^{-6}$ m^2

The controller and the plant set-up are as in Figure 3.6. During the learning process, a reference signal is fed into the system, and the ENFLICT network model takes as its input the error and the change of error. The output of the network is the control signal, u, and is fed into the plant. Θ is a vector of all the adjustable environment parameters, $\Theta = (C_{d1}, C_{d2}, a_1, a_2)^T$, and is adjusted at regular intervals to enable the network to adapt to the change and for the system to settle down to a stable level. For the learning algorithms and the mGA to follow the changes in the environment, only a small change in the environment is allowed, hence only one parameter is adjusted at any time.

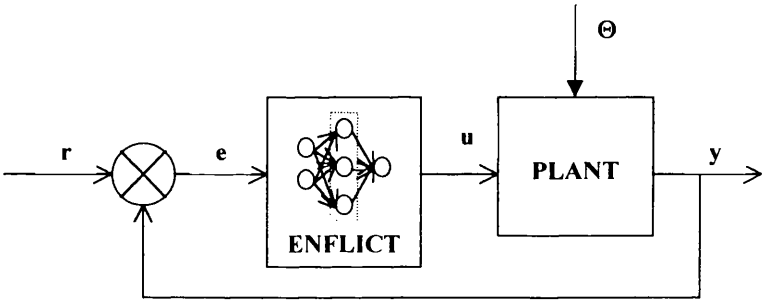


Figure 3.6. Control system set-up

The first step toward getting a self-learning controller is to learn the global structure of the network by mGA evolution. Before using mGA for the network optimisation, one must first to decide on how large the network should grow, and what type of activation to use. The universe of discourse to use for each input variable and output variable, is already known as they are all scaled to the interval [0,1]. The activation type used is the gaussian type described by equation (3.49). The advantage of this type is that it enables smooth transition between states and sub-regions. This is necessary since there is no information available on the way in which the environment changes.

$$Gaussian(x;\sigma,c) = e^{-\frac{(x-c)^2}{2\sigma^2}} \tag{3.49}$$

where c is the centre and σ the width of the activation function.

3.4.1 Global Structure Learning

To begin with, the network is initialised as a fully connected one with seven membership functions for each input variable and output variable. The choice on the number of memberships is entirely arbitrary. The inputs to the network are the error e between the set point and the output y, and the rate of change in error. A population size of 100 was used over 100 generations and 2 eras. At the start of the second era (i.e. at generation 55), the mGA was reinitialised and the population was reconstructed by filling half of it with the best members from the previous era and by randomly generating the other half. For the juxtapositional phase the cut and splice rates were set to 75% and 80% respectively. The hyper mutation had a baseline rate of 0.001 and an upper limit rate of 0.02 and the genes were mutated to a value not in the chromosome. The reference signal presented is a step-up-step-down signal. This is to ensure a greater degree of generality in the operating points in addition to the changes in the plant parameters. The cost function that the mGA tries to minimise is equation (3.12).

Figure 3.7(a) shows the response of the network to the step-up-step-down reference signal after the mGA has learned. Figure 3.7(b) shows the control action applied to achieve the response while Figure 3.7(c) shows the control surface which is equivalent to the network connectivity. Figure 3.7(d) shows the progress of the best-fit individual through the mGA learning. The reason for the sudden decrease in the quality of the individual's fitness is that at generation 55 the second era was executed where half of the old population members were carried forward to the next generation, and the other replaced by randomly generated population members.

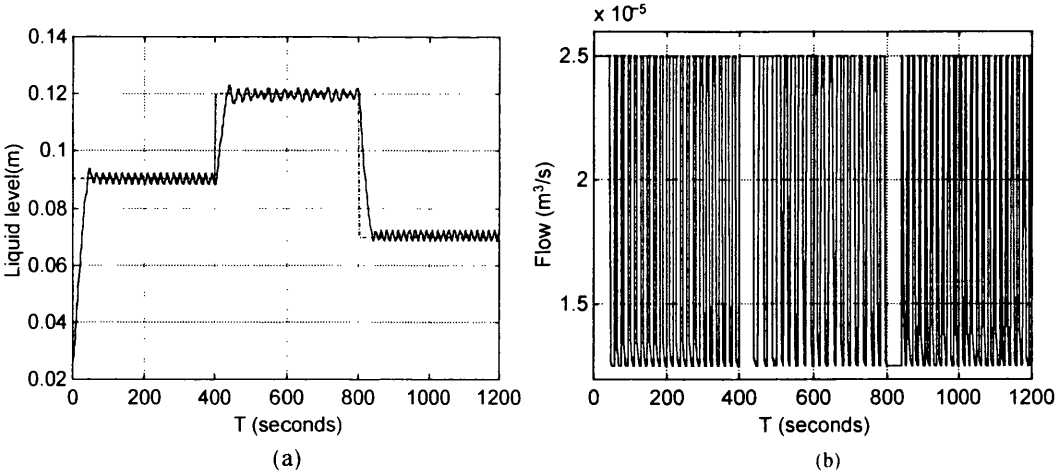


Figure 3.7 Network behaviour after mGA learning

(a) controller response, (b) control action

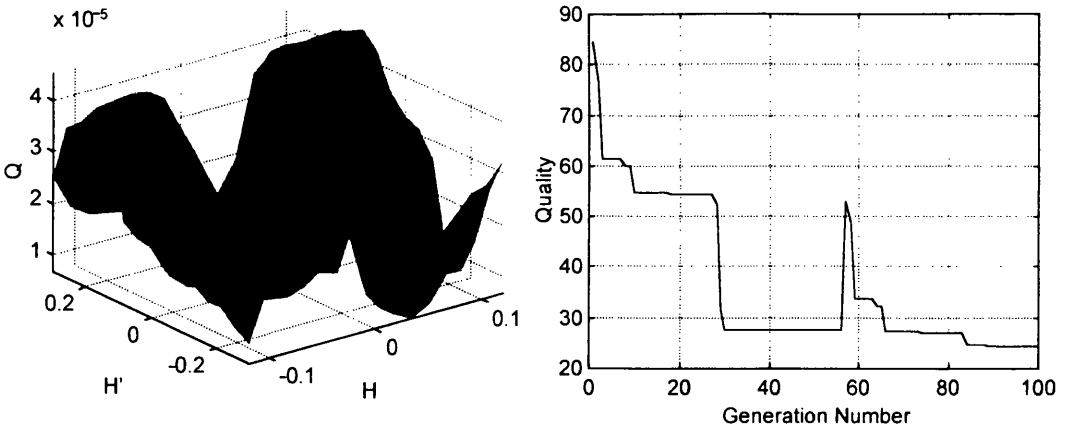


Figure 3.7 Network behaviour after mGA learning. (c) control surface, (d) fitness of best individual over 100 generations

Figure 3.8 displays the extracted equivalent membership functions of the network. It can be observed from the response of Figure 3.7(a) that the network is able to follow the reference signal. However, there is a lot of switching occurring at the operating points as reflected by the control action in Figure 3.7(b). The system being considered is a slow system, and hence it is possible under a simulation environment to achieve such a sudden change in the control action which otherwise may not be possible in real-time full-scale operation. This reinforces De Jong's statement about GAs wandering about near the global area (De Jong 1985). Hence further parameter tuning of the network is required.

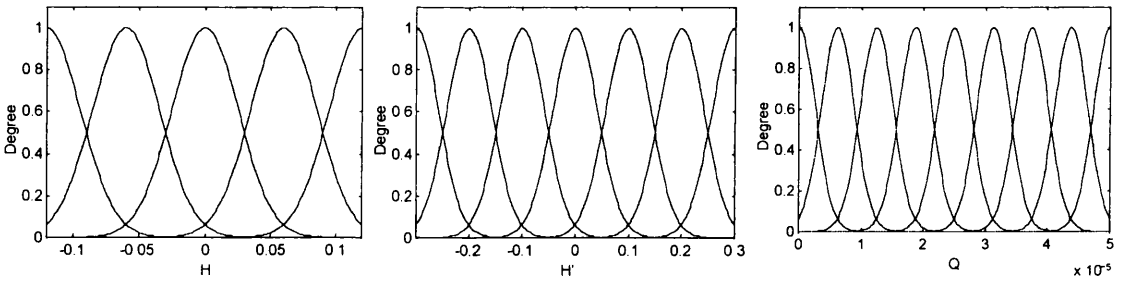


Figure 3.8 Extracted membership functions after mGA learning.

3.4.2 Parameter Pruning

While the mGA tries to obtain a network structure to give reasonable response, the network learning operates simply as a feedforward network. No parameter tuning of the weights is carried out. Only after the mGA phase has been completed is the parameter tuning carried out on the "best" network. The inputs to the network are held over an interval to enable the network to adapt to changes due to the previous inputs. The annealing rate is set to 0.95

with boundary [0.5,1.5]. The response of the controller due to various changes in the environment is shown in Figure 3.9-3.12. Figure 3.9 shows the response of the network as changes to the environment are introduced by varying the cross-sectional area of orifice 1. It is adjusted as follows.

$$a_1 = \begin{cases} 3.956e^{-5}m^2 & 0 < t < 200 \\ 3.056e^{-5}m^2 & 200 \le t < 600 \\ 4.512e^{-5}m^2 & 600 \le t < 1000 \\ 7.856e^{-5}m^2 & 1000 \le t \le 1200 \end{cases}$$

It is apparent from the response that the networks work very well with the learned parameters, Figure 3.9, except for the final third of the time period where the response remains rather coarse. During this period two incidences occur. First the operating point changes suddenly by a large amount and second, before the network can adapt to this sudden change, variations in the system parameters also occur by quite a large amount. This validates the statement that the learning algorithm is suitable only for small variations in the environment. The reason for this is that although the annealing rate adapts the network well, it can not change quickly enough to adapt the network for large environmental variations. Figure 3.10 shows the changes in the parameters of the network, taking place as it adapts to the variations in the environmental conditions. Each row of Figure 3.10 corresponds to a time period of 200 seconds in Figure 3.8. Further tests were carried out to validate the algorithm that it does indeed adapt to varying environmental conditions, and results are shown in Figure 3.11 and Figure 3.12. Figure 3.11 shows the network behaviour to a changing discharge coefficient while Figure 3.12 indicates the network behaviour to a sinusoidal reference signal. For Figure 3.12, no retraining of the network was necessary by mGA, and the same network topology as illustrated in Figure 3.7(c) was used.

To compare the performance of the ENFLICT network with other method, the learned network was tested against a PD controller and the ANFIS model (Jang 1993), Figure 3.13. The PD controller was manually tuned to follow the reference signal, and for ANFIS, training data was generated by simulating the model using the 4th order Runga-Kutta method. The ENFLICT response is that of the learned network (above) without any disturbance or parameter changes. As can be seen, all three controllers perform reasonably well. However, the PD controller was found to oscillate erratically when the step size changed, before recovering to the set point. It can also be seen that, with the same PD parameters, it is difficult to follow the reference properly. In contrast, the ENFLICT model, while having larger settling times, has smaller overshoots, more stable around the operating regions and follows the reference signal much more closely. Comparing ANFIS with ENFLICT, it can be seen that ANFIS is not as oscillatory as the PD controller, and appears to be just as good as the ENFLICT model.

However, this is only because it is using training data, while ENFLICT is performing without the training data. ENFLICT is learning by interacting with the system with one set of input data at any time instance. Hence, it takes longer to reach the desired set points. The point is that, in this case, description of the model was available, and it was possible to obtain training data for ANFIS. For instances where this is not possible, ENFLICT would be more suitable. Comparisons with varying system parameters are also difficult because modelling such changes in the process for generating training data would require system identification techniques. Whereas, as has been demonstrated in Figure 3.9-3.12, with ENFLICT, no such identification process is necessary.

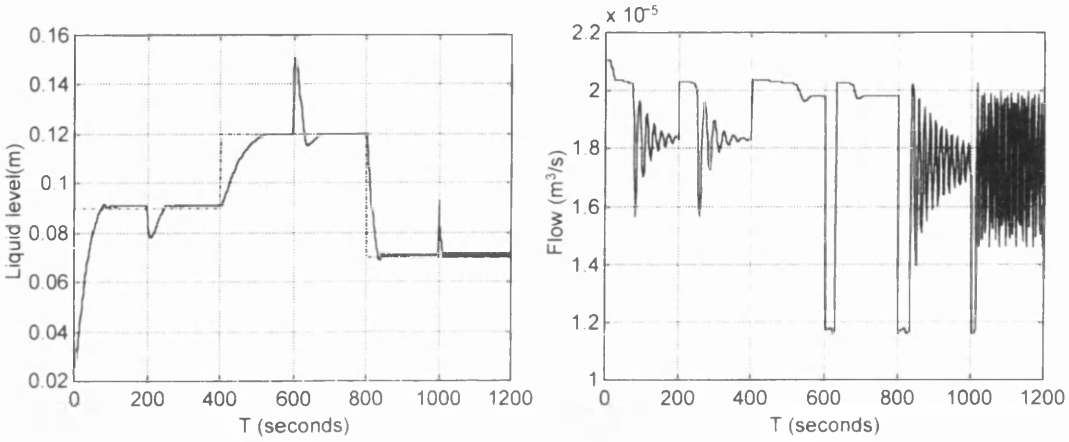


Figure 3.9 Network performance with changing cross sectional area of orifice 1.

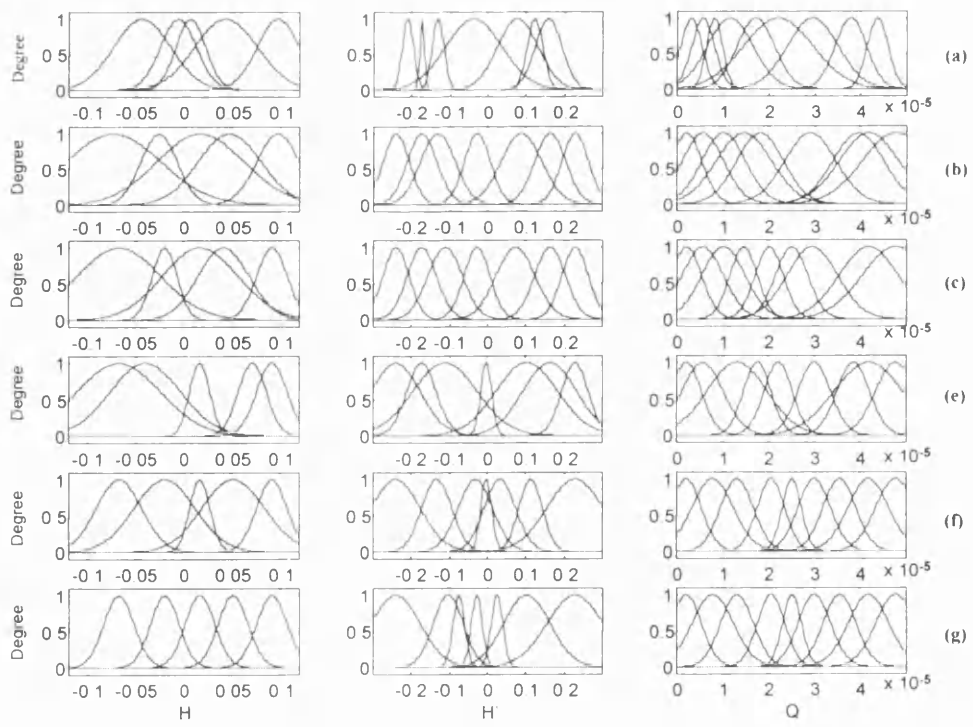


Figure 3.10 Network membership function adaptation to changing environmental conditions.

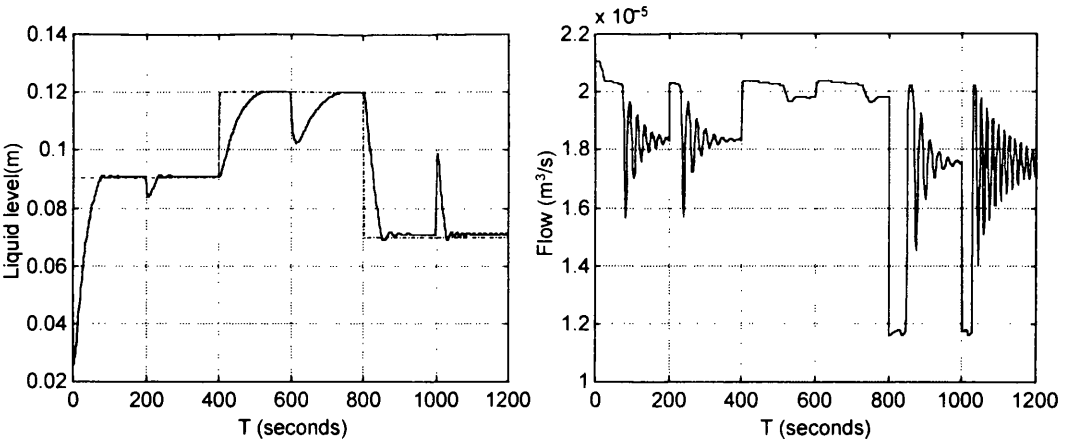


Figure 3.11 Network performance with changing discharge constant 1.

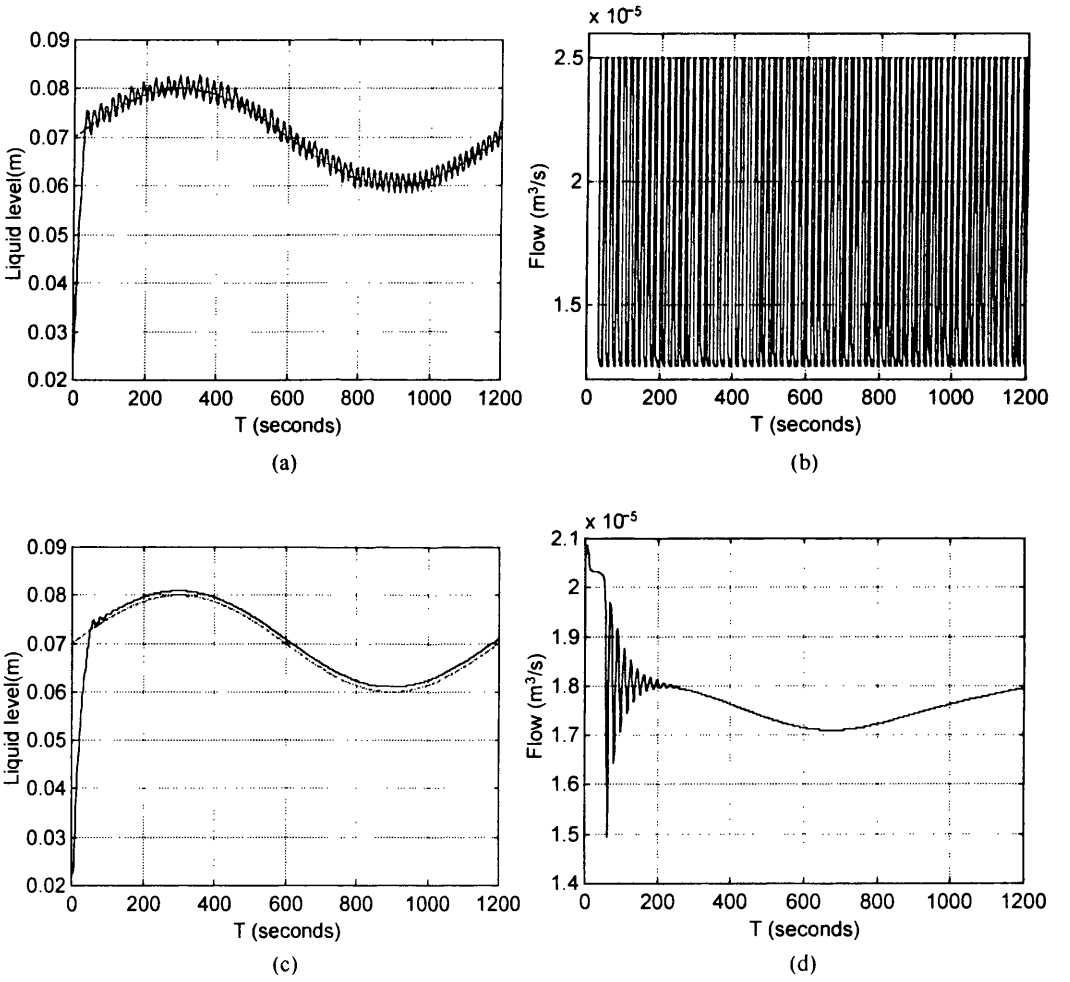


Figure 3.12 Network performance due to a sinusoidal reference signal (a) controller response before network parameters tuning (b) corresponding control action (c) controller response after parameters tuning (d) corresponding control action

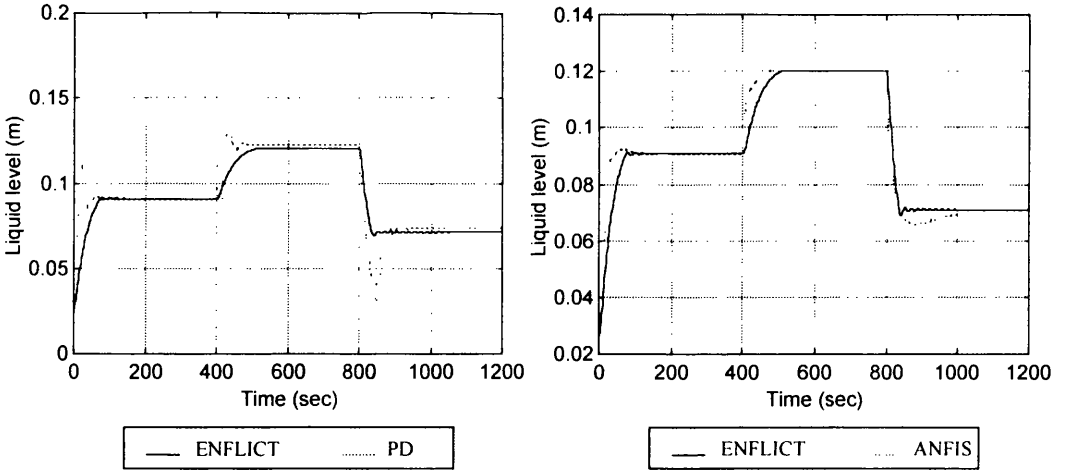


Figure 3.13 Comparison of ENFLICT with conventional PD and ANFIS structures

3.5 Case Study – Cart-Pole System

For the second case study, consider the example of the inverted pendulum in Appendix C, but with only one link. The problem is to control the motion of the cart along a horizontal line so that the pole will not fall down and will eventually stand at a desired angle. The problem is of particular interest because it resembles many practical engineering robot-arm like applications, such as ballistics, cranes, space shuttle arm, which depend on precision, stability and flexibility. There are four states associated with this model: cart position x, cart velocity v, pole angular position θ, and pole angular velocity ω. The pendulum is controlled by applying a force of varying magnitude to the cart's centre of mass. The simplified equations of motion are:

$$\dot{\theta} = \omega \tag{3.49}$$

$$\ddot{\theta} = \dot{\omega} = \frac{g\sin\theta + \cos\theta\left[\dfrac{-U - ml\dot{\theta}^2\sin\theta}{m+M}\right]}{l\left[\dfrac{4}{3} + \dfrac{m\cos^2\theta}{m+M}\right]} \tag{3.50}$$

$$\dot{x} = v \tag{3.51}$$

$$\ddot{x} = \frac{U + ml\left[\dot{\theta}^2\sin\theta - \ddot{\theta}\cos\theta\right]}{m+M} \tag{3.52}$$

For the tests, the following parameters were used:

g (acceleration due to gravity) = 9.81 m/sec^2

$0.1 \leq m$ (mass of pole) ≤ 1 kg

$0.5 \leq M$ (mass of cart) ≤ 2.0 kg

$0.5 \leq l$ (length of pole) ≤ 0.1 m

The first task is to control the pole and bring it to a vertical position by applying a force of varying magnitude to the centre of mass of the cart. The messy genetic optimisation was carried out and the fixed universe of discourse scheme was used. The initial template was defined as a fully connected network with seven memberships for each input and output domain, and the optimisation process was carried out on the neurofuzzy controller with with a single era of 100 generations and a population size of 200. The probability for cut and splice was set to 75% and 65% respectively, and mutation was set as 5%. The best network after the 100 generation was then mapped onto the ENFLICT structure for fine-tuning. Figure 3.14 shows the normalised error curve after mGA learning. The strength of the mGA is well demonstrated as near optima region is reached quite quickly.

For the online learning, the pendulum was reset with different settings. Figures 3.15 and 3.16 show the responses of the cart and pole system respectively for pole of length 0.5m, mass 0.3kg and a cart of mass 1kg, starting from the initial condition $\left(x, \theta, v, \dot{\theta}\right)$ (0.4, 5, 0, 0). It can be seen that the system is brought to its equilibrium point where the pole is balanced vertically and the cart is positioned to the middle of the track, i.e., the objectives have been reached.

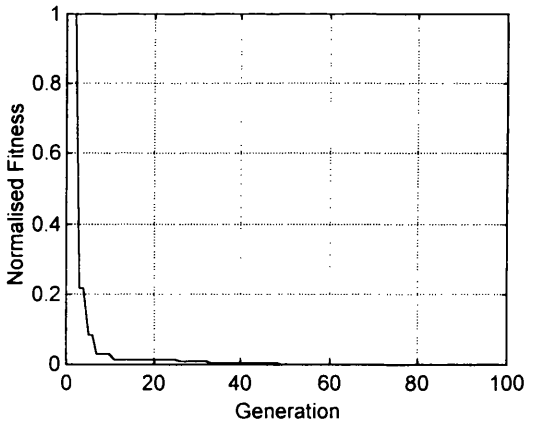
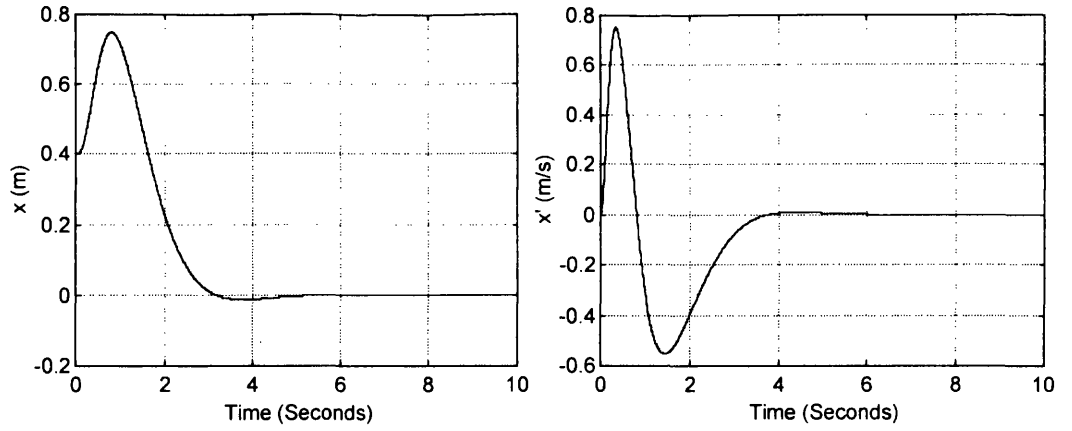
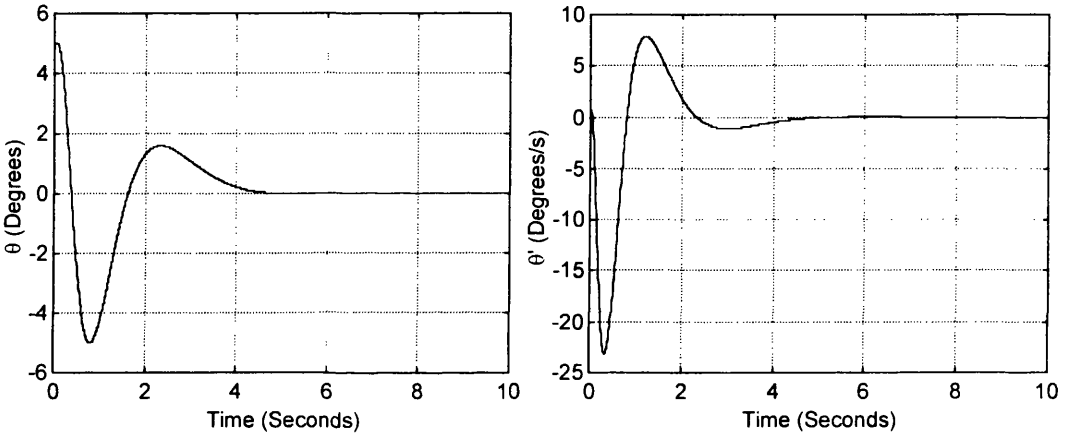


Figure 3.14 Normalised error measure



(a) (b)

Figure 3.15 (a) Position of cart for cart-pole system, (b) velocity of cart for cart-pole system



(a) (b)

Figure 3.16 (a) Angle of pole for cart-pole system, (b) Angular velocity of pole for cart-pole system

3.6 Summary and Discussion

In this chapter, a new model for designing intelligent controllers based on evolutionary and neurofuzzy technique has been developed. At the heart of this new model are portability, flexibility, usability, learning and evolution. While not based on any previously developed models elsewhere, the ENFLICT model can be applied to the same sort of problems as the Lin, ANFIS (Jang 1993), NeuFuz (Khan 1993), NN-FLC (Kaur and Lin 1998).

Flexibility implies freedom to choose from the various types of membership functions, the number of membership functions and the rule structure. Although the overall structure takes the form of a neural network, the underlying operation is simply a FLC with the ability to adapt to changing operating situations. Flexibility is achieved through the sophisticated messy genetic

algorithm. The variable length of the mGA chromosomes allows for growing network size (with certain preconditions), while the template and era operations allow for quick recovery of lost information. In order to accommodate the network connectivity, shape definitions and rule structures, the original mGA coding has been modified. In addition to the information that each gene now has, its structure is also different. In the developed model, the genes are represented by single integers. Of course, the resulting network obtained by the mGA lacks the ability to prune the network parameters, and hence its performance is coarse. Fine-tuning of the parameters is achieved through an on-line backpropagation learning algorithm.

Central to the working of the model is its learning properties. It was identified that the major drawback of other neurofuzzy models is that they are based on supervised batch learning. They depend on the operator providing quality network training data that can be expensive, unrepresentative and of poor quality. Despite various claims, such networks operate poorly outside the data they are trained with. The other difficulty that the ENFLICT model overcomes is that of on-line operation where the input pattern to the network is provided at random, and independent of the previous pattern. This has meant the need to modify the major parameter affecting the learning and convergence of the network. Therefore, the learning rate is replaced with an annealing rate, which is adaptive and reflects changes in the operating conditions so that the ENFLICT model operates on-line. The network structure optimisation needs to be carried out off-line due to the nature of genetic algorithms. In addition, the other drawback of this model is that it fails to operate satisfactorily to large changes in the operating conditions because the annealing rate can not adapt fast enough to reflect this change.

Finally, the ENFLICT model stands out from others for its usability. Usability can be tied up to flexibility because, although the backpropagation type learning is described for Mamdani type controllers, it can equally be applied for Sugeno type controllers. This is possible because the network operates at two levels. The main network remains unchanged and does not have any adjustable weights. However, by choosing the appropriate shape parameters at the various sub-networks, and selecting different defuzzification strategies, one can easily switch between the two types of controllers.

While ENFLICT is a two-phase model, the phases are inseparable, and comparisons have been made with optimisation techniques and neurofuzzy learning systems so that like for like comparisons can be made for flexibility, learning and optimisation. The main tasks and properties sought in optimisation are representation of fuzzy sets in terms of structure, flexibility of the rule base representation, tuning of the fuzzy sets, gene representation and ability to continuously learn and adapt to a changing environment. While many techniques exist for carrying out these tasks, most falls short from being a complete model that encompasses all properties. For example, binary representation of information as those of (Karr 1991) and (Cooper and Vidal 1994), mean that in order to keep the length of the string down, the amount

of information that can be encoded is restricted, whereas with integer encoding, a one-to-one gene-parameter representation can be used, and hence more information can be encoded.

ENFLICT takes the coding scheme further, but encoding all the information pertaining to a certain variable in one gene. This is very advantageous specially when using the variable length chromosomes. In contrast, other variable length representations require knowledge of the order of the gene and the linkage format because the order of the genes is important (Cooper and Vidal 1994, Carse *et al* 1996). Although in (Hoffman and Pfister 1995), the order of the gene is not important, a single integer representation was employed. Instead two integers were used to represent a single gene. This thus restricted the amount of information that could be encoded. For instance, in (Carse *et al* 1996), the genes consisted of only the centres and with of triangular fuzzy sets, and in (Hoffman and Pfister 1995) the genes made up only the rule base. In ENFLICT, the genes represented the type of fuzzy sets, the number of fuzzy sets and the mapping of the fuzzy sets. This means that high dimensional problems can be easily represented without any extra computational effort on the processor.

One of the main reasons for having the learning and optimisation phases separated is that, EAs are difficult to use for on-line processing, especially for fast and complex systems. Hence, most optimisations are applicable only for a set operating region for a set of parameters. For learning, universal approximators such as neural networks and fuzzy systems are better suited. By combining fuzzy systems to neural networks, it is possible to overcome the difficulties of both while retaining the advantages of both. The properties of neurofuzzy learning systems compared in §3.3.2 are symptomatic of neurofuzzy systems in general. That is, they are suitable either for off-line or online learning; applicable for either a Mamdani or Sugeno type controller; model dependent or independent based; use a single defuzzification and inferencing strategy. The reason for so many combinations being in existence is that as the authors of these works have shown, they are suitable for specific type of application. The aim of ENFLICT is not to be *ad hoc*, but to be suitable for any kind of system. Using the structure of ENFLICT presented, with its new annealing rate, it is possible to accomplish all the above tasks. The only limitation of ENFLICT when in the learning phase is that it works best for slow varying systems.

In the next chapters the ENFLICT model is taken a few steps further, and direct interaction with the environment through reinforcement learning is worked towards. This will lead to a completely model free, unsupervised and autonomous neurofuzzy control method for continuous time systems.

Chapter 4

Further Learning Through Reinforcements

Supposing is good, but finding out is better.
- Mark Twain in 'Eruption'

In this chapter reinforcement learning (RL) enhances the flexible evolutionary learning method for neurofuzzy control so that controllers can learn directly from the environment. The difficulty with on-line learning is that of knowing the exact future actions that will lead to global optimality. The standard dynamic programming based reinforcement learning uses an estimate of the value and advantage function hence does not actually produce global solutions. Evolutionary algorithms are identified to be a special type of reinforcement learning system. Evolutionary algorithms address the same set of problems as the dynamic programming RL. In the ENFLICT model unsupervised learning was preferentially chosen in order to overcome the undesirable property of having to rely on a teacher to provide correct answers to input patterns at the start of the problem. In unsupervised learning this is done by incorporating how to behave within the system. This is, in fact, undesirable too because it hinders the generality of the system. In this chapter, reinforcement learning techniques are used to overcome the difficulties associated with on-line learning. The model features are compared with other similar reinforcement learning methods and tested against some application.

4.1 The Need of Reinforcement Learning

Reinforcement learning is an approach to machine intelligence that combines unsupervised learning and dynamic programming to solve problems that neither of these disciplines are able to address alone (Barto *et al* 1983). Dynamic programming is a field of mathematics that has traditionally been used to solve problems of optimisation and control. A motivation for RL is that it is the primary learning method of biological systems. Animals learn and adapt daily with only reinforcement type error signals. Reinforcement learning studies therefore seek to capture similar capabilities in artificial systems. Just as artificial neural

networks are patterned after biological neural networks, RL systems strive to emulate animal learning.

Reinforcement learning is of interest in this chapter because first, the ENFLICT model in its present state requires a model of the system so that derivative information can be collected on its error measures so that the network parameters can be adjusted to minimise this error. However, often such derivative information is not available, and the actual description of the model is not always possible to obtain. Therefore, instead of using an error measure to indicate the performance of the network, RL can be used directly interact with the system to learn the network though reinforcements. The second reason for using RLs is that they are very similar to EAs in their semantics. However, RLs differ from EAs in two important ways: EAs search in a completely random fashion and hence ignore a lot of the information between state transitions, and secondly, EAs discards poor solutions in favour of good ones whereas RLs use this information in its decision making process. The similarities and differences of RL and EAs are highlighted in §2.5.

In the previous chapter a learning model was developed on the basis of neurofuzzy and genetic based methods. The system was based on off-line neurofuzzy structure optimisation and then on-line parameter tuning. The off-line learning was necessary because GA is not computationally viable for on-line implementation of fast and large systems. This leads on to the use of a simulation model of the system to be controlled, which has have discussed is not desirable as the simulated system cannot truly represent the actual system. There is, therefore, a need to modify the ENFLICT learning model so that not only are the network parameters fine-tuned, but also there is room for modifying the structure in terms of the control rule structure while on-line.

In the ENFLICT model unsupervised learning was preferentially chosen in order to overcome the undesirable property of having to rely on a teacher to provide correct answers to input patterns at the start of the problem. In unsupervised learning this is done by incorporating how to behave within the system. This is, in fact, undesirable too because it hinders the generality of the system.

In pursuit of the objective of this chapter an algorithm is used and extended that does not require a model to be given or learned, is fast and, perhaps most importantly, applicable to continuous systems. The RL algorithm is based on Harmon and Baird's (Harmon and Baird 1996) *advantage learning*, which is an enhancement of their *advantage updating* algorithm (Baird 1993) and requires the RL system to store only one type of information. This learning algorithm as it stands works only for the discontinuous case where a look-up table is used to guide the learning. The algorithm is therefore first extended for delayed reinforcement and on-line learning before applying to the ENFLICT model.

Finally, the procedures are compared with methods found in literature, and applied on some benchmark problems to demonstrate the stability and flexibility.

4.2 Continuous Time Reinforcement Advantage Learning

The goal of RL is to find a policy for selecting actions in a way that the selected sequence of actions will be optimal according to a certain evaluation (value) function. Since the actual outputs of the evaluation function involve future data not immediately available to the learning system, it leads to the fundamental question of almost all reinforcement learning research, i.e. how to devise an algorithm that will efficiently find the optimal value function?

Consider an RL-controller used to optimise a continuous system. Let $x(t)$ represent the system state at time t and $u(t)$ the action based on the state of the system and not on the previous ones. Suppose the system starts at $t = 0$, then $r(x(t), u(t))$ represents the reinforcement received by the system after performing action u at state x. Then the *value* is taken as the weighted sum of future reinforcements, which should be maximised for the system to perform optimally, and the value function for a given policy is defined as:

$$J(x) = \left\langle \sum_{t=0}^{\infty} \gamma^t r(x(t), u(t)) \middle| x(0) = x \right\rangle \tag{4.1}$$

where <·> is the expectation operator and γ the discount factor which represents the extent to which the learning system is concerned with future reinforcements of the control actions. The discount factor takes a value $0 \leq \gamma < 1$. The closer it is to 1, the greater the weight of future reinforcements, and $\gamma = 1$ implies infinite future weighting. The optimal value function $J^u(x)$ could then be calculated by:

$$J^u(x) : \max_u J(x) \forall x \tag{4.2}$$

4.2.1 Advantage Learning

Advantage learning is the RL algorithm used in this thesis to achieve the objective of learning through interaction. It is an algorithm that enhances *advantage updating* (Baird 93) by requiring only the learning update, and only the advantage function $A(x,u)$ needs to be stored. For each state-action pair (x,u), the advantage $A(x,u)$ is stored, representing the advantage of performing action u rather than the action currently considered best. The advantage in advantage learning is the sum of the value of the state plus the expected rate at which performing u increases the total discounted reinforcement. This advantage is so called because what is being considered is the advantage of receiving an increased overall weighted reinforcement by performing action u rather than the current action. The optimal advantage

function A^u can be defined in terms of the optimal value function J^u. The optimal value function $J^u(x)$ represents the true value of each state, and is defined as:

$$J^u(x) = \max_u A^u(x,u) \tag{4.3}$$

The advantage $A^u(x,u)$ for state x and action u is defined to be:

$$A^u(x,u) = J^u(x) + \frac{\langle R + \gamma^{\Delta t} J^u(x^+) \rangle - J^u(x)}{\Delta t} \tag{4.4}$$

where <·> represents the expected value over all possible results of performing action u in state x to receive immediate reinforcement R and to go to a next state x^+, and $\gamma^{\Delta t}$ is the discount factor per time step. For optimal actions the second term is zero, meaning the value of the action is also the value of the state; for sub-optimal actions the second term is negative, representing the degree of sub-optimality relative to the optimal action.

Advantage Learning and other RL algorithms such as Q-learning (Watkins 1989) and TD(0) (Sutton 1988) are generally classed as direct methods because they use a look-up table (Moore and Atkeson 1993) with a finite number of states. Each entry of the table has a state-action pair and various states are visited in any order and any number of times during each learning cycle. Convergence theories of most RL algorithms are based on such a finite look-up table structure. Although such methods are very fast and convergence for the finite space case is proven, the problem arises when the input space is continuous or infinite. In the look-up table, state and action spaces must be quantised into a finite number of cells. There are difficulties associated with determining an appropriate quantisation scheme to provide enough accuracy and low quantisation error. Many real-world applications are very large and very complex, and representing the states and actions is not a possibility because of the complications associated with trying to interpolate or identify values that are never seen. Look-up tables become impractical since the number of cells grows exponentially with the number of variables and geometrically with the number of quantisation levels, and convergence of the learning algorithm becomes extremely slow as the number of states and actions increases.

To overcome this, various function approximators such as neural networks (Anderson 1986, Thrun 1993, Gullapalli *et al* 1994) have been used because such approximators have generalisation properties and are able to perform reasonably steadily outside the input space in which they are trained. This is important because in systems with continuous state and action spaces, it is unlikely that the agent will experience exactly the same situation it has experienced before.

The direct method of advantage learning is implemented by combining with backpropagation based neural networks and shows convergence for a general neural network. The aim of the network is to adjust the network parameter to minimise an error function such as the mean squared error:

$$E = \tfrac{1}{2}\sum_{p}\left|\bar{y} - y\right|^2 \tag{4.5}$$

The equivalent Bellman mean square error is:

$$E = \tfrac{1}{2}\sum_{p}\left|\left\langle R + \gamma J(x^+)\right\rangle - J(x)\right|^2 \tag{4.6}$$

where for input x, the output of the network is $J(x)$ and the desired output is $\left\langle R + \gamma J(x^+)\right\rangle$. Between each transition from state x to x^+, the weights are updated according to

$$\Delta W = -\eta\frac{\partial E}{\partial W} = -\eta\left[R + \gamma J(x^+) - J(x)\right]\left[\frac{\partial}{\partial W}\gamma J(x^+)\right] \tag{4.7}$$

Although this method is more stable than the look-up table method, and performing gradient descent on the mean squared Bellman residual is guaranteed to converge to a local minimum, the method has a number of drawbacks. The first is that it is suitable only for off-line learning such as batch processing learning where the number of states to be learned is finite. The other noticeable feature is that it is an immediate RL algorithm.

4.2.2 Delayed Rewards

In an immediate RL, the agent receives reinforcements immediately after performing action u at state x. While this is desirable in some situations such as a robot trying to navigate a room, and it provides a lot of information, it is not possible in other situations. Consider the case of a surface-to-air missile control system. In such cases immediate reward is of little use because the performance measure is constantly changing. Instead it is more interesting to look at the reward or punishment several steps later such as at the point of impact between the missile and its target. This is referred to as learning with delayed reinforcements. Delayed RL is also very appropriate for situations where knowledge of the environment is incomplete or

unavailable. In other words, it is suitable for on-line learning where the environment could be very large and complex.

In delayed RL, the consequences of long-term compared with short term actions are adjusted by the discount factor γ. Recall that the discount factor γ lies between $0 < \gamma < 1$. The closer the discount factor is to 0 the more immediate the reinforcement. Unlike the immediate reinforcement method where each action-state is usually locally optimal, the delayed RL does not perform optimally for each state transition. Instead it is said that the RL system operates optimally on average.

4.3 Gradient Descent Delayed Advantage Reinforcement Learning

Having argued the benefits for gradient descent RL and delayed RL, the advantage reinforcement learning is extended for delayed RL. Immediate reinforcement is appealing because there is available the immediate information about the goodness or otherwise of taking an action at a state. This is not possible in delayed RL, and to achieve some sort of local optimality, an estimate of the optimal advantage function is used. Now, if an action u is taken at state x resulting in the next state x^+, then estimate $\hat{A}$ of the optimal A^* is taken as the immediate reinforcement of (x, a), and a good locally optimal policy is obtained. Now, how to obtain an estimate of A^u?

Identifying what is good or bad for each action is the problem that makes reinforcement learning difficult. In other words, the goal is to find an estimate, $\hat{J}(.;W)$, of J^u (.) in (4.3), where W is the parameter set of the neural network. A good estimation of J^u (.) is important since it could be used to check the optimality of the policy u and if necessary adapt it to get a better policy. Let n represent the number of time-steps elapsed after the system was in state x, and for brevity $r(t)$ instead of $r(x(t), u(x(t))$. Let an estimate of (4.1) be defined based on geometrically averaging $J^u_{(n)}(x)$. J^u_λ is defined as

$$J^u_\lambda(x) = (1-\lambda)\left[\sum_{n=1}^{\infty} \lambda^{n-1} J^u_{(n)}(x)\right] \tag{4.8}$$

with (1-λ) being a normalising term, and $0 \le \lambda \le 1$. Notice that the term $J^u_{(n)}(x)$ with smaller values for n is weighted more in the averaging process. This makes sense since the terms with a large n rely more heavily on future data and therefore should be weighted less in the average. Now define an n-step truncation of the sum in (4.1) as

$$J^u_{(n)}(x) = \sum_{r=0}^{n-1} \gamma^r r(\tau) \tag{4.9}$$

Since $r(0) = r(x,u(x))$, (4.8) can be rewritten recursively as

$$J^u_\lambda(x) = r(x,u(x)) + \gamma(1-\lambda)\hat{J}(x(1);W) + \gamma\lambda J^u_{(\lambda)}(x(1)) \tag{4.10}$$

with $x(1)$ being the system state one time-step after x. Using (4.10):

$$\lambda = 0 \rightarrow J^u_0(x) = r(x,u(x)) + \gamma\hat{J}(x(1);W) = J^u_1(x)$$
$$\lambda = 1 \rightarrow J^u_1(x) = r(x,u(x)) + \gamma J^u_1(x(1)) = J^u_\infty(x)$$

In other words, in order to calculate the discounted sum J^u, J^u_0 makes use of the immediate cost within one time-step plus the approximation of the rest of the sum. J^u_1, on the other hand, relies only on the actual costs to achieve the same goal. This learning method using J^u_λ is called TD(λ), with TD being the short form for temporal difference (Sutton 1988). Let a learning rule now be defined using J^u_λ:

$$\hat{J}(x;W) := \hat{J}(x;W) + \alpha\left[J^u_\lambda(x) - \hat{J}(x;W)\right] \tag{4.11}$$

where $\hat{J}$ is an approximation of J^u. To be able to use this rule, $J^u_\lambda(x)$ has to be calculated on-line without requiring a system model. As was seen in (4.1), the evaluation function is defined as the discounted sum of the future costs. The relation between two consecutive evaluations could easily be derived as:

$$J^u(x) = r + \gamma J^u(x(1)) \tag{4.12}$$

with $x(1)$ being the system state one time-step after x. However, the same relation should also hold for the predictions of $\hat{J}$ if it is a good approximation of J^u. If that is not the case then the difference between these predictions could be used to adapt $\hat{J}$. Now, $\varepsilon(.)$ is defined as the temporal difference between two successive predictions of the evaluation function

$$\varepsilon(x) = r(x,u(x)) + \gamma \hat{J}(x(1);W) - \hat{J}(x;W) \tag{4.13}$$

$\varepsilon(.)$ can be calculated using the temporal sequence of data available in each time step - hence the name temporal difference learning. The error used in the learning rule in (4.11) is a weighted sum of the temporal differences computed at each of the visited states.

$$J_\lambda^u(x) - \hat{J}(x;W) = \varepsilon(x) + (\gamma\lambda)\varepsilon(x(1)) + (\gamma\lambda)^2 \varepsilon(x(2)) + \ldots \tag{4.14}$$

Temporal differences are weighted exponentially with the earlier ones weighted more. Still, this value could not be used in the learning rule (4.11) since the calculation of all the terms except the first one involves data only available in the future. There are different ways to deal with this problem:

In (4.14) the terms on the right hand side could be truncated to select only the first N terms. This means that each term is calculated as the required information becomes available. The estimator $\hat{J}$ stays unchanged for N time-steps until the required error is accumulated, after which it will be used to update $\hat{J}$ using (4.11).

Alternatively, the effects of the temporal difference could be included as and when they occur in time. This can be implemented through the use of an *eligibility trace*, $e(x, t)$ for each visit, and using the rule at time t (Klopf 1988, Watkins 1989):

$$\hat{J}(x;W) = \hat{J}(x;W) + \eta e(x,t)\varepsilon(x(t)) \forall x \tag{4.15}$$

where the eligibility trace is adapted according to

$$e(x,t) = \begin{cases} 0 \text{ if } x \text{ is not seen} \\ \gamma\lambda e(x,t-1) \text{ if } x(t) \neq x \\ 1 + \gamma\lambda e(x,t-1) \text{ if } x(t) = x \end{cases} \tag{4.16}$$

Similarly the estimate for A^* and a learning rule are obtained,

$$\hat{A}(x,u;W) = \hat{A}(x,u;W) + \eta e_A(x,u,t)\varepsilon_A(x(t),u(t)) \forall (x,u) \tag{4.17}$$

where the trace is given by:

$$e_A(x,t)=\begin{cases} 0 \text{ if } (x,u) \text{ is not seen} \\ \gamma\lambda e_A(x,t-1) \text{ if } (x(t),u(t)) \neq (x,u) \\ 1+\gamma\lambda e_A(x,t-1) \text{ if } (x(t),u(t)) = (x,u) \end{cases} \tag{4.18}$$

and the temporal difference

$$\varepsilon_A = r(x,u)+\gamma\hat{A}(x_1,J^*(x_1);W)-\hat{A}(x,u;W) \tag{4.19}$$

The condition for using this procedure is that the trace be reset to zero if more than one trial is carried out, and that it is only implemented with connectionist methods such as backpropagation neural networks.

4.4 Application of Modified RL to Ship Control Regulation

The problem of manoeuvring a ship is challenging and of considerable interest because of the complexity in obtaining an accurate dynamic model. Various external forces such as wave motion and wind effects, allied with the coupled behaviour of the navigation, steering and auto pilot systems, make the control task very difficult. In this example the only point of interest is the design of a controller for regulating a cargo ship heading at a desired angle. A fuller description of the problem is given in (Åström and Källström 1976) and is summarised in Appendix D. It is also listed as IFAC benchmark problem number 89-08.

For straight-line motion the model of the ship under constant velocity is described as

$$\dot{x} = Ax + Bu \tag{4.20}$$

$$y = cx \tag{4.21}$$

where $x \in R^3$, $u \in R^1$, $y \in R^1$ are given as follows:

u = rudder angle
y = heading angle of ship
x_1 = sway velocity of ship
x_2 = turning yaw rate
x_3 = heading angle of ship

and the structure of A, B and C is given by

$$A = \begin{pmatrix} -0.895 & -0.286 & 0 \\ -4.367 & -0.918 & 0 \\ 0 & 1 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 0.108 \\ -0.918 \\ 0 \end{pmatrix} \qquad C = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$$

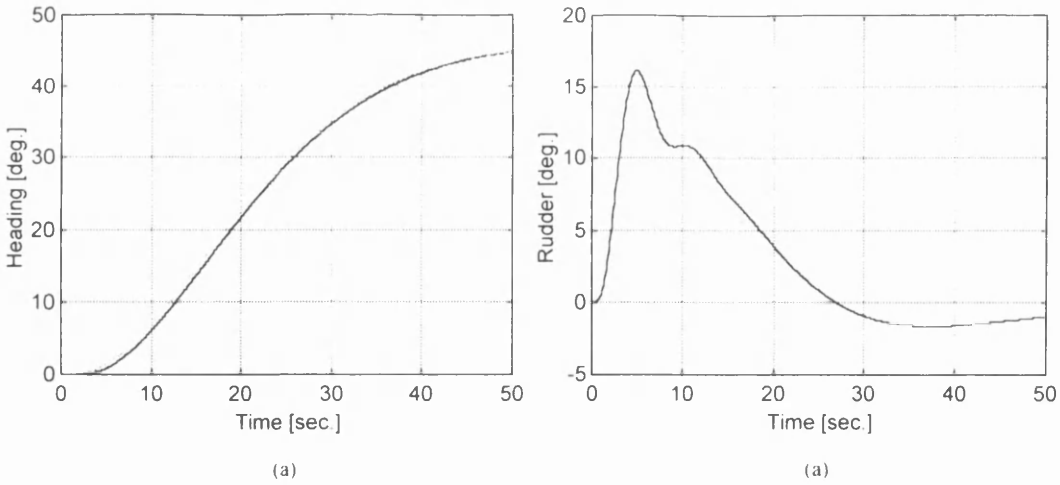
The objective is to find a controller of the system to control and regulate the heading angle of the ship to a desired angle of 12° such that no overshoot occurs for the heading angle, while the rudder motion is constrained by:

$$\|u\| < 40^o$$

The reinforcement function is defined as the difference between the actual heading and the desired angle. The amount of reinforcement received as a result of each state-action operation is inversely proportional to the amount by which the ship is away from the desired angle, with the amount overshot being penalised more:

$$r = \begin{cases} \dfrac{ref}{1-e} & \text{if no overshoot} \\ ref & \text{if desired} \\ \dfrac{ref}{1-5e} & \text{if overshoot} \end{cases} \tag{4.22}$$

where e is the difference between the desired and the actual heading angle, and ref is the desired reference that should be followed. The function approximator used to approximate the advantage function is a simple neural network with a single hidden layer. There are neurons in the hidden layer and each neuron has a sigmoidal activation function. The network is fully connected with three inputs and one output. The standard backpropagation learning algorithm is used to update the network parameters, error is minimised according to (4.6), and the weights updated according to (4.7). However, instead of using J, the estimate of the advantage function $\hat{A}$ needs to be used, where $\hat{A}$ is as (4.17). The trials are generated using the 4th order Runga-Kuta algorithm. The experiment was carried out over 200 trials where each trial consisted of 200 steps, and each trial terminates when overshoot occurs. Figure 4.1 shows the behaviour of a cargo ship of length 160m with a forward speed of 10 ms^{-1} required to follow a path of 45° to the horizontal. As can be seen the ship has learned to follow this path correctly. Figure 4.2 shows the learning curve of the network. After a slow start when the exploration space is large, the ship learns quite quickly to follow the objectives set out.



(a) (a)

Figure 4.1(a) Ship heading with respect to reference and (b) Rudder motion

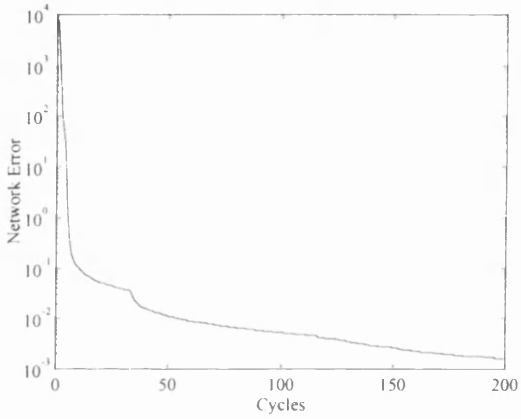


Figure 4.2 Performance measure at each learning cycle

4.5 Evolutionary Neurofuzzy Reinforcement Learning

In Chapter 3, the network structure was learned off-line through the messy genetic algorithm. Thereafter, the structure remained fixed while the network parameters were learned. In this section, a two-phase learning algorithm is developed where network structure is learned both on and off-line based on the dynamic programming and evolutionary reinforcement learning methods. The algorithm is summarised in Algorithm 4.1, and illustrated in Figure 4.3

Algorithm 4.1 Evolutionary RL of neurofuzzy networks

Stage 1: Off-line structural optimisation

1. Identify inputs and outputs
2. Identify approximate fuzzy subspaces for inputs variables
3. Obtain an approximate fuzzy rule base and network mapping
4. Optimise fuzzy subspaces and structure of the network using mGA

Stage 2: On-line network weight tuning

1. Start with the off-line network to obtain controlled actions
2. While the system is successful within limits apply new input states to the network
3. Record time to failure to reward and reinforce the best networks using Advantage Learning

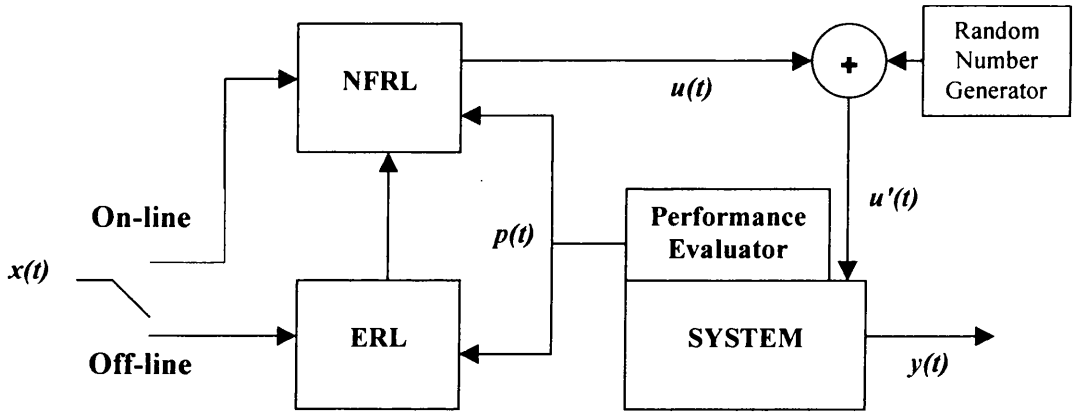


Figure 4.3 Block diagram of the evolutionary neurofuzzy RL algorithm

In the off-line process, the RL block is based on evolutionary algorithms, more specifically messy genetic algorithms. The on-line RL block is a gradient descent neurofuzzy network based on the ENFLICT model. In either case, the RL block accepts a state vector $x(t)$ and produces a control signal $u(t)$ which is then perturbed by adding a small signal generated by a random number generator. The performance $p(t)$ of the system due to this signal is then evaluated and fed back to the RL blocks. In the off-line process, the NFRL (neurofuzzy reinforcement learning) operates as a feed-forward network, and the ERL (evolutionary reinforcement learning) block determines its structure. The messy genetic algorithm (mGA) of the ERL block determines the shapes of the activation functions (fuzzy subspaces), the number of nodes in each layer and the interconnection of the network.

4.5.1 Off-line Learning

The first stage therefore deals with obtaining the structure of the network using the mGA procedure. The regions of fuzzy subspaces are defined according to the information available about the plant to the operator. Where the operating regions are known, a fixed universe of discourse with varying size membership functions is used. When the operating region range is not so clear, fixed size membership functions with a varying universe of discourse are used. The resulting network is one where the entire operating region is well covered with equally spaced overlapping membership functions, enabling smooth transition between states. At this stage, since no input-output data pattern is available, new input to the controller is obtained by

applying a 4th order Runga-Kutta algorithm to the system, and simulating the system over a certain time frame. The simulation would be carried out over a number of cycles to obtain a good general solution. The actual learning procedure using mGA is as described in §3.2.3.

On completion of the first stage, the best network structure is passed to the second stage where pruning and fine-tuning of the network is carried out and adjustments to the network structure made if necessary. Random initial states are applied to the network, which is then allowed to run until the system fails. If running off-line, new states to the controller are obtained as before using the RK algorithm. The number of successful (x, $u(x)$) (i.e. (*state*, *action*)) pairs are recorded and used as the overall reinforcement signal. The weights of the network are updated according to the number of hits they receive during each cycle of the learning algorithm.

4.5.2 On-line Learning

On-line learning is concerned with taking a near optimal network structure, plugging to a real system instead of a simulation and observing the system behaviour, and adapting the controller structure and parameters to these environmental changes. The learning algorithm employed is that of §4.3. Learning is through reward and penalty. If certain weights (corresponding to the parameters of the membership function (MF)) are used more often than others are, then they are rewarded such that the base width of the membership function is increased and its adjacent MFs are penalised by being reduced. Since no gradient information is being used, it does not make any sense to use the backpropagation algorithm to update the rules and network weights

Consider the equivalent fuzzy sets represented by the neurons at the *M*-nodes and *K*-nodes of the ENFLICT model. Let the shapes of these fuzzy sets be of triangular shape defined as

$$Triangle(x;a,b,c) = \max\left(\min\left(\frac{x-a}{b-a},\frac{c-x}{c-b}\right),0\right) \tag{4.23}$$

where a, b, c are parameters of the set as shown in Figure 4.4. $x\left(C_{S_l},C_{S_r}\right)$ and $u\left(C_{S_l},C_{S_r}\right)$ are defined as the [*state*, *action*] pair, and C_{S_l},C_{S_r} are the supports of the fuzzy set.

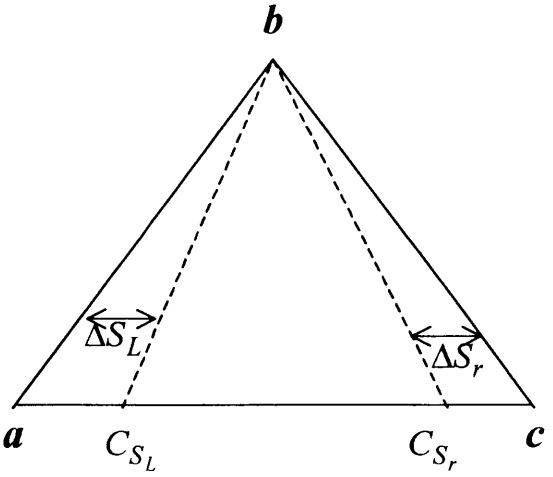


Figure 4.4 Triangular activation function

$$C_{S_L}(t) = \Delta S_L(t) C_{S_L}(t-1) \tag{4.24}$$

$$C_{S_r}(t) = \Delta S_r(t) C_{S_r}(t-1) \tag{4.25}$$

where ΔS is the amount the support is shifted,

$$\Delta S = \frac{R'}{R} N \tag{4.26}$$

where R' is the number of unique rules exciting set, R is the total number of rules as found at stage one of the process, and N is a weighting factor given by

$$N = \frac{\mu_s(m)}{\max(\mu_s(m))} \tag{4.27}$$

where s is the set under consideration, S is the full range of sets for the state and m is the linguistic rule. Finally to use the advantage learning of (4.17) the reinforcement signal is defined as

$$r_t = \sum_k \gamma^{k-1} r_{t+k} + \gamma^n f_t(x_{t+n}) \tag{4.28}$$

where f is the objective or value function and $\gamma(0.95)$ is the discount factor.

To illustrate the algorithm, consider Figure 4.5. Consider an initial network weight set-up such that the corresponding membership function set-up is as in Figure 4.5a. Now, assuming that the weights corresponding to the second membership function "NS" receive the most attention, then the base of this MF is spread out more into the regions of its adjacent MFs, "ZE" and "NM", Figure 4.5b. At the same time, as a consequence, the bases of its adjacent MFs are also reduced because they are playing a smaller role in the process. The amount of increase and decrease is proportional to the extent to which the weights are activated

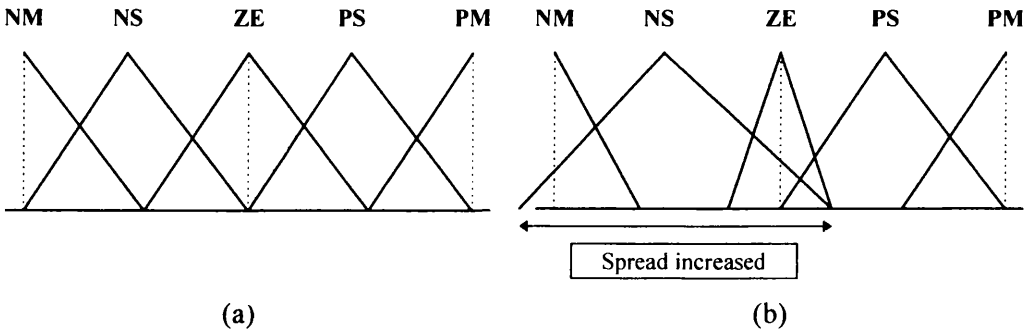


Figure 4.5 Extracted activation functions (a) before learning, (b) after learning

This sort of reward-penalty policy also has the advantage of removing redundant MFs. Consider Figure 4.5a again. In addition to "NS" receiving the most attention, assume "PS" is receiving more attention than "ZE" and "PM". The base of "PS" would then also be increased and the base of "ZE" and "PM" reduced. So "ZE" is gradually squeezed out from both sides. If as a result of increasing both "NS" and "PS", "ZE" is completely encompassed or not activated at all then "ZE" is removed, and with it any rules referring to "ZE" is ignored. The cycle of this second stage is repeated until the plant operates successfully to the operator's satisfaction (such as for a specified period).

4.6 Comparison of ENFLICT with other Reinforcement Learning Techniques.

When comparing the RL based ENFLICT, in addition to continuing the theme of learning fuzzy systems, the main focus is on the learning aspects. That is, whether delayed actions can be learned, continuous on-line learning is possible and flexibility exists to implement this learning fuzzy system for wide range of applications regardless of size and complexity. Table 4.1 compares ENFLICT with other fuzzy and neurofuzzy RL methods.

	ENFLICT	Nauck et al (1995)	Bonarini (1996)	Berenji (1992)	Lin (1995)	Zikidis & Vasilakos(1996)
Fuzzy set tuning	✓	✓	✓	✓	✓	✓
Rule modification	✓	✓	✓	×	✓	×
Mamdani controller	✓	✓	×	×	✓	×
Sugeno controller	✓	×	×	✓	×	✓
Non-symmetrical fuzzy sets	✓	×	✓	×	×	×
Different inferencing mechanism	✓	×	×	×	×	×
Different defuzzification process	✓	×	×	×	×	×
Unsupervised learning	✓	×	×	×	×	✓
Online learning	✓	×	×	×	×	×
Local learning	✓	✓	✓	×	×	×
Continuous time learning	✓	✓	✓	✓	×	✓
Delayed reinforcement	✓	×	×	✓	✓	✓
Model independent	✓	×	×	×	×	×

Table 4.1 Comparison of fuzzy reinforcement learning systems

As can been seen from Table 4.1, most work in literature have identified the need for continuous learning and delayed reinforcement. However, none of these mainstream methods are model independent as ENFLICT is. The main reason is that the neurofuzzy structures such as NEFCON (Nauck *et al* 1995, Zikidis and Vasilakos 1996) are not really neurofuzzy structures in the sense referred to in this thesis. That is they are not true representations of fuzzy systems as neural network structures. Instead, they are radial basis function networks that are functionally equivalent to fuzzy systems. This means that they are restricted to gaussian membership function type threshold functions, and the only parameters to learn are then the centres and widths of these functions, and only symmetrical fuzzy set tuning is possible. The other disadvantages of these systems are that they can't truly learn a fuzzy system. That is, these systems can not learn the type and shapes of the fuzzy sets, and only the rule base is learnt, and not the rule structure.

To learn the rule premise and consequents, Bonarini's (Bonarini 1996) ELF used an approach based on the Michigan evolutionary algorithm approach. This means that a group of rules, similar to a population of chromosomes, is used in the reinforcement learning approach. The problem of using this sort of EA based RL is that the appropriate premise to consequent mapping may always not be available, and thus result in sub-optimal solution. Thus, to begin with, the population is composed with full rule compliment, and reduced gradually. This implies that the operator has knowledge of the control surface to start off with. As (Nauck *et al* 1995) argues, this is also computationally expensive, as RL in general a slow learning process. In ENFLICT, although the rules are also deleted in the on-line phase, the approximate control surface has already been found by the off-line EA based RL, thus is less expensive. The Bonarini method also posses the problem on the other extreme. Since some rules are not available, the system can not cope with certain data. This problem is also shared with Nauck *et al*'s NEFCON approach. From this point of view, the ENFLICT approach is a compromise from both sides. No knowledge of the control surface is assumed, and constructed in the off-line phase, and rules are decremented during more localised learning.

Berenji's (Berenji 1992) GARIC model is one of the most cited works in this field because it was one of the very first systems to learn fuzzy systems through reinforcement learning. However, as can be seen, it only deals with fuzzy sets tuning, and does not have any rule modification properties. In terms of true comparisons, the closest system is that of Nauck *et al*'s NEFCON model. Although Lin's (Lin 1995) RFNC and the (Zikidis and Vasilakos 1996) model has a neurofuzzy structure, these are based on gradient decent learning, hence requiring knowledge of the derivative information needed to guide the learning. This is not the case with ENFLICT or NEFCON. However, as has mentioned already, NEFCON has certain restrictions, amongst which, the fuzzy weights must be implemented in such a way, that identical linguistic terms are represented by identical fuzzy sets. This symptomatic of other systems using the Mamdani control approach. In addition, in NEFCON, for different application areas different neurofuzzy models have to be derived. As has already been highlighted, this is not the case for ENFLICT.

To summarise, RL using ENFLICT presents a completely new approach to continuous time learning with relayed reinforcements for any kind of application without change of the underlying network structure. This model is able to learn the fuzzy rule base, rule structure, the fuzzy sets and the type of fuzzy sets.

4.7 Application to a Non-linear Coupled System

To see the operation of the algorithm, consider again the twin tank system used in the previous example. The objective of this control system is to drive, through the input to Tank 1, the liquid level at Tank 2 towards the desired level of 0.1 m in the first control cycle as fast as possible with minimal overshoots and steady-state errors. A second control cycle takes place from 600 s and the desired level in Tank 2 now is 0.2 m. The input states to the neurofuzzy network, are the tank height and rate of change of height and the output the pump flow rate.

For stage 1, the mGA was configured to accommodate a maximum of nine membership functions for each state variable. Since there are obvious limitations to the amount the liquid level can rise and the capacity of the pump, a fixed universe of discourse scheme was used. The activation type used was the triangular form. Since this system is very slow, to keep the computational time minimal only one type of activation is used. In addition, the triangular shape gives greater freedom when fine tuning the network in Stage 2 is carried out. The initial template was defined as a fully connected network with 9 memberships to describe each state/action variable. An initial population size of 200 was used which was halved after each era for 2 eras. This left a population size of 100 after the primordial phase, and the remaining population members were created randomly. For the juxtapositional phase the cut and splice rates were set to 75% and 80% respectively. Mutation rate was set to 10% and genes were

mutated to a value not in the chromosome. The resultant network of the system after stage 1 is illustrated in Figure 4.6 and figure 4.7, and the response is given in Figure 4.8. As can be seen the response is reasonable but not as smooth as one would wish. There appears to be excessive switching effect taking place as was experienced before and therefore again much tuning is required.

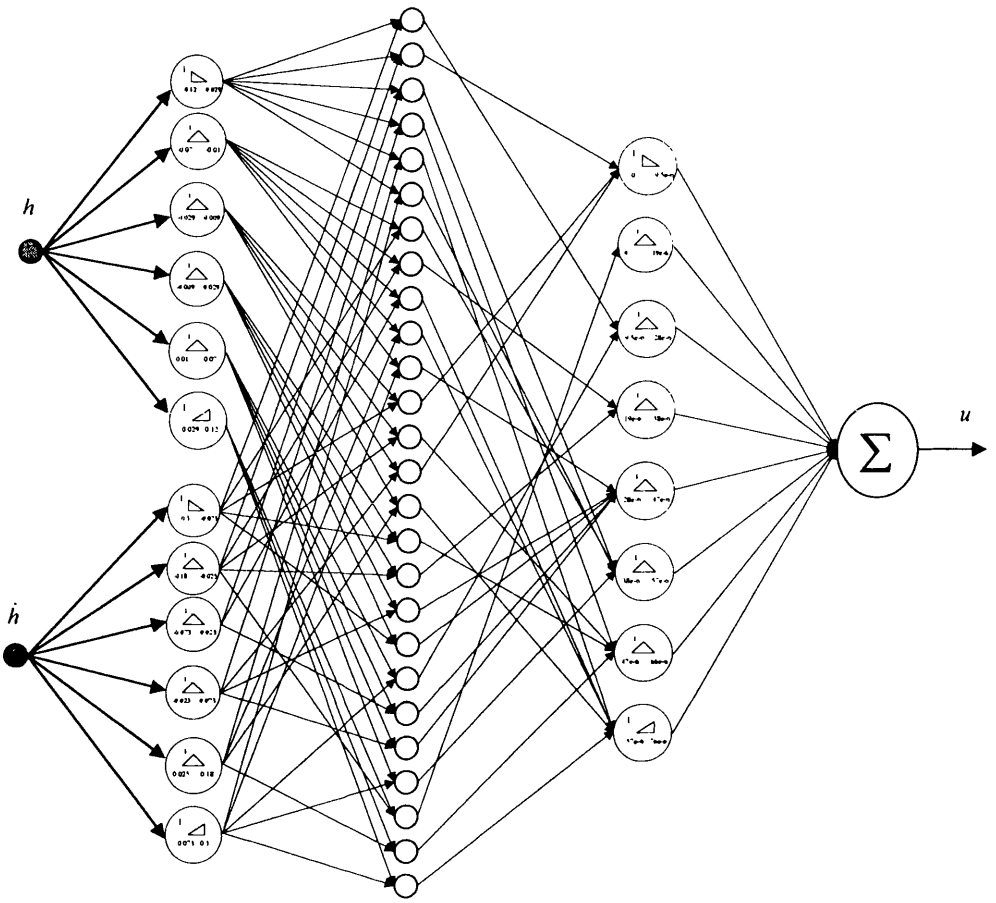


Figure 4.6 Neurofuzzy network of tank system after stage 1

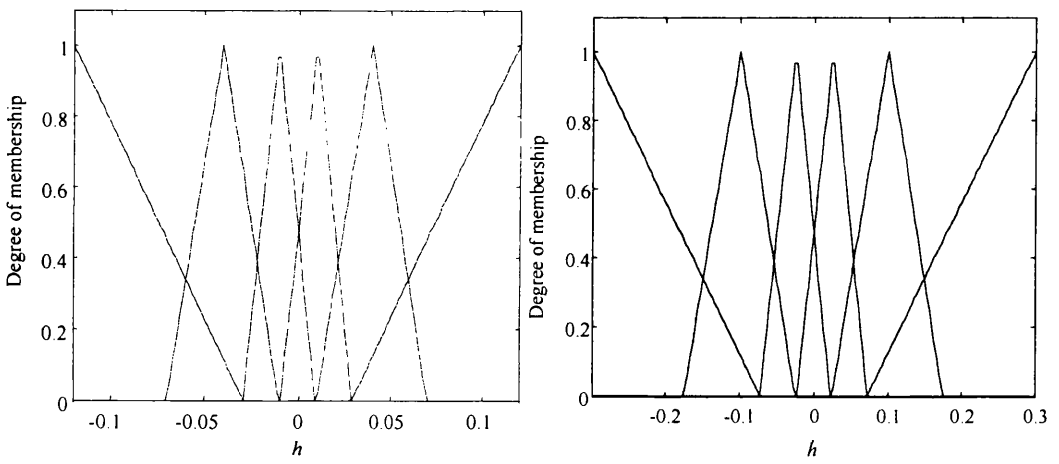


Figure 4.7 Extracted fuzzy sets from network of tank system after stage 1

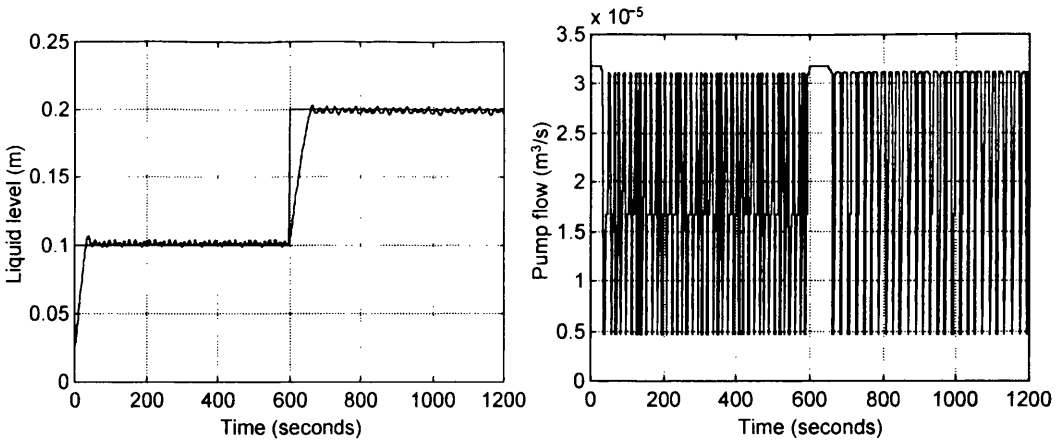


Figure 4.8 Closed Loop response of tank system after stage 1

This resulting "best" network if figure 4.6 is carried forward to the second stage for local learning and fine-tuning. For each training state if the network was able to deliver a level of liquid in tank 2 to within 5% of the desired level, it is rewarded with a score of 1, and −1 if outside the 5% boundary. With the objective not only to reach the desired height in the tank but also to maintain the level for a set period of time, the cumulative score after each run is used to reinforce the "local learning". The process is repeated until a satisfactory response is observed. Figures 4.9 and 4.10 show the resulting network of the system after local learning, and Figure 4.11 shows the response. Comparing with Figure 4.6, it is observed that in addition to the shape of the activation being affected, the number of neurons and the network connectivity is simplified. To test the robustness and ability in dealing with a non-linear system with varied operating conditions, the resultant controller was tested for different desired heights and this time a constant inflow disturbance of $8.33\times10^{-5}m^2/\text{sec}$ was applied at intervals of 300s. Note that there are no steady state errors and no switching effect. Figure 4.12 shows the responses for this test. Once again, observe that the oscillations are removed and the tank level is within 'acceptable' limits.

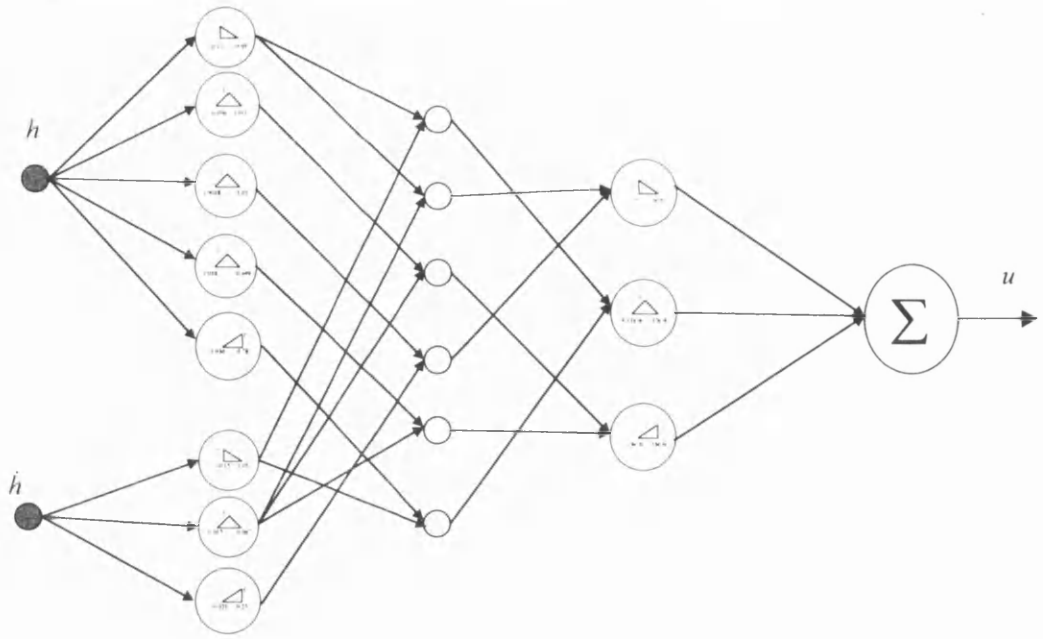


Figure 4.9 Neurofuzzy network of tank system after stage 2

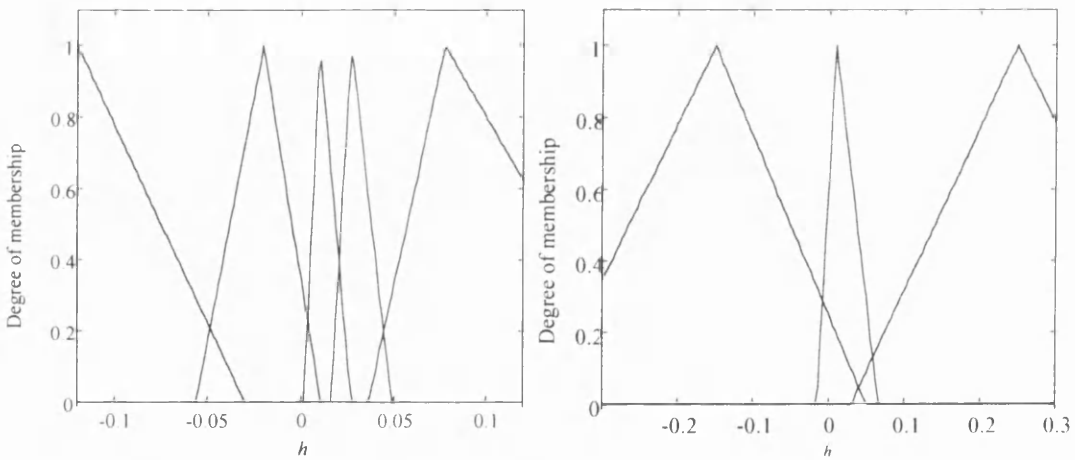


Figure 4.10 Extracted fuzzy sets from network of tank system after stage 2

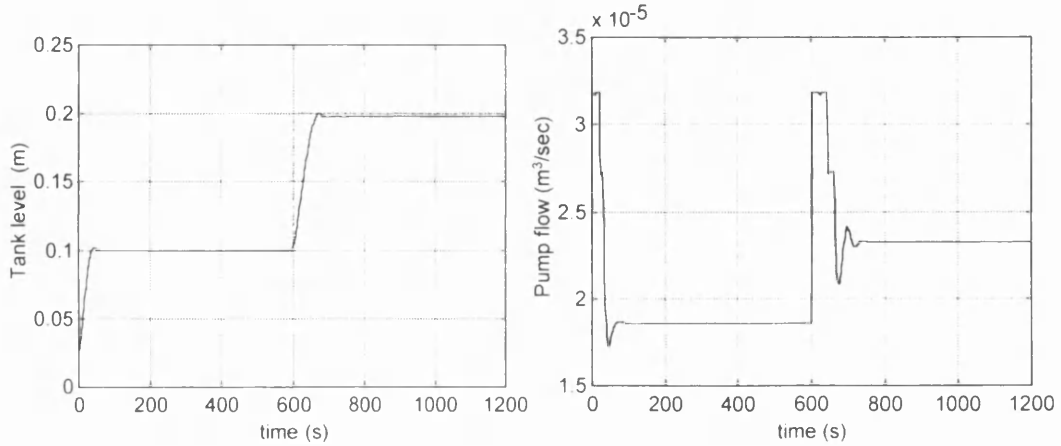


Figure 4.11 Closed Loop response of tank system after stage 2

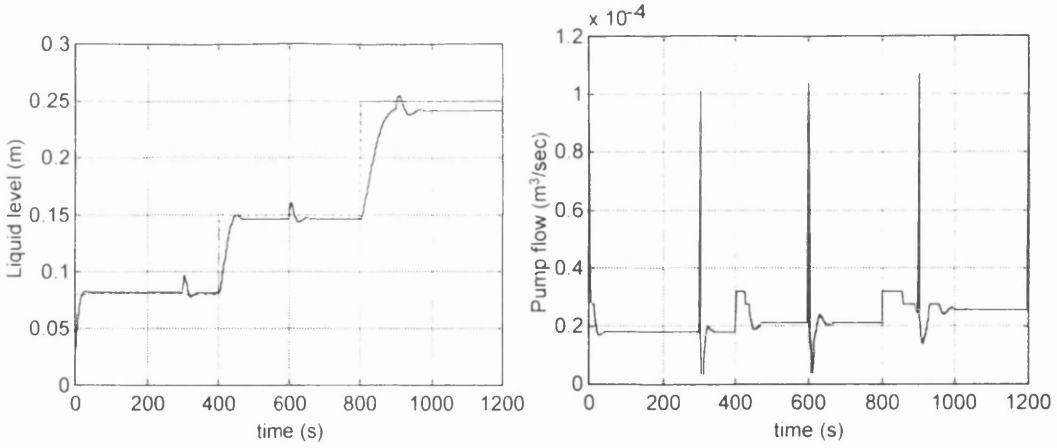


Figure 4.12 Closed loop response with disturbance

A comparison can also be made between ENFLICT and NEFCON as shown in Figure 4.13. As can be seen, NEFCON takes longer in comparison to ENFLICT to reach the desired levels of 0.1m and then 0.2m. This is because NEFCON has to build up the rule base as it learns, where as ENFLICT RL learning takes effect from a coarsely tuned network. As a result of not having the appropriate rules, there are more oscillations for NEFCON. However, since NEFCON learns from training data, it is more accurate, whereas at a level of 0.2m, there is a small steady state error for ENFLICT. This is as a consequence of the generalisation effect of the network, and can be solved using more localised learning.

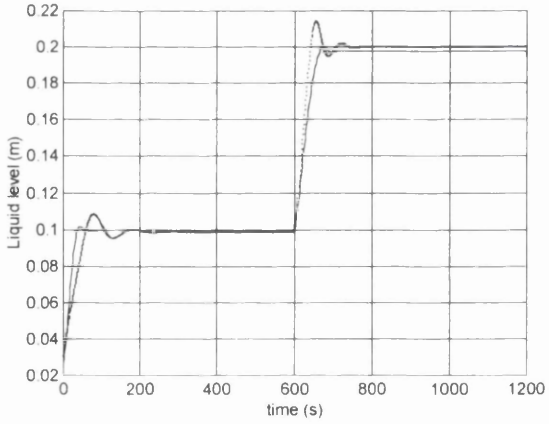


Figure 4.13 comparison between ENFLICT and NEFCON

4.8 Application to a Single-Input Multi-Output Non-linear System with On-line Learning

To illustrate the algorithm with another example,, consider the highly non-linear cart-pendulum system. However, instead of using the traditional single pendulum found in a lot of reinforcement learning and fuzzy control literature, and presented in Chapter 3, the more

complex double pendulum system is used as shown in Figure E.1. This non-linear control problem is often selected because of its similarity to many practical engineering applications, such as robot balancing, space shuttle arm, ballistics, and factory roof cranes, which require precision, stability and flexibility. The objective is to centre the cart on the track and balance both pendulums to vertical axis, by applying a control force to a cart centre of mass. The dynamic equations of the system are given below and full derivation can be found in Appendix E.

$$h_1\ddot{x} + h_2\ddot{\alpha}_1 \cos\ddot{\alpha}_1 + h_3\ddot{\alpha}_2 \cos\alpha_2 - h_2\dot{\alpha}_1^{\,2} \sin\alpha_1 - h_3\dot{\alpha}_2^{\,2} \sin\alpha_2 = u \tag{6.7}$$

$$h_2\ddot{x}\cos\alpha_1 + h_4\ddot{\alpha}_1 + h_5\ddot{\alpha}_2 \cos(\alpha_1 - \alpha_2) - h_5\dot{\alpha}_2^{\,2} \sin(\alpha_1 - \alpha_2) - h_7 \sin\alpha_1 = 0 \tag{6.8}$$

$$h_3\ddot{x}\cos\alpha_2 + h_5\ddot{\alpha}_1 \cos(\alpha_1 - \alpha_2) + h_6\ddot{\alpha}_2 - h_5\dot{\alpha}_1^{\,2} \sin(\alpha_1 - \alpha_2) - h_8 \sin\alpha_2 = 0 \tag{6.9}$$

where $h_1,\ldots,h_8$ be defined as below:

$$h_1 = m_c + m_1 + m_2 \qquad h_5 = m_2 l_2 L_1$$
$$h_2 = m_1 l_1 + m_2 L_1 \qquad h_6 = m_2 l_2^2 + J_2$$
$$h_3 = m_2 l_2 \qquad h_7 = m_1 l_1 g + m_2 L_1 g$$
$$h_4 = m_1 l_1^2 + m_2 L_1^2 + J_1 \qquad h_8 = m_2 l_2 g$$

where x is the cart position, α_1 and α_2 are pendulum link angles in radians, L_1 (=0.6m) and L_2 (=0.5m) are lengths of pendulum links, g (=9.81ms^{-2}) is the gravitational constant, l_1 and l_2 are the distances between the pivot and centre of mass of respective links, u is the control force, m_C (= 1.5 kg) is the mass of cart, m_1 (= 0.5 kg) and m_2 (=0.75 kg) are the masses of the first and second links and J_1 and J_2 (=0.0005kgm^2) are the inertia of the first and second links about their centre of mass.

The inputs to the neurofuzzy network are cart position, cart velocity, the pendulum link angles and their angular velocities and the output variable is the control force. For Stage 1, the mGA was configured to accommodate a maximum of nine membership functions for each state variable. This time the only bounds were the length of the track, which is set to 2m. Ideally the pendulum would be able to operate successfully from any given angle, but in practice this is not possible, hence a fixed universe of discourse scheme was used. The activation type used was the triangular and gaussian bell shaped form. Since there are 6 inputs, if a single controller in simple fuzzy form were to be used, a 6-D rule base would be needed, which would be incredibly complex to implement. However, this is not a problem with the neurofuzzy structure.

The off-line learning requires an initial template describing the network structure to be defined, and this was configured as a fully connected network with 9 memberships to describe each state/action variables. With the size of the problem in mind, an initial population size of 200 was used which was halved after each era and the other half refilled with random members

for 2 eras. The eras were executed at generation 20 and 45 respectively. For the juxtapositional phase the cut and splice rates were set to 80% and 80% respectively. Mutation rate was set to 15% and genes were mutated to a value not in the chromosome.

For the second phase (on-line learning), scores were awarded if the cart was within the track limits and the pendulums did not fail. An error margin of 5% was defined as "satisfactory". Figures 4.14-4.17 illustrates the behaviour of the cart-pendulum system with the optimised controller. Figure 4.18 shows the learning curves for the on-line and off-line stages. One can observe all the states reaching the desired states to a fast settling time and no oscillations. Only a snap shot of the entire time process is displayed to show the behaviour of the pendulums and cart before reaching the goal state.

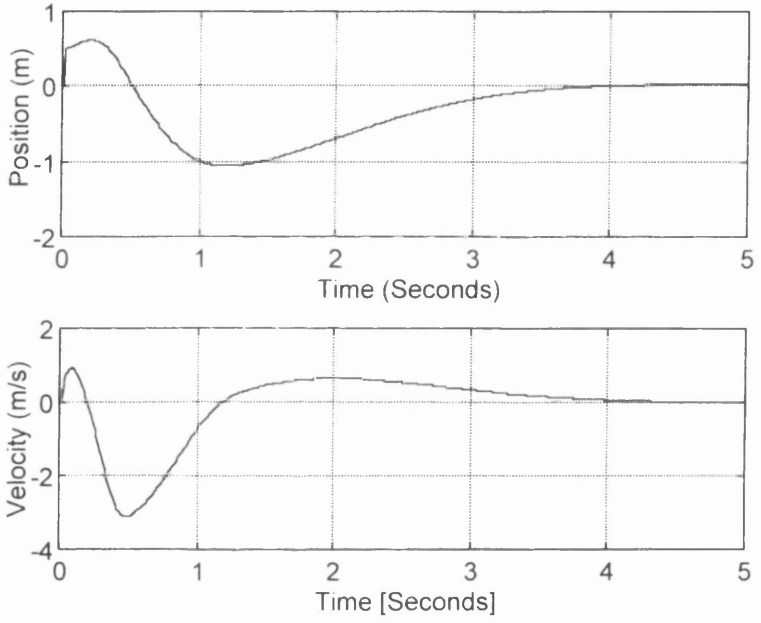


Figure 4.14 Cart position and velocity

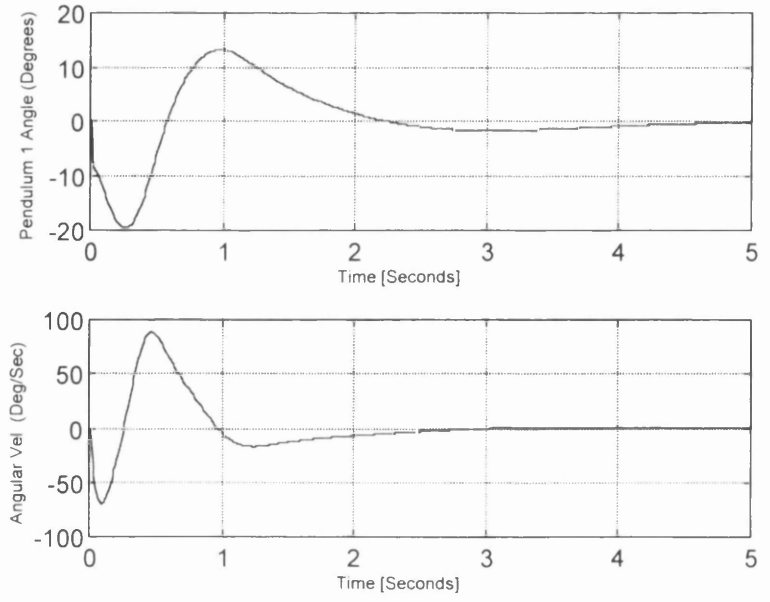


Figure 4.15 Pendulum 1 angle and angular velocity

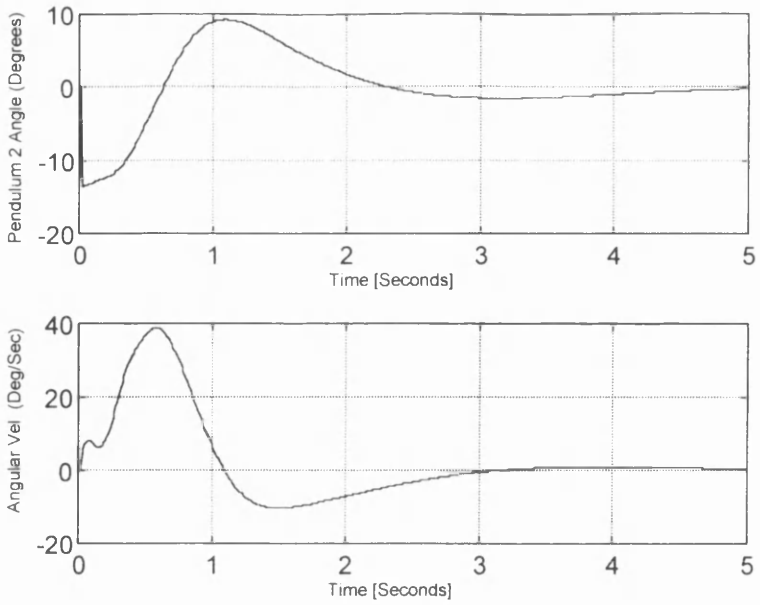


Figure 4.16 Pendulum 2 angle and angular velocity

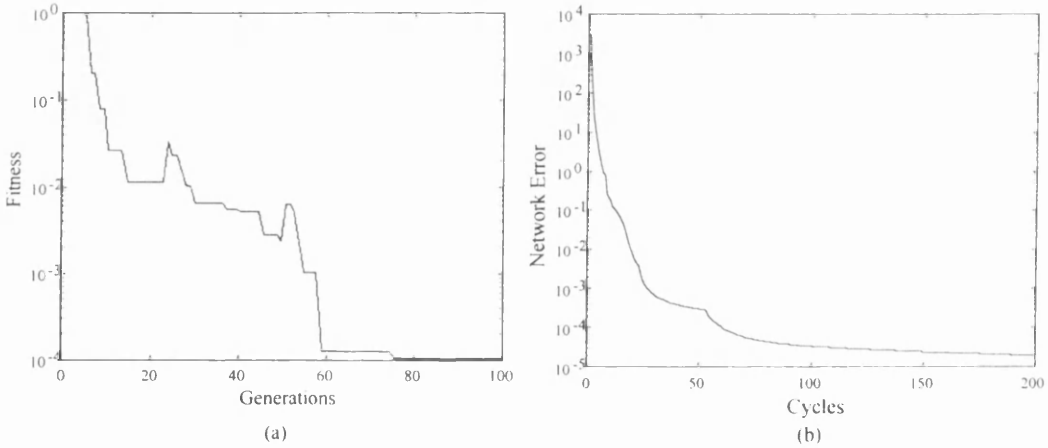


Figure 4.17 Network Learning Curves (a) off-line and (b) on-line

4.9 Summary and Discussion

In the previous chapter an unsupervised learning model was developed which was based on the backpropagation algorithm. However, it had a number of drawbacks that mean that the model in its existing format is unsuitable. The first is that the network structure grew with the size and complexity of the problem and the number of input and output domain variables. Another drawback is that once one has a network as presented after the evolution process, there is no scope to modify its structure. In other words there is no provision for modifying, rules. Unnecessarily large network structures can lead to slow operation of the model computationally, and also be detrimental to the convergence of the network parameters. In addition, being based on the backpropagation algorithm for updating of the network parameters implies that some

derivative information need to be available, like the direction of the annealing rate, to guide the learning. Finally, it was assumed that a model was present that could be used by the network to obtain a measure of its performance during learning. Therefore in this chapter reinforcement learning techniques were developed to solve these problems.

Reinforcement learning is a paradigm of artificial intelligence and is interested in systems that can adapt to their environment and experiences. A motivation for reinforcement learning is that it is the primary learning method of biological systems. Animals learn and adapt daily with only reinforcement type error signals. Reinforcement learning studies therefore seek to capture similar capabilities in artificial systems. Just as artificial neural networks are patterned after biological neural networks, reinforcement learning systems strive to emulate animal learning. Reinforcement learning combines elements of both supervised and unsupervised learning. Like supervised learning there is some training information available. However, this is not provided by an external teacher. Instead, as in unsupervised learning, there is a built-in critic that provides the training information. In addition as in evolutionary algorithms, it works around an evaluation function. In fact the correlation between evolutionary algorithms and reinforcement learning systems, will be studied. However, unlike EAs, the evaluation function does not tell the agent how it should change its behaviour. The agent simply tries to maximise or minimise the performance measure of the evaluation function.

In this thesis, Harmon and Baird's *advantage learning* algorithm (Harmon and Baird 1996) was used because it has been shown to function for continuous time systems without the need for a model definition. Some limitations of this algorithm were identified, viz., that it deals only with immediate reward RL, and uses a look-up table to guide the learning. Immediate reward is not suitable for on-line learning where the environment can be very large or complex. It is also unsuitable for situations where the system must be operating for considerable lengths of time before any information can be gathered regarding its relative performance. To overcome these limitations the gradient descent algorithm was extended to delayed reinforcement. This is a significant change because it is the only RL algorithm that can be truly applied to continuous time systems with a function approximator that is guaranteed to converge, and it is also suitable for on-line, model-free, implementation. This cannot be said for other RL algorithms. For instance, Q-learning and R-Learning do not work in continuous time and are sensitive to errors with small time steps. Other algorithms such as value iteration and SRV can work in continuous time. However value iteration requires a model to be learned and the calculation of the maximum of an infinite set of integrals to perform one update, and the SRV algorithm does not deal with delayed reinforcements.

After adapting the algorithm so that it can function for continuous time systems and on-line, the gradient descent advantage learning is combined with the ENFLICT model, and a two phase learning procedure is constructed that allows for off-line global network structure learning, and local on-line pruning of the network parameters. Since the underlying function

approximator is the ENFLICT model, the network is able to adapt to system parameter variations.

Comparisons with pure RL methods such as (Sutton 1988) and (Watkins 1989) are difficult as these are table based, supervised and applied to environments where all the information is available. Unfortunately, in the real world, this ideal environment does not exist. However, comparisons with other fuzzy-RL methods show the developed method to be much superior. To begin with, it is model independent unlike NEFCON (Nauck *et al* 1995), GARIC (Berenji 1992) and RFNC (Lin 1995). ENFLICT is a true integration of fuzzy and neural methods, and not some functional equivalent. Hence, all aspects of a fuzzy system can be learned, and different types of controller (Mamdani or Sugeno) can be used to suit the application without changing the network properties or structure. Unlike the likes of GARIC, ENFLICT is a single structure, hence only one simple structure needs to be learned. It is a compromise between the bottom up approach of NEFCON and top down approach of ELF (Bonarini 1996). With ELF, the Michigan style structure implies that rules may not exist for certain input, while using the bottom up approach also suffers the same problem.

Much has been achieved towards the aim of developing a flexible, autonomous, learning fuzzy control method. However, one important undesirable property has shown up in ENFLICT in its present state. While the network learns well for a specific set point, for a new set point, it performs sub-optimally, though not coarsely. Hence there is at least one further step that has to be taken, before it can satisfactorily be said that the method has achieved the aims of the thesis. The fact that EAs take a very long time (as does RL) for on-line operation means that there is a need for some approach that accommodates much more complex and higher order systems. An alternative approach to controlling such systems at a global level is to break the system up into sub-systems so that individual sub-systems can be treated locally, and then to connect up again through some hierarchical structure so that all the sub-systems when combined operate at a global level. In the next chapter this is further explored.

Chapter 5

Model-Free Design of FLCs

It's what you learn after you know it all that counts.
--John Wooden

This chapter focuses on localised learning aspects. Localised learning here refers to learning at different operating regions and also learning complex and coupled systems though hierarchical structures. In the last chapter, reinforcement learning was used to learn without a model and by direct interaction. However, to learn delayed actions, an estimate of the optimal action was used because knowledge of the next state was not immediately available. The effect of this was that the network performed sub-optimally around different operating regions. In this chapter, this problem is overcome by using plant step response data as the model, thus allowing immediate reinforcement to be used and more accurate networks to be obtained. The advantage of this approach is that learning can be done offline using data that is truly representative of the system. The method is then extended to deal with complex systems by breaking the system up in to sub-components and learning in a hierarchical structure. Then comparisons are made with well known complete evolutionary-neurofuzzy methods and the completed ENFLICT structure.

5.1 Autonomy and Ease of Design

Both evolutionary learning and reinforcement learning are good explorers and able to find good solution after a number of trials and some exploration. However, the main difficulty with such learning is that of knowing the plant behaviour in the form of a mathematical model and what the goal state or the next states should be. It is therefore important for the simulation model to be as accurate as the real plant itself so that the cost function and the fitness of a particular design reflect its true performance in the real world. Similarly for RL, having a good model and value function (especially if using an estimate) is of significant importance. This is, however, a challenging task in engineering practice.

Li *et al* (Li *et al* 1996) showed that it is possible to design linear controllers directly from plant response step data without the need for any mathematical model of the plant. When the

plant is non-linear it is shown that using such data is actually of a higher fidelity than any linearised model. In this chapter, this technique is extended to non-linear control system design and neurofuzzy system design in particular. The advantage of this is that the data can be treated as a model and the actual model need not be used at all. The disadvantage is that the data can be collected at any time for only one set point, but this is overcome by using the neurofuzzy structure to learn the controller. The generalisation property of the network means that it should be possible for the controller to operate at other areas outside the set-point that the data represents.

For more complex systems, where the system is highly coupled or the input and output domains are of higher order, the system can be broken down to sub-systems, and controllers may be obtained for each sub-system response data. Therefore, the other objective in this chapter is to construct a procedure for such a hierarchical structure.

5.2 Data as Model

In system design, the response to a step input is often utilised to analyse the system performance in terms of transient measures such as rise time, settling time and steady state errors. Given step-response data the controlled closed-loop output can be viewed as an open loop response of the system to the input filtered by a first order high pass and then convoluted by the step response of the plant. This arises from the fact that a system or plant is characterised by its unit impulse response, and the response of a plant can be obtained mathematically by convoluting the input waveform to that plant with its unit impulse response.

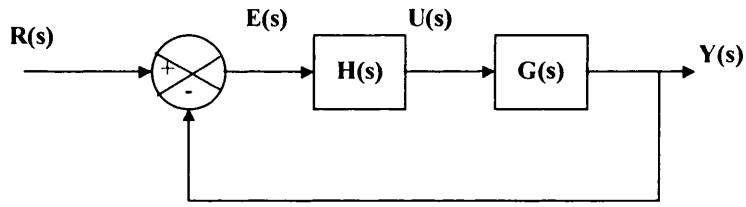


Figure 5.1 Schematic of unity feedback control system

Consider the plant set-up of Figure 5.1. If $U(s) = 1$, then for the open loop, the output is obtained by taking the inverse Laplace transform.

$$y(t) = L^{-1}\{G(s) \cdot U(s)\} = L^{-1}\{G(s)\} = g(t) \tag{5.1}$$

This can be generalised for any arbitrary input signal by applying the principle of superposition summation (Dorf 1989). Since a step signal is most common to obtain in the laboratory, let $U(s) = \frac{1}{s}$. Then the plant step response data will be given by:

$$Y_s(t) = \int_0^t g(\tau)d\tau \tag{5.2}$$

in other words,

$$g(t) = y_s(t) \tag{5.3}$$

Now consider a candidate controller is being designed, which provides a control signal $U(t)$. The plant output can then be simulated by:

$$y(t) = u(t) * g(t) = u(t) * y_s(t) = \int_0^t u(\tau) \cdot y_s(t-\tau)d\tau \tag{5.4}$$

It is clear to see from (5.4) that there is no mention of G or $g(t)$ i.e. the plant. Therefore, it is possible to simulate the control system and evaluate its performance directly from the plant response $y_s(t)$. Thus by taking this response to a step input, one can treat this like training data and use the evolutionary and reinforcement learning methods to evolve and learn a neurofuzzy controller. The difference between this and pure supervised learning is that the teacher has to generate manually the learning pattern for the network to learn in supervised learning, whereas in this case the data represents the true behaviour of the plant.

5.3 Hierarchical Control Approach

The drawback of this approach is that for a non-linear plant, such a method is valid only around the operating point. That is, the convolution approach can only be used for one operating region - which is undesirable since the aim is to have a controller that can perform over all operating regions. A possible way around this problem would be to build a controller, like a local controller network (Gawthrop 1996, Johansen and Foss 1992), around each operating region and then switch between each controller when in the appropriate region.

The objective of the hierarchical approach is to design a set of controllers which are locally optimal, and also when put together are general enough to perform efficiently at a global level. Therefore it is necessary to ensure that no single controller has priority over others, that the rewards and penalties are distributed equally amongst all the controllers. The whole

learning procedure can be broken up into three levels. The first is to obtain a global network structure that encompasses all the lower level controllers (LLCs) or networks; then to use a learning rule that adapts the 'global' network structure to environmental changes; and thirdly, to use the global information for tuning each local network. The hierarchical structure can also be applied for more complex and large systems. At the lower lever, individual controllers would be constructed for each local (or sub-system) level, and a network at the upper level structure would ensure that all the local networks would perform globally. Since EAs and, in the context of this thesis, a messy GA has been shown to be good global approximators, the upper level of the hierarchical structure is overseen by the mGA. At the lower lever, each local network controller is shaped by the reinforcement ENFLICT structure.

5.3.1 Global Network Structure Optimisation

Consider the case, at the top level of the hierarchy, of landing an aircraft. At a lower level, the sub-tasks may involve descending and taxiing the aircraft. Using the hierarchical structure, at least 2 controllers would be needed. Then for all local ENFLICT controllers, an upper lever structure consists of the combination of the networks such that it forms a global network as shown in Figure 5.2. As can be seen, this network resembles a feedforward structure.

In fact, this is another neurofuzzy network similar in structure to the ENFLICT model used for each local controller. The objective of the upper level is to perform a mapping from some input to some output space. For each state input, there is a switching policy that indicates the extent to which that state input will affect a certain local controller. This is similar to obtaining the degree of membership to which a certain fuzzy set is fired in a standard FLC. The task of the switching policy is to distribute the inputs to each of the controllers at the lower level. It does so by weighting the input according to the degree to which that input set is relevant to the sub-system.

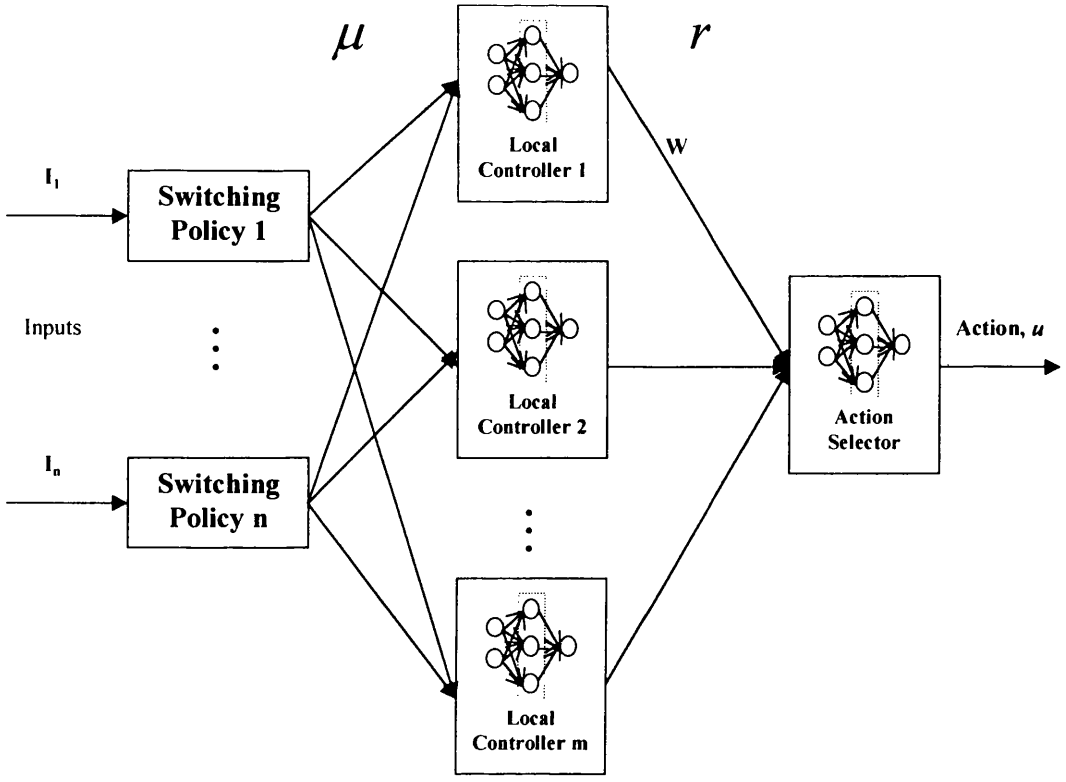


Figure 5.2 Global Network Structure for Local Controller Design

For example, for the 2 controllers there would be a policy selector, and its output is a measure, or degree of activation, of the immediate level of activation of the higher level expressed quantitatively as a scalar. This is an indication of the amount of influence that the upper level will have on the control action corresponding to the set-point during the control cycle. Therefore, the switching policy blocks resemble fuzzy subspaces, where the fuzzy subspaces correspond to the operating region of the sub-system it is hierarchy to. The fuzzy subspaces are defined by gaussian membership functions such that the centre of each membership function is a set point, and the spread is such that there is 50% overlap between adjacent membership functions, as shown in Figure 5.3. Here 'LLC1' stands for 'Lower Level Controller 1'. 'Lower' does not imply a lower performance, but locally refined. Therefore, the output of a policy selector is given by

$$\mu = e^{-\frac{(x-c)^2}{2\sigma^2}} \tag{5.5}$$

where x is the input, c is the centre of the fuzzy subspace, and σ is the spread of the fizzy subspace. The output of each sub-network, or lower controller, is a reinforcement signal that indicates how well that lower controller is performing for the given state inputs, and is given by (4.28).

Just as the policy switching block performs fuzzy operations, the *action selector* then performs defuzzification operation. Just as in the conventional defuzzification process, the result from all the rules is aggregated and defuzzified to provide crisp action, The action selector also defuzzifies the aggregate of all the output of 'lower' level controller in its hierarchy. This means that the output of the lower level networks have to be presented as fuzzy sets to the action selector. As a result, the defuzzification of the LLCs is carried out after their 'parent' defuzzification. Thus working up all the time, the output of the action selector of the highest level controller represents the overall behaviour of the global system. The overall output of the hierarchy is thus given by

$$V = \frac{\sum wr}{r} \tag{5.6}$$

where r is the reinforcement signal of each local network and w is the weight connecting the lower network to the action selector of its parent network. 5.6 is in fact the weighted average defuzzification process. Thus in essence the network is a neurofuzzy one. A normalised term is used to avoid under-generalisation. If normalisation is not used, then the sum of all the reinforcement signals may produce a value close to zero as a result of contributions of networks well outside the set point.

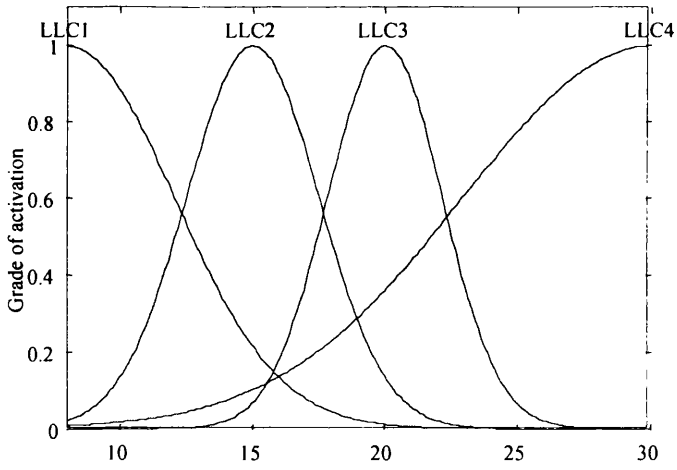


Figure 5.3 Lower level controller representation by gaussian fuzzy subspaces

It is worthwhile to note that while all the networks work in parallel, not all the networks will actually be fully functional for any set point. If the said state input is not in the neighbourhood of any LLC, or if the switching policy block has decided that during a certain control cycle a certain LLC does not necessarily have to function, then that LLC by default returns the worst reinforcement signal. Processing power is thereby saved and the time taken to obtain the corresponding action is reduced. This is equivalent to the LLC having zero rules

firing for the set-point in question. Since response data is being used as opposed to having the controller being plugged into the real system, it is now possible to use the messy genetic algorithm for continuous learning. Messy genetic algorithm is only used for the highest level in the hierarchy because the exploration space is largest at this level. It can used for LLCs, but, since the search space is narrower, RL is used instead.

The messy genetic algorithm codes and decodes as in the case of offline learning described in §3.2.3, but the interpretation of the decoded information is different. Recall the gene **(252)** which decodes to **[2 5 2]**. Previously, this was interpreted as

[2 5 2]: Input **2** connects to the **5**th *M*-node belonging to this domain, and this node has shape type **2**.

This now interprets as,

[2 5 2]: Input **2** connects to the **5**th controller immediately under its hierarchy, and the fuzzy subspace representing this local controller has shape type **2**.

Since the network output is given by the action selector, another difference is that each gene encodes and decodes for the input domain only,. As before, there has to be some sort of precedence rule, such as *first-come-first-served*, that governs the description of the fuzzy subspaces and the definition of the premise. Note also that there has to be a predefined limit on the number of local controllers, or else the mGA may end up with a very high dimensional global network structure prematurely. However, there is provision for adding further local networks should the need arise, and the rule for this is defined in the objective function for the mGA as follows:

- Determine the local network with the largest cumulative error, and call this network 1
- Determine the local controller adjacent to network 1 with the highest cumulative error and call this network 3.
- Create a new local controller, 2, in the middle of 1 and 3.
- Create a fuzzy subspace for 2 such that its centre is between that of 1 and 3, and its spread has 50% overlap with those for 1 and 3. Note that if 1 is such that its fuzzy subspace is at the boundary of the global operating condition, then 3 does not need to be scanned as it is the only controller adjacent to 1.

5.3.2 Local Network Structure and Parameter Learning

While the higher level behaviour is similar to a neurofuzzy structure in that it performs a mapping from some input space to some output space, the lower level maps the inputs to control outputs. Also, as the messy GA optimises the global structure, the local network structures and parameters are learned with the advantage learning procedure described above. It should be noted that each sub-system might itself have sub-systems immediately below it. To illustrate this, consider again the example of landing an aircraft. Under the sub-system dealing with the aircraft's descent, it is possible also to have the systems dealing with lifting of the wing flaps, the lowering of the wheels and controlling of the attitude. This is shown in Figure 5.4. Therefore, each sub-system with further nodes below it operates in the same fashion as the node at the highest level, except that the action selector returns a crisp value after its own parent has carried out the defuzzification process. The other difference is that the learning of the network structure at this level is carried out by not mGA but by the RL algorithm. In this case, in addition to the rule structures being adapted, the network parameters are also tuned. That is, the positions of the fuzzy subspaces are changed.

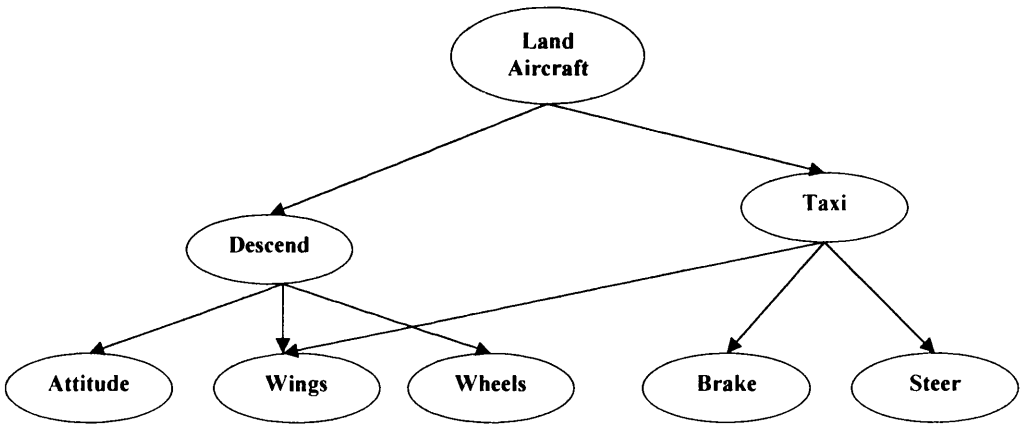


Figure 5.4 Hierarchical control structure for aircraft landing

This operation also applies for the lowest level of the hierarchy, where the operation is at a local region of the global search space. However, since only a local region of the entire exploration space is accessible to the network, the learning algorithm will be greatly handicapped in pursuit of the optimal policy because it may not have enough information about the system. There are two approaches to overcome this. The first is to remove some information from the system such as narrowing of the global operating regions. However, while this may make one local controller perform better around a certain operating region, it may render the other controllers ineffective or sub-optimal.

To avoid such interference, an alternative solution is to let each controller reach the best optimality it can in its separate operating regions. Thereafter, if further learning is required and more information is needed, detach the local controller from the hierarchical structure and let it explore further on the global search space. The advantage of this is that the starting controller is already near optimal and hence the time required for finding the optimal will be greatly reduced.

5.4 Comparing ENFLICT with Evolutionary Neurofuzzy Learning Systems

Thus far, any comparison between ENFLICT and methods found in other literatures, has been with evolutionary-fuzzy, neurofuzzy or fuzzy-reinforcement learning systems, and not with any global and complete evolutionary neurofuzzy systems. Now that the development is being concluded, it is a good moment to reflect on ENFLICT's properties and compare it with methods that it can be compared with as a single structure. The summary of the comparison is shown in Table 5.1.

	ENFLICT	Ishigami et al. (1995)	Fukuda et al. (1994)	Melikhov (1996)	Kim et al (1995)	Perneel et al (1995)	Russo (1998)
Fuzzy System Optimisation							
Rule base construction	✓	✓	×	×	×	×	×
Fuzzy set construction	✓	✓	×	×	×	×	×
Fuzzy set tuning	✓	✓	✓	✓	✓	✓	✓
Variable universe of discourse	✓	×	×	×	×	×	×
[G]lobal or [L]ocal fuzzy sets	G,L	G	G	G	G	G	G
Fuzzy set type definition	✓	×	×	×	×	×	×
Mamdani controller	✓	✓	?	✓	✓	✓	✓
Sugeno controller	✓	×	?	×	×	×	×
Non-symmetrical fuzzy sets	✓	✓	×	×	×	×	×
Different inferencing mechanism	✓	×	×	×	×	×	×
Different defuzzification process	✓	×	×	×	×	×	✓
Supervised learning	✓	✓	✓	✓	✓	✓	✓
Unsupervised learning	✓	×	×	×	×	×	×
Online learning	✓	×	×	×	×	×	×
Local learning	✓	×	✓	×	×	×	×
Model dependent	✓	✓	✓	?	✓	✓	✓
Model independent	✓	×	×	?	×	×	×
Hierarchical structure	✓	×	✓	×	×	×	×
Learning directly from data	✓	×	×	×	×	×	×
Learning through reinforcement	✓	×	×	×	×	×	×
EA Representation							
Integer encoding	✓	✓	?	✓	×	✓	×
Variable length representation	✓	×	×	×	×	×	×
Reproduction operator	cut and splice	crossover	crossover	crossover	crossover	crossover	crossover & hill climbing
Entropy cost function	✓	×	×	×	×	×	×
Single gene representation	✓	×	×	×	×	×	×

Table 5.1 Comparison of ENFLICT with evolutionary neurofuzzy methods

Comparisons are made against the major contributions of the thesis, that is, in the flexibility of the evolutionary algorithm representation. This involves coding of the rule base, the rule structure, gene representation, cost function and the number, type and shape of the fuzzy sets. In this thesis, a variable length chromosome is used that allows for representing the rule base and the rule mappings. In contrast, none of the systems being compared with exhibit this property. As has already been highlighted, using a fixed length chromosome is restrictive on the size of the problem and the amount of information that can be represented in the genes. Genes in the ENFLICT structure are integer encoded and a single gene coding is used, thus allowing a gene-to-parameter representation and making the order of the genes in the chromosome irrelevant. Therefore, knowledge of the linkage format is not necessary. In contrast to ENFLICT, all the other methods were found to be using some variation of the quadratic error function as the cost function. In ENFLICT, an entropy function speeds up the learning process and also is less resistant to getting trapped in local optima.

In the learning process, the focus is on how flexible and accommodating the network structure is. It comes as no surprise that most methods employ the Mamdani type controller because the Sugeno type uses crisp values to represent the output that is often difficult to predict. However, as has been highlighted in §3.3, there are situations where Sugeno type is useful. ENFLICT is the only model that allows for both types of controllers. However, what is surprising is that with the exception of (Ishigami *et al* 1995), none of the other methods are concerned with optimising the rule base. These methods are only concerned with optimising symmetrical fuzzy sets. As has been illustrated in §2.2, the shape and type of fuzzy sets can affect the performance of the controller. From this point of view, ENFLICT is much more flexible as it is the only model that allows mixed type non-symmetrical fuzzy sets.

A third area of contribution of the thesis is in the type of learning. Reinforcement learning was used for unsupervised learning through direct interaction with the environment. As can bee seen from Table 5.1, all these methods are based on supervised learning where training data is used to train the network. In contrast, ENFLICT allows one to learn the network on-line, unsupervised and independent of a model.

Final area of contribution is in local learning and learning of complex systems through hierarchical structures. ENFLICT uses plant step response data to represent the model, thus being able to learn off-line using data that is a true representation of the model. None of the methods being compared with exhibited this property.

In conclusion, whether being compared with individual components or as a whole structure, the methods developed around ENFLICT for learning fuzzy systems is the most flexible structure developed.

5.5 Case Studies

5.5.1 Single inverted pendulum

To illustrate the above procedures, consider the example of the inverted pendulum in appendix E, but with only one link, first studied in Chapter 3. The problem is to control the motion of the cart along a horizontal line so that the pole will not fall down and will eventually stand at a desired angle. As before, there are four states associated with this model: cart position x, cart velocity v, pole angular position θ, and pole angular velocity ω. The pendulum is controlled by applying a force of varying magnitude to the cart's centre of mass. The hierarchical structure of the system is illustrated in Figure 5.5. For the tests, the following parameters were used:

g (acceleration due to gravity) = 9.81 m/sec^2

$0.1 \leq m$ (mass of pole) ≤ 1 kg

$0.5 \leq M$ (mass of cart) ≤ 2.0 kg

$0.5 \leq l$ (length of pole) ≤ 0.1 m

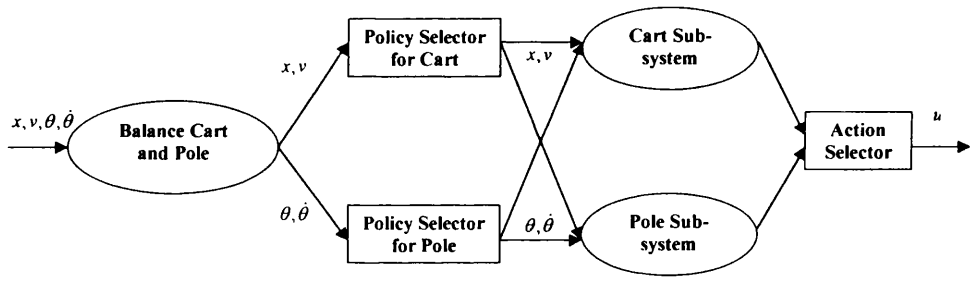


Figure 5.5 Hierarchical structure for cart-pole system

The aim is to find the control force required such that x(t) and θ(t) converge towards the desired centre position on the track, and an angle of zero to the vertical axis respectively in the shortest possible time for system parameter variations. As can be seen, there are two lower level controllers, one deals with the cart and the other the pole. The first task is to set up the global controller for the messy GA to optimise. This consists simply of two switching policy blocks with two membership function definitions, and the hidden layer consisting of two nodes, each representing a lower level controller. The evolutionary optimisation was carried out on the global controller alone with 100 generations over a single era and a population size of 200. The probabilities for cut, splice and mutation were set at 65%, 70% and 5% respectively. For computational efficiency, the mGA was only allowed to select between triangular and gaussian membership shapes.

For the local learning procedure, the local networks were each set initially with 5 fuzzy subspaces, equally spaced out for each of the input and output domains. To keep the

computational burden down only the centres and widths of each membership function were learned. Figure 5.6 and 5.7 show the two LLCs, and 5.8 and 5.9 shows the global controller input memberships. Figure 5.10-5.13 illustrates the response of the system to the various points used during and after the learning procedure. As can be seen, both the sub-system network structures have changed during the learning operation, and have found some nodes to be unnecessary and removed them from the structure. The responses are, as described, for the highest level structure, which consists of the two local controllers and the two switching policy structures of Figure 5.8 and 5.9. As can be seen, the required objectives have been achieved. Comparing the responses with those of the identical set-up for the model based system it can be seen that this model free approach procedure is comparable.

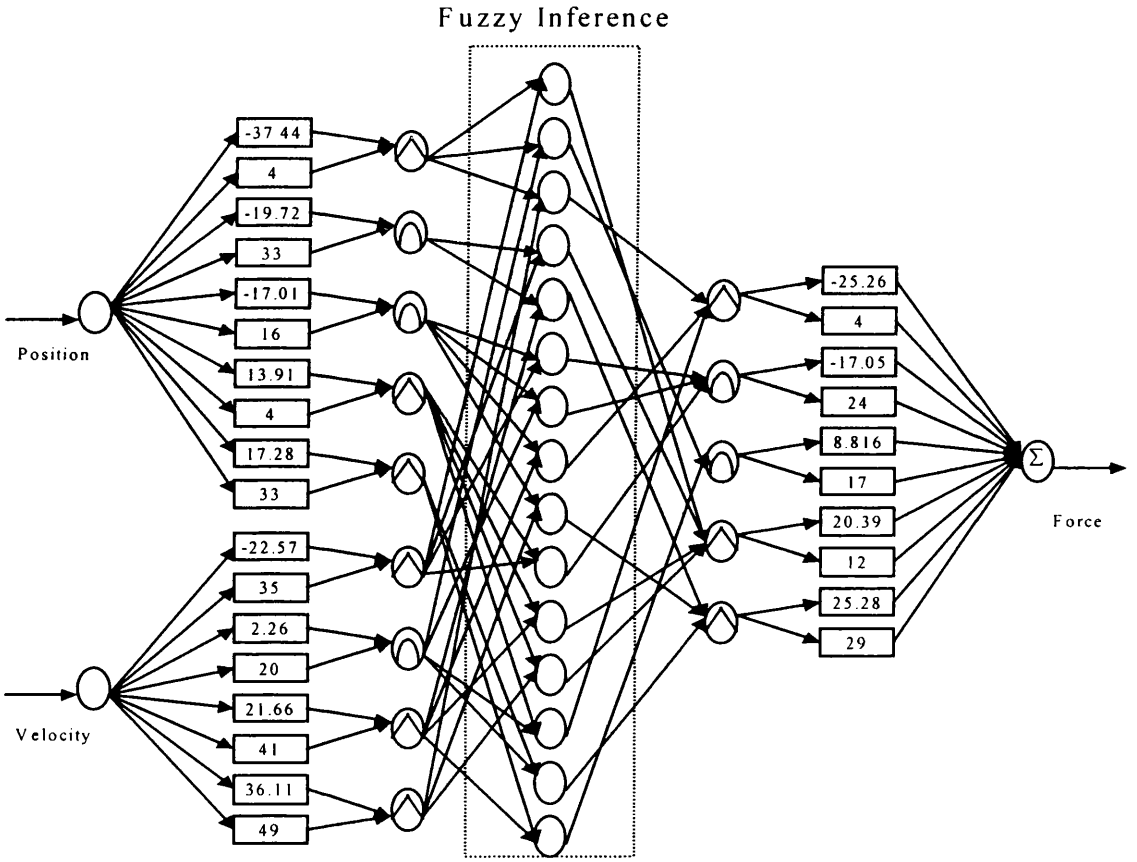


Figure 5.6 Controller for cart sub-system

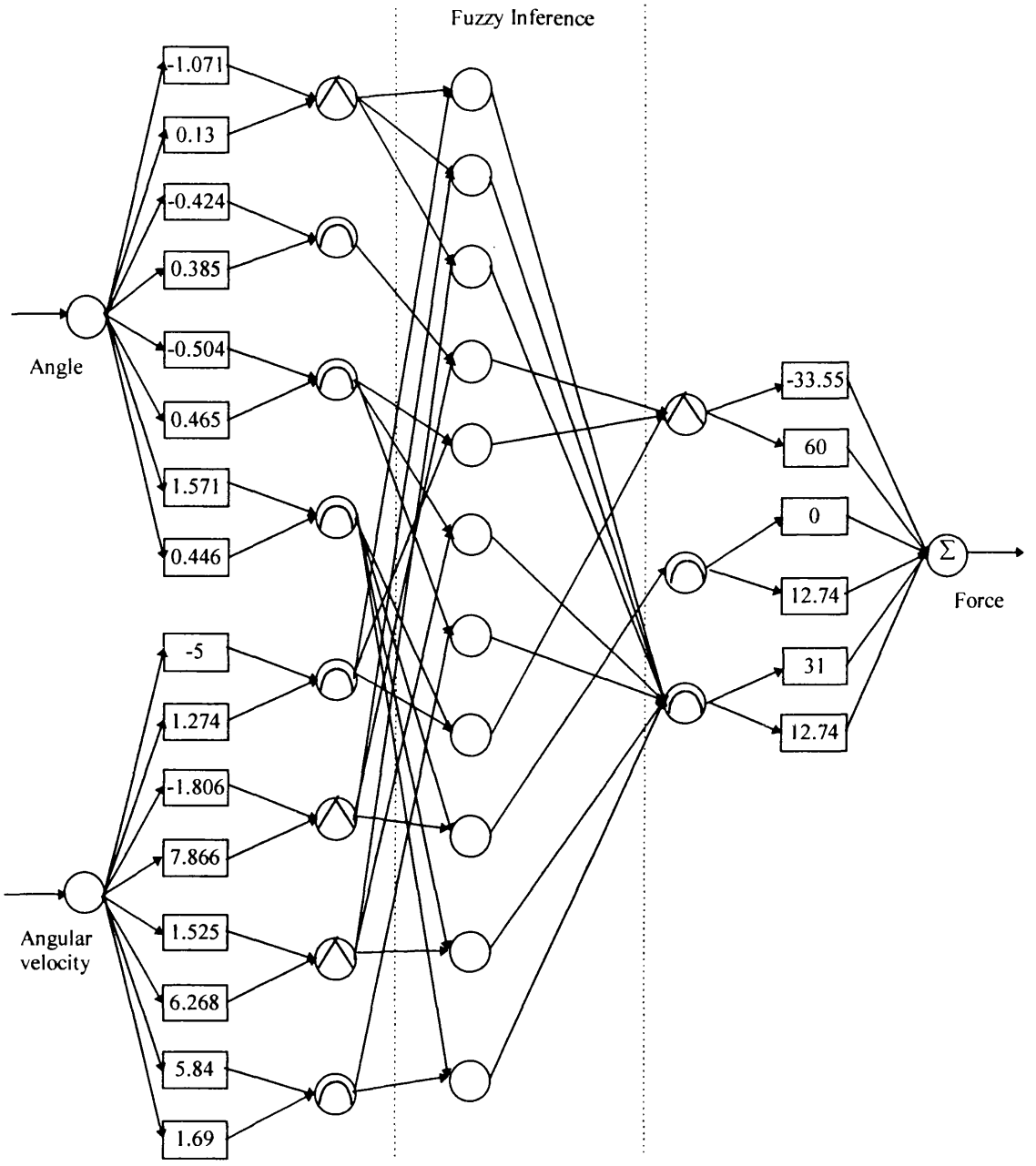


Figure 5.7 Controller for pole sub-system

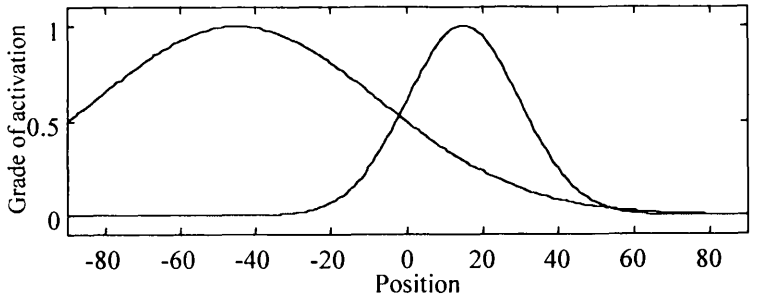


Figure 5.8 Fuzzy subspaces pertaining to cart switching policy box

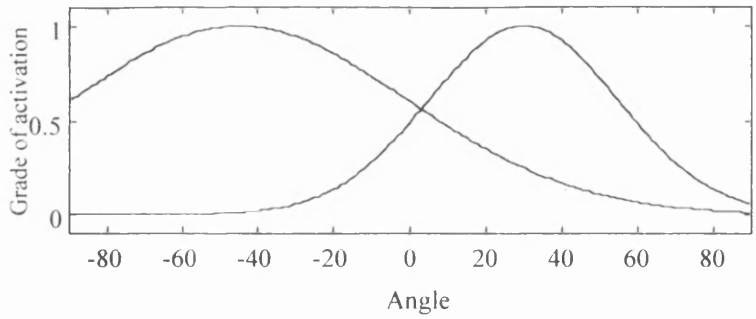


Figure 5.9 Fuzzy subspaces pertaining to the pole switching policy box

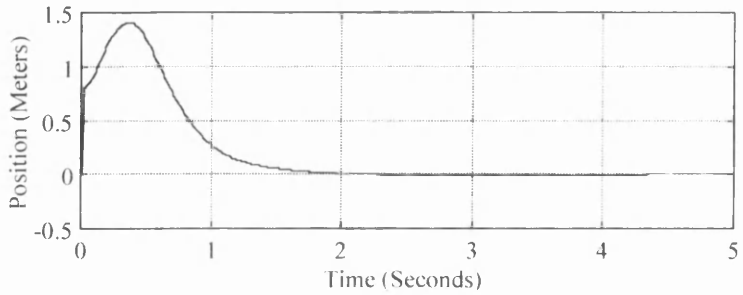


Figure 5.10 Position of cart for initial conditions (0;0;0;0)

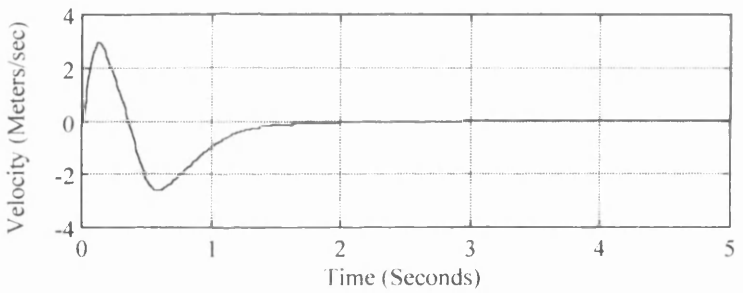


Figure 5.11 Velocity of cart for initial conditions (0;0;0;0)

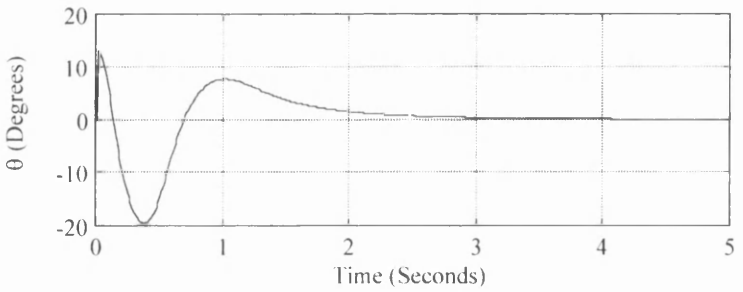


Figure 5.12 Angle of pole for initial conditions (0;0;0;0)

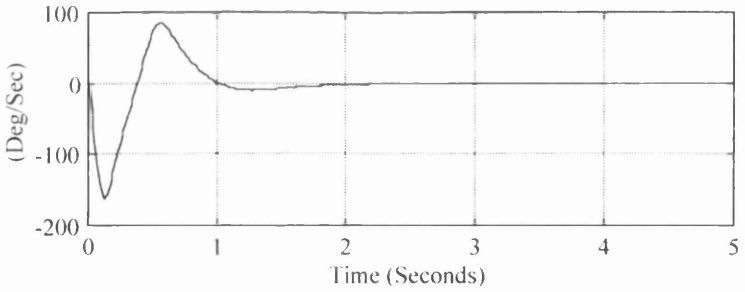


Figure 5.13 Angular velocity of pole for initial conditions (0;0;0;0)

5.5.2 Case Study – Liquid Level Control Example

As a second example, consider again the liquid level control system first encountered in Chapter 3. Simulations were first carried out on the tank system in open loop. Sample response to a step input signal was obtained by applying 4th order Runga-Kutta to the system equations. Figure 5.14 illustrates the response of the system for a step-input signal in open loop for the model and that obtained with the convolution method.

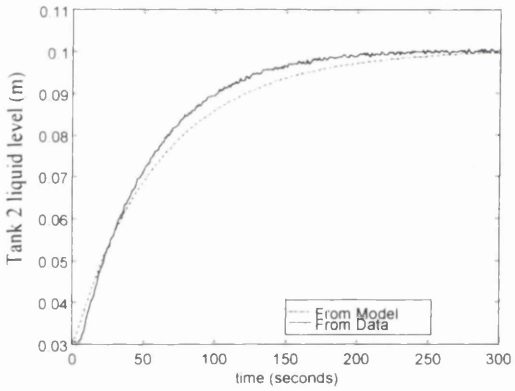


Figure 5.14 Plant response data for non-linear model and data obtain by convolution

For the local neurofuzzy networks, the type of activation functions were limited to two types, gaussian and triangular. However, this time the sub-components are different set-points which the system must follow, instead of physical sub-components, as shown in figure 5.15. As before, the inputs to the networks are the tank level and the rate of change of the level, and the output is the pump flow rate. Since the system is very slow and hence to keep the computational time to a minimal possible, an upper limit of 7 nodes per layer was imposed. For the higher level network, an initial population size of 400 was used which was halved after each era for 2 eras. This left a population size of 200 after the primordial phase, and the remaining population members were filled randomly. For the juxtapositional phase the cut and splice rates

were set to 75% and 80% respectively. Mutation rate was set to 10% and genes were mutated to a value not in the chromosome.

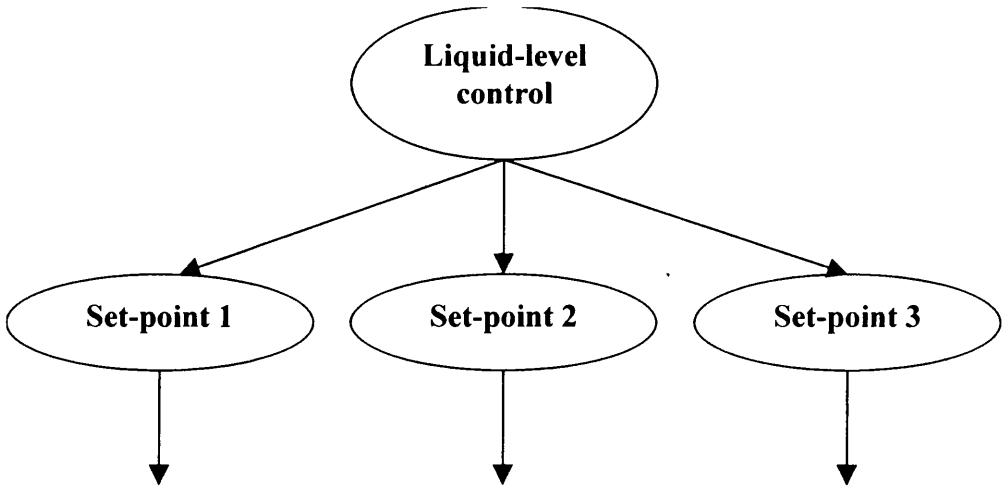


Figure 5.15 Hierarchical structure for liquid level control system

Figure 5.16 shows the tank liquid level responses and the control actions needed for various set points. It can be observed that the response obtained through the convolution method using the neurofuzzy based on mGA and RL combination has faster rise time and settling time than that obtained through the model. The neurofuzzy approach seemed to have a small steady state error for set points different from that with which the training was carried out. This is due to the difficulties stated, with the local controllers needing more information available to them for optimal performance. This suggests the limitation of using the linear convolution data to replace the non-linear plant. Nevertheless, it shows that a neurofuzzy controller designed from the data can provide a good controller for an unknown non-linear plant. For comparison, a single controller designed around a single operating point was also constructed, and Figure 5.17 shows the behaviour of the tank. As can be see, as expected, the hierarchical method performs better because it is able to obtain controllers that are more representative of the global system. The set-point used to obtain the single controller is 0.1m, and it can be seen that for a set-point of 0.075m, which is in the neighbourhood of 0.1m, the controller does a good job, deviating only slightly from the desired point. On the other hand, for a set-point of 0.15m, the deviation from the desired is very significant. Comparing it with 5.16, the hierarchical structure has yielded a better global performance.

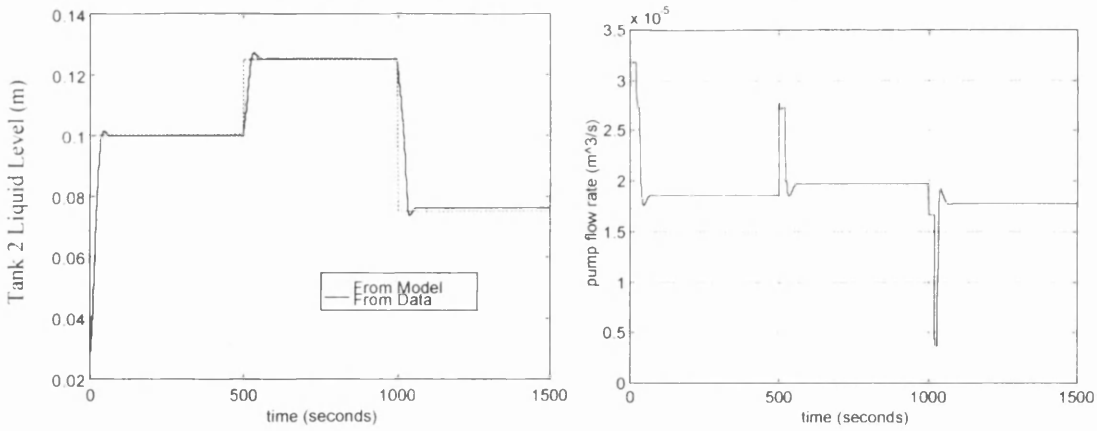


Figure 5.16 Tank performance at various set points

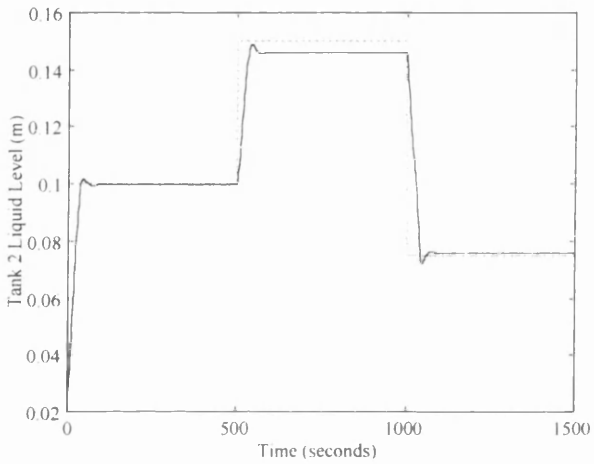


Figure 5.17 Response of system to a single controller around a single set-point

5.6 Summary and Discussion

In this chapter, the learning through direct interaction with the environment procedure was extended and a method for working with complex and large systems was developed. First, the environment was replaced with plant response data. Using the convolution approach means that no identification of the plant is necessary. The response data of the system to a step can be taken as the model. However it was identified that since the input-output data is not representative of the whole operating range of the plant, a single controller performed inadequately for set-points outside the response data.

To overcome this, a hierarchical structure for controller design was developed. This not only allowed for learning controllers outside the step response data, but can also be used to learn larger and more complex systems. The idea is to decompose the system into subsystems, or in the case of operating under different set-points, decompose the operating range into sub-regions and work at a lower level on each subsystem or sub-region.

All the levels in the hierarchy are neurofuzzy networks where the higher level performs a mapping from an input space to an output space, and the lower level maps the inputs to control outputs. At the top level, the global search space is handled by the messy genetic algorithm, and it regulates the flow of information to the lower levels, while at lower levels RL is used to learn network structure and tune parameters of the lower level networks. This enables the system to compensate for local changes in the environment. Through examples, it has been shown that this adaptive hierarchical structure is a flexible and efficient approach to autonomous and globally optimal fuzzy control design.

Chapter 6

Conclusion and Future Research

One never notices what has been done;
one can only see what remains to be done.
- Marie Curie

6.1 Conclusions

The aim of this thesis was not to reinvent fuzzy control but to help control designers build better fuzzy controllers more easily. This has been achieved through Soft Computing (SC) techniques. SC is a discipline that compliments the distinct methodologies of fuzzy logic, neural network, evolutionary algorithms and learning theory. The underlying principle of SC is exploitation of impricision and uncertainty to achieve flexibility, stability and robustness.

The reason for this work is associated with the fact that there appears to exist no systematic design methodology for fuzzy control. The objective is also set out that the design and the designed system should be flexible, autonomous and interactive. This is important because many applications are very complex and time varying, and representing them by traditional mathematical means may be impossible or impractical.

To achieve these goals, it was argued that the learning fuzzy control system be modelled around SC. The important characteristic of SC is that it is not a mixture of FL, NN, EA, RL and other learning theory, but that it is a complementary partnership in which each of the individual methods contribute a distinct methodology for addressing problems in its domain. Initially, the learning fuzzy control system is modelled around a neural network framework because such a network has been established as a fast learner with generalisation capabilities, and it is simple to map onto. However, the existing neural and fuzzy hybrid structures, such as the ANFIS (Jang 1993), NEFCON (Nauck and Kruse 1994) and NEUFUZ (Khalid *et al* 1994) models, have been found to have a number of limitations. For example, the ANFIS model can deal with Sugeno type controllers only, while the Neufuz model lacks the rule modification ability and its performance with slow systems is rather poor. To overcome these inadequacies, the ENFLICT (Evolutionary NeuroFuzzy Learning Intelligent Control Technique) has been developed. The model is flexible in terms of:

- the type of FLC it can represent
- the type of fuzzification
- the defuzzification strategy
- the inferencing mechanism
- the type of memberships that can be used to define the linguistic expressions
- the rule structure that can be used to define the control behaviour.

This is in contrast to other methods where the focus of attention is mainly on fuzzy tuning (Lin and Lee 1991), where only one type of symmetrical fuzzy set can be used (Harris *et al* 1996, Bruske *et al* 1993, Khan 1993), and which is restricted to a single type of inferencing and defuzzification process.

The developed model is more than just a neurofuzzy model. As the name suggests, it also exhibits evolutionary optimisation properties, where global and flexible learning of the FLC and its structure is achieved. Although there have been a number of attempts at using flexible chromosome representations to overcome some of the problems associated with traditional chromosomes, the existing results are not as flexible as ENFLICT even excluding the neural learning feature. For example, a variable length chromosome technique was developed by Carse *et al* to tune the fuzzy sets by adjusting the width and centres of triangular fuzzy sets (Carse *et al* 1996). Since only the centres and widths are adjusted, the fuzzy sets must be predefined symmetrical. Further, the requirement of reordering genes in this approach implies that the number of rules and fuzzy sets per variable is limited.

Hoffman and Pfister (Hoffman and Pfister 1995) used a messy genetic algorithm to encode the rule base using two integer representations. However, in this work the size of the rule base was predefined and there exists no option for tuning the fuzzy sets. An improved version of Goldberg's mGA (Goldberg 1989b) has been used in this thesis because of its attractive flexible coding properties. The improvement is in the gene and chromosome representation. In the improved version, each gene uses integer encoding instead of binary and each single integer value can also hold more than one piece of information about the FLC. This is to overcome the rigid, such as fixed-length, coding deficiency in existing evolutionary-fuzzy hybrids (Takagi and Lee 1993, Kinzel *et al* 1994, Ng 1995, Herrera *et al* 1995). The extension from the pair to three-parameter gene set has meant that it is possible to represent more information within each gene without affecting convergence, while at the same time avoiding chromosomes of large dimensions and redundant code. This also allows ENFLICT to accommodate such non-symmetrical fuzzy sets. ENFLICT assumes no a priori knowledge and requires no restrictions on the number of fuzzy set, rules and shapes, and the types of the fuzzy sets are determined through the evolutionary process.

The ENFLICT model has therefore been developed for the purpose of fine tuning the network parameters. In addition to the flexible structure, significant differences between other neurofuzzy models and the ENFLICT model are that ENFLICT is able to learn:

- online as well as offline
- local and global fuzzy sets
- Mamdani and Sugeno type controllers
- mixed membership functions
- non-symmetrical fuzzy sets
- with different fuzzification and defuzzification strategies
- a global structure followed by local fine tuning.

Learning online has the advantage that the network is able to adapt to real environmental changes, which would otherwise have been difficult to simulate. It also implies that one does not need to collect training data that may be unrepresentative or inaccurate. However, despite these significant strides forward, the cost of using this global and flexible evolutionary algorithm is that it is almost impossible to implement in real time. It is only possible to optimise offline the network connectivity and the type of activation for each node or neuron. This results in a controller that is coarse and further tuning is necessary. The second problem left to tackle is that the standard ENFLICT learning procedure depended on a model being available. One way to overcome these is to alter the rule structure online by directly interacting with the environment. For this purpose reinforcement learning techniques have been employed. Since RL is similar to an evolutionary algorithm, it naturally and smoothly expands the EA's capability into online and offline fine and fast tuning.

There exists a number of reinforcement learning techniques, and it has been argued that the one best suited to the purpose of this work is that of Advantage Reinforcement Learning. This is because it requires no models to be defined and it can work with continuous time systems. Using the ENFLICT model, its gradient descent backpropagation algorithm can be switched on when needed in conjunction with the extended reinforcement learning method.

The problems with learning online are fully addressed in Chapter 4. In supervised learning the goal state from any action is provided by the teacher. However, this can be very time-consuming and inappropriate as what the goal state or the successive should be is unknown. In online or unsupervised learning the system will only know this through exploration. On the other hand, the aim of all methods developed is to make the system behave optimally at each state. Therefore, one is caught compromising between exploration and immediate optimal behaviour. If exploration is compromised, the optimal policy may not be found, and if it is not, then immediate optimality is not likely to be achieved. Although it has been argued that supervised learning is not desirable because of the various difficulties

highlighted in Chapters 2 and 3, it seems that it should not be sidelined altogether. In fact, comparison tests with methods from literature show such evidence. The resulting RL model exhibits the following properties:

- continuous online learning without derivative information
- optimal behaviour at each state
- online control surface modification
- non-lookup table delayed reinforcement exploration

Learning from real plant data as opposed to from a model has been achieved in Chapter 5. This has various advantages:

- the data is truly representative of the plant
- one does not have to use expensive equipment and set-up to learn the controller
- the response to step input allows system performance analysis in terms of transient measures such as steady state errors, rise and settling times.

However, since such response is taken around a set point, it limits the controller to learn to only one set point. To expand to include an entire operating envelope, a Local Controller Network (Gawthrop 1996) has been constructed to allow operation in different conditions or to carry out different tasks. Thus it is possible to learn to control the most complex of systems by breaking it up into sub-components and learning each sub-component independently. While there is a separate controller for each sub-goal, they do not operate sequentially. Therefore, learning for all conditions is continuous until error thresholds are satisfied for all conditions.

6.2 Future Research

Although it is hoped that the aims of this thesis have been fulfilled, it is believed that there is still scope for further work in this area.

- **Stability and Robustness Analysis:** In this work as well as in work reported elsewhere in this field, stability and robustness are guarded by the design criteria in the form of fitness or cost functions, which are validated through simulations in the design optimisation (Li *et al* 1996). Although this is sufficient by human experience and linguistic expression of control behaviour, to a sceptic of fuzzy control it is mathematically insufficient. This is probably one of the major factors that have dogged the progress of fuzzy control, and any work towards a general and formalised design procedure. Therefore, it is believed that a step

forward from the point to which the work has progressed, is to carry out theoretical analysis on robustness and stability for all the methods developed.

- **Digital Control:** Fuzzy control is increasingly implemented by digital means. It is less expensive, more flexible and easier to implement. It is technologically feasible nowadays to implement a FLC on a single silicon chip using the modern system level integration technology. Therefore, another possible direction that can be followed from the work in this thesis is to work towards automation and code generation in addition to the mathematics of an optimal FLC. For example, for portability, platform independence and object orientation, generating Java code for plugging in to a Java interpretable board could be the first step towards the eventual goal of "fuzzy control systems on a chip".

References

AKBARZADEH, M.R., Kumbla, K.K., and Jamshidi, M., 1995. Genetic algorithms in learning fuzzy hierarchical control of distributed parameter systems. In *Proc. IEEE Conference on Systems, Man and Cybernetics*, 1995, pp. 4027-4032. Vancouver.

ALIEV, R.A., Aliev, F.T. and Babaev, M.D., 1992. The synthesis of a fuzzy co-ordinate-parametric automatic control system for an oil-refinery unit. *Fuzzy Sets and Systems*, 1992, vol. 42, no. 2, pp. 157-162.

ALTROCK, C.V., Arend, H.O. and Zimmermann, H.J.,, 1994. Adaptive fuzzy control applied to home heating systems. *Fuzzy Sets and Systems*, 1994, vol. 61, no. 1, pp. 29-36.

ANDERSON, C.W., 1986. *Learning and Problem Solving with Multilayer Connectionist Systems*, PhD thesis, 1986. Amberst, MA: University of Massachussetts,

ANDERSON, C.W., 1989. Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, April 1989, pp. 31-37.

ANDERSON, C.W., 1993. Q-Learning with hidden-unit restarting. In Hanson, S.J., Cowan, J.D. and Giles, C.L. (eds.), *Advances in Neural Information Processing Systems 5*, 1993, pp. 81-88. San Mateo, CA: Morgan Kauffmann

ARJONA, D., 1998. A hybrid neuro-genetic approach to flow calculation based on the representation of an electrical power system by critical switches. *Computers and Artificial Intelligence*, 1998, vol. 17, no. 2-3, pp. 231-247.

ÅSTRÖM, K.J. and Kållström, C.G, 1976. Identification of ship steering dynamics. *Automatica*, 1976, vol. 12, pp. 9-22. Pergamon Press.

BAAKLINI, N., 1976. *Automated Learning Control Using Fuzzy Logic*, Ph.D. Thesis. London, UK: London University.

BACHARACH, J.R., 1992. *Connectionist modeling and control of finite state environments*, PhD Thesis, 1992. Amherst, MA: University of Massachussetts.

BACK, T. and Kursawe, F., 1995. Evolutionary algorithms for fuzzy logic, a brief overview. In Bouchon-Meunier B., Yager R. R., and Zadeh L. A. (eds) *Fuzzy Logic and Soft Computing*, 1995, pp. 3-10. World Scientific.

BÄCK, T., Hoffmeister, F. and Schwefel, H.-P., 1991. A survey of evolution strategies. In Belew, R. and Booker, L., (eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms*, 1991, pp. 2-9. San Mateo, CA, Morgan Kaufmann

BAIRD, L.C., 1993. *Advantage Updating.* (Technical Report WL-TR-93-1146), 1993. Wright-Patterson Air Force Base Ohio: Wright Laboratory.

BAIRD, L.C., 1995. Residual Algoriths: Reinforcement Learning with Function Approximation. In Armand Prieditis and Stuart Russell (eds.), *Machine Learning:*

Proceedings of the Twelfth International Conference, 9-12 July 1995. San Francisco, CA: Morgan Kauffmann

BAKER, J.E., 1985. Adaptive selection methods for genetic algorithms. In Grefenstette, J.J. (ed.), *Proceedings of the First International Conference on Genetic Algorithms*, 1985, pp. 101-111. Hillsdale, NJ: Lawrence Erlbaum Associates.

BALAZINSKI, M., Czogala, E and Bellerose, M., 1994. Application of fuzzy logic techniques to the selection of cutting parameters in machine processes. *Fuzzy Sets and Systems*, 1994, vol. 63, no. 3, pp. 307-317.

BANHRAMI, M, 1994. A new method of training direct neuro-controllers. *1994 IEEE International Conference on Neural Networks*, 1994, vol. 1-7, no. 881, pp. 2655-2660.

BARTO, A.G., 1992. Reinforcement learning and adaptive critic methods. In White, D.A., and Sofge, D.A. (eds.), *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, 1992, pp. 469-491.

BARTO, A.G., Sutton, R. and Anderson, C.W., 1983. Neuron-like elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 1983, vol. 13, pp. 835-846.

BASTIAN, A., 1995. A genetic algorithm for tuning membership functions. In *Proc. Fourth European Conference on Intelligent Techniques and Soft Computing*, 1995, pp. 494-498. Aachen.

BAVARIAN, B., 1988. Introduction to neural networks for intelligent control. *IEEE Control Systems Magazine*, 1988, vol. 8, no. 2, pp. 3-8.

BERENJI, H.R., 1990. Neural networks and fuzzy logic in intelligent control. *Proceedings of IEEE International Symosium Of Intelligent Control*, 1990, pp. 916-919.

BERENJI, H.R., 1992. A reinforcement learning based architecture for fuzzy logic control. *International Journal of Approximate Reasoning*, 1992, vol. 6, no. 2, pp. 267-292

BERENJI, H.R. and Khedkar, P, 1992. Learning and tuning fuzzy logic controllers through reinforcements. *IEEE Trans. on Neural Networks*, 1992, vol. 3, no. 5, pp. 724-740.

BERNARD, J.A., 1988. Use of rule-based system for process control. *IEEE Control Systems Magazine*, 1988, vol. 8, no. 5, pp. 3-13.

BERTRAND, T., 1993. Design of a fuzzy controller for a rotating oscillator. *International Journal of Systems Science*, 1993, vol. 24, no. 10, pp. 1923-1934.

BONARINI, A., 1996. Delayed reinforcement, fuzzy Q-learning and fuzzy logic controllers. In Herrera, F. and Verdegay, J.L. (eds.), *Genetic Algorithms and Soft Computing*, 1996. Physica-Verlag.

BOUSLAMA, F. and Ichiwaka, A., 1992. Fuzzy control rules and their natural control laws. *Fuzzy Sets and Systems*, 1992, vol. 48, no. 1, pp. 65-86.

BROWN, M. and Harris, C.J. 1991. A nonlinear adaptive controller: a comparison between fuzzy logic control and neurocontrol, *IMA Journal Math. Control Inform.*, 1991no 8, pp. 239-265

BROWN, M. and Harris, C.J., 1994. *Neurofuzzy Adaptive Modelling and Control.* Prentice Hall International.

BROWN, M. and Harris, C.J., 1995. A perspective and critique of adaptive neurofuzzy systems used for modelling and control applications. *International Journal of Neural Systems*, 1995, vol. 6, no. 2, pp. 197-220.

BRUSKE, J., Puttkamer, E. and Zimmer, U.R., 1993. SPIN-NFDS learning and pre-set knowledge for surface fusion – a neural fuzzy decision system. *Proceedings of the ANZIIS '93.* 1993, Perth, Australia.

BUCKLEY, J.J. and Hayashi, Y., 1993. Numerical relationship between neural networks, continuous functions and fuzzy systems. *Fuzzy Sets and Systems*, 1993, vol. 60, no. 1, pp. 1-8.

BUCKLEY, J.J. and Hayashi, Y., 1994. Fuzzy genetic algorithms and applications. *Fuzzy Sets and Systems*, 1994, vol. 61, no. 2, pp. 129-136.

BUCKLEY, J.J., 1989. Fuzzy vs. non-fuzzy controllers. *Control and Cybernetics,* 1989, vol. 18, no. 2, pp. 127-130.

BUIJTENEN, M. van, Schram, G., Babuška, R. and Verbruggen, H.B., 1998. Adaptive fuzzy control of satellite attitude by reinforcement learning. *IEEE Transactions on Fuzzy Systems*, May 1998, vol. 6, no. 2, pp. 185-195

CAPONETTO, R., Lavorgna, M., and Presti, M.L., 1996. Automatic fuzzy controller design via GA and neurofuzzy networks. In Chiang, W. and Lee, J. (eds.), *FUZZYLogic for the Application to Complex Systems,* 1996, pp. 380-385. Singapore: World Scientific.

CARSE, B., Fogarty, T.C., and Munro, A., 1996. Evolving fuzzy rule based controllers using genetic algorithms. *Fuzzy Set and Systems*, 1996, vol. 80, pp. 273-294.

CASTRO, J.L., Dalgedo, M., and Herrera, F., 1993, A learning method of fuzzy reasoning by ms, *Proceedings of the First European Congress on Fuzzy and Intelligent Technologies,* 1993, pp. 804-809, Aachen

CHEN, C.L. and Chang, F.Y., 1996. Design and analysis of neural/fuzzy variable structural PID control-systems. *IEEE Proceedings – Control Theory and Applications*, 1996, vol. 143, no. 2, pp. 200-208

CHEN, C.L., Chen, C.K., and Lin, J.M., 1993. Synthesis of fuzzy logic controllers using genetic algorithm. *Journal of Control Synthesis and Technology*, 1993, vol. 1, no. 2, pp. 163-171.

CHEN, C.L. and Chen, W.C.H., 1994. Fuzzy controller design by using neural network techniques. *IEEE Transaction on Fuzzy Systems*, 1994, vol. 2, no. 3.

CHEN, J.Q., Lu, J.H. and Chen, L.J., 1994. An on-line identification algorithm for fuzzy systems. *Fuzzy Sets and Systems*, 1994, vol. 64, no. 1, pp. 63-72.

CHIU, S. and Togai, M., 1988. A fuzzy logic programming environment for real-time control. *International Journal of Approximate Reasoning*, 1988, vol. 2, no. 2, pp. 163-176.

CHO, B.H. and No, H.C., 1996. Design of stability-guaranteed neurofuzzy logic-controller for nuclear steam-generators, *Nuclear Engineering and Design*, 1996, vol. 166, no. 1, pp. 17-29

CHOWDHURY, M and Li, Y, 1996. Messy genetic algorithm based new learning method for structurally optimised neuro-fuzzy controllers. *Proceedings of the IEEE International Conference on Industrial Technology*, December 1996, pp. 274-279. Shanghai, China.

COOPER, M. G., 1995. Evolving a rule based fuzzy controller. *Simulation,* 1995, vol. 65, no. 1, pp. 67-72.

COOPER, M.G. and Vidal, J.J., 1994. Genetic design of fuzzy logic controllers, the cart and jointed pole problem. In *Proc. Third IEEE International Conference on Fuzzy Systems*, 1994, pp. 1332-1337. Orlando.

CORDON, O. and Herrera, F., 1995. A general study on genetic fuzzy systems. In Periaux J., Winter G. Galan M., and Cuesta P. (eds), *Genetic Algorithms in Engineering and Computer Science*, 1995, pp. 33-57. John Wiley and Sons

CORDON, O., Herrera, F., Herrera-Viedma, E., and Lozano M., 1996. Genetic algorithms and fuzzy logic in control processes. *Archives of Control Science*, 1996, vol. 5, no. 1-2, pp. 135-168.

COTTA, C., Alba, E., and Troya, J.M., 1996. Evolutionary design of fuzzy logic controllers. In *Proc. ISIC'96 Conference*, 1996, pp. 127-132. Detroit.

CZOGALA, E. and Rawlik, T., 1989. Modelling of a fuzzy controller with application to the control of biological processes. *Fuzzy Sets and Systems*, 1989, vol. 31, no. 1, pp. 13-22.

DAGLI, C.H. and Schierholt, K., 1997. Evaluating the performance of the genetic neuro scheduler using constant as well as changing crossover and mutation rates. *Computers & Industrial Engineering*, 1997, vol. 33, no. 1-2, pp. 253-256.

DALEY, S. and Gill, K.F., 1987. Attitude control of a spacecraft using an extended self-organising fuzzy logic controller. *Proceedings of the Institute of Mechanical Engineers*, 1987, vol. 201, no. C2, pp. 97-106.

DALEY, S. and Gill, K.F., 1989. Comparison of fuzzy logic controller with a P+D control law. *Transaction of ASME,* 1989, vol. 111, pp. 128-137.

DASGUPTA, D. and McGregor, D.R., 1993. Genetically designing neuro-controllers for a dynamic system. *IJCNN '93-Nagoya: Proceedings of 1993 International Joint Conference on Neural Networks*, 1993, vol. 1-3, no. 718, pp. 2951-2954.

DASGUPTA, D., 1998. Evolving neuro-controllers for a dynamic system using structured genetic algorithms. *Applied Intelligence*, 1998, vol. 8, no. 2, pp. 113-121.

DAVIS, L. (ed.), 1991. *Handbook of Genetic Algorithms*, 1991. New York: Van Nostrand Reinhold.

DAVIS, L. and Steenstrup, M., 1987. Genetic algorithms and simulated annealing: an overview. In Davis, L. (ed.), *Genetic Algorithms and Simulated Annealing*, 1987. San Mateo, CA: Morgan Kaufmann.

DE JONG, K.A., 1985. Genetic algorithms: a 10 year perspective. Grefenstette, J.J., (Ed), *Proceedings of the First International Conference on Genetic Algorithms*, 1985, pp. 169-177. Hillsdale, NJ: Lawrence Erlbaum Associates.

DE JONG, K.A., 1990. Genetic-algorithm-based learning. Kodratoff, Y. and Michalski, R., *Machine Learning: An Artificial Approach*, 1990, vol. 3, pp. 611-638. San Mateo, CA: Morgan Kaufmann.

DESHPANDE, N.A. and Gupta, M.M., 1998. Inverse kinematic neuro-control of robot systems. *Engineering Applications of Artificial Intelligence*, 1998, vol. 11, no. 1, pp. 55-66.

DJUKANOVIC, M.B., Calovic, M.S., Vesovic, B.V. and Sobajic, D.J., 1997. Neuro-fuzzy controller of low head hydropower plants using adaptive-network based fuzzy inference system. *IEEE Transactions on Energy Conversion*, 1997, vol. 12, no. 4, pp. 375-381.

DORF, R.C., 1989. *Modern Control Systems, 5th ed.*, 1989. Addison-Wesley Publishing

DRACOPOULOS, D.C. and Jones, A.J., 1997. Adaptive neuro-genetic control of chaos applied to the attitude control problem. *Neural Computing & Applications*, 1997, vol. 6, no.2, pp. 102-115.

DU, T.C.T. and Wolfe, P.M., 1997. Implementation of fuzzy logic systems and neural networks in industry. *Computers in Industry*, 1997, vol. 32, no. 3, pp. 261-272.

ESHELMAN, L.J., Caruana, R.A., and Schaffer, J.D., 1989. Biases in the crossover landscape. In Schaffer, J. (ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, 1989, pp. 115-122. San Mateo, CA : Morgan Kaufmann.

FENG, W., Chowdhury, M., Brune, T. and Li, Y., 1998. Benchmarks for testing evolutionary algorithms. *Third Asia-Pacific Conference on Control and Measurement*, August 1998, pp. 134-138. Dunhaung, China.

FILIPIC, B. and Juricic D., 1996. A genetic algorithm to support learning fuzzy control rules from examples. In Herrera, F. and Verdegay, J. (eds), *Genetic Algorithm and Soft Computing*, 1996, pp. 403-418. Physica Verlag.

FOGEL, D.B., 1994. An introduction to simulated evolutionary optimization. *IEEE transactions on Neural Networks, special issue on EP*, 1994, vol. 1, no. 5.

FOGEL, D.B., 1995. *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*, 1995. Piscataway, NJ: IEEE Press.

FOSS, B.A. and Johansen, T.A., 1993. On local and fuzzy modelling. In: *3rd Int. Conf. on Industrial Fuzzy Control and Intelligent Systems*, 1993. Houston, Texas.

FUKUDA, T. and Ishigami, H., 1993. Structure optimisation of fuzzy neural network by genetic algorithms. *Proccedings of the Fifth International Fuzzy Systems Association World Congress*, 1993, pp. 964-967. Seoul.

FUKUDA, T., and Shibata, T., 1994. Synthesis of Fuzzy, Artificial Intelligence And Neural Networks For Hierarchical intelligent Control, *Neural and Fuzzy Systems*, 1994, pp. 57-83.

FUKUDA, T., Hasegawa, Y., and Shimojima, K., 1995a. Structure organisation of hierarchical fuzzy model using by genetic algorithm. In *Proceedings of Fourth IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'95)*, 1995, pp. 295-299. Yokohama.

FUKUDA, T., Hasegawa, Y., Shimojima, K., and Saito, F., 1995b. Reinforcement learning method for generating fuzzy controller. In *Proceedings of Second IEEE Conference on Evolutionary Computation (ECIEEE'95)*, 1995, vol. 1, pp. 273-278. Perth

GAWTHROP, P. J., (1996), Continuous-time local model networks, Zbikowski, R and Hunt, K.J., (eds.), *Neural Adaptive Control Technology*, World Scientific Series in Robotics and Intelligent Systems, 1996, vol. 15, p 41-70, Singapore: World Scientific.

GEGOV, A., 1994. Multilevel intelligent fuzzy control of over-saturated urban traffic networks, *International Journal of Systems Science*, 1994, vol. 25, no. 6, pp. 967-978

GLORENNEC, P. Y., 1994. Fuzzy Q-learning and evolutionary strategy for adaptive fuzzy control. In *Proceedings of Second European Conference on Intelligent Techniques and Soft Computing (EUFIT'9J)*, 1994, pp. 35-40. Aachen.

GOLDBERG, D.E., 1985a. Genetic algorithms and rule learning in dynamic control system. In Grefenstette, J.J., (ed.), *Proceedings of the First International Conference on Genetic Algorithms*, 1985, pp. 8-15. Hillsdale, NJ: Lawrence Erlbaum Associates.

GOLDBERG, D.E., 1985b. Dynamic system control using rule learning and genetic algorithms. *Proceedings of the International Joint Conference on Artificial Intelligence*, 1985, vol. 9, pp. 588-592

GOLDBERG, D.E., 1989a. *Genetic algorithms in Search, Optimization and Machine Learning*, 1989. Reading, MA: Addison-Wesley

GOLDBERG, D.E., 1989b. Messy genetic algorithms: motivation, analysis, and first results. *Complex Systems*, 1989, vol. 3, pp. 493-530.

GOLDBERG, D.E., 1990. Messy genetic algorithms revisited: studies in mixed size and scale. *Complex Systems*, 1990, vol. 4, pp. 415-444

GOLDBERG, D.E., and Smith, R.E., 1987. Non-stationary function optimization using genetic algorithms with dominance and dipliody. *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, 1987, pp. 59-68.

GOLDBERG, D.E., Deb, K., and Korb, B., 1991. Do not worry, be messy. Belew, R., and Booker, L., (eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms*, 1991, pp. 24-30. San Mateo, CA: Morgan Kaufmann.

GOMIDE, F., Rocha, A., Albertos, P., 1992. Neurofuzzy controllers. *Low Cost Automation 1992 Techniques Components and Instrumentation Applications*, 1992, vol 13, pp. 13-26.

GONZALEZ, A. and Perez, R., 1996. A learning system of fuzzy control rules based on genetic algorithms. Herrera, F. and Verdegay, J. (eds.), *Genetic Algorithms and Soft Computing*, 1996, pp. 202-225. Physica Verlag.

GRAHAM, B.P. and Newell, R.B., 1988. Fuzzy identification and control of a liquid level rig. *Fuzzy Sets and Systems*, 1988, vol. 26, no. 3, pp. 255-273.

GREFENSTETTE, J.J., 1986. Optimization of control parameters for genetic algorithm. *IEEE Trans. On Systems, Man and Cybernetics*, 1986, vol. 16, no. 1, pp. 122-128.

GREFENSTETTE, J.J., 1992. Genetic algorithms for changing environments. In Manner, R., and Manderick, B. (eds.), *Parallel Problem Solving From Nature*, 1992, vol. 2, pp. 137-144.

GUILLEMIN, P., 1994. Universal motor control with fuzzy logic. *Fuzzy Sets and Systems*, 1994, vol. 63, no. 3, pp. 339-348.

GULLAPALLI, V., Franklin, J.A. and Benbrahim, H., 1994. Acquiring robot skills via reinforcement learning. *IEEE Control Systems Magazine*, February 1994, pp. 13-24.

GUPTA, M.M., 1991. Theory of T-norms and fuzzy inference methods. *Fuzzy sets and systems*, 1991, vol. 40, pp. 431-450.

GUPTA, M.M., 1997. Fuzzy-neural computing systems: recent developments and future directions. *Lecture Notes in Computer Science*, 1997, vol. 1226, pp. 82-91.

HAN, J. and Ham, W., 1995. A GA-fuzzy controller with sliding mode. In *Proceedings of First Korea-Australia Joint Workshop on Evolutionary Computation*, 1995, pp. 199-205. Taejon.

HANEBECK, U. and Schmidt, G., 1994. Optimization of fuzzy networks via genetic algorithms. In *Proceedings of Second European Conference on Intelligent Techniques and Soft Computing*, 1994, pp. 1011-1013. Aachen.

HARMON, M.E., and Baird, L.C., 1996. Multi-player residual advantage learning with general function approximation. *Technical Report WL-TR-96-1065*, 1996. Wright-Patterson Air Force Base Ohio: Wright Laboratory.

HARRIS, C.J., Moore, G.C and Brown, M, 1994, *Intelligent Control: Aspects of Fuzzy Logic and Neural Nets*, 1994, World Scientific, Singapore

HARRIS, C.J., Brown, M., Bossley, K.M., Mills, D.J. and Ming, F., 1996. Advances in neurofuzzy algorithms for real-time modeling and control. *Engineering Applications of Artificial Intelligence*, 1996, vol. 9, no. 1, pp. 1-16.

HERRERA, F., Lozano, M., and Verdegay, J. L, 1994. Applying genetic algorithms to fuzzy optimization problems. *Fuzzy Systems and Artificial Intelligence*, 1994, vol. 3, no. 1, pp. 39-52.

HERRERA, F., Lozano, M., and Verdegay, J. L., 1995a. Tuning fuzzy logic controllers by genetic algorithms. *International Journal of Approximate Reasoning,* 1995, vol. 12, no. 3, pp. 293-315

HERRERA, F., Lozano, M., and Verdegay, J. L., 1995b. Design of a control rule base based on genetic algorithms. In *Proceedings of Sixth International Fuzzy Systems Association World Congress,* 1995, vol. 1, no. 265-268. Sao Paulo.

HIROTA, K. (ed.), 1993. *Industrial Applications of Fuzzy Technology*, 1993. Springer-Verlag.

HIROTA, K., Arai, Y. and Hachisu, S., 1989. Fuzzy controlled robot arm playing two-dimensional ping-pong game. *Fuzzy Sets and System*, 1989, vol. 32, no. 2, pp. 149-159

HISHIYAMA, T., Takagi, T., Yager, R.R., and Nakanishi, S., 1995. Automatic generation of fuzzy inference rules by genetic algorithm. In *Proceedings of Eighth Fuzzy System Symposium,* 1995, pp. 237-240. Hiroshima.

HOFFMANN, F. and Pfister, G., 1994. Automatic design of hierarchical fuzzy controllers using genetic algorithms. In *Proceedings of Second European Conference on Intelligent Techniques and Soft Computing (EUFIT'9X),* 1994, pp. 1516-1522. Aachen.

HOFFMANN, F. and Pfister, G., 1995. A new learning method for the design of hierarchical fuzzy controllers using messy genetic algorithms. In *Proceedings of Sixth International FUZZYSystems Association World Congress,* 1995, vol. 1, pp. 249-252. Sao Paulo.

HOLLAND, J.H., 1975. *Adaptation in Natural and Artificial Systems*, 1975. Ann Arbor: University of Michigan Press.

HOLMBLAD, L.P. and Ostergaard, J.J., 1982. Control of a cement kiln by fuzzy logic. In: Gupta, M.M. and Sanchez, (eds.), *Fuzzy Information and Decision Processes*, 1982, pp. 389-399. North-Holland.

HOMAIFAR, A. and McCormick, V.E., 1992. Full design of fuzzy controllers using genetic algorithms. In Chen, S. S. (ed.), *Neural and Stochastic Methods in Image and Signal* Processing, 1992, vol. 1766, pp. 393-404. San Diego: The International Society of Photo-Optics Instrumentation Engineers.

HORIKAWA, S., Furuhashi, T, and Uchikawa, Y., 1992. On fuzzy modelling using fuzzy neural networks with the back-propagation algorithm. *IEEE Transactions on Neural Networks*, September 1992, vol. 3, no. 4.

HRYCEJ, T., 1992. *Modular Learning in Neural Networks: A Modularised Approach to Neural Network Classification*, 1992. John Wiley and Sons.

HSU, Y. and Cheng, C., 1993. A fuzzy controller for generator excitation control. *IEEE Transactions on Systems, Man, and Cybernetics,* 1993, vol. 23, no. 2, pp. 532-539.

HUANG, S.J. and Hung, C.C., 1996. Genetic-evolved fuzzy systems for inverted pendulum controls. In Chiang, W. and Lee, J. (eds), *Fuzzy Logic for the Applications to Complex Systems*, 1996, pp. 35-40. Singapore: World Scientific.

HUNT, K. J. , Haas, R. and Murray-Smith, R., Extending the functional equivalence of radial basis function networks and fuzzy inference systems, *Trans. IEEE on Neural Networks*, 1996, vol. 7, pp. 776-781.

HUNT, K. J., and Johansen, T. A., Design and analysis of gain-scheduled control using local controller networks, *International Journal Of Control*, 1997, vol. 66, no. 5, pp. 619-651.

HUNT, K.J., 1995. Neurofuzzy. *International Journal of Neural Systems*, 1995, vol. 6, no. 2, pp. 143.

HUNT, K.J., Zbikowski, R., Sbarbaro, D. and Gawthrop, P.J., 1992. Neural networks for control systems – a survey. *Automatica*, 1992, vol. 22, no. 16, pp. 1083-1112.

HWANG, W. R and Thompson, W. E., 1994. Design of fuzzy logic controllers using genetic algorithms. In: *Proceedings of Third IEEE International Conference on Fuzzy Systems*, 1994, pp. 1383-1388. Orlando.

ICHIHASHI, H., Miyoshi, T., Nagasaka, K., Tokunaga, M. and Wakamatsu, 1995. A neurofuzzy approach to variational-problems by using Gaussian membership functions. *International Journal of Approximate Reasoning*, 1995, vol. 13, no. 4, pp. 287-302.

ISHIGAMI, H., Fukuda, T., Shibata, T. and Arai, F., 1995. Structure optimization of fuzzy neural networks by genetic algorithm. *Fuzzy Sets and Systems*, 1995, vol. 71, no. 3, pp. 257-264.

ISHIGAMI, H., Hasegawa, Y., Fukuda, T., and Shibata, T., 1994. Automatic generation of hierarchical structure of fuzzy inference by genetic algorithm. In: *Proceedings of IEEE International Congress on Neural Network*, 1994, pp. 1566-1570. Orlando.

JANG, R., 1993. ANFIS: Adaptive Network-based Fuzzy Inference System. *IEEE Transaction on Systems Man and Cybernetics*, 1993, vol. 23, no. 3, pp. 665-685.

JIA, L.M. and Zhang, X.D., 1994. Distributed intelligent railway traffic control based on fuzzy decisionmaking. *Fuzzy Sets and Systems*, 1994, vol. 62, no. 3, pp. 255-265.

JOHANSEN, T.A., and Foss, B.A., 1992. Nonlinear local model representation for adaptive systems. In: *Proceeding of the Singapore Int. Conf. On Intelligent Control and Instrumentation*, 1992, vol. 2, pp. 677-682.

JUANG, C.F. and Lin, C.T., 1998. An on-line self-constructing neural fuzzy network and its applications. *IEEE Transactions on Fuzzy Systems*, 1998, vol. 6, no. 1, pp. 12-32

JUANG, C.F., and Lin, C.T., 1998. An on-line self-constructing neural fuzzy inference network and its applications. *IEEE Transactions on Fuzzy Systems*, Feb. 1998, vol.6, no. 1.

KAELBLING, L.P., 1991. *Learning in Embedded Systems*, 1991. Cambridge, MA: MIT Press.

KARR, C.L., Freeman, L.M., and Meredith, D.L., 1989, Improved Fuzzy Process Control of Spacecraft Autonomous Rendezvous Using a Genetic Algorithm, *SPIE Workshop on Intelligent control and Adaptive Systems*, 1989, ch 62, vol 1196, pp. 274-288

KARR, C.L., 1991a. Design of a cart-pole balancing fuzzy logic controller using a genetic algorithm. In: *Proceedings of SPIE Conference on the Applications of Artificial Intelligence*, 1991, pp. 26-36. Bellingham.

KARR, C.L., 1991b. Design of an adaptive fuzzy logic controller using a genetic algorithm. In: *Proceedings of Fourth International Conference on Genetic Algorithms*, 1991, pp. 450-456. San Diego

KARR, C.L., 1991c. Genetic algorithms for fuzzy controllers. *AI Expert*, 1991, vol. 6, no. 2, pp. 26-33

KARR, C.L., 1993. Real time process control with fuzzy logic and genetic algorithms. In: *Proceedings of Symposium on Emerging Computer Techniques for the Minerals Industry*, 1993, pp. 31-37. Littleton

KARR, C.L., 1994. Adaptive control with fuzzy logic and genetic algorithms. In Yager, R.R. and Zadeh L.A. (eds.), *Fuzzy Set, Neural Networks, and Soft Computing*, 1994, pp. 345-367. New York: Van Nostrand Reinhold.

KAUR, D. and Lin, B., 1998. On the design of neural-fuzzy control system. *International Journal of Intelligent Systems*, 1998, vol. 13, no. 1, pp. 11-26.

KHALID, M., Omatu, S., Yusof, R., 1994. Adaptive fuzzy control of a water bath process with neural networks. *Engineering Applications of Artificial Intelligence*, 1994, vol. 7, no. 1, pp. 39-52

KHAN, E. and Venkatapuram, P., 1993, NEUFUZ - neural-network-based fuzzy-logic design algorithms, *Second IEEE International Conference On Fuzzy Systems*, 1993, vols 1 and 2, ch 243, pp. 647-654

KHOSLA, R. and Dillon, T., 1997. Learning knowledge and strategy of a neuro-expert system architecture in alarm processing. *IEEE Transactions on Power Systems*, 1997, vol. 12, no. 4, pp. 1610-1618.

KICKERT, W.J.M. and Van Nauta Lemke, H.R., 1976. Application of a fuzzy controller in a warm water plant. *Automatica*, 1976, vol. 12, no. 4, pp. 301-308.

KIM K. C. and Kim J. H., 1995 Evolutionary programming based multicriteria fuzzy expert system. *In Proc. Eight Australian Joint Conference on Artificial Intelligence*, 1995, p 475-482

KIM, S.H, Kim, Y.H., Sim, K.B. and Jeon, H.T., 1993. On developing an adaptive neural-fuzzy control system. *Proceedings of the 1993 international Conference on Intelligent Robots and Systems*, 1993, vol. 1-3, no. 313, pp. 950-957.

KINZEL, J., Klawonn, F. and Kruse, R., 1994 Modifications of Genetic Algorithms for Designing and Optimizing Fuzzy Controllers. *In Proc. IEEE International Conference on Evolutionary Computation*. 1994. Orlando,

KIUPEL, N. and Frank, P.M., 1993. Fuzzy control of steam turbines. *International Journal of Systems Sciences*, 1993, vol. 24, no. 10, pp. 1905-1914.

KLIR, G.J. and Yuan, B., 1995, Fuzzy sets and fuzzy logic, *Prentice Hall Inc.*, New Jersey 1995

KLOPF, A.H., 1988. A neuronal model of classical conditioning. *Psychobiology*, 1988, vol. 16, pp. 85-125.

KOHONEN, T., 1990. The self-organizing map. *Proceedings of the IEEE*, 1990, vol. 78, pp. 1464-1480.

KOSKO, B., 1991. *Neural Networks and fuzzy Systems: A Dynamical System Approach to Machine Intelligence*, 1991. Prentice Hall.

KOZA, J.R., 1992. *Genetic Programming*, 1992. Cambridge, MA: MIT Press.

KOZA, J.R., 1994. *Genetic Programming 2*, 1994. Cambridge, MA: MIT Press.

KRAUSE, B., Altrock, C.V., Limper, K. and Schäfers, W., 1994. A neuro-fuzzy adaptive control strategy for refuse incineration plants. *Fuzzy Sets and Systems*, 1994, vol. 63, no. 3, pp. 329-337

KROPP, K. and Baitinger, U.G., 1993. Optimization of fuzzy logic controller inference rules using a genetic algorithm. In: *Proceedings of First European Congress on Fuzzy and Intelligent Technologies,* 1993, pp. 1090-1096. Aachen

KUNG, Y.S. and Liaw, C.M., 1994. A fuzzy controller improving a linear model following controller for motor drives. *IEEE Transactions on Fuzzy Systems*, 1994, vol. 2, no. 3, pp. 194-202

LARSEN, P.M., 1980. Industrial application of fuzzy logic control. *International Journal of Man Machine Studies*, 1980, vol. 2, no. 1, pp. 3-10.

LEE M.A. and Takagi H., 1993a. Integrating design stages of fuzzy systems using genetic algorithms. In *Proceedings of the Second IEEE International Conference on Fuzzy Systems,* 1993, vol. 2, pp. 612-617. San Francisco.

LEE, C.C., 1990. Fuzzy logic in control systems: fuzzy logic controller – part 1. *IEEE Transactions On Systems, Man and Cybernetics*, 1990, vol. 20, no. 2, pp. 404-418

LEE, H.C. and Jeon, G.J., 1998. A neuro-controller for synchronization of two motion axes, *International Journal of Intelligent Systems*, 1998, vol. 13, no. 6, pp. 571-586.

LEE, K.T., Jean, K.T., and Chen, Y.Y., 1995. Genetic-based reinforcement learning of fuzzy logic control systems. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics,* 1995, vol. 2, pp. 1057-1060. Vancouver

LEE, M.A. and Takagi, H., 1993. Dynamic control of genetic algorithms using fuzzy logic techniques. In: *Proceedings of the Fifth International Conference on Genetic Algorithms*, 1993, pp. 76-83. San Mateo, CA.

LEE, M.H., Lee, S.K.Y and Park, C.H., 1995. Neurofuzzy controller-design using neurofuzzy identifier. *International Journal of Approximate Reasoning*, 1995, vol. 13, no. 4, pp. 269-285.

LEITCH, D. and Probert, P., 1994. Genetic algorithms for the development of fuzzy controllers for autonomous guided vehicles. In: *Proc. Second European Conference on Intelligent Techniques and Soft Computing*, 1994, pp. 464-469. Aachen.

LEITNER, J., Calise, A. and Prasad, J.V.R., 1998. A full authority helicopter adaptive neuro-controller. *1998 IEEE Aerospace Conference Proceedings*, 1998, vol. 2, no. 43, pp. 117-126.

LEUNG, T.P., Qijie, Z., Zhongyuan, M., and Dejing, Y., 1995. An optimization design method of fuzzy logic controller. *Control Theory and Applications (China)*, 1995, vol. 12, no. 4, pp. 491-496.

LI, C.S. and Priemer, R., 1996. Self-learning general purpose PID controller. *Journal of the Franklin Institute – Engineering and Applied Mathematics*, 1996, vol. 334B, no. 2, pp. 167-189.

LI, W., 1997. A method for design of a hybrid neuro-fuzzy control system based on behaviour modeling. *IEEE Transactions on Fuzzy Systems*, February 1997, vol. 5, no. 1, pp. 128-137.

LI, Y and Häußler, A., 1996, Artificial evolution of neural networks and its application to feedback control, *Artificial Intelligence in Engineering*, 1996, no 10, pp. 143-152.

LI, Y. and Ng, K.C., 1995. Genetic algorithm based techniques for design automation of three term fuzzy systems. In *Proceedings of the Sixth International Fuzzy Systems Association World Congress*, 1995, vol. 1, pp. 261-264. Sao Paulo.

LI, Y. and Ng, K.C., 1996. Uniform approach to model-based fuzzy control system design and structural optimisation. In: Herrera, F. and Verdegay, J. (eds.), *Genetic Algorithms and Soft Computing*, 1996, pp. 129-151. Physica Verlag.

LI, Y., Tan, K. and Marionneau, C., 1996. Direct design of uniform LTI controllers from plant I/O data using a parallel evolutionary algorithm. *UKACC Control '96*, 1996, vol. 1, no. 427, pp. 680-686.

LI, Y., Tan, K.C., Marionneau, C., 1996, Direct Design of Uniform LTI Controllers from Plant I/O Data Using a Parallel Evolutionary Algorithm, *UKACC Control '96*, 1996, vol 1, no 427, pp. 680-686.

LI, Y.F. and Lau, C.C., 1989. Development of fuzzy algorithms for servo systems. *IEEE Control Systems Magazine*, 1989, vol. 9, no. 3, pp. 65-72.

LIN, C.T., 1995. A neural fuzzy control-system with structure and parameter learning. *Fuzzy Sets and Systems*, 1995, vol. 70, no. 2-3, pp. 183-212.

LIN, C.T., 1996. A fuzzy adaptive learning control network with on-line structure and parameter learning. *International Journal of Neural Systems*, 1996, vol. 7, no. 5, pp. 569-590

LIN, C.T., and Kan, M.C., 1998. Adaptive fuzzy command acquisition with reinforcement learning. *IEEE Transactions on Fuzzy Systems*, February 1998, vol. 6, no.1.

LIN, C.T. and Lee, C.S.G., 1991. Neural-network-based fuzzy logic control and decision system. *IEEE Transactions on Computers*, 1991, vol. 40, no. 12, pp. 1320-1336.

LIN, C.T. and Lee, C.S.G., 1994. Reinforcement structure/parameter learning for neural network based fuzzy logic control system. *IEEE Trans. Fuzzy Systems*, 1994, vol. 2, no. 1, pp. 46-63

LIN, C.T. and Lu, Y.C., 1996. A neural fuzzy system with fuzzy supervised learning. *IEEE Transactions on Systems Man and Cybernetics Part B – Cybernetics*, 1996, vol. 26, no. 5, pp. 744-763

LIN, L.J., 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 1992, vol. 8, no. 3-4, pp. 293-321.

LIN, S.C. and Chen, Y.Y., 1995a. A GA-based fuzzy controller with sliding mode. In: *Proc. Fourth IEEE International Conference on Fuzzy Systems*, 1995, pp. 1103-1110. Yokohama.

LIN, S.C. and Chen, Y.Y., 1995b. On GA-based optimal fuzzy control. In *Proc. Second IEEE Conference on Evolutionary Computation (EC-IEEB'95)*, 1995, vol. 2, pp. 846-851. Perth.

LING, C, and Edgar, T.F., 1993. A new fuzzy scheduling algorithm for process control. *Asia-Pacific Engineering Journal*, 1993, vol. 3, no. 1-2, pp. 129-142.

LINKENS, D.A. and Nie, J.H., 1994. Backpropagation neural-network-based fuzzy controller with a self-learning teacher. *International Journal of Control*, 1994, vol. 60, no. 1, pp. 17-39

LINKENS, D.A. and Nyongesa, H.O., 1995a. Evolutionary learning in fuzzy neural control systems. In: *Proc. Third European Conference on Intelligent Techniques and Soft Computing*, 1995, pp. 990-995. Aachen

LINKENS, D.A. and Nyongesa, H.O., 1995b. Evolutionary learning in fuzzy neural control systems. *Proceedings of the Third European Congress on Intelligent Techniques and Soft Computing, EUFIT '95*, 1995, pp. 28-31. Aachen

LINKENS, D.A. and Okola, H., 1992. A real time genetic algorithm for fuzzy control. In *Proc. IEE Colloquium on Genetic Algorithm for Control and Systems Engineering*, 1992, pp. 106, London.

LISKA, J. and Melsheimer, S.S., 1994. Complete design of fuzzy logic systems using genetic algorithms. In: *Proc. Third IEEE International Conference on Fuzzy Systems*, 1994, pp. 1377-1382. Orlando

LIU, T.S. and Wu, J.C., 1993. A model for rider-motorcycle system using fuzzy control. *IEEE Transactions on Systems, Man, and Cybernetics*, 1993, vol. 23, no. 1, pp. 257-276.

MAGDALENA, L. and Velasco, J.R., 1996. Fuzzy rules-based controllers that learn by evolving their knowledge base. In: Herrera, F. and Verdegay, J. (ed.), *Genetic Algorithms and Soft Computing*, 1996, pp. 172-201. Physica Verlag

MAMDANI, E.H., 1974. Applications of fuzzy algorithm for simple dynamic plant. *Proceedings IEE*, 1974, vol. 121, no. 12, pp. 1585-1588

MATTHEWS, N.D., An, P.E., Roberts, J.M. and Harris, C.J., 1998. A neurofuzzy approach to future intelligent driver support systems. *Proceedings of the Institution of Mechanical Engineers Part D – Journal of Automobile Engineering*, 1998, vol 212, no. D1, pp. 43-58

MEDDAH, D.Y. and Benallegue, A., 1997. A stable neuro-adaptive controller for rigid robot manipulators. *Journal of Intelligent & Robotic Systems*, 1997, vol. 20, no. 2-4, pp. 181-193

MICHALEWICZ, Z. and Janikow, C., 1991. Genetic algorithms for numerical optimization. *Statistics and Computing*, 1991, vol. 1, no. 1.

MICHALEWICZ, Z. and Janikow, C., 1992. GENOCOP: A genetic algorithm for numerical optimization problems with linear constraints. *Communications of the ACM*, 1992

MICHALEWICZ, Z., 1996. *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd Revised and Extended Edn., 1996. Springer-Verlag.

MILLAN, J.D.R., and Torras, C., 1992. A reinforcement connectionist approach to robot path finding in non maze-like environments. *Machine Learning*, 1992, vol. 8, pp. 363-395.

MOHAMMADIAN, M. and Stonier, R.J., 1994. Tuning and optimisation of membership functions of fuzzy logic controllers by genetic algorithms. In: *Proc. Third IEEE International Workshop on Robot and Human Communication*, 1994, pp. 356-361. Nagoya

MOORE, A.W., and Atkeson, C.G., 1993. Memory-based reinforcement learning: Efficient computation with prioritized sweeping. In Hanson, S.J., Cowan, J.D., and Giles, C.L. (eds.), *Advances in Neural Information Processing Systems 5*, 1993, pp. 263-270. San Mateo, CA: Morgan Kauffmann.

MURAKAMI, S. and Maeda, M., 1985. Automobile speed control system using a fuzzy logic controller. In: Sugeno, M. (Ed.), *Industrial Applications of Fuzzy Control*, 1985, 105-123. Amsterdam: North Holland.

MURRAY-SMITH, R., 1994. Local model networks and local learning. In: *Fuzzy Duisburg, '94*, 1994, pp. 404-409

NAUCK, D., 1997. Neuro-fuzzy systems: review and prospects. In: *Proc. 5th European Congress on Intelligent Techniques and Soft Computing (EUFIT '97)*, Sept. 1997, pp. 1044-1053. Aachen.

NAUCK, D. and Kruse, R., 1994. NEFCON-I: An X-Window based simulator for neural-fuzzy controllers. *Proc. IEEE Int. Conference on Neural Networks*, June 1994, pp. 1638-1643. Orlando, Florida.

NAUCK, D. Kruse, R, and Stellmach, R., 1995, New Learning Algorithms for the Neuro-Fuzzy Environment NEFCON-I, *Third German GI-Workshop "Fuzzy-Neuro-Systeme'95"*, Darmstadt, Germany, November 1995, pp. 15-17,

NEDUNGADI, A., 1993. A fuzzy logic-based robot controller. *Journal of Intelligent & Fuzzy Systems*, 1993, vol. 1, no. 3, pp. 243-251

NG, K.C., 1995. *Switching Control Systems and Their Design Automation via Genetic Algorithms*, PhD Thesis, 1995. University of Glasgow.

NG, K.C. and Li, Y., 1994. Design of sophisticated fuzzy logic controllers using genetic algorithms. In: *Proc. Third IEEE International Conference on Fuzzy Systems*, 1994, pp. 1708-1712. Orlando. Florida.

NG, K.C. and Li, Y., 1995, Genetic Algorithm Based Techniques for Design Automation of Three-Term Fuzzy Systems, *Proceedings. 6th International Conference on Fuzzy Systems*, August 1995, Sao Paulo, Brazil

NG, K.C., Li, Y., Murray-Smith, D.J. and Sharman, K.C., 1995. Genetic algorithms applied to fuzzy sliding mode controller design, *First IEE/IEEE International Conference on GA in Engineering System: Innovations and Applications*, University of Sheffield, 1995a, pp. 220-225

NG, K.C., Li, Y., and Murray-Smith, D.J., Performance based linear control system design by genetic evolution with simulated annealing, *34th IEEE Conference on Decision and Control*, Dec. 1995b, New Orleans, LA

NGO, C.Y. and Li, V.O.K., 1994. Freeway traffic control using fuzzy logic controllers. *Information Sciences: Applications*, 1994, vol. 1, no. 2, pp. 59-76

NIE, J. and Linkens, D.A., 1993. Learning control using fuzzified self-organizing radial basis function network. *IEEE Transactions on Fuzzy Systems*, 1993, vol. 1, no. 4, pp. 280-287.

NOMURA, H., Hayashi, I. and Noboru, W., 1992. A learning method of fuzzy inference rules by descent method. In: *Proceedings of IEEE International Conference on Fuzzy Systems*, 1992, pp. 203-210, San Diego.

NORIEGA, J.R. and Wang, H., 1998. A direct adaptive neural network control for unknown nonlinear systems and its applications. *IEEE Transactions On Neural Networks*, 1998, vol. 9, no. 1, pp. 27-34

OHTANI, Y and Yoshimura, T., 1994. Fuzzy control of a manipulator using the switching motion of brakes. *International Journal of Systems Science*, 1994, vol 25, no 6, pp. 979-989

OMATU, S. and Ide, T., 1994. Stabilization of inverted pendulum by neuro-control. *1994 IEEE International Conference on Neural Networks*, 1994, vol. 1-7, no. 881, pp. 2367-2372

OSTERTAG, E. and Carvalho-Ostertag, M.J., 1993. Fuzzy control of an inverted pendulum with fuzzy compensation of friction forces. *International Journal of Systems Sciences*, 1993, vol. 24, no. 10, pp. 1915-1922.

PALM, R, 1989. Fuzzy controller for a sensor guided robot manipulator. *Fuzzy Sets and Systems*, 1989, vol. 31, no. 2, pp. 133-149.

PALM, R., Driankov, D., Hellendoorn, H., 1996. *Model Based Fuzzy Control*, 1996. Springer-Verlag

PAPPIS, C.P. and Mamdani, E.H., 1977. A fuzzy logic controller for a traffic junction. *IEEE Transactions on Systems, Man, and Cybernetics*, 1977, vol. 7, no. 10, pp. 707-717

PERNEEL C., Themlin J. M., Renders J. M., and Acheroy M. 1995, Optimization of fuzzy expert systems using genetic algorithms and neural networks. *IEEE Transactions on Fuzzy Systems*, 1995, vol 3, no 3, pp. 300--312.

POPOVIC, D. and Xiong N., 1996. Design of flexible structured fuzzy controllers using genetic algorithms. In *Proc. Fifth IEEE International Conference on Fuzzy Systems*, 1996, vol. 3, pp. 1682-1685. New Orleans

PROCYK, T. and Mamdani, E., 1979. A linguistic self-organising process controller. *Automatica*, 1979, vol. 15, pp. 15-30.

RAO, D.H. and Gupta, M.M., 1994. Neuro-fuzzy controller for control and robotics applications. *Engineering Applications of Artificial Intelligence*, 1994, vol. 7, no. 5, pp. 479-491

RATTRAY, M. and Saad, D., 1997. Globally optimal on-line learning rules for multi-layer neural networks, *Journal of Physics*, 1997, vol 30, pp. 771-776

RAY, K.S. and Majumder, D.D., 1984. Application of circle criteria for stability analysis of linear SISO and MIMO systems associated with fuzzy logic controller. *IEEE Transactions on Systems, Man, and Cybernetics*, 1984, vol. 14, no. 2, pp. 345-349

RAY, K.S. and Majumder, D.D., 1985. Fuzzy logic control of a non-linear multivariable steam generating unit using decoupling theory. *IEEE Transactions on Systems, Man, and Cybernetics*, 1985, vol. 15, no. 4, pp. 539-558

RENDERS, J.-M., and Bersini, H., 1994. Hybridizing genetic algorithms with hill-climbing methods for global optimization: two possible ways. In: Michalewicz, Z, Schaffer, D., Schwefel, H.-P., Fogel, D. and Kitano, H. (eds.), *Proceedings of the First IEEE*

International Conference on Evolutionary Computation, 27-29 June 1994, vol. 1, pp. 312-317. Orlando

RENHOU, L. and Yi, Z., 1996. Fuzzy logic controller based on genetic algorithms. *Fuzzy Sets and Systems*, 1996, vol. 83, pp. 1-10.

ROSS, T.J., 1995. *Fuzzy Logic With Engineering Applications*, 1995. McGraw-Hill

ROSS, T.J., Hasselman, T.K., Chrostowski, J.D. and Verzi, S.J., 1993. Fuzzy set methods for assessing uncertainty in the modelling and control of space structures. *Journal of Intelligent & Fuzzy Systems*, 1993, vol. 1, no. 2, pp. 135-155

ROY, A., Govil, S. and Miranda, R., 1997. A neural network learning theory and a polynomial time RBF algorithm, *IEEE Transactions on Neural Networks*, 1997, vol. 8, no. 6, pp. 1301-1313

SANCHEZ, E., 1992. Genetic algorithms, neural networks and fuzzy logic systems. In: *Proc. Second International Conference on Fuzzy Logic and Neural Network*, 1992, pp. 17-19. Iizuka.

SANNER, R.M., and Slotine, J.J., 1992. Gaussian networks for direct adaptive control. *IEEE Transactions on Neural Networks*, 1992, vol. 2, pp. 837-863

SASAKI, T. and Akiyama, T., 1988. Traffic control process of expressway by fuzzy logic, *Fuzzy Sets and Systems*, 1988, vol26, no 2, pp. 165-178.

SCHARF, E.M., and Mandic, N.J., 1985. The application of a fuzzy controller to the control of a multi-degree-of-freedom robot arm. In: Sugeno, M. (ed.), *Industrial Applications of Fuzzy Control*, 1985, pp. 41-61. Amsterdam: North Holland.

SCHMITZ, G.P.J. and Aldrich, C., 1998. Neurofuzzy modeling of chemical process systems with ellipsoidal radial basis function neural networks and genetic algorithms. *Computers & Chemical Engineering*, 1998, vol. 22, no. SS, pS1001-S1004

SCHWEFEL, H-P., 1995. *Evolution and Optimum Seeking*, 1995. Chichester, UK: John Wiley.

SHAOUT, A. and Quail, A., 1997. Applications of fuzzy logic in household appliances. *International Journal of Computer Applications in Technology*, 1997, vol. 10, no. 5-6, pp. 361-369.

SHEPANSKY, J.F., and Macy, S.A., 1987. Teaching artificial neural systems to drive: Manual training techniques for autonomous systems. In *Proceedings of the First Annual International Conference on Neural Networks*, 1987. San Diego, CA.

SHIJOJIMA, K., Hasegawa, Y., and Fukuda, T., 1995. Unsupervised/supervised learning for rbf-fuzzy system, adaptive rules, membership functions and hierarchical structure by genetic algorithm. In Furuhashi, T. (ed), *Advances in Fuzzy Logic, Neural Networks and Genetic Algorithms: Proc. 199J IEEE/Nayoya University World Wide Wisepersons Selected Papers, LNAI 1011*, 1995, pp. 127-147. Berlin: Springer-Verlag.

SINGH, S.P., 1992. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 1992, vol. 8, no. 3-4, pp. 323-329.

SOLLA, S.A., Levin, E., and Fleisher, M, 1988, Accelerated Learning in Layered Neural Networks, *Complex Systems*, 1988, vol 2, pp. 625-639.

SPINRAD, M.D., 1991. Self-learning fuzzy control. *Proceedings of the ISA 91 International Confernce and Exhibition*, 1991, vol. 46, no. 169, pp. 1247-1260.

SPOONER, J.T. and Passino, K., 1996. Stable adaptive control using fuzzy systems and neural networks. *IEEE Transactions on Fuzzy Systems*, 1996, vol. 4, no. 3, pp. 339-359

SRINIVAS, M. and Patnaik, L.M., 1994. Genetic algorithms, a survey. *IEEE Computer*, 1994, vol. 27, no. 6, pp. 17-26

SUBUDHI, B. and Swain, A.K., 1995. Genetic algorithm based fuzzy logic controller for real time liquid level control. *Journal Inst. Engineering*, 1995, vol. 76, pp. 96-100.

SUGENO, M. (ed.), 1985. *Industrial Applications of Fuzzy Control*, 1985, North-Holland.

SUGENO, M. and Kang, G.T., 1986. Fuzzy modelling and control of multilayer incinerator. *Fuzzy sets and Systems*, 1986, vol. 18, no. 3, pp. 329-346

SUGENO, M. and Takagi T., 1985. Fuzzy identification of systems and its applications to modelling and control. *IEEE Transactions on Systems, Man and Cybernetics*, 1985, vol. 15, no. 1, pp. 116-132

SUNDARESHAN, M.K. and Condarcure, T.A., 1998. Recurrent neural-network training by a learning automation approach for trajectory learning and control system design. *IEEE Transactions on Neural Networks*, 1998, vol. 9, no. 3, pp. 354-368.

SURMANN, H., Kanstein, A., and Goser, K., 1993, Self-organising and genetic algorithms for an automatic design of fuzzy control and decision systems, *Proceedings of the First European Congress on Fuzzy and Intelligent Technologies*, 1993, pp. 1097-1104, Aachen

SUTTON, R.S., 1988. Learning to predict by the method of temporal differences. *Machine Learning*, 1988, vol. 3, pp. 9-44.

SYSWERDA, G., 1989. Uniform crossover in genetic algorithms. Schaffer, J., (ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, 1989, 2-9. San Mateo, CA: Morgan Kaufmann.

TAKAGI, H., 1993. Fusion techniques of fuzzy systems and networks and fuzzy systems and genetic algorithms. *Proc. SPIE Int. Soc. Opt. Eng.*, 1993, vol. 2061, pp. 402-413

TAKAGI, H. and Hayashi, I., 1991. NN-driven fuzzy reasoning. *International Journal of Approximate Reasoning*, 1991, vol. 5, pp. 191-212

TAKAGI, H. and Lee, M.A., 1993. Neural network and genetic algorithm approaches to auto-design of fuzzy systems. In Klement, E.P. and Slany, W. (eds.), *Lecture Notes in Artificial Intelligence*, 1993, vol. 695, no. 6F79. Springer-Verlag.

TAKAGI, T. and Sugeno M., 1983. Derivation of fuzzy control rules from human operator's control actions. *Proceedings of the IFAC Symposium on Fuzzy Information, Knowledge Representation and Decision Analysis*, 1983, 55-60

TANG, K.L. and Mulholland, R.J., 1987. Comparing fuzzy logic with classical controller designs. *IEEE Transactions on Systems, Man, and Cybernetics*, 1987, vol. 17, no. 6, pp. 1085-1087.

TARNG, Y.S., Yeh, Z.M., and Nian, C.Y., 1996. Genetic synthesis of fuzzy logic controllers in turning. *Fuzzy Sets and Systems*, 1996, vol. 83, pp. 301-310.

TETTAMANZI, A.G., 1995. An evolutionary algorithm for fuzzy controller synthesis and optimisation. In: *Proc. IEEE Conference on Systems, Man and Cybernetics*, 1995, pp. 4021-4026. Vancouver

THAM, C.K. and Prager, R.W., 1994. A modular Q-learning architecture for manipulator task decomposition. In Cohen, W.W. and Hirsch, H. (eds.), *Machine Learning: Proceedings of the Eleventh International Conference*, 1994. NJ: Morgan Kauffmann.

THRIFT, P., 1991, Fuzzy logic synthesis with genetic algorithms, *Proceedings of the Fourth International Conference on Genetic Algorithms*, 1991, pp. 509-513, Morgan Kaufmann

THRUN, S.B., 1993. Issues in using function approximation for reinforcement learning. In *Proceedings of the Fourth Connecticut Models Summer School*, 1993. Hillsdale, NJ: Lawrence Erlbaum.

TOBI, T. and Hanafusa, T., 1991. A practical application of fuzzy control for an air-conditioning system. *International Journal of Approximate Reasoning*, 1991, vol. 5, no. 3, pp. 331-348.

TOBI, T., Hanafusa, T., Ito, S. and Kashiwagi, N., 1992. The application of fuzzy control to a coke oven gas cooling plant. *Fuzzy Sets and Systems*, 1992, vol. 46, no. 3, pp. 373-381

TOKHI, M.O. and Wood, R., 1997. Active noise control using multi-layered perceptron neural networks. *Journal of Low Frequency Noise Vibration and Active Control*, 1997, vol. 16, no. 2, pp. 109-144

TONG, R.M., Beck, M.B. and Latten, A., 1980. Fuzzy control of the activated sludge wastewater treatment process. *Automatica*, 1980, vol. 16, no. 6, pp. 695-701.

TÖNSHOFF, H.K. and Walter, A, 1994. Self-tuning fuzzy-controller for process control in internal grinding. *Fuzzy Sets and Systems*, 1994, vol. 63, no. 3, pp. 359-373

UHRIG, R.E. and Tsoukalas, L.H., 1998. Neurofuzzy approaches and their applications to nuclear power systems. *Computers and Artificial Intelligence*, 1998, vol. 17, no. 2-3, pp. 169-188

UMBERS, I.G. and King, P.J., 1980. An analysis of human decision-making in cement kiln control and the implications for automation. *International Journal of Man-Machine Studies*, 1980, vol. 12, no. 1, pp. 11-23

URAGAMI, A., Mizumoto, M., and Tanaka, K., 1976. Fuzzy robot controls. *Journal of Cybernetics*, 1976, vol. 6, no. 1-2, pp. 39-64

UTKIN, V.J., 1977. Variable structure systems: a survey. *IEEE Trans. Automatic Control*, 1977, vol. 22, pp. 212-222.

ZIKIDIS, K.C., and VASILAKOS, A.V., 1996, A novel, neuro-fuzzy architecture for fuzzy computing, based on functional reasoning, *Fuzzy Sets and Systems*, 1996, vol.83, no.1, pp.63-84

VELASCO, J.R. and Magdalena, L., 1995. Genetic learning applied to fuzzy rules and fuzzy knowledge bases. In: *Proc. Sixth International Fuzzy Systems Association World Congress*, 1995, vol. 1, pp. 257-260. Sao Paulo.

VONALTROCK, C., 1994. Neurofuzzy technologies. *Computer Design*, 1994, vol. 33, no. 8, pp. 82-83.

WANG , L., 1994. *Adaptive Fuzzy Systems and Control Design and Stability Analysis*, 1994. Prentice Hall.

WANG, L.X., 1993. Stable adaptive fuzzy control of nonlinear systems. *IEEE Transactions on Fuzzy Systems*, 1993, vol. 1, no. 2, pp. 146-155

WANG, P. and Kwok, D. P., 1992. Optimal fuzzy PID control based on genetic algorithms. In *Proc. International Conference* on *Industrial Electronics, Control, and Instrumentation*, 1992, vol. 2, pp. 977-981. San Diego.

WANG, Y.Y. and Kim, H., 1995. Implementing adaptive fuzzy logic controllers with neural networks: a design paradigm. *Journal of Intelligent and Fuzzy Systems*, 1995, vol. 3, pp. 165-180

WATKINS, C.J., 1989. *Learning from Delayed Rewards*, PhD Thesis, 1989. Cambridge, UK: Cambridge University

WATTA, P.B., Hassoun, M.H. and Meisel, J., 1996. Design of optimal neuro-controllers for the separately excited dc motor using a hybrid genetic algorithm -- neural network approach. *Proceedings of the Society of Photo-Optical Instrumentation Engineers (SPIE)*, 1996, vol. 2760, no. 79, pp. 230-241.

WHITELY, D., 1993. Genetic reinforcement learning for neurocontrol problems. *Machine Learning*, 1993, vol. 13, pp. 259-284

WHITLEY, D., 1994. Genetic algorithms: a tutorial. In Michalewicz, Z. (Ed.), *Statistics & Computing*, Special issue on Evolutionary Computation, 1994.

WIELAND, F. and Aliev, F., 1996. Neuro-fuzzy-genetic adaptive control system. In *Proc. Fourth European Conference on Intelligent Techniques and Soft Computing (EUFIT'96)*, 1996, pp. 747-751. Aachen.

WOLF, T., 1994. Optimization of fuzzy systems using neural networks and genetic algorithms. In: *Proc. Second European Conference on Intelligent Techniques and Soft Computing*, 1994, pp. 544-551. Aachen.

WONG, C., Chou, C. and Mon, D., 1993. Studies on the output of fuzzy controller with multiple inputs. *Fuzzy Sets and Systems*, 1993, vol. 57, no. 2, pp. 149-158.

WU, J.C. and Liu, T.S., 1996. A sliding-mode approach to fuzzy control design. *IEEE Transactions on Control Systems Technology*, 1996, vol. 4, no. 2, pp. 141-151.

WU, Z.Q., 1990. The application of fuzzy control theory to an oil-fueled annealing surface. *Fuzzy Sets and Systems*, 1990, vol. 36, no. 1, pp. 145-156.

XU, C.W., 1989. Decoupling fuzzy relational systems: an output feedback approach. *IEEE Transactions on Systems, Man, and Cybernetics*, 1989, vol. 19, no. 2, pp. 414-418

YAGER, R.R., 1992. Implementing fuzzy logic controllers using a neural network framework. *Fuzzy Sets and Systems*, 1992, vol. 48, no. 1, pp. 53-64

YAMAZAKI, T. and Sugeno, M., 1984. Self organizing fuzzy controller, 1984. *Transactions of the Society for Instrument Control Engineers*, 1984, vol. 20, no. 8, pp. 720-726.

YING, H., 1994. Analytical structure of a two-input two-output fuzzy controller and its relation to PI and multilevel relay controllers. *Fuzzy Sets and Systems*, 1994, vol. 63, no. 1, pp. 21-33

YU, C., Cao, Z. and Kandel, A., 1990, Application of fuzzy reasoning to the control, of an activated sludge plant. *Fuzzy Sets and Systems*, 1990, vol. 38, no. 1, pp. 1-14

ZADEH, L.A., 1965. Fuzzy sets. *Information and Control*, 1965, vol. 8, pp. 338-353

ZADEH, L.A., 1973. Outline of a new approach to the analysis of complex systems and decision process. *IEEE Transactions On Systems, Man and Cybernetics*, 1973, vol. 3, no. 1, pp. 28-44

ZEINSABATTO, S., Alsmadi, O. and Kuschewski, J.G., 1996. An intelligent neuro-controller based on system parameter-estimation. *Proceedings of the IEEE SouthEastCon '96*, 1996, vol. 146, pp. 517-520

ZHANG, Y.Q. and Kandel, A., 1998a. Compensatory neurofuzzy systems with fast learning algorithms. *IEEE Transactions on Neural Networks*, 1998, vol. 9, no. 1, pp. 83-105

ZHANG, Y.Q. and Kandel, A., 1998b, Compression and expansion of fuzzy rule bases by using crisp-fuzzy neural networks. *Cybernetics and Systems*, 1998, vol. 29, no. 1, pp. 5-34

ZHOU, Y.S. and Lai, L.Y., 1995. Optimal design of fuzzy controllers by genetic algorithms. In: *Proc. IEEE/IAS International Conference on Industrial Automation and Control, Emerging Technologies,* 1995, pp. 429-435. Taipei

Appendix A

The Backpropagation Algorithm

Suppose that a given feed-forward network in layered representation has L-layers and layer i (i = 0,1, ..., L; i=0 represents the input layer) has $N(i)$ nodes. Then the output and function nodes $i(i = 1, \ldots, N(i))$ of layer i can be represented as $x_{l,i}$ and $f_{l,i}$, respectively, as shown in Figure A.1. Without loss of generality, we assume there are no jumping links, that is, links connecting non-consecutive layers.

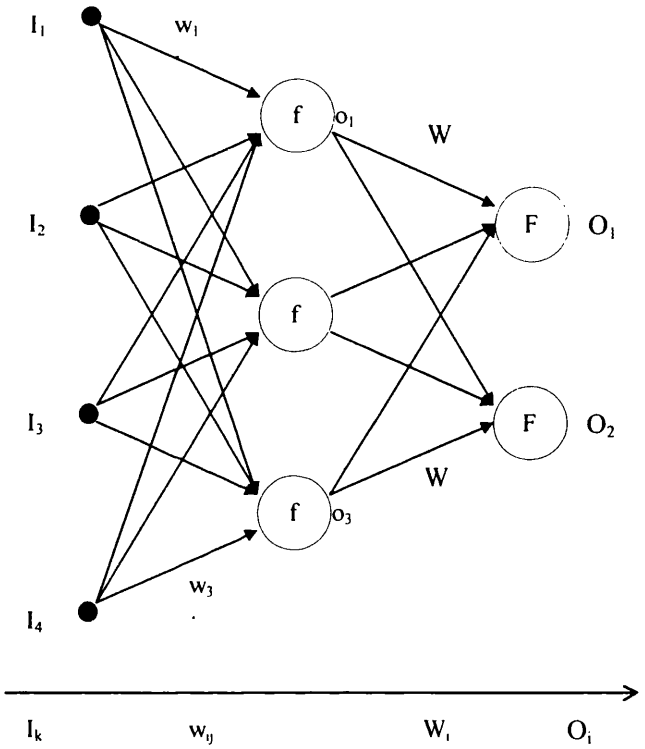


Figure A.1 2-layer feed-forward network

Given an input vector P, the input at the hidden unit j is

$$h_j^P = \sum_k w_{jk} I_k^P \tag{A.1}$$

where I_k^p is the input signal k belonging to an input vector P. The output from the same hidden unit j is obtained by

$$o_j^p = f(h_j^P) = f\left(\sum_k w_{jk} I_k^P\right) \tag{A.2}$$

Output unit I thus receives a net input

$$h_i^P = \sum_j W_{ij}\, o_j^P = \sum_j W_{ij} f\left(\sum_k w_{jk}\, I_k^P\right) \tag{A.3}$$

which produces the final output

$$O_i^P = F(h_i^P) = F\left(\sum_j W_{ij}\, o_j^P\right) = F\left(\sum_j W_{ij} f\left(\sum_k w_{jk}\, I_k^P\right)\right) \tag{A.4}$$

The overall error measure at the output is

$$E = \frac{1}{2}\sum_P \sum_i \left[T_i^P - O_i^P\right]^2 = \frac{1}{2}\sum_P \sum_i \left[T_i^P - F\left(\sum_j W_{ij} f\left(\sum_k w_{jk}\, I_k^P\right)\right)\right]^2 \tag{A.5}$$

Provided the activation functions are differentiable, using the chain rule the change in weights at the output layer is found from

$$\Delta W_{ij} = -\eta \cdot \frac{\partial E}{\partial W_{ij}} = \eta \sum_P \left[T_i^P - O_i^P\right] \cdot \frac{\partial F\left(h_i^P\right)}{\partial h_i^P} \cdot \frac{\partial h_i^P}{\partial W_{ij}} \tag{A.6}$$

where η is known as the learning rate and is used to decide how fast the weights are allowed to change for each time step. Substituting equation A.3 into A.6

$$\Delta W_{ij} = \eta \sum_P \left[T_i^P - O_i^P\right] \cdot \frac{\partial F\left(h_i^P\right)}{\partial h_i^P} \cdot o_j^P = \sum_P \delta_i^P o_i^P \tag{A.7}$$

where

$$\delta_i^P = \left[T_i^P - O_i^P\right] \cdot \frac{\partial F\left(h_i^P\right)}{\partial h_i^P} \tag{A.8}$$

The updated weight W_{ij} is found from $W_{ij} = W_{ij} + \Delta W_{ij}$. The change in weight in the hidden layer is found in a similar way

$$\Delta w_{jk} = -\eta \cdot \frac{\partial E}{\partial w_{jk}} = \eta \sum_P \sum_i \left[T_i^P - O_i^P\right] \cdot \frac{\partial F\left(h_i^P\right)}{\partial h_i^P} \cdot \frac{\partial h_i^P}{\partial W_{jk}} \tag{A.9}$$

Using the chain rule, and from equations A.1, A.2 and A.3

$$\Delta w_{jk} = \eta \sum_P \sum_i \left[T_i^P - O_i^P\right] \cdot \frac{\partial F\left(h_i^P\right)}{\partial h_i^P} \cdot W_{ij} \cdot \frac{\partial f\left(h_j^P\right)}{\partial h_j^P} \cdot \frac{\partial h_i^P}{\partial W_{jk}} \tag{A.10}$$

$$= \eta \sum_P \sum_i \left[T_i^P - O_i^P\right] \cdot \frac{\partial F\left(h_i^P\right)}{\partial h_i^P} \cdot W_{ij} \cdot \frac{\partial f\left(h_j^P\right)}{\partial h_j^P} \cdot I_k^P \tag{A.11}$$

$$= \eta \sum_P \sum_i \delta_i^P \cdot W_{ij} \cdot \frac{\partial f\left(h_j^P\right)}{\partial h_j^P} \cdot I_k^P \tag{A.12}$$

$$= \eta \sum_P \delta_{ji}^P \cdot I_k^P \tag{A.13}$$

The updated weight w_{jk} is found from $w_{jk} = w_{jk} + \Delta w_{jk}$. The update rule can be generalised for an arbitrary number of layers

$$\Delta w_{mn} = \eta \sum_P \delta_{output} \cdot V_{input} \tag{A.14}$$

where V_{input} is the input to the layer being considered, and m and n are the connection of the two ends being considered.

Appendix B

The Coupled Tank System

The system consists of a glass container divided at the centre to form two areas that represent the two tanks. Water is pumped into the first tank through a variable speed pump, and the flow rate is measured by a flow meter. The water out of the second tank is recycled to provide the supply reservoir for the pump. The depth of fluid is measured using differential pressure sensors that provide an analogue direct current signal for control purpose. Figure B.1 illustrates the set-up.

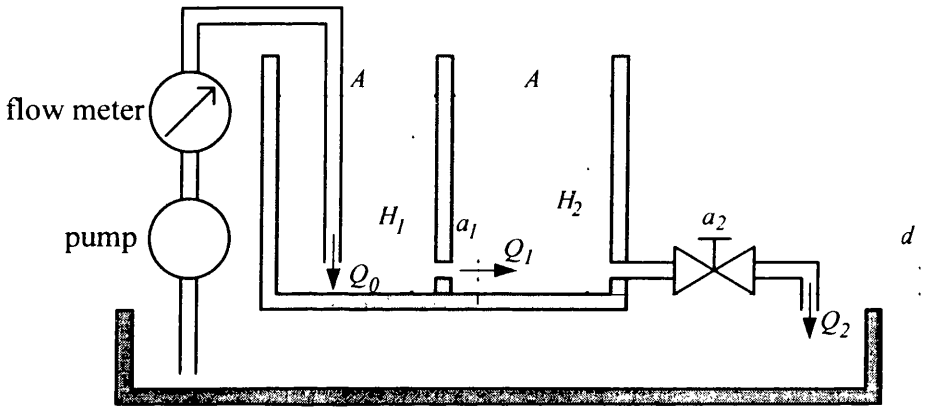


Figure B.1 Schematic of liquid-level system

Consider the tank set-up, as depicted in Figure B.1. A fluid balance about each tank delivers:

$$Q_0 - Q_1 = A\frac{dH_1}{dt} \tag{B.1}$$

$$Q_1 - Q_2 = A\frac{dH_2}{dt} \quad . \tag{B.2}$$

From Bernoulli's equation can be derived:

$$gH_1 = gH_2 + \frac{1}{2}u_1^2 \tag{B.3}$$

$$gH_2 = gd + \frac{1}{2}u_2^2 \tag{B.4}$$

where u_1 and u_2 are the flow velocities through orifice 1 or 2, respectively. With

$$Q_1 = u_1 a_1 \tag{B.5}$$

$$Q_2 = u_2 a_2 \tag{B.6}$$

the following equations are obtained:

$$Q_1 = c_{d1} a_1 \sqrt{2g(H_1 - H_2)} \tag{B.7}$$

$$Q_2 = c_{d2} a_2 \sqrt{2g(H_2 - d)} \tag{B.8}$$

where c_{d1} and c_{d2} are the discharge coefficients of orifices 1 and 2, respectively.

Substituting equation (B.7) and (B.8) into (B.1) and (B.2) gives:

$$A\frac{dH_1}{dt} = Q_0 - c_{d1} a_1 \sqrt{2g(H_1 - H_2)} \tag{B.9}$$

$$A\frac{dH_2}{dt} = c_{d1} a_1 \sqrt{2g(H_1 - H_2)} - c_{d2} a_2 \sqrt{2g(H_2 - d)} \quad . \tag{B.10}$$

Equations (B.9) and (B.10) describe the system dynamics in their true non-linear form. The discharge coefficients c_{d1} and c_{d2} can be determined from experimental results as follows:

Discharge coefficient c_{d1}:

Assuming there is no flow into tank 1 ($Q_0 = 0$) and the drain tap is closed ($Q_2 = 0$), from equation (B.1) and (B.2),

$$-Q_1 = A\frac{dH_1}{dt} \tag{B.11}$$

$$Q_1 = A\frac{dH_2}{dt} \quad . \tag{B.12}$$

Subtracting these equations gives:

$$-2Q_1 = A\frac{d}{dt}(H_1 - H_2) \quad . \tag{B.13}$$

Substituting Q_1 by equation (B.7) leads to:

$$-2c_{d1}a_1\sqrt{2g(H_1 - H_2)} = A\frac{d}{dt}(H_1 - H_2) \quad . \tag{B.14}$$

Equation (B.14) can be rearranged to give:

$$-\frac{2c_{d1}a_1\sqrt{2g}}{A}d\ t = \frac{dH_\Delta}{\sqrt{H_\Delta}} \tag{B.15}$$

where $H_1 - H_2$ is substituted by H_Δ.

Integration of (B.15) from $t = t_0$ to $t = t_0 + T$ and from $H_\Delta(0) = H_1(t_0) - H_2(t_0)$ to $H_\Delta(t_0 + T) = H_1(t_0 + T) - H_2(t_0 + T)$, respectively, delivers the final result:

$$c_{d1} = \frac{A}{a_1 T\sqrt{2g}}\left[\sqrt{H_1(t_0) - H_2(t_0)} - \sqrt{H_1(t_0 + T) - H_2(t_0 + T)}\right] \tag{B.16}$$

where $H_1(t_0)$, $H_2(t_0)$, $H_1(t_0 + T)$ and $H_2(t_0 + T)$ are to be measured at $t = t_0$ and $t = t_0 + T$, respectively. T is a known time interval.

Discharge coefficient c_{d2}

Assuming the tanks are one, i.e. with the partitions removed, with cross-sectional area $2A$, the following flow balance can be written:

$$-Q_2 = 2A\frac{dH_2}{dt} \quad . \tag{B.17}$$

Substituting Q_2 by equation (B.8) gives:

$$-c_{d2}a_2\sqrt{2g(H_2-d)} = 2A\frac{dH_2}{dt} \quad . \tag{B.18}$$

As with discharge coefficient c_{d1} equation (B.18) can be rearranged and integrated to give:

$$c_{d2} = \frac{4A}{a_2T\sqrt{2g}}\left[\sqrt{H_2(t_0)-d} - \sqrt{H_2(t_0+T)-d}\right] \tag{B.19}$$

where $H_2(t_0)$, $H_2(t_0+T)$ are to be measured at $t=t_0$ and $t=t_0+T$, respectively. T is a known time interval.

Appendix C

Dynamics of Cart-Pole system

The coordinate system for the single inverted pendulum is as Figure C.1

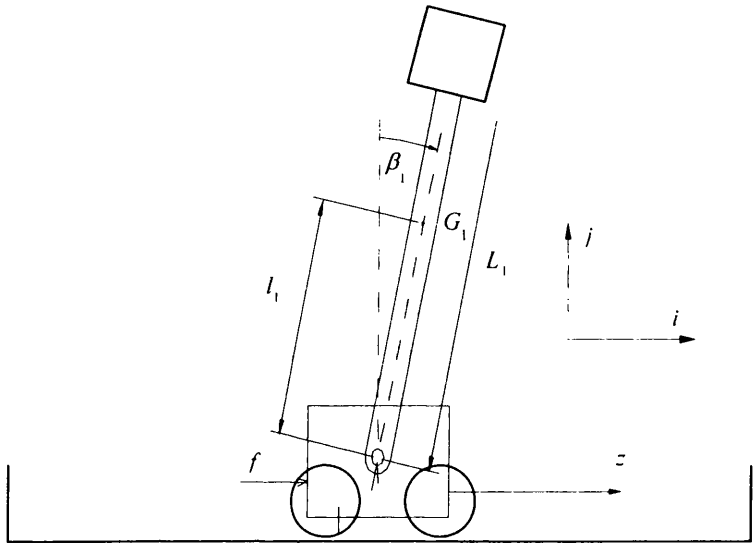


Figure C.1 Cart-pole co-ordinate system

where

z - cart position, m

β_1 - pendulum link angle, radians

L_1 - length of pendulum link, m

l_1 - distance between pivot and centre of mass, m

f - force exerted on cart, N

G_1 - centre of mass of primary link

General system equations:

$$M\begin{bmatrix}\ddot{z}\\ \ddot{\beta}_1\end{bmatrix}+C\begin{bmatrix}\dot{z}\\ \dot{\beta}_1\end{bmatrix}+N=\begin{bmatrix}f\\ 0\end{bmatrix} \tag{C.1}$$

Inertia matrix M, damping matrix C and nonlinear terms N are:

$$M=\begin{bmatrix} m_c+m_1 & m_1 l_1 \cos\beta_1 \\ m_1 l_1 \cos\beta_1 & m_1 l_1^2 + J_1 \end{bmatrix},\quad C=\begin{bmatrix} v & 0 \\ 0 & C_1 \end{bmatrix},\quad N=\begin{bmatrix} -m_1 l_1 \sin\beta_1 \dot{\beta}_1^2 \\ -m_1 l_1 g \sin\beta_1 \end{bmatrix}$$

Considering motor equations, force is:

$$f=\frac{K}{r}i_a-\frac{J_a}{r^2}\ddot{z}-\frac{D_m}{r^2}\dot{z} \tag{C.2}$$

The matrices M and C become:

$$M=\begin{bmatrix} m_c+m_1+\frac{J_a}{r^2} & m_1 l_1 \cos\beta_1 \\ m_1 l_1 \cos\beta_1 & m_1 l_1^2 + J_1 \end{bmatrix},\quad C=\begin{bmatrix} v+\frac{D_m}{r^2} & 0 \\ 0 & C_1 \end{bmatrix}$$

$$\det(M)=\left(m_c+m_1+\frac{J_a}{r^2}\right)\left(m_1 l_1^2+J_1\right)-m_1^2 l_1^2 \cos^2\beta_1$$

In current control mode

$$i_a = k_i u$$

where the u is voltage. Introducing state variables:

$$x=\begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T=\begin{bmatrix} z & \beta & \dot{z} & \dot{\beta} \end{bmatrix}^T \tag{C.3}$$

$$\dot{x}_1 = x_3 \tag{C.4}$$

$$\dot{x}_2 = x_4 \tag{C.5}$$

$$\begin{aligned}\dot{x}_3 =& -\frac{m_1 l_1^2 + J_1}{\det(M)}\left(v x_3 - m_1 l_1 \sin x_2 x_4^2 - \frac{K k_i}{r}u + \frac{D_m}{r^2}x_3\right) \\ &+\frac{m_1 l_1 \cos x_2}{\det(M)}\left(-m_1 l_1 g \sin x_2 + C_1 x_4\right)\end{aligned} \tag{C.6}$$

$$\dot{x}_4 = \frac{m_1 l_1 \cos x_2}{\det(M)}\left(vx_3 - m_1 l_1 \sin x_2 x_4^2 - \frac{Kk_i}{r}u + \frac{D_m}{r^2}x_3\right) - \frac{m_c + m_1 + \frac{J_a}{r^2}}{\det(M)}\left(-m_1 l_1 g \sin x_2 + C_1 x_4\right) \tag{C.7}$$

Appendix D

Ship Dynamics

The equations describing ship dynamics are obtained from Newton's laws expressing conservation of linear and angular momentum. The forces are in general complicated functions of the ship's motion, i.e. the time history of the velocity, angular velocity and the rudder motion. Consider the ship as a rigid body with 6 degrees of freedom corresponding to translations in 3 directions and rotation around 3 axes. Neglecting sensor and actuator dynamics, the ship can thus be modelled as a 12-order system. Additional dynamics are also introduced by the rudder servo. It is customary at least for tankers and similar ships to neglect the coupling between the yaw motion and the pitch and roll motions. To describe the equations of motion the co-ordinate system fixed to the ship shown in Figure. D.1 is used.

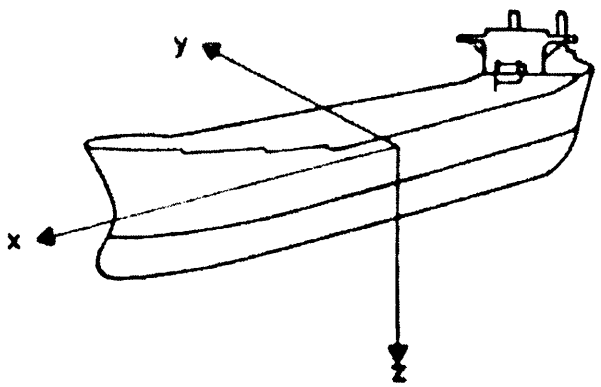


Figure D.1. Definition of co-ordinates fixed to the ship. Translation along the co-ordinate axes are called surges sway and heave and rotation around the co-ordinate axes are called rote pitch and yaw, respectively.

Let v be the projection of the ship's velocity on the y-axis, and r the component of the angular velocity on the z-axis, Figure D.2. The projection of the ship's velocity on the x-axis is assumed to be constant and equal to u_0. The equations for the yaw motion are then given by the laws of conservation of linear and angular momentum:

$$\left.\begin{aligned} m\left(\frac{dv}{dt} + ru_0 + x_G\frac{dr}{dt}\right) &= Y, \\ I_z\frac{dr}{dt} + mx_G\left(\frac{dv}{dt} + ru_0\right) &= N, \end{aligned}\right\} \qquad \text{(D.1)}$$

where m is the mass of the ship, I_Z its moment of inertia about the z-axis, Y the component of the hydrodynamic forces on the y~axis, N the z-component of the torque due to the hydrodynamic forces and x_G the x co-ordinate of the centre of mass. It is assumed that the centre of mass is located in the x-z plane. The hydrodynamic force Y and the torque N are complicated functions of the motion. It is usually assumed that

$$\begin{aligned} Y &= Y(v,r,\delta,\dot{v},\dot{r}), \\ N &= N(v,r,\delta,\dot{v},\dot{r}), \end{aligned} \tag{D.2}$$

where δ is the rudder deflection. The functions Y and N will also depend on trim and draught, and the results will hold for one loading condition only.

Stationary solutions

Assuming that the rudder is kept constant at the centre position the steady-state solution to the equations of motion is given by

$$\left.\begin{aligned} f(v,r) &= Y(v,r,0,0,0) - mru_0 = 0, \\ g(v,r) &= N(v,r,0,0,0) - mx_G ru_0 = 0, \end{aligned}\right\} \tag{D.3}$$

For a ship which is symmetric around its centre plane the force Y and the torque N will vanish for a motion with $v = 0$ and $r = 0$. The stationary solution to (D.3) is then given by $v = 0$ and $r = 0$. Depending on the properties of the functions Y and N there may, however, also be other solutions. These are obtained from a graph of the functions f and g. in Figure D.3. The case of one stationary point Q only as shown in Figure D.3(A) is most common. For very large tankers the case shown in Figure D.3(B) can, however, occur. In such a case the solution $v = 0$ and $r = 0$. point Q in Figure D.3(B), which corresponds to a straight line motion is unstable while the solutions $v = v_0$, $r = -r_0$ and $v = v_0$, $r = r_0$, point P_1 and P_2 respectively in Figure D.3(B), which correspond to circular motions are stable. A ship with these properties cannot be kept on a straight course with zero rudder. It will either go into a port yaw or into a starboard yaw and the motion will tend to a stationary circular motion.

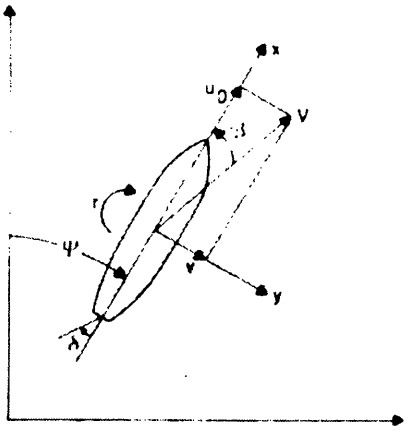


Figure D.2. Variables used to describe the linearised yaw motion of a ship. Notice that different conventions for the sign of δ are used in the literature.

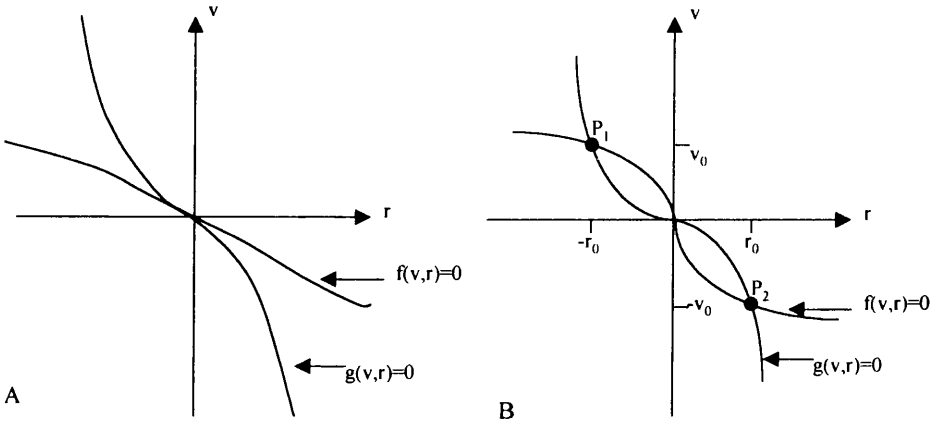


Figure D.3 The determination of the stationary motions as the intersections of the curves $f(v, r)=0$ and $g(v, r)=0$. In case A the curves intersect at the origin only but in case B there are 3 stationary solutions.

Normalisation and linearisation

It is customary to normalise the equations by introducing dimension free quantities. This can be done in several different ways. In the 'prime' system, which is most common, the length unit is the length of the ship, L, the time unit is L/V, where V is the ship's speed, and the mass unit is $\rho L^3/2$, where ρ is the mass density of water. The normalised variables are denoted by introducing a 'prime' on the non-normalised variables. To linearise the equations it is necessary to introduce the partial derivatives of the force Y and the torque N. The partial derivative

$$Y_v = (\delta / \delta v) Y(v, r, \delta, \dot{v}, \dot{r})$$

where the right hand side is evaluated at arguments zero, is called a hydrodynamic derivative. The derivatives $Y_r, Y_\delta, Y_{\dot{v}}, Y_{\dot{r}}, N_v, N_r, N_\delta, N_{\dot{v}}$ and $N_{\dot{r}}$ are defined analoguously. Linearisation of (D.1) around the stationary solution v=0, r=0 and normalization gives

$$\begin{bmatrix} m'-Y'_{\dot{v}} & m'x'_G - Y'_{\dot{r}} \\ m'x'_G - N'_{\dot{v}} & I'_z - Y'_{\dot{r}} \end{bmatrix} \frac{d}{dr'} \begin{bmatrix} v' \\ r' \end{bmatrix} = \begin{bmatrix} Y'_v & Y'_r - m' \\ N'_v & N'_r - m'x'_G \end{bmatrix} \begin{bmatrix} v' \\ r' \end{bmatrix} + \begin{bmatrix} Y'_\delta \\ N'_\delta \end{bmatrix} \delta, \qquad \text{(D.4)}$$

where all parameters and variables are dimension free. Notice that it has been assumed that u_0/V=1. The derivatives $Y'_{\dot{v}}$ and $N'_{\dot{r}}$ are negative. Notice that they appear in the equations in the same way as the mass and the inertia. These terms are therefore sometimes called added mass and added inertia.

State equations

The normalised equations of motion (D.4) are converted to standard state space notation by solving for the derivatives dv'/dt' and dr'/dt'. This gives the following model for the yaw motion of the ship

$$\frac{d}{dt'} \begin{bmatrix} v' \\ r' \\ \varphi \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} v' \\ r' \\ \varphi \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{21} \\ 0 \end{bmatrix} \delta, \qquad \text{(D.5)}$$

where the heading ψ defined by $d\psi/dt'$ has also been introduced as an extra variable. The heading ψ is shown in Figure D.2. The linearised yaw motion of a ship can thus be described as a third-order dynamical system where the state variables can be chosen as

1. v' the sway velocity, i.e. the component of the ship velocity on the y-axis in the co-ordinate system fixed to the ship,
2. r' the ship's angular velozity about the z-axis,
3. ψ the deviation in heading angle

Other state variables are sometimes chosen. The angle of attack, i.e. β in Figure D.2, can be used instead of the sway velocity v.

Appendix E

Double Inverted Pendulum Model

The mechanic of the system consists of 3 rigid bodies, as shown in Figure D.1, and has3 degrees of freedom, 1 translational and 2 rotational

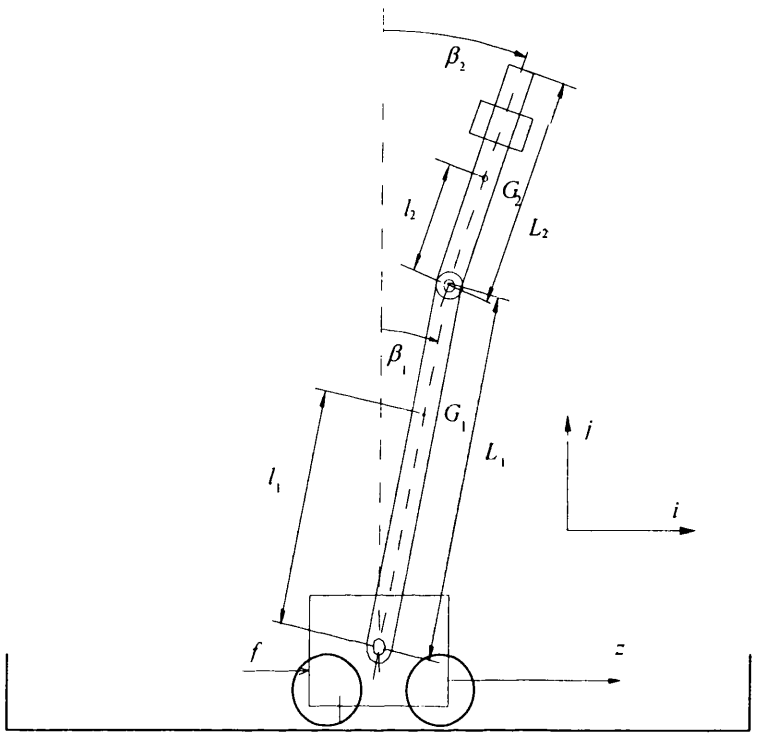


Figure E.1 Double pole system coordinates

Where, z is the cart position; β_1, β_2 are the pendulum link angles, L_1, L_2 are the lengths of pendulum links; l_1, l_2 are the distances between the pivot and centre of mass of respective links; f is the force exerted on cart; G_1 is the centre of mass of primary link and G_2 is the centre of mass of second link

Equations of Motion

Using a Lagrangian approach, the system equations are developed by determining the kinetic and potential energies of system components in terms of generalised co-ordinates. Defining the Lagrangian Equation:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right)-\frac{\partial L}{\partial q}+\frac{\partial D}{\partial \dot{q}}=Q_q \tag{E.1}$$

where $L=T-V$; T is the kinetic energy; V is the potential energy; D is the Rayleigh dissipation function; Q_q is the generalised forces not taken into account in T, V, and D; q is the generalised co-ordinates

$$q=\begin{bmatrix} z \\ \beta_1 \\ \beta_2 \end{bmatrix} \tag{E.2}$$

The system kinetic energy, T is:

$$T=T_{cart}+T_{pendulum1}+T_{pendulum2}=T_1+T_2+T_3 \tag{E.3}$$

with

$$\begin{aligned} T_1 &= \frac{1}{2}m_c\dot{z}^2 \\ T_2 &= \frac{1}{2}m_1v_{G_1}^2+\frac{1}{2}J_1\dot{\beta}_1^2 \\ T_2 &= \frac{1}{2}m_2v_{G_2}^2+\frac{1}{2}J_2\dot{\beta}_2^2 \end{aligned} \tag{E.4}$$

where m_c is the mass of cart ; m_1 the mass of first link; m_2 is mass of second link; J_1 is the inertia of first link about centre of mass; J_2 is inertia of second link about centre of mass; v_{G_1}, v_{G_2} are the linear velocities of G_1 and G_2.

Velocities v_{G_1}, v_{G_2} can be found from radius vectors

$$\vec{v}_{G_1}=\frac{d\vec{r}_1}{dt} \text{ and } \vec{v}_{G_2}=\frac{d\vec{r}_2}{dt} \tag{E.5}$$

If $\vec{r}_1=r_{1i}\vec{i}+r_{1j}\vec{j}$ and $\vec{r}_2=r_{2i}\vec{i}+r_{2j}\vec{j}$, then

$$v_{G_1}^2=\dot{r}_{1i}^2+\dot{r}_{1j}^2 \text{ and } v_{G_2}^2=\dot{r}_{2i}^2+\dot{r}_{2j}^2 \tag{E.6}$$

where the derivatives are

$$\begin{aligned} \dot{r}_{1i} &= \dot{z} + l_1\dot{\beta}_1 \cos\beta_1 \\ \dot{r}_{1j} &= -l_1\dot{\beta}_1 \sin\beta_1 \end{aligned} \tag{E.7}$$

and

$$\begin{aligned} \dot{r}_{2i} &= \dot{z} + L_1\dot{\beta}_1 \cos\beta_1 + l_2\dot{\beta}_2 \cos\beta_2 \\ \dot{r}_{2j} &= -L_1\dot{\beta}_1 \sin\beta_1 - l_2\dot{\beta}_2 \sin\beta_2 \end{aligned} \tag{E.8}$$

Therefore the velocities are given by

$$\begin{aligned} v_{G_1}^2 &= \left(\dot{z} + l_1\dot{\beta}_1 \cos\beta_1\right)^2 + l_1^2 \sin^2\beta_1\dot{\beta}_1^2 \\ v_{G_2}^2 &= \left(\dot{z} + l_2\dot{\beta}_2 \cos\beta_2 + L_1\dot{\beta}_1 \cos\beta_1\right)^2 + \left(l_2\beta_2 \sin\beta_2 + L_1\dot{\beta}_1 \sin\beta_1\right)^2 \end{aligned} \tag{E.9}$$

Finally

$$\begin{aligned} T_1 &= \frac{1}{2}m_c\dot{z}^2 \\ T_2 &= \frac{1}{2}m_1\left[\left(\dot{z} + l_1\dot{\beta}_1 \cos\beta_1\right)^2 + l_1^2 \sin^2\beta_1\dot{\beta}_1^2\right] + \frac{1}{2}J_1\dot{\beta}_1^2 \\ T_3 &= \frac{1}{2}m_2\left(\dot{z} + l_2\dot{\beta}_2 \cos\beta_2 + L_1\dot{\beta}_1 \cos\beta_1\right)^2 + \\ &\frac{1}{2}m_2\left(l_2\dot{\beta}_2 \sin\beta_2 + L_1\dot{\beta}_1 \sin\beta_1\right)^2 + \frac{1}{2}J_2\dot{\beta}_2^2 \end{aligned} \tag{E.10}$$

The expression for the system potential energy , V is:

$$V = V_{cart} + V_{pendulum1} + V_{pendulum2} = V_1 + V_2 + V_3 \tag{E.11}$$

where:

$$\begin{aligned} V_1 &= 0 \\ V_2 &= m_1 g l_1 \cos\beta_1 \\ V_3 &= m_2 g\left(L_1 \cos\beta_1 + l_2 \cos\beta_2\right) \end{aligned} \tag{E.12}$$

where g is the gravitational constant 9.807 m/s^2. For general viscous friction forces, Rayleigh's dissipation function is defined as a quadratic function of generalised velocities $\dot{q}_i$

$$D = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}\dot{q}_i\dot{q}_j \tag{E.13}$$

so that the viscous friction generalised force in the direction of the i^{th} co-ordinate becomes

$$Q_{v,i} = -\frac{\partial D}{\partial \dot{q}_i} = -\sum_{j=1}^{n} C_{ij}\dot{q}_j \tag{E.14}$$

For the pendulum at hand, $c_{ij} = 0$, for $i \neq 0$. Also, for convenience of notation we let $c_{11} = v$, $c_{22} = C_1$, $c_{33} = C_2$ so that:

$$D = \frac{1}{2}v\dot{z}^2 + \frac{1}{2}C_1\dot{\beta}_1^2 + \frac{1}{2}C_2\dot{\beta}_2^2 \tag{E.15}$$

where v is the coefficient of sliding viscous friction between track and cart; C_1 the coefficient of viscous friction at the pivot of first link; C_2 is coefficient of viscous friction at the pivot of second link. The expression for generalised forces, Q is:

$$Q_q = \begin{pmatrix} Q_z \\ Q_{\beta_1} \\ Q_{\beta_2} \end{pmatrix} = \begin{pmatrix} f \\ 0 \\ 0 \end{pmatrix} \tag{E.16}$$

where f is a force exerted on the cart by the actuator. Now we can write the Lagrangian equations for L_2 system as

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{z}}\right) - \frac{\partial L}{\partial z} + \frac{\partial D}{\partial \dot{z}} &= Q_z \\ \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\beta}_1}\right) - \frac{\partial L}{\partial \beta_1} + \frac{\partial D}{\partial \dot{\beta}_1} &= Q_{\beta_1} \\ \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\beta}_2}\right) - \frac{\partial L}{\partial \beta_2} + \frac{\partial D}{\partial \dot{\beta}_2} &= Q_{\beta_2} \end{aligned} \tag{E.17}$$

Let $h_1, \ldots, h_8$ be defined as below:

$$\begin{aligned} h_1 &= m_c + m_1 + m_2 \\ h_2 &= m_1 l_1 + m_2 L_1 \\ h_3 &= m_2 l_2 \\ h_4 &= m_1 l_1^2 + m_2 L_1^2 + J_1 \end{aligned} \tag{E.18a}$$

$$\begin{aligned} h_5 &= m_2 l_2 L_1 \\ h_6 &= m_2 l_2^2 + J_2 \\ h_7 &= m_1 l_1 g + m_2 L_1 g \\ h_8 &= m_2 l_2 g \end{aligned} \tag{E.18b}$$

After simplifications the equations of motion become

$$\begin{aligned} &h_1 \ddot{z} + h_2 \ddot{\beta}_1 \cos\beta_1 + h_3 \ddot{\beta}_2 \cos\beta_2 - h_2 \dot{\beta}_1^2 \sin\beta_1 - h_3 \dot{\beta}_2^2 \sin\beta_2 = f \\ &h_2 \ddot{z} \cos\beta_1 + h_4 \ddot{\beta}_1 + h_5 \ddot{\beta}_2 \cos(\beta_1 - \beta_2) + h_5 \dot{\beta}_2^2 \sin(\beta_1 - \beta_2) - h_7 \sin\beta_1 = 0 \\ &h_3 \ddot{z} \cos\beta_2 + h_5 \ddot{\beta}_1 \cos(\beta_1 - \beta_2) + h_6 \ddot{\beta}_2 - h_5 \dot{\beta}_1^2 \sin(\beta_1 - \beta_2) - h_8 \sin\beta_2 = 0 \end{aligned} \tag{E.19}$$

The system equations can be written in compact form:

$$M(q)\ddot{q} + C\dot{q} + N(q,\dot{q}) = Q_q \tag{E.20}$$

where the matrices M, C and N are defined as follows:

$$M(q) = M = \begin{bmatrix} h_1 & h_2 \cos\beta_1 & h_3 \cos\beta_2 \\ h_2 \cos\beta_1 & h_4 & h_5 \cos(\beta_2 - \beta_1) \\ h_3 \cos\beta_2 & h_5 \cos(\beta_2 - \beta_1) & h_6 \end{bmatrix} \tag{E.21}$$

$$C = \begin{bmatrix} v & 0 & 0 \\ 0 & C_1 & 0 \\ 0 & 0 & C_2 \end{bmatrix} \tag{E.22}$$

All nonlinear terms are collected into N:

$$N(q,\dot{q}) = N = \begin{bmatrix} -h_2 \dot{\beta}_1^2 \sin\beta_1 - h_3 \dot{\beta}_2^2 \sin\beta_2 \\ h_5 \dot{\beta}_2^2 \sin(\beta_1 - \beta_2) - h_7 \sin\beta_1 \\ -h_5 \dot{\beta}_1^2 \sin(\beta_1 - \beta_2) - h_8 \sin\beta_2 \end{bmatrix} \tag{E.23}$$