METHODS FOR DESIGNING AND OPTIMIZING FUZZY CONTROLLERS

THESIS
Submitted in partial fulfillment of the
Requirements for the Degree of
MASTER of SCIENCE
of Rhodes University

by

Andre Michael Swartz

SUPERVISOR: Prof. W. Kotze

December 1999

ACKNOWLEDGEMENTS

I wish to express my sincere appreciation and gratitude to:

My supervisor, Prof. W. Kotze for his unfailing commitment, guidance, encouragement and patience.

Prof. F. Klawonn for his valued assistance and suggestions.

The staff of the Mathematics Department at Rhodes University, who always made learning mathematics a pleasure.

Baron Peterssen who made financial support through Telkom S.A. possible.

My family for being a source of constant inspiration and encouragement. It is to them that I dedicate this thesis.

Contents

List of Tables

List of Figures

Abstract

We start by discussing fuzzy sets and the algebra of fuzzy sets. We consider some properties of fuzzy modeling tools. This is followed by considering the Mamdani and Sugeno models for designing fuzzy controllers. Various methods for using sets of data for desining controllers are discussed. This is followed by a chapter illustrating the use of genetic algorithms in designing and optimizing fuzzy controllers.Finally we look at some previous applications of fuzzy control in telecommunication networks, and illustrate a simple application that was developed as part of the present work.

Chapter 1

Introduction

The first controllers

During the 1970's the first fuzzy logic controller was developed by Abraham Mamdani and a research assistant of his, Seto Assilian [23]. Mamdani was attempting to create an adaptive system that could learn how to control an industrial process. After trying conventional approaches to control and failing, his assistant suggested using fuzzy logic. It produced better results than the previously attempted approaches. During the 1980's Michio Sugeno developed his own version of a fuzzy controller that could control the motion of a car[53][54]. Using 20 fuzzy if - then rules Sugeno's controller was able to drive a model car through angled corridors after a learning session.

Thus fuzzy controllers were initially developed as systems that could automatically emulate the control process of a skilled human operator. In developing a fuzzy controller a human operator is required to express his expertise in the form of rules in a natural language. Whereas controllers developed by traditional strategies require exact knowledge, in the development of a fuzzy controller, vagueness of information creates no problem and is in fact desired. Fuzzy systems represent a step in the direction of modeling human decision making processes. A fuzzy system creates an interface of communication between humans and systems which is clearly one of the objectives of Zadeh's seminal papers on Fuzzy Sets and Fuzzy Logic[65][66].

While the West has been slow in accepting the new technology in Japan, a wide range of problems have been solved using it. Applications range from industrial robots and machinery to consumer products like video cameras, washing machines and T.V.'s

The need for fuzzy theory

Probability is the mathematical tool for dealing with stochastic uncertainty. In a statement like "the probability of getting a tail on the flip of a coin is 0.5" the uncertainty of getting a tail is modeled by the number 0.5. The event of getting a tail is a well - defined event.

The aim of fuzzy theory is modeling a different kind of uncertainty namely lexical or linguistic uncertainty - the type of uncertainty or vagueness which is inherent in natural language. It would be difficult if not impossible to model a

concept like "comfortable temperature" using a set with crisp boundaries. The problem is that a temperature arbitrarily close to the boundary of such a set of comfortable temperatures would not be considered comfortable at all. This does not conform to our experience of how temperatures change. The problem is that the event of a temperature being comfortable is less well - defined than a probability as in the above example. Also, in deciding whether a given temperature is comfortable a certain amount of subjectivity is inevitable. Classes arising in natural language like "comfortable temperature" or "tall men" have been termed subjective categories by psycholinguists. Membership of a thing to a subjective category is a matter of degree - the degree to which the thing satisfies the criteria that define the category. Similarly, elements have a degree of belonging to a fuzzy set instead of it being a simple matter of inclusion or exclusion as in the case of a crisp set. This gradual transition from non - membership to membership allows one to represent subjective categories as fuzzy sets, i.e. the idea of a fuzzy set is the modification to the concept of set that one needs to deal with linguistic uncertainty.

Controversy

The development of Fuzzy Theory has been clouded by controversy and Zadeh has been severely critisized for his ideas. For example, according to Kalman [59] the major problems facing systems analysts - developing a deep insight into the nature of systems - is not addressed by Fuzzy Theory and "... Zadeh's solution has no chance to contribute to this basic problem ..." Even today, after hundreds of successful applications of the theory, this attitude persists. One reason, strange as it may be, seems to be the name "Fuzzy" itself. It is notable that Van Altrock in [59] mentions that the Japanese language does not have a negative connotation to the word "fuzzy". Maybe it creates the impression that the reasoning itself is vague. This is not true. The theory is developed to deal with vagueness but is based on solid mathematical foundations. Another reason may be what Zadeh has termed the "hammer" principle, according to which if you have a hammer in your hand everything starts to look like a nail.

Traditional controllers run the risk of becoming so simple that they are unrealistic or so complicated that they become impractical. Fuzzy controllers have often been compared with traditional controllers in studies. These studies generally prove that fuzzy controllers are more robust, have slower rise times and faster settling times. Their control signals are generally smoother as well. Considering such benefits it becomes difficult to understand the contoversy surrounding fuzzy systems.

Aim and development of the work

This work was commisioned as part of Telkom (S.A.)'s COE programme. From this one can assume that their research departments are becoming aware of the importance of fuzzy solutions to complex problems. ITAS (Integrated Technologies Application Strategies) - the division where such solutions would presumably first be researched, is almost exclusively populated by researchers who have not been exposed to fuzzy theory. Someone wishing to develop a fuzzy controller for use in one of Telkom's networks would need to become exposed to the background theory in this area. They would need material on the different methods that have been developed to create fuzzy solutions as well as examples of previous applications of the theory in telecommunication networks. Thus the topics discussed were chosen with the needs of such a prospective developer in mind. Hence the focus is on the mathematical structures and tools that support the development of fuzzy solutions. It is hoped that the work will serve the purpose of demystifying the subject and to create an appreciation of the beauty and simplicity of a field of study that we thoroughly enjoyed working in.

In the second chapter we will start by defining the basic concepts and all of the operators necessary for the developments to come. Concepts from fuzzy logic that are required will also be briefly discussed. The chapter will also introduce the structure and operational details of the two main types of controllers. It will end with a discussion on some mathematical properties of fuzzy controllers.

Chapter 3 looks at the use of sets of input - output data in designing fuzzy controllers. Chapter 4 discusses the use of Genetic Algorithms in the design and optimization of fuzzy controllers. Chapter 5 looks at the ways in which fuzzy systems have been used in Network Management in Telecommunications Engineering and discusses an application which was developed to illustrate possible uses of some of the theory. Chapter 7 contains the appendices. The first of these contains the proofs of two statements made in Chapter 3, the second a number of results from the application in Chapter 5 and finally the third appendix contains the programmes that were written to demonstrate the application.

Chapter 2

Background Theory

Definitions

The algebra of fuzzy sets

Fuzzy logic is an extention of two valued logic. Instead of the truth value of a statement being only 0 or 1 any value in [0, 1] is possible. In two valued logic we study how (true or false) statements are connected and how we can make inferences from these connections. To connect statements we use the conjunction and disjunction operators, while we require the implication operator to make inferences (together with rules for valid implications, for example modus ponens).

The algebra of fuzzy sets was constructed in such a way the crisp Set Theory and crisp Logic become special cases of the now more general Fuzzy Set Theory. In the development of Fuzzy Logic the logical operators of conjunction and disjunction were extended to t - norms and t - conorms respectively.

Definition 1 *A triangular norm (or t - norm) is a map:*

$$T : [0,1]^2 \to [0,1] \ satisfying :$$

1. $T(a,b) = T(b,a)$ *(T is commutative)*
2. $T(a,T(b,c)) = T(T(a,b),c)$ *(T is associative)*
3. $a \leq b \Rightarrow T(a,c) \leq T(b,c)$ *(T is monotone)*
4. $T(a,1) = a$ *(boundary condition)* $\forall a,b,c \in [0,1]$

Definition 2 *A triangular conorm (or an s - norm or a t - conorm) is a map:*

$$S : [0,1]^2 \to [0,1] \ satisfying$$

1. $S(a,b) = S(b,a)$ (S is commutative)
2. $S(a,S(b,c)) = S(S(a,b),c)$ (S is associative)
3. $a \leq b \Rightarrow S(a,c) \leq S(b,c)$ (T is monotone)
4. $S(a,0) = a$ $\forall a,b,c \in [0,1]$

There are infinely many t - norms and t - conorms. Only a few have been used in practical control. Some of these are:

Example 3

$$T_m(x,y) = \min\{x,y\}$$
$$S_m(x,y) = \max\{x,y\}$$

$$T_p(x,y) = x \times y$$
$$S_p(x,y) = x + y - x \times y$$

$$T_L(x,y) = \max\{x+y-1,0\} \tag{2.1}$$
$$S_L(x,y) = \min\{x+y,1\}$$

Definition 4 *Let X be an arbitrary set. A fuzzy subset A of X is characterised by a membership function:*

$$\mu_A : X \to [0,1]$$

Remark 1 *In the traditional literature a fuzzy subset A of X is usually denoted by greek letters eg. μ_A. $A(x)$ (or $\mu_A(x)$) is understood to be the "degree" to which x belongs to A, or the truth value of the statement $x \in A$. X is called the universe of discourse. The unit interval $[0,1]$ is in what follows often denoted by I and the family of all fuzzy subsets of X by I^X.*

Definition 5 *The support of a fuzzy set A is the set:*

$$\text{supp(A)} = \{x \in X : \mu_A(x) > 0\}$$

Definition 6 *The core of a fuzzy set A is the set:*

$$core(A) = \{x \in X : \mu_A(x) = 1\}$$

Definition 7 *A fuzzy singleton is a fuzzy set A s.t.:*

$$\text{supp(A)} = \{x\} \text{ for some } x \in X$$

Definition 8 *A fuzzy number is a fuzzy set A satisfying:*

1. $max\{\mu_A(x) : x \in X\} = 1$

2. $\mu_A(\lambda x_1 + (1-\lambda)x_2) \geq \min\{\mu_A(x_1); \mu_A(x_2)\}$ *for* $x_1, x_2 \in X, \lambda \in [0,1]$

Definition 9 *For A and B fuzzy subsets of X we can now define:*

1. $\mu_{A\cap B}(x) = T(\mu_A(x), \mu_B(x))$
2. $\mu_{A\cup B}(x) = S(\mu_A(x), \mu_B(x))$
3. $\mu_{A'}(x) = 1 - \mu_A(x)$

for T and S a t - norm and co - norm respectively.

Definition 10 *A fuzzy relation R is a fuzzy subset of a product space:*

$$\mu_R : X_1 \times X_2 \times \cdots \times X_n \rightarrow [0,1]$$

Definition 11 *If $A_1, A_2, ..., A_n$ are fuzzy subsets of $X_1, ..., X_n$ respectively then the cartesian product of $A_1, ..., A_n$ is a fuzzy set on the product space defined as:*

$$\begin{aligned}
\mu_{A_1\times A_2\times ...\times A_n}(x_1, ..., x_n) &= \min\{\mu_{A_1}(x_1), ..., \mu_{A_n}(x_n)\} \\
&\quad or \\
\mu_{A_1\times A_2\times ...\times A_n}(x_1, ...x_n) &= \mu_{A_1}(x_1) \cdot \mu_{A_2}(x_2) \cdot ... \cdot \mu_{A_n}(x_n) \\
or \text{ if } n &\leq 2 \\
\mu_{A_1\times A_2}(x_1, x_2) &= T(\mu_{A_1}(x_1), \mu_{A_2}(x_2))
\end{aligned}$$

Definition 12 *For A a fuzzy subset of X, R a fuzzy relation on $X \times Y$ and T a t-norm*

$$\mu_{A\circ R}(y) = \sup\{T(\mu_A(x), \mu_R(x,y)) : x \in X\} \tag{2.2}$$

In the case of A being a fuzzy relation on $X \times Y$ and B being a fuzzy relation on $Y \times Z$ this definition changes to:

$$\mu_{A\circ B}(x,z) = \sup\{T(\mu_A(x,y), \mu_B(y,z)) : y \in Y\}$$

The above definition is Zadeh's Compositional Rule of Inference. It is used in the inference process of a fuzzy controller as explained later.

Linguistic variables and fuzzy if - then rules

As stated earlier the concept of a fuzzy set is what one needs to deal with linguistic uncertainty. A related concept is that of a linguistic variable. According to Zadeh this is a variable whose values are sentences in a natural or artificial language. More specifically:

Definition 13 *A linguistic variable is characterised by a quintuple*

$$(x, \Im(x), U, G, M) \text{ where}$$

x is the name of the variable;

$\Im(x)$ is the term set of x or the set of linguistic values of x;

Each linguistic value of x represents a fuzzy subset of the universe of discourse U;

G is a syntactic rule for generating the names, X of values of x.

M is a semantic rule for associating with each value X of x its value $M(X)$, a fuzzy subset of U.

This definition is very general and is a bit more than what we will require. A simple example will illustrate the terms which we find necessary.

Example 14 *Suppose one wanted to design a fuzzy controlled air - conditioner. Here one linguistic variable might be x = temperature, which might have the term set:*

$$\Im(x) = \{cool; warm; hot\}$$

The elements of $\Im(x)$ are the linguistic values of x and each has an associated fuzzy set; as illustrated in Figure 2.1.

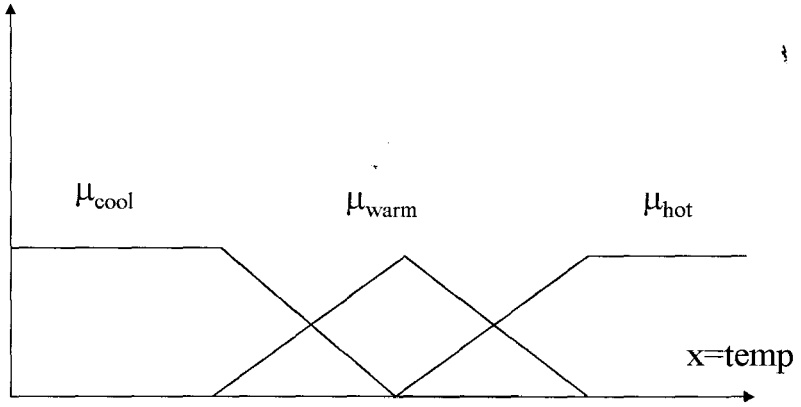


Figure 2.1: Fuzzy sets on x=temperature

We refer to "warm" as the linguistic label of the fuzzy set μ_{warm}. Similarly for the rest.

Fuzzy Rules

The essential part of a fuzzy controller is a collection of conditional statements called fuzzy if - then rules. The condition part of a rule specifies a collection of conditions on input variables. Not every input variable need be mentioned in any particular rule though in most controllers each rule contains a condition for every variable. Such a condition is given in the form of a linguistic term of a linguistic variable. These conditions are aggregated in the rule in the form of sentence connectives (either "and" or "or"). In the case of use of the sentence connective "or" such a rule can be (and is for inference purposes) rewritten in the form of "and" rules.

There are mainly two types of fuzzy controllers which differ in terms of the form of the consequence of the rules they use. The Mamdani controller has linguistic labels of fuzzy sets in the conditions in both the antecedent and consequent of a rule. The Sugeno controller has linguistic labels in the antecedent of each rule. The consequent of a rule is a functional relationship between the output variable and the collection of input variables as will be seen clearly below.

Fuzzy Logic and the Mamdani controller

Fuzzy implication functions

Three main categories of fuzzy implication operators have been defined. These are fuzzy conjunction, fuzzy disjunction and the fuzzy implication operators.

The fuzzy conjunction is defined using a triangular norm. Suppose A and B are fuzzy sets on the domains X and Y respectively. In the following T will denote a t - norm and S a t - conorm. Then the truth value of the statement

$$A \rightarrow B$$

can be defined by means of a fuzzy conjunction as

$$\mu_{A\rightarrow B}(x,y) = T(\mu_A(x), \mu_B(y))$$

or by means of a fuzzy disjunction as

$$\mu_{A\rightarrow B}(x,y) = S(\mu_A(x), \mu_B(y))$$

Examples of Fuzzy implication functions are

1 Material implication:

$$\mu_{A\rightarrow B}(x,y) = S(\mu_{A'}(x), \mu_B(y))$$

2 Propositional Calculus:

$$\mu_{A\to B}(x,y) = S(\mu_{A'}(x), T(\mu_A(x), \mu_B(y)))$$

If one restricts to crisp membership values and S = "or" and T = "and" then (2) recovers the implication operater in two - valued logic.

3 Generalization of modus ponens:

$$\mu_{A\to B}(x,y) = \sup\{c \in [0,1] : T(\mu_A(x), c) \leq \mu_B(y)\}$$

4 Generalization of modus tollens:

$$\mu_{A\to B}(x,y) = \inf\{t \in [0,1] : S(\mu_B(y), t) \leq \mu_A(x)\}$$

Clearly now one can define many different fuzzy implication operators by using different t - norms and t - conorms. For example each rule in a Mamdani rulebase can be interpreted as a fuzzy conjunction with the minimum operator as the chosen t - norm.

Inference in the Mamdani Controller

Suppose that we have an n input - 1 output system to be controlled by a Mamdani controller. Suppose that $x_i \in X_i$ for $1 \leq i \leq n$. The rule base of such a controller consists of rules of the form:

$$R_r : if\ x_1 \text{ is } A_{r,1} \text{ and } x_2 \text{ is } A_{r,2} \text{ and } ...\text{and } x_n \text{ is } A_{r,n} \text{ then } y \text{ is } B_r$$

Here $1 \leq r \leq k$. Each $A_{r,i}$ and B_r is the linguistic label of some fuzzy set on X and Y respectively $\forall i \in \{1, 2, ..., n\}$ and $\forall r \in \{1, 2, ..., k\}$.

At this point the notation can be simplified to:

$$R_r : if\ \bar{x} \text{ is } \bar{A}_r \text{ then } y \text{ is } B_r \qquad \forall r \in \{1; ...; k\} \tag{2.3}$$

Such a rule can be represented as a fuzzy implication (conjunction) as:

$$\bar{A}_r \to B_r$$

The truth value of the implication is calculated as:

$\mu_{\bar{A}_r \to B_r}(\bar{x}, y) = \min\{\mu_{A_{r,1}}(x_1); \mu_{A_{r,2}}(x_2); ...; \mu_{A_{r,n}}(x_n); \nu_{B_r}(y)\}$ for a given vector $(\bar{x}, y)$.

We now combine the collection of rules R_r into a fuzzy relation as:

$$\mu_R(\bar{x}, y) = \max\{\min\{\mu_{\bar{A}_r}(\bar{x}); \nu_{B_r}(y)\} : r \in \{1; 2; ...; k\}\}$$

A given crisp input from the process is first fed into the fuzzifier module of the fuzzy controller. Here it is transformed into a fuzzy set as follows. Suppose the crisp input is $\bar{x}^0 = (x_1^0, x_2^0, ..., x_n^0)$. For each $i \in \{1; 2; ...; n\}$ the fuzzifier creates the fuzzy set A^i defined on X_i with membership function:

$$\mu_{A^i}(x) = \begin{cases} 0 & \text{if } x \neq x_i^0 \\ 1 & \text{if } x = x_i^0 \end{cases}$$

So a crisp input $(x_1^0, ..., x_n^0)$ creates a collection of fuzzy sets $A^i, i = 1, ..., n$ on the different universes of discourse. Simplifying the notation again, let

$$\bar{A} = \Pi_{i=1}^n A_i$$

This input fuzzy set is now composed with the fuzzy relation R to produce the output fuzzy set B according to the compositional rule of inference (CRI):

$$\begin{aligned} B &= R \circ A \text{ which is as before} \qquad (2.4) \\ \nu_B(y) &= \max_{\bar{x} \in \Pi_{i=1}^n X_i} \{\min\{\mu_R(\bar{x}, y); \mu_{\bar{A}}(\bar{x})\}\} \end{aligned}$$

Simplifying the last equation as follows gives a working definition that leads to an easier calculation process.

Suppose $\bar{x} \neq (x_1^0, x_2^0, ..., x_n^0)$. Then $\mu_{\bar{A}}(\bar{x}) = 0$

Suppose $\bar{x} = (x_1^0, x_2^0, ..., x_n^0)$. Then $\mu_{\bar{A}}(\bar{x}) = 1$. In this case

$$\begin{aligned} \min\{\mu_R(\bar{x}, y); \mu_{\bar{A}}(\bar{x})\} &= \mu_R(\bar{x}, y) \geq 0, \text{ hence} \\ &\max_{\bar{x} \in \Pi_{i=1}^n X_i} \{\min\{\mu_R(\bar{x}, y), \mu_{\bar{A}}(\bar{x})\}\} \end{aligned}$$

occurs when $\bar{x} = (x_1^0, ..., x_n^0)$ and clearly:

$$\begin{aligned} \nu^{output}(y) &= \mu_R(\bar{x}, y) \qquad (2.5) \\ &= \max_{r=1;...;k} \{\min\{\mu_{r,1}(x_1^0); ...; \mu_{r,n}(x_n^0); \nu_{B_r}(y)\}\} \qquad (2.6) \end{aligned}$$

Often observed data are disturbed by random noise. In such a case a different fuzzification interface has been found useful. In [76] such a data set containing noise is fuzzified into an isosceles triangle. The vertex of the triangle corresponds

to the mean of the data set, while the base of the triangle is twice the standard deviation.

The above CRI is the so - called max - min CRI. It is by far the most commonly used in fuzzy controller design. Other forms of the CRI that have been used are:

the sup - product operation due to Kaufmann;

the sup - bounded product and

the sup - drastic product operation (both due to Mizumoto)

The simplified inference procedure is now illustrated below using two rules:

Suppose the rules are:

$$\begin{aligned} R_1 &: \text{ if } x_1 \text{ is } A_{11} \text{ and } x_2 \text{ is } A_{12} \text{ then } y \text{ is } B_1 \\ R_2 &: \text{ if } x_1 \text{ is } A_{21} \text{ and } x_2 \text{ is } A_{22} \text{ then } y \text{ is } B_2 \end{aligned}$$

where A_{r1} and A_{r2} are linguistic labels of fuzzy sets on X_1 and X_2 respectively for $r = 1, 2$. B_1, B_2 are linguistic labels of fuzzy sets on Y.

For a crisp input (x_1^0, x_2^0) the inference process now proceeds as follows:

1) Calculate $\tau_r = \min\{\mu_{A_{r1}}(x_1^0), \mu_{A_{r2}}(x_2^0)\}$

2) Now define $\nu_r^{output}(y) = \begin{cases} \nu_{B_r}(y) & \text{if } \nu_{B_r}(y) \leq \tau_r \\ \tau_r & \text{otherwise} \end{cases}$

3) Calculate the union of $\nu_1^{output}(y)$ and $\nu_2^{output}(y)$ which is the output fuzzy set to be defuzzified.

The first two steps are shown below:

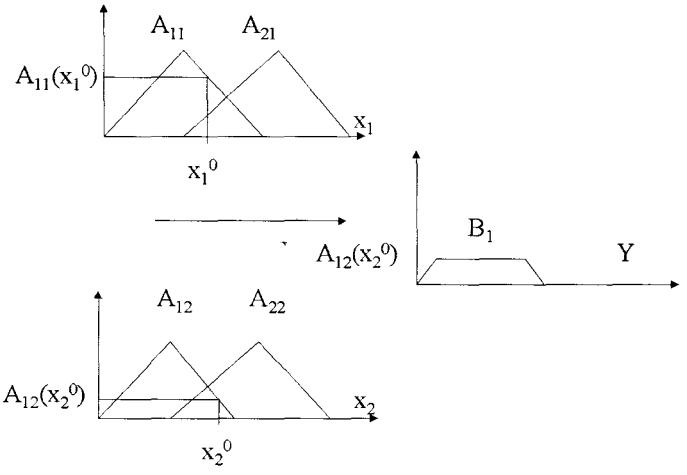


Figure 2.2: Inference in the Mamdani controller

The union of the two fuzzy sets on Y, B' in the figure is then the final fuzzy output to be defuzzified:

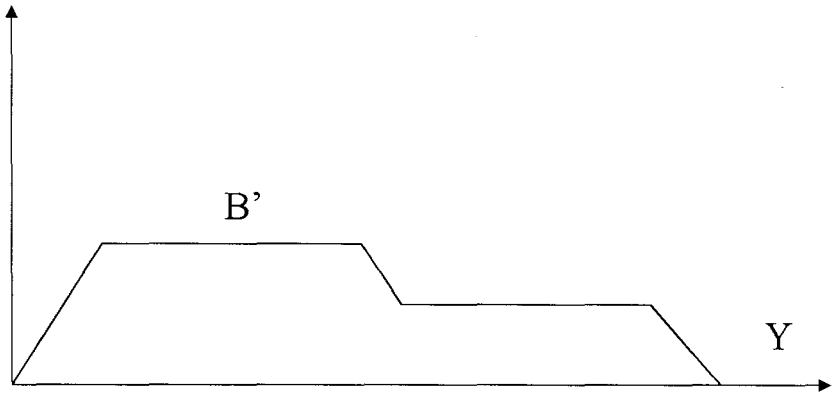


Figure 2.3: Output fuzzy set to be defuzzified

Defuzzification

Defuzzification is the process by which the output fuzzy set is transformed into a crisp value that can be used by a controller. The two most commonly used defuzzification strategies are:

Centre of Area Criterion

This method divides the first moment of the area under $\nu^{output}(y)$ in half:

$$y_0 = \frac{\int_Y y\nu^{output}(y)dy}{\int_Y \nu^{output}(y)dy}$$

The Maximum Criterion

This method chooses the smallest y value at which $\nu^{output}(y)$ reaches a maximum value.

Mean of Maxima Criterion

This method considers the collection of values for which $\nu^{output}(y)$ reaches a maximum value and calculates the centre of area of such values.

The Mean of Maxima is reported in to lead to harsh discontinuities. Braae and Rutherford in [3] present a study on defuzzification strategies and conclude that the COA leads to superior results.

The Sugeno Controller

The rules in a Sugeno Controller have form:

$R_r : if\ x_1$ is $A_{r,1}$ and x_2 is $A_{r,2}$ and ... and x_n is $A_{r,n}$ then $y = f_r(x_1, ..., x_n)$ $(r = 1, ..., k)$

Only the sets $X_1, X_2, ..., X_n$ are partitioned by fuzzy sets. f_r usually has form:

$$f_r(x_1, ..., x_n) = a_1^r \cdot x_1 + ... + a_n^r \cdot x_n + a_0^r$$

where $a_i^r \in \mathbb{R}\ \forall i \in \{0; 1; ...; n\}$.

For a crisp input value $\bar{x}^0 = (x_1^0, ..., x_n^0)$ the output of the fuzzy controller is calculated as:

First the degree of satisfaction (or degree of truth) of each rule is calculated as:

$$\tau_r = T(\mu_{A_{r,1}}(x_1^0), ..., \mu_{A_{r,n}}(x_n^0))$$

Secondly the final output of the controller is given by:

$$y = \frac{\sum_{r=1}^{k} \tau_r \cdot f_r(x_1^0, ..., x_n^0)}{\sum_{r=1}^{k} \tau_r}$$

The t-norm used in the calculation of τ_r is usually the minimum or product.

The Approximating Capabilities of Fuzzy Systems

Since the first fuzzy controllers were designed by Mamdani and Sugeno fuzzy control has been applied to an increasingly wider range of problems.

When one considers the wide range of applications one question that comes to mind is "What kind of systems and processes can be controlled by a fuzzy controller?" Another is "If a system is controllable (using conventional control) can one find a fuzzy controller that is capable of controlling the process as well?"

Some answers to questions like these were produced in [60][9]

In [60] the author shows that a certain class of Fuzzy Controllers are Universal Approximators. By this is meant that given a continuous function:

$$f : U \to \mathbb{R}$$

with

$$U \subset \mathbb{R}^n,\ U\ compact$$

there exists a fuzzy controller that can approximate the function f to an arbitrary degree of accuracy.

The proof as presented in [60] follows:

Let us first consider the design parameters of fuzzy systems. These are:

1) the number of fuzzy sets defined on the input and output universes of discourse.

2) the membership functions of these fuzzy sets.
3) the number of fuzzy rules in the rule base.
4) the linguistic statements of the fuzzy rules.
5) the decision making logic in the inference procedure.
6) the defuzzification method.

Thus there is a range of different "classes" of fuzzy controllers, each of which has a particular choice for each of the design parameters noted above. In what follows we focus on the class of fuzzy controllers with the following parameters:

a) The fuzzy rules have the form of rules in a Mamdani rule base.

b) all membership functions have the following Gaussian form:

$$\mu^i_{k_i,r}(x_i) = a^r_i \exp(-\frac{1}{2}(\frac{x_i - \bar{x}^r_i}{\sigma^r_i})^2)$$

with $i = 0, 1, ..., n$ and $r = 1, 2, ..., p$. Here $i = 0$ represents the membership functions for the output space, i.e.

$$A^0_{k_0,r} = B_{j_r}$$

Also $0 \leq a^r_i \leq 1$ and $\bar{x}^r_i$ is the point in the input or output space where the fuzzy set $\mu^i_{k_i,r}$ achieves its maximum membership value.

c) product inference logic is used.

d) the centre of area defuzzification method is used.

In the following we denote the set of fuzzy systems with the above parameters by **F**

i.e. let $U \subset \mathbb{R}^n$, then

$$\mathbf{F} = \{f : U \to \mathbb{R} \text{ s.t. } f(\bar{x}) = \frac{\sum^p_{r=1}(\zeta^r \Pi^n_{i=1}\mu^i_{k_i,r}(x_i))}{\sum^p_{r=1}(\Pi^n_{i=1}\mu^i_{k_i,r}(x_i))}\}$$

Here ζ^r is the point in the output space $\mathbb{R}$ at which ν_{j_r} achieves its maximum value. We assume that $p \geq 1$ and that U is compact.

We now make **F** into a metric space by defining:

$$d_\infty(f_1, f_2) = \sup_{\bar{x} \in U} (\mid f_1(\bar{x}) - f_2(\bar{x}) \mid) \qquad \forall f_1, f_2 \in F$$

It is clear that **F** is non - empty since $p \geq 1$. Also the denominator of the expression defining $f \in F$ is never zero since the Gaussian functions are never zero. The last two facts show that $(\mathbf{F}, d_\infty)$ is well - defined.

According to the Stone - Weierstrass theorem :

If Z is a set of real continuous functions on a compact set U s.t.:

1) Z is an algebra;

2) Z separates points on U and

3) $\forall$ point $u \in U \; \exists \; z \in Z$ which does not vanish at that point.

then (Z, d_∞) is dense in $(C[U], d_\infty)$. The proof follows in the form of three lemmas:

Lemma 15 $(\mathbf{F}, d_\infty)$ *is an algebra*

Proof.

Let $f_1, f_2 \in \mathbf{F}$.

Hence

$$f_1(\bar{x}) = \frac{\sum_{r_1=1}^{p_1}(\zeta^{r_1}\prod_{i=1}^{n}\mu^i_{k_i^1,r_1}(x_i))}{\sum_{r_1=1}^{p_1}(\prod_{i=1}^{n}\mu^i_{k_i^1,r_1}(x_i))}$$

and

$$f_2(\bar{x}) = \frac{\sum_{r_2=1}^{p_2}(\zeta^{r_2}\prod_{i=1}^{n}\mu^i_{k_i^2,r_2}(x_i))}{\sum_{r_2=1}^{p_2}(\prod_{i=1}^{n}\mu^i_{k_i^2,r_2}(x_i))}$$

Then

$$f_1(\bar{x}) + f_2(\bar{x}) = \frac{\sum_{r_1=1}^{p_1}\sum_{r_2=1}^{p_2}(\zeta^{r_1}+\zeta^{r_2})(\prod_{i=1}^{n}\mu^i_{k_i^1,r}(x_i)\mu^i_{k_i^2,r}(x_i))}{\sum_{r_1=1}^{p_1}\sum_{r_2=1}^{p_2}(\prod_{i=1}^{n}\mu^i_{k_i^1,r_1}(x_i)\mu^i_{k_i^2,r_2}(x_i))}$$

Since $\mu^i_{k_i^1,r_1}$ and $\mu^i_{k_i^2,r_2}$ are both Gaussian their product is Gaussian as well. Hence $f_1 + f_2 \in \mathbf{F}$.

Also

$$f_1(\bar{x}) \cdot f_2(\bar{x}) = \frac{\sum_{r_1=1}^{p_1}\sum_{r_2=1}^{p_2}(\zeta^{r_1}+\zeta^{r_2})(\prod_{i=1}^{n}\mu^i_{k_i^1,r}(x_i)\mu^i_{k_i^2,r}(x_i)}{\sum_{r_1=1}^{p_1}\sum_{r_2=1}^{p_2}(\prod_{i=1}^{n}\mu^i_{k_i^1,r}(x_i)\mu^i_{k_i^2,r}(x_i))}$$

which also has the required form, hence $f_1 f_2 \in \mathbf{F}$.

Finally, for arbitrary $c \in \mathbb{R}$:

$$cf_1(\bar{x}) = \frac{\sum_{r=1}^{p}(c\zeta^{r})(\prod_{i=1}^{n}\mu^i_{k_i,r}(x_i))}{\sum_{r=1}^{p}(\prod_{i=1}^{n}\mu^i_{k_i,r}(x_i))}$$

which shows that

$$cf_1 \in \mathbf{F}$$

as well.

Lemma 16 $(\mathbf{F}, d_\infty)$ *separates points on* U.

Proof.

$(\mathbf{F}, d_\infty)$ separates points on $U \iff$ for arbitrary $\bar{x}^0, \bar{y}^0 \in U$, s.t. $\bar{x}^0 \neq \bar{y}^0$ $\exists\, f \in \mathbf{F}$ *s.t.*

$$f(\bar{x}^0) \neq f(\bar{y}^0)$$

Given $\bar{x}^0, \bar{y}^0 \in U$, s.t. $\bar{x}^0 = (x_1^0, x_2^0, ..., x_n^0)$, $\bar{y}^0 = (y_1^0, y_2^0, ...y_n^0)$, we construct the required $f \in \mathbf{F}$ as follows.

$\forall i \in (1; ...; n\}$ we define 2 fuzzy sets:

$$\begin{aligned} \mu_i^1(x_i) &= \exp\left[-\frac{1}{2}(x_i - x_i^0)^2\right] \text{ with linguistic label } A_i^1 \text{ and} \\ \mu_i^2(x_i) &= \exp\left[-\frac{1}{2}(x_i - y_i^0)^2\right] \text{ with linguistic label } A_i^2 \end{aligned}$$

If $x_i^0 = y_i^0$ then $A_i^1 = A_i^2$ and the i-th subspace of U has only one fuzzy set defined on it.

We also define two fuzzy sets μ_1 and μ_2 with linguistic labels B_1 and B_2 respectively on $\mathbb{R}$:

$$\nu_j(z) = \exp\left[-\frac{(z - \bar{z}^j)^2}{z}\right], \text{ for } j = 1, 2$$

and $\bar{z}^j$ will be specified later. Our controller will run on two rules:

$$R_r: \text{ if } x_1 \text{ is } A_1^r \text{ and } x_2 \text{ is } A_2^r \text{ and } ... \text{ and } x_n \text{ is } A_n^r \text{ then } z \text{ is } B_r$$

Now

$$\begin{aligned} f(\bar{x}^0) &= \frac{\bar{z}_1 + \bar{z}_2 \Pi_{i=1}^n \exp(-\frac{(x_i^0 - y_i^0)^2}{2})}{1 + \Pi_{i=1}^n \exp(-\frac{(x_i^0 - y_i^0)^2}{2})} = \alpha \bar{z}_1 + (1 - \alpha)\bar{z}_2 \\ f(\bar{y}^0) &= \frac{\bar{z}_2 + \bar{z}_1 \Pi_{i=1}^n \exp(-\frac{(x_i^0 - y_i^0)^2}{2})}{1 + \Pi_{i=1}^n \exp(-\frac{(x_i^0 - y_i^0)^2}{2})} = \alpha \bar{z}_2 + (1 - \alpha)\bar{z}_1 \end{aligned}$$

where

$$\alpha = \frac{1}{1 + \Pi_{i=1}^n \exp\left[-\frac{(x_i^0 - y_i^0)^2}{2}\right]}$$

Now $\bar{x}^0 \neq \bar{y}^0 \Longrightarrow \exists i : x_i^0 \neq y_i^0$, *hence*

$$\begin{aligned} \Pi_{i=1}^n \exp\left[-\frac{(x_i^0 - y_i^0)^2}{2}\right] &\neq 1 \text{ or} \\ \alpha &\neq 1 - \alpha \end{aligned}$$

choose

$$\begin{aligned} \bar{z}_1 &= 0 \text{ and } \bar{z}_2 = 1, \text{ then} \\ f(\bar{x}^0) &= 1 - \alpha \neq \alpha = f(\bar{y}^0) \end{aligned}$$

Lemma 17 $(\mathbf{F}, d_\infty)$ *vanishes at no point of* U.

Proof.

Considering the form of the equations in (1) and the form of the inference function f, the required $f \in (F, d_\infty)$ is constructed by choosing all $\bar{z}_j > 0$ $(j = 1, ..., K)$

The three lemmas thus prove that for any given real continuous g defined on the compact $U \subset \mathbb{R}^n$ and arbitrary $\varepsilon > 0, \exists\ f \in (\mathbf{F}, d_\infty)$ *s.t.* :

$$\sup_{\bar{x} \in U} \mid g(\bar{x}) - f(\bar{x}) \mid < \varepsilon$$

The above shows that a certain class of fuzzy controllers is capable of approximating any real continuous function on a compact domain to arbitrary accuracy. More work on the mathematical properties of fuzzy controllers has been done . For example, in [7] the author proves a similar result for Sugeno type controllers. In [6] he develops a theory of the fuzzy controller and in [8] looks at relationships between neural networks, continuous functions and fuzzy systems.

Chapter conclusion

This chapter served the purpose of introducing the notation and terminology that we will require in the rest of the work. Another important part of the chapter is the discussion of Mamdani's model used to build fuzzy controllers. The final section on the approximating capabilities of fuzzy controllers was included to give some indication of the theoretical developments in the research areas in fuzzy systems.

Chapter 3

Designing fuzzy controllers

Introduction

This chapter focuses on the determination of the parameters of a fuzzy controller. This requires us to determine the fuzzy partitions of the input and output spaces as well as finding a set of fuzzy if-then rules that will drive the inference process of the fuzzy controller.

Semantics play an important role in fuzzy controller design. The linguistic terms of fuzzy sets used to partition the spaces are intended to form a link between the imprecise knowledge of an expert and the inference process implemented on a computer. Hence initial ideas on the design process of a fuzzy controller centred around a control expert expressing his knowledge of the system in terms of linguistic terms. These linguistic terms are then translated into fuzzy sets to be used in the inference process.

The above approach, however, suffers from a number of drawbacks. A number of papers [23], [53], [54] comment on the difficulty that process controllers experience in trying to express their knowledge of the system verbally. Also, in [23] the author notes that the system can actually learn the operators' failures as well as their successes.

After those attemps a number of other approaches have been developed. Many of these depend on analysing pairs of input-output data for the system to be controlled. The data will connect the desired output for a given input, hence can be seen as specifying a partially defined control function. The data set is then used to design a fuzzy controller that extends the partially defined control function to the rest of the input space.

A telecommunications network seems ideal for gathering data and using this approach. Once the variables have been decided upon the necessary software can be added to the nodes of the network. In this way the nodes of the network can record the values of the identified variables, perhaps as a function of time, during the operation of the network.

For the above reason we focus on methods that use pairs of input -output data.

The first method we consider starts out with a fuzzification of the crisp notion of an equivalence relation and other related concepts. The outcome of the development is an arrival at Mamdani's inference process with a new mathematical structure to support it.

Fuzzy control using fuzzy equivalence relations

Fuzzy equivalence relations on single spaces

Definition 18 *A fuzzy equivalence relation (w.r.t. the t-norm T) on the set X is a map:*

$$\begin{aligned} E &: X \times X \to I \text{ satisfying:} \\ E1)\ E(x,x) &= 1 \\ E2)\ E(x,y) &= E(y,x) \\ E3)\ E(x,z) &\geq T(E(x,y),E(y,z)) \qquad \forall x,y,z \in X \end{aligned}$$

Property $E1$ is referred to as reflexivity while $E2$ makes E symmetric. The last property means that E is transitive with respect to T

When the T-norm above is the Lukasiewicz t-norm then the following interesting relationship exists between fuzzy equivalence relations and pseudometrics on X.

Proposition 19 *Let T_L denote the Lukasiewicz conjunction as before. Then:*

(i) If E is a fuzzy equivalence relation on X with respect to T_L then $\delta_E = 1 - E$ is a pseudometric on X, and

(ii) If $\delta : X \to I$ is a pseudometric on X then $E_\delta(x,y) = 1 - \min(\delta(x,y),1)$ is a fuzzy equivalence relation on X w.r.t. T_L.

Proof.

(i) Suppose E is a fuzzy equivalence relation on X w.r.t. T_L. Then

(a) $\delta(x,y) = 1 - E(x,y) \geq 0 \qquad \forall x,y \in X$ is clear.

(b) $\delta(x,y) = 1 - E(x,y) = 1 - E(y,x) = \delta(y,x)$

(c) From the transitivity of E w.r.t. T_L get:

$$\begin{aligned} T_L(E(x,y),E(y,z)) &\leq E(x,z) \Rightarrow \\ \max(E(x,y)+E(y,z)-1,0) &\leq E(x,z), \text{ hence} \\ E(x,y)+E(y,z) &\leq E(x,z)+1, \text{ hence} \\ -E(x,z) &\leq 1-(E(x,y)+E(y,z)) \end{aligned}$$

Now

$$\begin{aligned} \delta_E(x,y)+\delta_E(y,z) &= 1-E(x,y)+1-E(y,z) \\ &= 2-(E(x,y)+E(y,z)) \\ &= 1+1-(E(x,y)+E(y,z)) \\ &\geq 1-E(x,z) \\ &= \delta_E(x,z) \end{aligned}$$

Hence δ_E is a pseudometric on X.
(ii) Suppose δ is a pseudometric on X. Then

$$\delta'(x,y) = \min\{\delta(x,y), 1\}$$

is also a pseudometric and so satisfies the triangle inequality

$$\delta'(x,z) \leq \delta'(x,y) + \delta'(y,z)$$

E1)

$$\begin{aligned} E_\delta(x,x) &= 1 - \min(\delta(x,x), 1) \\ &= 1 - \min(0,1) \\ &= 1 \end{aligned}$$

E2)

$$\begin{aligned} E_\delta(x,y) &= 1 - \min(\delta(x,y), 1) \\ &= 1 - \min(\delta(y,x), 1) \\ &= E_\delta(y,x) \end{aligned}$$

E3)

$$\begin{aligned} T_L(E_\delta(x,y), E_\delta(y,z)) &= \max(E_\delta(x,y) + E_\delta(y,z) - 1, 0) \\ &= \max(1 - \delta'(x,y) + 1 - \delta'(y,z) - 1, 0) \\ &= \max(1 - (\delta'(x,y) + \delta'(y,z)), 0) \\ &\leq \max(1 - \delta'(x,z), 0) \\ &= E_\delta(x,z). \end{aligned}$$

This completes the proof.

An ordinary (or crisp) equivalence relation $\approx$ on a set X defines a partition of X into equivalence classes s.t. if M is a part of the partition we have for $x, y \in X$

$$x \in M \text{ and } x \approx y \;\Rightarrow\; y \in M.$$

The following concept generalizes the above to the setting of fuzzy equivalence relations and fuzzy sets. Where necessary we will use the notation

$$xTy \text{ instead of } T(x,y)$$

for the action of a t-norm.

Definition 20 *A fuzzy set $\mu \in I^X$ is called extensional with respect to the fuzzy equivalence relation E (w.r.t. a t-norm T) on $X \Leftrightarrow$*

$$T(\mu(x), E(x,y)) \leq \mu(y) \qquad \forall x, y \in X$$

Definition 21 *Let E be a fuzzy equivalence relation on X and let $\mu \in I^X$. The fuzzy set*

$$\overline{\mu} = \wedge\{\nu : \mu \leq \nu \text{ and } \nu \text{ is extensional wrt } E\} \tag{3.1}$$

is called the extensional hull of μ wrt E.

Proposition 22 *Let E be a fuzzy equivalence relation on X and let $\mu \in I^X$. Then*

$$\text{(i) } \overline{\mu}(x) = \vee\{\mu(y)\ T\ E(x,y) : y \in X\} \tag{3.2}$$

$$\text{(ii) } \overline{\mu} \text{ is extensional wrt } E \tag{3.3}$$

$$\text{(iii) } \overline{\overline{\mu}} = \overline{\mu}$$

Proof.

(i) Let $\tilde{\mu}(x) = \vee\{\mu(z)\ T\ E(x,z) : z \in X\}$

Now

$$\begin{aligned}
\tilde{\mu}(x)\ T\ E(x,y) &= E(x,y)\ T\ \tilde{\mu}(x) \\
&= E(x,y)\ T \vee \{\mu(z)\ T\ E(x,z) : z \in X\} \\
&= \vee\{\mu(z)\ T\ E(x,y)\ T\ E(x,z)\} \\
&\leq \vee\{\mu(z)\ T\ E(y,z) : z \in X\} \\
&= \tilde{\mu}(y)
\end{aligned}$$

Hence $\tilde{\mu}$ is extensional wrt E.

Also

$$\tilde{\mu}(x) = \vee\{\mu(y)\ T\ E((x,y) : y \in X\} \geq \mu(x)\ T\ E(x,x) = \mu(x).$$

Now $\overline{\mu}(x)$ is the smallest extensional fuzzy set s.t. $\mu \leq \overline{\mu}$. Since $\tilde{\mu} \geq \mu$ and $\tilde{\mu}$ is extensional, we get $\tilde{\mu} \geq \overline{\mu}$.

To see that $\tilde{\mu} \leq \overline{\mu}$, let ν be an arbitrary fuzzy set extensional wrt E, s.t. $\mu \leq \nu$. Since

$$\nu(x) \geq \nu(y)\ T\ E(x,y) \geq \mu(y)\ T\ E(x,y) \qquad \forall y \in X$$

we get

$$\nu(x) \geq \vee\{\mu(y)\ T\ E(x,y) : y \in X\} = \tilde{\mu}(x).$$

Hence

$$\begin{aligned}
\overline{\mu}(x) &\geq \tilde{\mu}(x) \\
\text{and so } \overline{\mu} &= \tilde{\mu}.
\end{aligned}$$

(ii) proved in (i)

(iii) follows from (ii) and the definition of $\overline{\mu}$. This completes the proof.

Example 23 *We now consider the extensional hulls of some fuzzy sets that are common in control problems.*

1. Let $X \subseteq \mathbb{R}$, $x_0 \in X$

   For $x, y \in X$, we define

$$\begin{aligned} E(x,y) &= 1 - \min\{|\, x - y \,|, 1\}, \\ \mu_{x_0}(x) &= \begin{cases} 1 & \text{if } x = x_0 \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \\ T_L &: I \times I \to I \text{ is the Lukasiewicz t-norm} \end{aligned}$$

   We calculate the extensional hull of μ_{x_0} using 3.2 of the proposition above

$$\overline{\mu}_{x_0}(x) = \sup\{\max\{\mu_{x_0}(y) + E(x,y) - 1\} : y \in X\} \tag{3.4}$$

.Now $y \neq x_0 \Rightarrow \mu_{x_0}(y) = 0 \Rightarrow \max\{\mu_{x_0}(y) + E(x,y) - 1, 0\} = 0$.

Clearly now the sup in 3.4 occurs when $y = x_0$, hence

$$\begin{aligned} \overline{\mu}_{x_0}(x) &= \max\{\mu_{x_0}(x_0) + E(x_0,x) - 1, 0\} \\ &= E(x_0,x) \\ &= 1 - \min\{|\, x_0 - x \,|, 1\} \\ &= \begin{cases} 1 - (x - x_0) & \text{if } x \in [x_0, x_0 + 1) \\ 1 - (x_0 - x) & \text{if } x \in (x_0 - 1, x_0) \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

   which is the isosceles triangle with base length of 2 and vertex at $(x_0, 1)$.

   More generally, if we use $E(x,y) = 1 - \min\{k \mid x - y \mid, 1\}$ for a fixed $k > 0$ (this will also be a fuzzy equivalence relation on X w.r.t. T_L since $\min\{k \mid x - y \mid, 1\}$ is a pseudometric on X. - Proposition 19) we obtain for $\overline{\mu}_{x_0}(x)$ an isosceles triangle with base length $\frac{2}{k}$.

2. For the same fuzzy equivalence relation and t-norm as above we consider $M \subseteq X$, where M is an interval. Again, μ_M is the characteristic function of M. :

$$\mu_M(x) = \begin{cases} 1 & \text{if } x \in M \\ 0 & \text{otherwise} \end{cases}$$

$$\overline{\mu}_M(x) = \sup\{\max\{\mu_M(y) + E(x,y) - 1, 0\} : y \in X\}. \tag{3.5}$$

   Suppose that $x \in M$, then

$$\overline{\mu}_M(x) = E(x,x) = 1.$$

Let

$$\begin{aligned} \underline{x} &= \inf M \text{ and} \\ \overline{x} &= \sup M. \end{aligned}$$

If $x \in (\underline{x} - 1, \underline{x})$, then the sup in 3.5 occurs for some $y \in M$, since $\mu_M(y) = 0$ elsewhere. Also

$$E(x, y) = 1 - \min\{\mid x - y \mid, 1\}$$

To maximize $E(x, y)$ we need $y = \underline{x}$, leading to

$$\overline{\mu}_M(x) = E(x, \underline{x})$$

It is also easy to see that for $x \in (\overline{x}, \overline{x} + 1)$ we get

$$\overline{\mu}_M(x) = E(x, \overline{x})$$

The two extensional hulls are illustrated below.

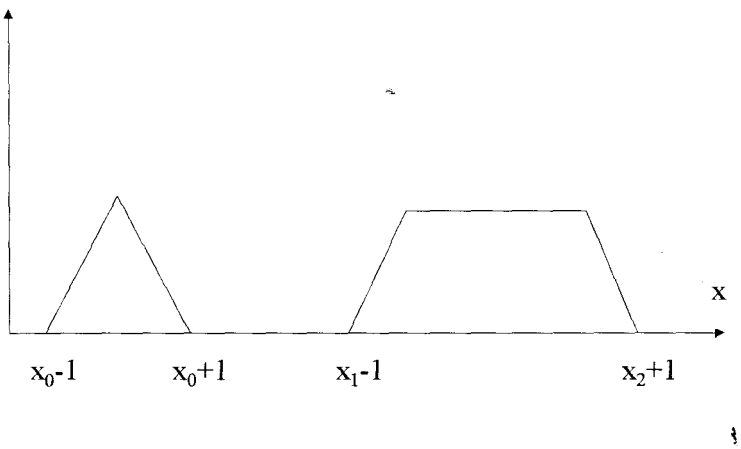


Figure 3.1: Extensional hulls of μ_{x_0} and μ_M.

Product spaces and mappings

We need now to extend the notion of fuzzy equivalence relations from a single space to a product of spaces. The fuzzy equivalence relations on single spaces are combined to form a fuzzy equivalence relation on the product space.

Suppose now that the spaces X and Y have the fuzzy equivalence relations E and F defined on them. Let H be the fuzzy equivalence relation to be defined on $X \times Y$.

What would one require from this H? Firstly that

$$H((x, y), (x', y'))$$

which is the degree of similarity of (x, y) and (x', y') depends only on $E(x, x')$ and $F(y, y')$ and not on the specific choice of x, x', y, y'. So we define

$$H : (X \times Y)^2 \to I$$

as

$$H((x, y), (x', y')) = h(E(x, x'), F(y, y'))$$

for some function $h : I^2 \to I$ that satisfies:

h1) $h(\alpha, \beta) = h(\beta, \alpha)$
h2) $h(\alpha, 1) = \alpha$
h3) $\alpha \leq \gamma \Rightarrow h(\alpha, \beta) \leq h(\gamma, \beta) \qquad \forall \alpha, \beta, \gamma \in I$

Clearly axioms E1 and E2 of definition 18 of 3.1.1 are satisfied by H defined in this way.

Here (h1) ensures that E and F have the same influence on H, while (h2) ensures that

$$H((x, y), (x', y)) = E(x, x')$$

(h3) requires that the degree of similarity between (x, y) and (x', y') does not exceed the degree of similarity between (x'', y) and (x''', y') if the degree of similarity between x and x' is less than or equal to the similarity degree between x'' and x'''.

Of course axiom E3 of Definition 18 of 3.1.1 (transitivity) must also be satisfied for H to be an equivalence relation on $X \times Y$. We consider the following cases:

Proposition 24 *Let E and F be fuzzy equivalence relations on X and Y respectively, w.r.t. a t-norm T. Then*

1.

$$H_T((x, y), (x', y')) = E(x, x') \; T \; F(y, y')$$

is a fuzzy equivalence relation on $X \times Y$ w.r.t T satisfying (h1)-(h3). In particular for $T = T_m$ (see example3) :

$$H_m((x, y), (x', y')) = \min\{E(x, x'), F(y, y')\}$$

is a fuzzy equivalence relation on $X \times Y$ with respect to T_m satisfying (h1)-(h3).

2. If H is a fuzzy equivalence relation with respect to T which satisfies (h1)-(h3), then

$$H_T \leq H \leq H_m$$

(cf. the observation under example3)

Proof.

1) H_T clearly satisfies (h1)-(h3). To show that H is an equivalence relation w.r.t. T, firstly observe that for a t-norm

$$T(T(a,b),T(c,d)) = T(T(a,c),T(b,d)) \tag{3.6}$$

since both sides reduce to $T(T(a,T(b,c)),d)$ by virtue of the associative law for t-norms. So:

$$\begin{aligned} H((x,y),(x',y')) &= T(E(x,x'),F(y,y')) && (3.7)\\ &\geq T(T(E(x,z),E(z,x')),T(F(y,z'),F(z',y'))) && (3.8)\\ &= T(T(E(x,z),F(y,z')),T(E(z,x'),F(z',y')))\\ &= T(H((x,y),(z,z')),H((z,z'),(x',y'))) && (3.9) \end{aligned}$$

Here 3.8 holds since E and F are equivalence relations with respect to T and 3.9 holds due to 3.6 above.

2) To prove 2 assume that $H((x,y),(x',y')) = h(E(x,x'),F(y,y'))$. On the one hand:

$$\begin{aligned} H((x,y),(x',y')) &\geq H((x,y),(x',y))\ T\ H((x',y),(x',y'))\\ &= h(E(x,x'),F(y,y))\ T\ h(E(x',x'),F(y,y'))\\ &= E(x,x')\ T\ F(y,y') \end{aligned}$$

On the other hand we need to show that

$$h(\alpha,\beta) \leq \min\{\alpha,\beta\}$$

From (h3) and (h2) get

$$h(\alpha,\beta) \leq h(\alpha,1) = \alpha$$

Also, using (h1) obtain

$$h(\alpha,\beta) \leq \beta$$

This completes the proof.

Given two spaces X and Y with similarity relations E and F respectively, we want to consider maps between X and Y that preserve the similarity of points.

Definition 25 *Let E and F be fuzzy equivalence relations on X and Y respectively. A mapping*

$$\varphi : X \to Y$$

is called extensional w.r.t. E and F if $E(x,x') \leq F(\varphi(x),\varphi(x')) \qquad \forall x,x' \in X.$

Thus the extensionality of a map requires that the degree of similarity of the images of two points be at least as great as the degree of similarity of the points.

Note1 This concept should be carefully distinguished from that of extensionality of a fuzzy set on X w.r.t. a fuzzy equivalence relation on X. (Definition 20)

Note2 If we have a fuzzy set $\mu : X \to I$ on a pseudometric space (X,δ) and the usual metric on I, then $E_\delta(x,x') = 1 - \min(\delta(x,x'),1)$ is a fuzzy equivalence relation on X w.r.t. T_L and so is $F(r_1,r_2) = 1 - \min(\mid r_1 - r_2 \mid, 1)$ on I. (See Proposition 19). Then saying that μ is extensional w.r.t. E_δ and F i.t.o. the Definition 25 above means that

$$\begin{aligned} 1 - \min(\delta(x,x'),1) \leq 1 - \min(\mid \mu(x) - \mu(x') \mid, 1) \qquad & \forall x,x' \in X \\ \text{or } \min(\mid \mu(x) - \mu(x') \mid, 1) \leq \min(\delta(x,x'),1) & \\ \text{or } \mid \mu(x) - \mu(x') \mid \leq \min(\delta(x,x'),1) & \end{aligned}$$

which is equivalent to ordinary continuity of μ.

Definition 26 *Let $E_1, E_2, ..., E_n$ be fuzzy equivalence relations on $X_1, ..., X_n$ respectively, w.r.t. the t-norm T. Define*

$$\begin{aligned} E_{1,...,n} : (X_1 \times ... \times X_n)^2 &\to I \\ ((x_1,...,x_n),(x_1',...,x_n')) &\mapsto \min\{E_1(x_1,x_1'),...,E_n(x_n,x_n')\} \end{aligned}$$

Then we have the following :

1. $E_{1,...,n}$ is a fuzzy equivalence relation on $X_1 \times X_2 \times ... \times X_n$.
2. The projection

$$\begin{aligned} \Pi_i : X_1 \times X_2 \times ... \times X_n &\to X_i \\ (x_1,...,x_n) &\mapsto x_i \end{aligned}$$

   is extensional w.r.t. $E_{1,...,n}$ and $E_i \qquad \forall i \in \{1;2;...n\}$.

3. If E is a fuzzy equivalence relation on $X_1 \times X_2 \times ... \times X_n$ s.t. all projections Π_i are extensional, then

$$E \leq E_{1,...,n}.$$

Proof.

1) $E_{1,...,n}$ is clearly reflexive and symmetric. It is also transitive w.r.t. T since:

$$\begin{aligned}
& E_{1,...,n}((x_1,...,x_n),(x'_1,...,x'_n))\ T\ E_{1,...,n}((x'_1,...x'_n),(x''_1,...,x''_n)) \\
= \ & \min\{E_i(x_i,x'_i) : i \in \{1;...;n\}\}\ T\ \min\{E_j(x'_j,x''_j) : j \in \{1;...;n\}\} \\
\leq \ & \min\{E_i(x_i,x'_i)\ T\ E_i(x'_i,x''_i)\} \\
\leq \ & \min\{E_i(x_i,x''_i)\} \\
= \ & E_{1,...,n}((x_1,...,x_n),(x''_1,...,x''_n)).
\end{aligned}$$

2)

$$\begin{aligned}
E_{1,...,n}((x_1,...,x_n),(x'_1,...,x'_n)) \ &\leq \ E_i(x_i,x'_i)) \\
&= \ E_i(\Pi_i(x_1,...,x_n),\Pi_i(x'_1,...,x'_n)).
\end{aligned}$$

3) Since Π_i is extensional w.r.t. E and E_i, get:

$$\begin{aligned}
E((x_1,...,x_n),(x'_1,...x'_n)) \ &\leq \ E_i(x_i,x'_i)\ \forall i \in \{1;...;n\} \text{ and hence} \\
E((x_1,...x_n),(x'_1,...,x'_n)) \ &\leq \ E_{1,...,n}((x_1,...,x_n),(x'_1,...,x'_n))
\end{aligned}$$

This completes the proof.

Mamdani's Model Revisited

Design of a controller based on similarity

The control expert must first decide on canonical representations of the linguistic variables which make up the k rules, i.e. he has to provide k $(n+1)$ tuples:

$$((x_1^r,...,x_n^r),y^r) \in (X_1 \times ... \times X_n) \times Y \ : r \in \{1;2;...;k\}$$

such that for example, x_1^r is a point in X_1 which can be truly considered "large" if that is the linguistic variable in that case, etc. In other words the rules can be expressed as follows:

R_r : If x_1 is approximately x_1^r; x_2 is approximately x_2^r; ...; x_n is approximately x_n^r; then y is approximately y^r

On each X_i and Y we decide on fuzzy equivalence relations $E_1,...,E_n$ and F with respect to a t-norm. Then we use on the product space $X_1 \times X_2 \times \cdots \times X_n \times Y$ the equivalence relation

$$E((x_1,...,x_n,y),(x_1',...,x_n',y')) = \min\{E_1(x_1,x_1');...;E_n(x_n,x_n');F(y,y')\}.$$

of Definition 26. In other words, in view of Proposition 27, we choose the coarsest equivalence relation on the product space that will make the projections extensional.

Now for a crisp input $(x_1,x_2,...,x_n) \in X_1 \times X_2 \times \cdots X_n$ and an arbitrary $y \in Y$, $(x_1,x_2,...,x_n,y)$ is equivalent to the pre-assigned $(x_1^r,x_2^r,...,x_n^r,y^r)$ to the degree

$$E((x_1^r,...x_n^r,y^r),(x_1,x_2,...x_n,y))$$

which we can write as

$$O_r(x_1,...,x_n)(y).$$

This is a fuzzy set on Y which represents the output through rule r as a result of the crisp input $(x_1,...,x_n)$.

Clearly, the rule which gives the largest value (degree of equivalence) is the most significant for the specific input $(x_1,...,x_n)$.

So we consider the final output as:

$$\begin{aligned} O(x_1,...,x_n)(y) &= \max_{1\le r\le n} O_r(x_1,...,x_n)(y) \\ &= \max_r \min\{E_1(x_1^r,x_1),...,E_n(x_n^r,x_n),F(y^r,y)\}. \end{aligned}$$

If we use the equivalence relations $E_1,E_2,...,E_n,F$ w.r.t. T_L as in Example 23, then, as we have seen in that example, $E_1(x_1^r,x)$ is the extensional hull $\overline{\mu}_{x_1^r}(x)$ of the crisp point x_1^r.

So we get

$$O(x_1,...,x_n) = \max_r \min\{\overline{\mu}_{x_1^r}(x_1),...,\overline{\mu}_{x_n^r}(x_n),\overline{\mu}_{y^r}(y)\}$$

Now one sees the connection with Mamdani's model if we compare this last equation with equation 2.6 from Chapter 2.

Fuzzy Control using fuzzy relational equations

We now refer back to Chapter 2, where the controlled system is modelled as a fuzzy relational equation of form

$$B = A \circ R$$

For this section we let X_r and Y_r (for $r \in \{1;...;k\}$) denote fuzzy sets on X and Y respectively, while R is a fuzzy relation on the product space $X \times Y$. For each r, X_r and Y_r satisfy the above equation, hence

$$Y_r = X_r \circ R \qquad (3.10)$$

Thus we consider a collection of fuzzy sets X_r, as input measurements to a process controller. Each X_r produces an output fuzzy set by application of 3.10.

The problem we focus on now is the following. Given a collection $\{(X_r, Y_r) : r \in \{1; ...; k\}\}$, we are interested in finding a fuzzy relation R that satisfies all k equations simultaneously. We are also interested in finding maximal solutions to such a set of fuzzy relational equations. We will prove a theorem regarding the maximal solution for such a system of equations. We need some new concepts in order to develop our approach.

Definition 27 *Let L be a lattice, $a, b \in L$. We define the operation α as follows. Let $c = a\alpha b$. Then c is the greatest element in L s.t.*

$$a \wedge c \leq b \tag{3.11}$$

We call c the *relative pseudocomplement* of a in b.

The following expression can now be seen to be equivalent to the definition above.

$$a\alpha b = \begin{cases} 1 & \text{if } a \leq b \\ b & \text{if } a > b \end{cases} \tag{3.12}$$

We now need to be able to apply this operation to fuzzy sets and relations, hence we extend the definition to these cases as

Definition 28 *Let A and B be fuzzy sets on X and Y respectively, R a fuzzy relation on $X \times Y$.*

1. If

$$B = A\overline{\alpha}R \tag{3.13}$$

then B has membership function

$$\mu_B(y) = \wedge\{\mu_A(x)\alpha\mu_R(x, y) : x \in X\} \tag{3.14}$$

2.

$$G = A\overline{\alpha}B \tag{3.15}$$

has membership function

$$\mu_G(x, y) = \mu_A(x)\alpha\mu_B(y) \qquad \forall x \in X, \forall y \in Y \tag{3.16}$$

Now that we have the necessary definitions in place, let us prove some lemmas that we need for the main results

$$a \wedge (a\alpha b) \leq b \tag{3.17}$$

2

$$a\alpha b \geq b \tag{3.18}$$

3

$$a\alpha(a\alpha b) \geq b \tag{3.19}$$

4

$$a\alpha(b \vee c) \geq a\alpha b \ \forall a, b, c \in L \tag{3.20}$$

5

$$R \subseteq X_r \overline{\alpha}(X_r \circ R) \tag{3.21}$$

6

$$X_r \circ (X_r \overline{\alpha} Y_r) \subseteq Y_r \tag{3.22}$$

Proof.

1.

$$a \wedge (a\alpha b) = \begin{cases} a \wedge 1 & \text{if } a \leq b \\ a \wedge b & \text{if } a > b \end{cases}$$

Suppose that $a \leq b$. Then

$$a \wedge (a\alpha b) = a \wedge 1 = a \leq b$$

Suppose $a > b$. Then

$$a \wedge (a\alpha b) = a \wedge b = b$$

Hence the result.

2.

$$a\alpha b = \begin{cases} 1 & \text{if } a \leq b \\ b & \text{otherwise} \end{cases}$$

Hence the result is clear.

3.

$$a\alpha b' = \begin{cases} 1 & \text{if } a \leq b \\ b & \text{if } a > b \end{cases} \tag{3.23}$$

hence

$$\begin{aligned} a\alpha(a\alpha b) &= \begin{cases} a\alpha 1 & \text{if } a \leq b \\ a\alpha b & \text{if } a > b \end{cases} \\ &= \begin{cases} 1 & \text{if } a \leq b \\ a\alpha b & \text{if } a > b \end{cases} \\ &= \begin{cases} 1 & \text{if } a \leq b \\ b & \text{if } a > b \end{cases} \end{aligned}$$

which proves that the result holds.

4.

$$a\alpha(b \vee c) = \begin{cases} 1 & \text{if } a \leq (b \vee c) \\ b \vee c & \text{if } a > (b \vee c) \end{cases}$$

Now if $a\alpha(b \vee c) = 1$ then $a\alpha(b \vee c) \geq a\alpha b$ is clear.
If $a > (b \vee c\ )$ then $a > b$ and $a > c$. Then

$$a\alpha(b \vee c) = b \vee c = \begin{cases} b & \text{if } b \geq c \\ c & \text{if } c > b \end{cases}$$

Now

$$b \vee c = b = a\alpha b \text{ (since } a > b)$$

hence

$$a\alpha(b \vee c) = a\alpha b$$

and

$$b \vee c = c > b = a\alpha b$$

hence

$$a\alpha(b \vee c) > a\alpha b$$

5. Let

$$\beta = \mu_{X_r \overline{\alpha}(X_r \circ R)}(x_0, y_0) = \mu_{X_r}(x_0)\alpha\mu_{(X_r \circ R)}(y_0)$$
$$= \begin{cases} 1 & \text{if } \mu_{X_r}(x_0) \leq \gamma \\ \gamma & \text{if } \mu_{X_r}(x_0) > \gamma \end{cases}$$

where $\gamma = \vee\{\mu_{X_r}(x) \wedge \mu_R(x, y_0) : x \in X\}$
Now suppose that $\beta = 1$. In this case the result is immediate.
Otherwise, suppose

$$\beta = \gamma = \vee\{\mu_{X_r}(x) \wedge \mu_R(x, y_0) : x \in X\}$$

Then

$$\mu_{X_r}(x_0) > \vee\{\mu_{X_r}(x) \wedge \mu_R(x, y_0) : x \in X\} \geq \mu_{X_r}(x_0) \wedge \mu_R(x_0, y_0)$$

hence

$$\begin{aligned} \mu_{X_r}(x_0) &> \mu_{X_r}(x_0) \wedge \mu_R(x_0, y_0) \text{ hence} \\ \mu_{X_r}(x_0) \wedge \mu_R(x_0, y_0) &= \mu_R(x_0, y_0) \text{ or} \\ \mu_R(x_0, y_0) &= \mu_{X_r}(x_0) \wedge \mu_R(x_0, y_0) \\ &\leq \vee\{\mu_{X_r}(x) \wedge \mu_R(x, y_0) : x \in X\} = \gamma = \beta \end{aligned}$$

This completes the proof.

6

$$\mu_{X_r \circ (X_r \overline{\alpha} Y_r)}(y) = \vee\{\mu_{X_r}(x) \wedge \mu_{(X_r \overline{\alpha} Y_r)}(x, y) : x \in X\}$$
$$= \mu_{X_r}(x_0) \wedge \mu_{(X_r \overline{\alpha} Y_r)}(x_0, y), \text{ for some } x_0 \in X$$

let

$$\beta = \mu_{X_r}(x_0) \wedge [\mu_{X_r}(x_0) \alpha \mu_{Y_r}(y)]$$

suppose $\mu_{X_r}(x_0) \leq \mu_{Y_r}(y)$
then

$$\beta = \mu_{X_r}(x_0) \wedge 1 = \mu_{X_r}(x_0) \leq \mu_{Y_r}(y)$$

otherwise $\mu_{X_r}(x_0) > \mu_{Y_r}(y)$ Then

$$\beta = \mu_{X_r}(x_0) \wedge \mu_{Y_r}(y) \leq \mu_{Y_r}(y)$$

Hence the result
We are now in a position to state and prove the theorems that we need.

Theorem 29 *Given the equation $Y_r = X_r \circ R$, the least upper bound solution to it is given by:*

$$\widehat{R} = X_r \overline{\alpha} Y_r$$

Proof.
Let

$$T = \{R \in \Im(X \times Y) : Y_r = X_r \circ R\}$$

We show

1. $\widehat{R} \in T$
2. $R \in T \Rightarrow R \subseteq \widehat{R}$
3. if $R_0 \in \Im(X \times Y)$ and $R_0 \geq R \; \forall R \in T$ then $\widehat{R} \subseteq R_0$

1.

$$X_r \circ (X_r \overline{\alpha} Y_r) \subseteq Y_r \text{ from lemma 6}$$
$$\text{hence } X_r \circ \widehat{R} \subseteq Y_r$$

Also

$$R \subseteq X_r \overline{\alpha}(X_r \circ R) \text{ from lemma 5}$$
$$\text{hence } X_r \circ R \subseteq X_r \circ (X_r \overline{\alpha}(X_r \circ R))$$
$$\text{hence } Y_r \subseteq X_r \circ (X_r \overline{\alpha} Y_r)$$
$$\text{hence } Y_r \subseteq X_r \circ \widehat{R}$$

hence the result.

2. From lemma 5 we get

$$\begin{aligned} R &\subseteq X_r \overline{\alpha}(X_r \circ R) \\ \text{and } R \in T &\Rightarrow Y_r = X_r \circ R \\ \text{hence } R &\subseteq X_r \overline{\alpha} Y_r \\ \text{and } \widehat{R} &= X_r \overline{\alpha} Y_r \\ \text{hence } R &\subseteq \widehat{R} \end{aligned}$$

3. This is clear since $\widehat{R} \in T$. This completes the proof.

Application in controller design

One can now design a controller in which the inference of the output fuzzy set(s) for a given input fuzzy set occurs as the calculation of a fuzzy relational equation. We illustrate this with a simple example.

Example 30 *Suppose we have a single input - single output system where the input space is* $X = \{x_1; x_2; ...; x_5\}$ *and the output space is* $Y = \{y_1; y_2; ...; y_5\}$. *We have the following fuzzy sets defined on the two spaces:*

$$X_S = [1\ .8\ .6\ .2\ 0] \tag{3.24}$$
$$X_M = [.2\ .6\ 1\ .6\ .2]$$
$$X_L = [.2\ .4\ .6\ .8\ 1]$$
$$Y_S = [.2\ .4\ .8\ 1\ 1] \tag{3.25}$$
$$Y_M = [.2\ .4\ 1\ .4\ .2]$$
$$Y_L = [1\ .8\ .6\ .4\ .2]$$

The controller that we design operates on the following three rules:

$$\begin{aligned} r_1 &: \text{If } x \text{ is } S \text{ then } y \text{ is } L \\ r_2 &: \text{If } x \text{ is } M \text{ then } y \text{ is } M \\ r_3 &: \text{If } x \text{ is } L \text{ then } y \text{ is } S \end{aligned}$$

In the case of this controller designed using fuzzy relational equations the rules become:

$$\begin{aligned} r_1 &: Y_L = X_S \circ R_1 \\ r_2 &: Y_M = X_M \circ R_2 \\ r_3 &: Y_S = X_L \circ R_3 \end{aligned}$$

We now think of the relations $R_1 - R_3$ as translating each input fuzzy set to an output fuzzy set. At this point we use the definitions of the input and output fuzzy sets to determine the relations $R_1 - R_3$ as follows:

We make use of the operation defined in 3.16 and calculate:

1.

$$R_1(x, y) = X_S(x) \alpha Y_L(y)$$

which turns out to be

$$R_1 = \begin{bmatrix} 1 & .8 & .6 & .4 & .2 \\ 1 & 1 & .6 & .4 & .2 \\ 1 & 1 & 1 & .4 & .2 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

2.

$$R_2(x, y) = X_M \alpha Y_M$$

which is

$$R_2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ .2 & .4 & 1 & .4 & .2 \\ .2 & .4 & 1 & .4 & .2 \\ .2 & .4 & 1 & .4 & .2 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

and the last relation is

$$R_3 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ .2 & 1 & 1 & 1 & 1 \\ .2 & .4 & 1 & 1 & 1 \\ .2 & .4 & 1 & 1 & 1 \\ .2 & .4 & .8 & 1 & 1 \end{bmatrix}$$

The last step of this process is to take the intersection of the three fuzzy relations to produce a single relation that is used for any input fuzzy set. This turns out to be

$$R = \begin{bmatrix} 1 & .8 & .6 & .4 & .2 \\ .2 & .4 & .6 & .4 & .2 \\ .2 & .4 & 1 & .4 & .2 \\ .2 & .4 & 1 & .4 & .2 \\ .2 & .4 & .8 & 1 & 1 \end{bmatrix}.$$

For a given input fuzzy set $A = [.2\ .5\ .6\ .3\ .4]$ we now calculate the output fuzzy set as:

$$\begin{aligned} B(y) &= (A \circ R)(y) \\ &= \sup_{x \in X} [A(x) \wedge R(x,y)] \\ &= [.2\ .4\ .6\ .4\ .4] \end{aligned}$$

This then completes this section dealing with some of the theory of fuzzy relational equations and their application to controller design.

The method of Wang and Mendel

Wang and Mendel developed a general method for using numerical data to develop a fuzzy controller. Their five step procedure is quite simple and is outlined below.

The system

Let us consider an n-input m-output system where one datum is represented as:

$$(x_{1r}, x_{2r}, ..., x_{nr}, y_{1r}, y_{2r}, ..., y_{mr}) \tag{3.26}$$

with

$$x_{ir} \in X_i\ \forall i \in \{1; 2; ...; n\} \text{ and } y_{jr} \in Y_j\ \forall j \in \{1; 2; ...; m\}$$

Also, we assume that we have k such input output pairs of data available, i.e.

$$r \in \{1; 2; ...; k\}$$

The steps

Step1

During this step the input and output spaces are partitioned by fuzzy sets. The decision as to how many of these is left up to the developer, as well as the shapes of membership functions. Let us denote the membership functions on set X_i by

$$\mu_{i1}, \mu_{i2}, ..., \mu_{it(i)} \forall i$$

and the ones on Y_j by

$$\nu_{j1}, \nu_{j2}, ..., \nu_{js(j)} \forall j$$

Let $A_{i1}, ..., A_{it(i)}$ and $B_{j1}, ..., B_{js(j)}$ be the corresponding linguistic labels for the fuzzy sets above.

Step2

During this step we start to construct the rule base for the controller. We consider an input - output tuple as in 3.25 We determine the fuzzy set on domain X_i to which x_{ir} has the highest membership value as the fuzzy set (or equivalently the linguistic label) to occur in position i of rule r. This is done for every coordinate entry of the input - output tuple. It leads to a rule:

$$R_r \ : \ \text{if } x_1 \text{ is } A_{1\beta_r(1)} \text{ and } x_2 \text{ is } A_{2\beta_r(2)} \text{ and ...and } x_n \text{ is } A_{n\beta_r(n)}$$
$$\text{then } y_1 \text{ is } B_{1\gamma_r(1)} \text{ and ...and } y_m \text{ is } B_{m\gamma_r(m)}$$

This is done for each of the k pairs of data, leading to k rules of the above form.

Step3

We now have a collection of k rules - one derived from each datum. Clearly it is possible to generate inconsistent rules, i.e. two rules are inconsistent if they specify the same antecedent fuzzy sets but different consequent fuzzy sets. To overcome this problem we define and calculate a degree of validity for each rule. From a collection of inconsistent rules we then simply choose the rule with highest validity and discard the rest. The most natural way of defining the degree of validity of rule r is

$$d_1(r) = \Pi_{i=1}^{n} \mu_{i\beta_r(i)}(x_{ir}) \times \Pi_{j=1}^{m} \nu_{j\gamma_r(j)}(y_{jr})$$

If an expert on the system is available we can now also define a subjective degree of validity for each datum. This gives a measure to which we believe that the particular datum is representative of the system. Thus data points that give the desired relationship between input and output get higher scores for the subjective degree of validity than those that give a less realistic relationship. Let us label the second degree of validity as $d_2(r)$. Finally

$$d(r) = d_1(r) \times d_2(r)$$

is the final value for the degree of validity of rule r.

Step4

In addition to the rules derived from the data the expert can also give linguistic rules based on his/her experience of the system under control. The rules derived in the previous steps are all "and" rules, i.e. rules where every condition in the antecedent must be satisfied for the rule to fire. In this step it is possible that "or" rules, i.e. rules where only one condition in the antecedent has to be satisfied for the rule to fire, are added to the list of rules. The process of comparing the degrees of validity of different rules is repeated and the best rules are chosen again.

With the rules determined above in place the following defuzzification procedure is used.

Defuzzification

We define the *centre* of a fuzzy set $\nu_{j\gamma_r(j)}$ as

$$centre(\nu_{j\gamma_r(j)}) = y^0_{j\gamma_r(j)} \in core(\nu_{j\gamma_r(j)}) :| y^0_{j\gamma_r(j)} |\leq| y \mid \forall y \in core(\nu_{j\gamma_r(j)})$$

Then for rule r and input $(x_1, ..., x_n)$ we define:

$$\deg(r) = \Pi_{i=1}^{n} \mu_{i\beta_r(i)}(x_i)$$

The output is then calculated as

$$y_j = \frac{\sum_{r=1}^{k} \deg(r) \times y^0_{j\gamma_r(j)}}{\sum_{r=1}^{k} \deg(r)}$$

for each coordinate j.

The authors of [41] illustrate their approach by designing a controller for the truck backer-upper problem. The problem consists of designing a controller that can reverse a truck into a loading dock position. They also design a controller that predicts the next element in a chaotic time series. The method is clearly very simple and their two applications show good results.

Fuzzy clustering

Fuzzy clustering has emerged as another method for designing fuzzy controllers.

Clustering techniques

Fuzzy clustering algorithms are used to search for patterns in data.

Given a collection:

$$X = \{x_1; x_2; ...; x_n\}$$

of vectors $x_k \in \mathbb{R}^m$, a hard or crisp clustering algorithm will produce a partition of X. An element $x_k \in X$ will have a membership degree of 1 to one of the parts and zero to the rest. Fuzzy clustering fuzzifies this notion by producing a number of fuzzy clusters. Each point $x_k \in X$ will have a degree of membership (ranging in $[0, 1]$) to each fuzzy cluster. In this section we consider two fuzzy clustering algorithms and discuss how these can be used in designing fuzzy controllers.

The Fuzzy c-Means Algorithm

This algorithm classifies the data set into c clusters. Note that c has to be at least 2, otherwise all points belong to the same (and only) cluster. Also c=n would imply that each cluster contains a single point. Hence for the problem to be non-trivial we require that

$$2 \leq c < n$$

The algorithm works best when the data are approximately evenly distributed around distinct cluster centers. We think of a cluster center v_i ($i = 1; ...; c$) as being a prototype of the elements in cluster i. Note that v_i need not be a point in cluster i. The FCM algorithm aims to minimize the sum of distances of points from prototypes.

Letting μ_{ik} be the membership degree of datum k to cluster i, v_i be the prototype for cluster i, and d_{ik} be the distance of datum k to the $i-th$ cluster center, v_i, we seek values of μ_{ik} and v_i such that the value of the objective function:

$$J(c) = \sum_{k=1}^{n} \sum_{i=1}^{c} (\mu_{ik})^w (d_{ik})^2 \tag{3.27}$$

is minimized. In 3.27 w is a weight on membership values, referred to as the fuzziness index. We place the following two constraints on membership values:

1.

$$\sum_{k=1}^{n} \mu_{ik} > 0 \ \forall \ i \in \{1; ...; c\} \text{ and}$$

2.

$$\sum_{i=1}^{c} \mu_{ik} = 1 \ \forall k \in \{1; ...; n\}$$

The first constraint ensures that each cluster has at least one point with non-zero membership degree to it. The second constraint means that the sum of membership degrees for each element k must be one. In the Appendix we prove that the membership values μ_{ik} that leads to a minimum value of $J(c)$ are given by the expression:

$$\mu_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{d_{ik}}{d_{jk}}\right)^{\frac{2}{w-1}}} \tag{3.28}$$

A necessary condition on the prototypes is:

$$v_i = \frac{\sum_{k=1}^{n} (\mu_{ik})^w \cdot x_{kj}}{\sum_{k=1}^{n} (\mu_{ik})^w} \tag{3.29}$$

We collect together the membership degrees μ_{ik} in a partition matrix

$$U = \begin{bmatrix} \mu_{11} & \cdot & \cdot & \cdot & \mu_{1n} \\ \cdot & \cdot & & & \cdot \\ \cdot & & \cdot & & \cdot \\ \cdot & & & \cdot & \cdot \\ \mu_{c1} & \cdot & \cdot & \cdot & \mu_{cn} \end{bmatrix}$$

The FCM algorithm iterates the following steps:

1. Given the desired number of clusters, c, the fuzziness index, w, we make an initial guess at the partition matrix, U^0.

2. Calculate the cluster centers (prototypes) using:

$$v_{ij}^{(t)} = \frac{\sum_{k=1}^{n}(\mu_{ik}^{(t)})^w \cdot x_{kj}}{\sum_{k=1}^{n}(\mu_{ik}^{(t)})^w}$$

3. Compute the distances from each element in the set to each cluster center, using:

$$d_{ik}^{(t)} = \|x_k - v_i^{(t)}\| = \left(\sum_{j=1}^{m}(x_{kj} - v_{ij}^{(t)})^2\right)^{\frac{1}{2}}$$

for each cluster $i = 1; ...; c$ and elements $k = 1; ...; n$

4. Update the membership values of each data point. The updated values μ_{ik} of element k in cluster i are computed by:

$$\mu_{ik}^{(t+1)} = \frac{1}{\left(\sum_{j=1}^{c}(\frac{d_{ik}^{(t)}}{d_{jk}^{(t)}})^{\frac{2}{w-1}}\right)}$$

and the partition matrix U is updated.

5. The iterative process stops when it has converged under some selected norm; otherwise a new iteration is performed (set $t = t + 1$ and return to step 2). The norm for checking convergence might be:

$$\max_{i,k} \mid \mu_{ik}^{(t+1)} - \mu_{ik}^{(t)} \mid \leq \varepsilon$$

for some predifined value of $\varepsilon > 0$.

Deriving rules from fuzzy clusters

In order to derive if-then rules from the fuzzy clusters we project each cluster to each coordinate space. This is done by taking the i-th coordinate of each data point and assigning to it the membership value of the original data point to the cluster. This yields a discrete fuzzy set on the $i-th$ coordinate space. This fuzzy set can be extended to the whole space by a piecewise linear fuzzy set defined on the basis of the discrete points, or an enveloping fuzzy set. Each fuzzy cluster will then induce a rule of form:

$$if\ \xi_1 \text{ is } \mu_1 \text{ and } ...\text{and } \xi_{n-1} \text{ is } \mu_{n-1} \text{ then } \xi_n \text{ is } \mu_n$$

Here, the μ_i denotes the extension of the $i-th$ projection of the considered cluster and $\xi_1, ..., \xi_{n-1}$ are input variables, while ξ_n is the output variable. The above rule would clearly be for a multi input - single output controller. It is easy to see how it can be extended to the multi input multi output case.

As mentioned above the FCM algorithm produces the best results when the data are approximately spherically distributed around the cluster centre. What of cases where a linear model better approximates the data? In these cases we can make use of fuzzy c-lines and fuzzy c-elliptotypes.

Fuzzy c-lines and fuzzy c-elliptotypes

For a collection of points X in three dimensions we calculate the centre of gravity of the points as:

$$v = \frac{1}{n}\sum_{k=1}^{n} x_k$$

This point v has the property that the sum of squared distances of data points to v is minimal. We are interested in a line that best represents the data points. We would thus require the line with the property that the sum of squared distances from the data points to the line is minimal. For a line l, we now define the moment of inertia of the line w.r.t. the data points as:

$$I_l = \sum_{k=1}^{n} (d_{kl})^2$$

where d_{kl} is the distance of the point k to the line l. The line with smallest moment of inertia passes through the centre of gravity. We refer to this line as the *principal axis of inertia* of the data set and we find its direction by determining the largest eigenvector of the matrix:

$$J = \sum_{k=1}^{n} (x_k - v)(x_k - v)^T$$

Here $(x_k - v)$ is a column vector. The expression for the distance d_{kl} from point x_k to line l is:

$$d_{kl} = \sqrt{\|x_k - v\|^2 - ((x_k - v) \bullet u)^2}$$

The figure below illustrates this distance.

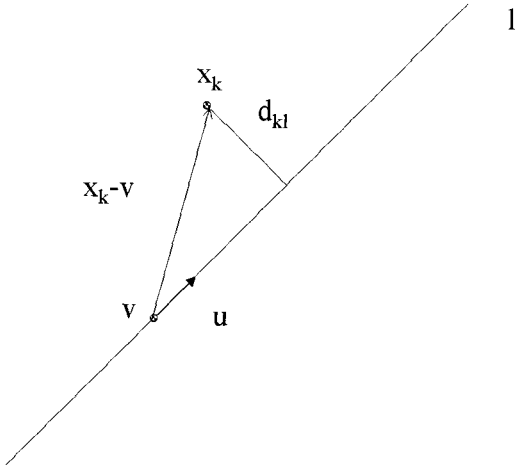


Figure 3.2: Calculating distances for the fuzzy c-elliptotypes

The dot indicates scalar product, i.e. the projection of the vector $x_k - v$ in the direction of u. The vector u is determined as said above as the largest eigenvector of the matrix J. Suppose now that we are searching for a number, c clusters again. We can now drop the restriction on the dimensionality, i.e. the data need not necessarily be three-dimensional. We modify the above formulae as:

$$d_{kl} = \sqrt{\|x_k - v_i\|^2 - \sum_{j=1}^{q} ((x_k - v_i) \bullet u_{ij})^2} \qquad (3.30)$$

Here v_i is the centre of cluster i, while u_{ij} is the $j-th$ largest eigen-value of the matrix:

$$J_i = \sum_{k=1}^{n} (\mu_{ik})^w (x_k - v_i)(x_k - v_i)^T$$

Here μ_{ik} denotes the membership of point x_k in fuzzy set i, and the power w is again the fuzziness index as in the FCM case. Note that in 3.30 we now do not project $(x_k - v_i)$ to one axis only. Instead we project it to q axes. This produces inertia ellipses in two dimensions, ellipsoids in three dimensions and elliptotypes in m dimensions. Thus the value q selected for the summation can at most be m. The iterative fuzzy c-lines algorithm can now be generalized from the FCM as follows:

1. Given an initial set of membership values μ_{ik} for all points k in the clusters i, compute an iterated approximation to the cluster centres, using:

$$v_i = \frac{\sum_{k=1}^{n} \left(\mu_{ik}^{(0)}\right)^w \cdot x_k}{\sum_{k=1}^{n} \left(\mu_{ik}^{(0)}\right)^w}$$

2. Next, find the largest eigen-value (or eigen-values) of matrix J_i and the corresponding eigen-vector(s) u_{ij} for each cluster, and compute the distances d_{ik}.

3. Update the assumed membership values for the next iteration, using:

$$\mu_{ik}^{(t+1)} = \frac{1}{\sum_{j=1}^{c} \left(\frac{d_{ik}^{(t)}}{d_{jk}^{(t)}}\right)^{\frac{2}{w-1}}}$$

4. We continue iterating until there is almost no more change in the membership values between successive iterations. A criterion for stopping the iterations can be, for example:

$$\max_{i,k} \|\mu_{ik}^{(t+1)} - \mu_{ik}^{(t)}\| \leq \varepsilon \text{ for some } \varepsilon > 0$$

The authors of [64] use a combination of the above methods to search for clusters in a set of synthetic 2 dimensional data points. Each cluster identifies a specific range of values of the variables of the system which are associated with it. Hence each cluster becomes a local model. Also, the model is direction free in the sense that no distinction is made regarding the character of the variables. After clustering one is free to decide on which variables are to be treated as input and which as output variables. Suppose now we have a point $x \in \mathbb{R}^m$ and we have decided that the input variables will be the first m_1 coordinates of x and that the remaining ones are output variables. We determine an output for the input $x' = (x_1, x_2, ..., x_{m_1})$ as follows.

1. We calculate the distance from the point x' to each cluster. Here the distances will be calculated in $\mathbb{R}^{m_1}$. Let δ_i be the distance from x' to cluster i $\forall\, i : 1 \leq i \leq c$.

2. The smallest distance corresponds to the best local linear model for this point:

$$\delta_{i_0} = \min_{1 \leq i \leq c} \delta_i \tag{3.31}$$

$$\overline{\delta}_{i_0} = \frac{\delta_{i_0}}{m_1} \tag{3.32}$$

The value of 3.32 is now used to determine a subset of the output space where an output is found for the input x'. Let $\gamma_{i_0}(y)$ now represent the distance from the point $y \in \mathbb{R}^{m_2}$ to the centre of the identified cluster (measured in $\mathbb{R}^{m_2}$). We choose the output from the set:

$$O = \{y \in \mathbb{R}^{m_2} : \gamma_{i_0}(y) \leq m_2 \bar{\delta}_{i_0}\}$$

Unfortunately the authors do not discuss how to reach a single output. Maybe one first determines the set O and then pick a point y_0 that has the smallest distance to the identified local model. They test their approach on the above mentioned synthetic data set. The data is non-linearly (approximately linearly) spread across the plane. The procedure outlined above is then applied to the data and three fuzzy clusters are identified that approximates the non-linear function quite closely.

The authors of [13] developed an algorithm that identifies the parameters for a simplified Takagi - Sugeno controller. Similar to [64] their approach combines FCM clustering with the fuzzy c-lines clustering. They develop four objective functions which are used to optimize both the premise and consequence parameters for the simplified model of the Takagi - Sugeno controller.

Chapter Conclusion

Chapter 3 then considered three of the main directions that have in recent years emerged as methods of designing fuzzy controllers. The area of fuzzy relational equations is of course much wider than the above discussion indicates. The operators were chosen because of their application in controller design. Similarly, fuzzy clustering has developed in various directions. For example measures of cluster validity have not been discussed. The work on fuzzy equivalence relations is too mathematically pleasing to have been omitted. The section on Wang and Mendel's method is probably easiest to directly apply.

Chapter 4

The use of Genetic Algorithms

Introduction

Genetic Algorithms are computer programs that search a space of solutions to a problem in order to produce optimal or near optimal solutions. The set of solutions on which the G.A. acts at time t is called the population or generation at time t, denoted $G(t)$. The solutions are usually coded in as strings of bits although other coding schemes are also fairly common. Often the parameters to be optimized are not coded and the actual real or integer values of the parameters are used in the string representation. Gray codes have also been used. One such coded solution is called a chromosome while a substring is called a gene and a single bit has been termed an allele in G.A. terminology.

The strings in the population are subjected to genetic operators called selection, crossover and mutation in an attempt to produce strings that represent increasingly better solutions to the problem considered.

G.A.'s have been used to solve a wide range of different kinds of problems. In the last decade or so they have become more and more important as means for designing and optimizing fuzzy controllers. They are interesting in this respect since they are successful in producing good controllers and represent probably one of the first systematic tools for the design of fuzzy controllers.

This chapter focuses on the different ways in which G.A.'s have been used to design and optimize fuzzy controllers. We present the material that is required by someone who wishes to optimize an existing controller. Theoretical aspects of G.A.'s are hence not of importance to us.

Applying G.A.'s

Three main directions have emerged here namely:

1. G.A.'s have been used to tune the scaling factors of the input and output variables of a controller.

2. G.A.'s have been used to produce high performance membership functions, and,

3. G.A.'s have been used to produce optimal rule bases.

Combinations of the above are common as we shall see in the following.
G.A.'s differ from one another in terms of:

1. The way in which the solution to the problem is coded.
2. The specific forms of the reproduction operators, i.e. selection, crossover and mutation.
3. The function used to evaluate how good a solution to the problem a specific string represents, called the fitness function.

We will start the discussion by looking at the most commonly used reproduction operators.

Genetic Operators

Selection

As stated above each string has an associated fitness value which is an indication of its relative worth as a solution to the problem. Selection is the process by which copies of high fitness strings are selected from the population at time t, $P(t)$ and promoted to $P(t+1)$. The number of copies of a string promoted to $P(t+1)$ is usually proportional to the fitness of the string.

If f_i is the fitness of string i in the present population then the probability of selecting string i is often given by:

$$p_{select_i} = \frac{f_i}{\sum f_j}$$

where $\sum f_j$ is the sum of fitness values for all strings in $P(t)$.

Some schemes automatically promote the string with highest fitness value to the next generation. Such schemes use so-called elitist selection procedures.

The newly selected strings form the mating pool and await the application of the other two genetic operators.

Crossover

This process allows two strings to swop information. Simple crossover proceeds in three steps:

1. two strings are randomly selected from the mating pool.
2. a position along the length of the strings is selected uniformly at random.
3. the substrings following the crossing site are swopped.

A simple example illustrates this:
A = 10100 101110 B = 00101 100011 (before crossover)
A'= 10100 100011 B'= 00101 101110 (after crossover)
Here it can be seen that position 5 was chosen as the crossing site. More complex crossover oprators have been employed and will be discussed later.

Mutation

Mutation randomly, with a given probability alters the value of a bit. In the case of binary coding a zero becomes a one and vice versa. In the case of integer coding the value of the integer increases or decreases by one.

Mutation is an attempt to prevent the permanent loss of any single bit. The reasoning is as follows: Suppose that in some generation a particular bit is absent in all strings. For example all strings may have a zero in position 5 while a one in position 5 may be crucial to a good solution. In such a case the other operators will never produce the required bit in successive generations.

Techniques for coding information

When a G.A. is applied to a controller design problem the first step requires us to code the parameters to be optimized in strings so that the genetic operators can be applied to the strings. A string could represent a fuzzy set on a particular domain or the collection of rules in a rule base, or even the whole data base of the fuzzy controller. The following examples illustrate different strategies that have been used by researches.

Methods for coding rule bases

The simplest method for coding a rule base as a bit string is to let each bit position represent a rule. A 0 switches a rule off and a 1 switches it on. Genetic operators are applied to these strings to produce optimal rule bases. This method was used in [42] Notice that here the output membership functions of each rule stay constant and are not affected by the learning process.

The output fuzzy sets of rules can also be included in the rule base coding and thus be subject to change during the learning process. In this regard an often used technique is to number the fuzzy sets on the output space. For example, suppose we have the rule table below:

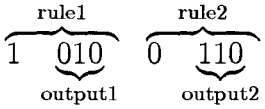
This rule base can now be coded as the string (1, 2, 4, 6, 0, 3, 1, 6, 3). The mutation operator applied to this string will increase or decrease an integer by one or set it to zero. One could also have a combination of the above two approaches as shown in the next example.

Example 31 *In [50] the output fuzzy sets are numbered from 0 to 6, i.e. NB = 0,*

	A_{11}	A_{12}	A_{13}
A_{21}	B_1	B_2	B_4
A_{22}	B_6	–	B_3
A_{23}	B_1	B_6	B_3

Table 4.1: Example of coded rule table

NM = 1, ..., PB = 6. The FLC parameters are represented in a string consisting of 3 substrings. Substring 1 codes the rules in the rule base. Each rule is coded using 4 bits. The first bit switches a rule on or off. The three bits following it gives the binary code of the number of the output fuzzy set occurring in that rule. Thus substring 1 consists of a collection of strings each of which codes a rule as shown below:



Here the codes for rules 1 and 2 are shown. Rule 1 is switched on and its output fuzzy set is the fuzzy set with binary code 010 i.e. NS. Similarly, Rule 2 is switched off and the binary code of its output fuzzy set is 110 or PM.

Example 32 *In [22] the rule base of a controller is optimized. A rule base is represented as $C_r = C_{r1}C_{r2} \cdots C_{rm}$. For each $i : 1 \leq i \leq m$, C_{ri} represents a rule of the rule base r. C_{ri} is a substring coding the parameters of the membership functions occurring in rule i. Each membership function is assumed to be trapezoidal, hence needs four parameters to code it. Hence:*

$$C_{ri} = (c_{i1}, a_{i1}, b_{i1}, d_{i1}, ..., c_{in}, a_{in}, b_{in}, d_{in}, c_i, a_i, b_i, d_i)$$

codes a rule in an n input - 1 output system. The substring $c_{i1}, a_{i1}, b_{i1}, d_{i1}$ codes the parameters for the fuzzy set on the first domain and so on.

Example 33 *In [31] the designers work with a 2 input - 1 output system. The crossover operator is applied directly to the rule base in the form of a rule table. The operator is called a "point - radius" operator. It's operation is illustrated below:*

One can see here that the cells with capital letter entries have been swopped. In this case the point is the address (ze,ns) and the radius is 1. If one is working with a system with more than 2 inputs the rule base can not be represented as a rule table. In such a case one would need an extension of the method illustrated above. The extension for a 3 input - 1 output system would look like this:

	NM	NS	ZE	PS	PM
NM	ps	pm	ps	ze	nm
NS	nm	PS	ns	ps	pm
ZE	PM	PS	ZE	pm	nm
PS	nm	PM	ps	nm	ns
PM	ps	pm	nm	ns	ns

Table 4.2: Rule Base 1 before crossover

	NM	NS	ZE	PS	PM
NM	pm	ps	pm	nm	nm
NS	ns	NS	nm	ns	pm
ZE	PS	NM	ZE	ps	ns
PS	ns	PM	nm	ns	nm
PM	pm	nm	ns	ps	ns

Table 4.3: Rule Base 2 before crossover

	NM	NS	ZE	PS	PM
NM	ps	pm	ps	ze	nm
NS	nm	NS	ns	ps	pm
ZE	PS	NM	ZE	pm	nm
PS	nm	PM	ps	nm	ns
PM	ps	pm	nm	ns	ns

Table 4.4: Rule Base 1 after crossover

	NM	NS	ZE	PS	PM
NM	pm	ps	pm	nm	nm
NS	ns	PS	nm	ns	pm
ZE	PM	PS	ZE	ps	ns
PS	ns	PM	nm	ns	nm
PM	pm	nm	ns	ps	ns

Table 4.5: Rule Base 2 after crossover

Suppose that the input variables are labeled x_i for $1 \leq i \leq 3$. Also, let the number of linguistic terms on domain i be n_i. Instead of using letters like ps in the address of a cell in the rule base, let us number the linguistic terms on domain i as $1, 2, ..., n_i$. So a vector like (a_1, a_2, a_3) with $a_i \in \{1; ...; n_i\}$ for $i = 1, 2, 3$ gives the address of a cell in the rule base that is filled with the output fuzzy set of the particular rule. Suppose two rule bases are to be crossed over and the point is (a_1, a_2, a_3) and the radius is n_0.

Then we identify in the two rule bases the cells with addresses :

$(a_1, a_2, a_3 + 1), (a_1, a_2, a_3 + 2), ..., (a_1, a_2, a_3 + n_0), (a_1, a_2, a_3 - 1), (a_1, a_2, a_3 - 2), ..., (a_1, a_2, a_3 - n_0)$;

$(a_1, a_2 + 1, a_3), (a_1, a_2 + 2, a_3), ..., (a_1, a_2 + n_0, a_3), (a_1, a_2 - 1, a_3), ..., (a_1, a_2 - n_0, a_3)$;

and a similar expression with the first entry changing and the other two staying constant. These cells in the two rule bases are then swopped.

Example 34 *Tan and Hu in [56] design a rule base and membership functions for a controller that balances an inverted pendulum.The chromosome string to which the G.A. is applied consists of four substrings. The first substring codes the membership functions for the first input variable, the second substring codes the membership functions for the second input variable. The third substring codes the rule base and the final substring codes the output membership functions. Each membership function is coded as a 24 bit string consisting of three 8 - bit substrings, one substring for each of the left base, centre and right base of a triangular fuzzy set.*

The rule base was coded as follows. Each domain (input and output) has 8 linguistic terms defined on it. The membership functions on the output domain are numbered from 0 to 7 as in example 2. Each rule is assigned a position in the rule string. This position is filled with a 4 - bit code that gives the number of the output fuzzy set appearing in that rule.

Thus the rule base requires 32 bytes while each of the other substrings require 24 bytes, giving a total of 104 bytes to represent the system.

Coding of fuzzy sets

Example 35 *This example extends the work described in example 5 by Kinzel et al. In order to code a collection of fuzzy sets the authors represent a domain by a string of genes. Each gene lists the values of a fuzzy set at a particular point of the domain. Suppose domain X_i has left boundary b_l and right boundary b_r. Suppose also that the number of fuzzy sets on X_i is n_i. Then the fuzzy partition on X_i is coded as:*

$$\underbrace{\begin{pmatrix} \mu_{i1}(b_l) \\ \mu_{i2}(b_l) \\ \cdot \\ \cdot \\ \cdot \\ \mu_{in_i}(b_l) \end{pmatrix} \cdots \begin{pmatrix} \mu_{i1}(b_r) \\ \mu_{i2}(b_r) \\ \cdot \\ \cdot \\ \cdot \\ \mu_{in_i}(b_r) \end{pmatrix}}_{\text{coding of domain } X_i}$$

A two point crossover operator is used which exchanges the ranges of two given chromosomes. The simple example below illustrates this:

$\begin{pmatrix} 0 \\ 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ crossed over with
$\begin{pmatrix} 0 \\ 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \end{pmatrix}$

yields

$\begin{pmatrix} 0 \\ 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ and
$\begin{pmatrix} 0 \\ 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \end{pmatrix}$

where it can be seen that the fourth vectors in the strings were swopped. This crossover operation on the strings can be seen as:

Example 36 *In a number of papers Karr discusses the use of G.A.'s to determine the optimal fuzzy sets to be used with a fixed rule base. The fuzzy sets are either triangular or trapezoidal. A trapezoidal fuzzy set is represented by coding the four parameters on its domain that define it, while a triangular fuzzy set needs three parameters to specify it. Karr uses a 7 bit code for each parameter. For each parameter the user defines a minimum and maximum value, P_{min} and P_{max}. In general if an m-bit string is used for the coding and b is the integer value of the binary code for P then b is determined via the equation:*

$$P = P_{\min} + \frac{b}{2^m - 1}(P_{\max} - P_{\min})$$

The coded parameters for each fuzzy set are simply concatenated into a single string. These strings are again concatenated to represent the whole collection of input and output fuzzy sets. This is a fairly common technique and Karr calls it a concatenated, mapped, unsigned, binary coding.

Example 37 *In this example the G.A.is applied to produce an optimal Data Base. The Data Base contains the parameters of the FLC, normalization limits of the variables and the definition of membership functions.*

1. The collection of FLC parameters are:

$$\{Parameters\} = \{N; M; n = (n_1, n_2, ..., n_N); m = (m_1, m_2, ..., m_M)\}$$

where N = number of input variables; M = number of output variables;
n_i = number of fuzzy sets on domain i $\forall i : 1 \leq i \leq N$ and
m_j = number of fuzzy sets on domain j $\forall j : 1 \leq j \leq M$.

2. The normalization limits of input and output variables are collected together in an (N + M) × 2 matrix

3. Each membership function is coded as a trapezoidal fuzzy set, hence 4 parameters define it. The collection of all fuzzy sets is coded as an L × 4 matrix, (where the total number of fuzzy sets on input and output domains is L) with each row giving the parameters that define a fuzzy set.

Now let L_a = sum of linguistic terms on all input domains and
L_c = sum of linguistic terms on all output domains
Then

$$\begin{aligned} L_a &= \sum_{i=1}^{N} n_i \text{ and} \\ L_c &= \sum_{j=1}^{M} m_j \text{ and} \\ L &= L_a + L_c \end{aligned}$$

A rule string consists of two substrings, one of length L_a and one of length L_c. Hence each linguistic term on each domain is assigned one bit position. If a linuistic term occurs in a rule then that linguistic term's bit is 1, if not it is 0.

To illustrate the above suppose we have a system with the following parameters

$$N = 3; M = 1; n = (3, 4, 3); m = (5)$$

Then the rule

$$if\ x_1 \text{ is } A_{12} \text{ and } x_2 \text{ is } A_{24} \text{ and } x_3 \text{ is } A_{31} \text{ then } y \text{ is } B_4$$

is coded as:

010 0001 100 - 00010

Other Genetic Operators

Most of the G.A.'s used in the reports use simple crossover and mutation operators. The following examples look at novel operators that have been defined for learning rule bases and membership functions.

Example 38 *For this example refer back to example 4 above. As stated there a rule is represented by a string:*

$$C_{ri} = (c_{i1}, a_{i1}, b_{i1}, d_{i1}, ..., c_{in}, a_{in}, b_{in}, d_{in}, c_i^{'}, a_i^{'}, b_i^{'}, d_i^{'})$$

So each string C_{ri} codes a rule and has $(n+1) \times 4$ entries. For the discussions to follow we relabel the parameters in the string as:

$$\begin{aligned} C_{ri} &= (c_1, c_2, ..., c_H) \\ \text{where } H &= (n+1) \times 4 \end{aligned}$$

For c_h a gene in C_{ri}, i.e. $\forall h \in \{1, 2, ..., H\}$ we define an interval $[c_h^l,\ c_h^r]$, in which we adjust the value of c_h by means of the genetic operators.

If $t \equiv 1 (\mathrm{mod}\, 4)$, i.e. if $4 \mid t-1$ then c_t is the first in a list of four real values that codes a fuzzy set, namely c_t, c_{t+1}, c_{t+2}, c_{t+3}. The intervals of performance of the above real parameters are:

$$\begin{aligned} c_t &\in [c_t^l,\ c_t^r] = [c_t - \frac{c_{t+1} - c_t}{2},\ c_t + \frac{c_{t+1} - c_t}{2}] \\ c_{t+1} &\in [c_{t+1}^l,\ c_{t+1}^r] = [c_{t+1} - \frac{c_{t+1} - c_t}{2},\ c_{t+1} + \frac{c_{t+2} - c_{t+1}}{2}] \\ c_{t+2} &\in [c_{t+2}^l,\ c_{t+2}^r] = [c_{t+2} - \frac{c_{t+2} - c_{t+1}}{2},\ c_{t+2} + \frac{c_{t+3} - c_{t+2}}{2}] \\ c_{t+3} &\in [c_{t+3}^l,\ c_{t+3}^r] = [c_{t+3} - \frac{c_{t+3} - c_{t+2}}{2},\ c_{t+3} + \frac{c_{t+3} - c_{t+2}}{2}] \end{aligned}$$

Michalewicz defines a mutation operator that uses a function:

$$\Delta : \mathbb{R}^2 \to \mathbb{R}$$

that has the properties:

- $\Delta(t,\ y) \in [0,\ y]$
- the probability that $\Delta(t,\ y)$ is close to zero increases as t insreases.
- the operator does a uniform search in the initial space when t is small and when t is larger does a local search.

If $C_v^t = (c_1, ..., c_k, ..., c_H)$ is a chromosome and c_k was selected for mutation then the result of the mutation is a vector:

$$C_v^{t+1} = (c_1, ..., c_k^{'}, ..., c_H)$$

with

$$k \in \{1, ..., H\} \text{ and}$$
$$c_k^{'} = \left\{ \begin{array}{l} c_k + \Delta(t,\ c_k^r - c_k) \text{ if a randomly chosen digit is 0} \\ c_k - \Delta(t,\ c_k - c_k^l) \text{ if a randomly chosen digit is 1} \end{array} \right\}$$

This kind of mutation operator has been termed soft mutation, i.e. soft mutation changes the shape of the fuzzy set. On the other hand hard mutation changes the variable occurring in the rule, for example a part of a rule ... x_2 is A... might change to ...x_3 is A... when hard mutation is applied to the rule.

In the same work [22] an interessting crossover operator, called max-min arithmatical crossover is also defined. It is used in addition to the simple crossover commonly used and described earlier. Max-min arithmatical crossover works as follows.

Suppose C_v^t and C_w^t are two chromosomes to be crossed. Four chromosomes are now generated, namely:

$$C_1^{t+1} = aC_w^t + (1-a)C_v^t$$
$$C_2^{t+1} = aC_v^t + (1-a)C_w^t$$
$$C_3^{t+1} \text{ which has } c_{3k}^{t+1} = \min\{c_k, c_k^{'}\}$$
$$C_4^{t+1} \text{ which has } c_{4k}^{t+1} = \max\{c_k, c_k^{'}\}$$

Of these two offspring the two with highest fitness function values are chosen to be promoted to the next generation. The parameter a can be fixed or made to depend on t.

Fitness Functions

The fitness functions used are generally not explicitly discussed in the literature. The following examples show the forms of some of the fitness functions used in the above-mentioned examples.

Consider again the work done by Karr in example 7 above. The example describes how fuzzy sets are coded to be optimized for a liquid level system. The system consists of a cylindrical vessel into which and out of which liquid flows at rates Q_{in} and Q_{out} respectively. The rates Q_{in} and Q_{out} are controlled by the fuzzy controller. The aim of the fuzzy controller is to drive the height of the liquid in the vessel to some specified setpoint as soon as possible.

To determine the fitness of a string, the coded fuzzy sets are used to run the controller. The squared error between the liquid level height and the setpoint is

then calculated for the first 20 seconds of controller runtime. These measurements are then added. This is done for four different starting heights of the liquid level. The sums of squared errors for the four cases are then added. Thus the equation used is:

$$f = \sum_{i=case1}^{case4} \left[\sum_{j=0s}^{20s} (25 - h_{ij})^2 \right]$$

Karr also designs a system that controls the titration of an acid-base solution. The aim here is to get the pH of the solution to the value 7 (neutral) as soon as possible by controlling the inflow rates of acid and base into the solution mixture. The fitness function for this GA is:

$$f = \sum_{i=1}^{2} \left[\sum_{j=0s}^{50s} (pH_{ij} - 7)^2 \right]$$

It is clear that fitness functions will be application dependent. However, general types of fitness functions exist. For example the above two are examples of a type where some parameter has to achieve some special value, c. Another example of this type is that discussed by Kinzel et al in example . In that work the controller was designed to balance an inverted pendulum. Let $\varphi(t)$ be the angle the pendulum makes with the vertical, while $\dot{\varphi}$ represents the angular velocity of the pendulum. If $x(t) = (\varphi\ (t), \dot{\varphi}\ (t))$ and $c = (0, 0)$ then:

$$f = \sum_{t=start}^{\max time} t(x(t) - c)^2$$

and is clearly similar in form to the previous two examples.

Tan et al in [56] also design a fuzzy controller for the inverted pendulum problem. Instead of the fitness function used by Kinzel et al they used the following function:

For this system there are three possible outcomes namely, either the pendulum falls or the time necessary to balance it expires or the pendulum balances (using some ϵ-criterion). The fitness function is split up into these three cases as follows:

$$fitness = \left\{ \begin{array}{c} f_1(t) \text{ if the pole balanced} \\ reward \text{ if time expires} \\ f_2(t) \text{ if pole fell over} \end{array} \right\}$$

where the functions $f_1(t)$ and $f_2(t)$ and *reward* are illustrated in Figure 4.1 below:

One can see from this fitness function that the longer a solution string keeps the pendulum from falling over the higher it's fitness is. Also, the longer a string

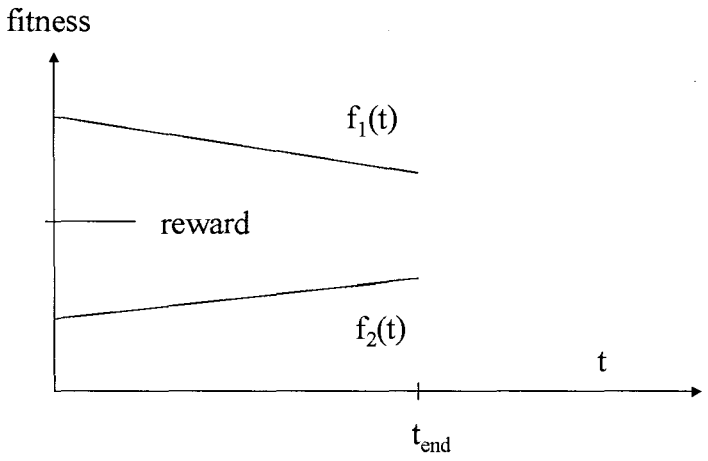


Figure 4.1: Fitness function for the pendulum example

takes to balance the pendulum the smaller it's fitness value is. This is as one would expect as the controller is supposed to balance the pendulum in minimum time.

In [42] a rule base is optimized. In order to keep the rule base as small as possible each string is penalized for the number of rules that are active in it.

Some implementational details

G.A.'s running in parallel

Pham and Karaboga in [48] describe their implementation of a Genetic fuzzy system. They use a G.A. to optimize the relation matrix for a fuzzy controller. The scheme consists of starting with a number, n say, of initial populations of strings, each of which is generated by a random number generator. The initial populations have G.A.'s applied to them which are executed in parallel. After a fixed number, max_1 of iterations of the G.A. the first phase of the learning process is completed. A second phase then starts. This consists of choosing the fittest $\frac{1}{n}$ strings of each solution set and collecting them together into a new initial population. The G.A. is then applied to this new initial population, this time for a number, max_2 of iterations. After that the string with highest fitness value is chosen as the desired fuzzy relation matrix.

Gray codes

Traditional binary coding has the drawback that in order to change a decimal number by one often requires a change of more than one digit in the binary form

Decimal Number	Binary Code	Gray Code
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Table 4.6: Three bit Gray code

of the number. For example changing from 3 to 4 translates to changing from 011 to 100. This creates the problem that a string that is close to an optimum solution cannot move closer to that optimum by mutation. Gray codes have been used to solve this problem. The Gray codes for three bit strings are shown below:

As can be seen changing a number by one is always a one bit change in a Gray coding scheme as is required.

Chapter Conclusion

This chapter is a summary of some of the different methods developed for applying G.A.'s to the tasks of designing and optimizing fuzzy controllers that have been investigated in the literature. As stated above G.A.'s represent effective systematic tools for optimizing fuzzy controllers and as such are indispensible tools for the developer of a Fuzzy Logic Controller. We tried to concentrate on the practical aspects of this area in order to make application in fuzzy controller design straightforward. Of course hundreds of papers have been written on the topic and many other techniques have been developed that we have not touched upon.

Chapter 5

Application

Introduction

In this chapter we look at some applications of Fuzzy Control in Telecommunication systems. The first section considers some of the reported work in the literature. The second section considers one of the methods in congestion control commonly used in Intelligent Networks and fuzzify the algorithm. Afterwards we look at the results of the fuzzy method of controlling congestion and compare it with results for the crisp version of the algorithm.

Previous Work

ATM is an emerging set of standards and protocols that have been designed in order to support a range of different types of communication traffic on a single network. The different types of traffic are to include voice, video and data traffic together with a range of future possible types.

The different traffic types differ in terms of traffic characteristics like sensitivity to delay, burstiness, and average holding time. Various models and assumptions are used in designing networks supporting conventional telephone traffic. These include methods based on Poisson arrival processes, Bernoulli assumptions, packet train models and fluid flow models [10].

However, because of the bursty and non-linear nature of the heterogeneous traffic envisaged for the ATM network, these assumptions no longer lead to an accurate analysis of the network.

Video traffic is particularly bursty. This means that the arrival of packets at the destination machine often occurs in large bursts with relatively few packets arriving at other times. Hence it becomes important for the destination machine to predict how the traffic arrives. This is done in an attempt to prepare the destination machine to prevent cells being lost due to buffer overflow.

The authors of [51] designed a fuzzy mechanism that predicts the amount of video traffic to arrive at discrete points in time.

The fuzzy learning algorithm used is based on building a fuzzy relation using adaptive clustering. The relation is used to estimate the possible system response to a new and unknown set of system inputs.

The delayed values of ATM traffic, $y(k-1)$ and $y(k-x)$ are used as inputs to the model (with x defined as the delay between the first and second observation).

After a learning phase, the fuzzy rules are used with the observed values, to predict the future value of traffic, $y(k)$, i.e. one observation point in advance. An indicator of prediction accuracy is the mean squared error between the actual traffic and predicted traffic as defined below:

$$MSE = \frac{1}{N}\sum_{k=1}^{N}(y_n(k) - y_{n_{est}}(k))^2$$

where:

$y_n(k)$ is the normalized value of observed traffic intensity

$y_{n_{est}}(k)$ is the normalized value of estimated traffic and N is the number of observations.

The fuzzy prediction scheme was compared with conventional methods and showed satisfactory results.

As stated above, ATM networks are designed to handle a range of traffic types. When a number of such bursty traffic sources add cells to the network the network will inevitably be subject to congestion. Traditional approaches to congestion management include admission control algorithms, smoothing functions and the use of finite sized buffers with queue management techniques.

In admission control, upon arrival of a new call or message, the network predicts the performance degradation that may result, based on current network traffic and the traffic characteristics of the new call. It accepts the call only if the desired performance requirements are met. The admission control policy accepts or rejects an entire call or message as opposed to individual cells of the call.

The traffic smoothing function reduces congestion by buffering incoming cells and injecting them into the network at a slower speed.

In the case of finite sized buffers with queue management techniques predefined buffer content thresholds are used to guide the discard of cells and to adapt the service process to the occupancy of the buffer. Some queue management techniques use two threshold values, L_1 and L_2 say, where $L_1 < L_2$. The buffer is 'full' as soon as the buffer occupancy exceeds L_2. All incoming cells and messages are then blocked until the buffer occupancy becomes less than L_1.

Whether the queue management technique uses a single or a double threshold value, it partitions the buffer into two states, admit or block. The choice of this crisp cut-off value is clearly critical. For a low value, i.e. cells are only accepted if the buffer occupancy is very low, 25% say, buffer utilization may be very poor. The buffer may be basically empty and still block incoming calls. On the other hand a high threshold value may lead to problems in the case of bursty traffic.

For the above reasons, the authors of [2] decided to experiment with the use of fuzzy thresholds. The controller has the occupancy level (as % of total buffer capacity) as a universe of discourse. Two fuzzy sets are defined, one called "Degree of Blocking" and the second called "Degree of Admittance". These are sigmoidal

and inverses of each other. Degree of Blocking is zero at 20% occupancy and rises to flatten out (at a value of 1) at around 80% occupancy.

The fuzzy thresholding scheme differs from fixed thresholding schemes in that it blocks a fraction of incoming cells and not necessarily all incoming cells from a new connection. The fraction of cells blocked is determined by the value of the fuzzy set "Degree of Blocking" at the occupancy level of the buffer at the time of arrival of the call.

The authors report on simulations that were run to compare the performance of the fuzzy thresholding scheme with that of crisp cell blocking methods. The results indicate that the fuzzy version adapts well to sharp changes in cell arrival rates and maximum burstiness of bursty traffic sources, yielding lower cell discard rates, high throughput of cells and lower cell blocking rates.

Fuzzy control has also been applied in the area of traffic routing in telecommunications networks.

In most networks traffic is routed under fixed rules. In order to cater for periods of peak traffic fixed routing schemes have to provide over-capacity. Such networks have often been shown to be unable to accomodate demands with the required grade of service.

Adaptive routing methods have also been developed. These depend on the availability of network resources. One such routing scheme is implemented by AT&T and is called Real-Time Network Routing (RTNR). For each origin-destination node pair, it considers the number of idle circuits between the node pair and determines a level of availability of the node-pair link. The routing then depends on the availability levels of possible routes together with the class of service of the call.

The authors of [12] have fuzzified the RTNR technique. Fuzzy adaptive routing of telephone traffic uses a set of routing rules that order and select paths from an origin switch to a destination switch. The paths always have only two "legs" at most, i.e. are composed of two chained circuit groups for a stream going from an origin node to a destination node.

It determines, for each origin-destination node pair, the availability of all paths and the quality of all routes, and decides on the best route for routing the current traffic.

The fuzzy controller consists of two inference engines. The first rule base uses the number of idle cicuits in a circuit group and determines a fuzzy set describing the availability of that circuit group. This fuzzy output from the first inference engine is fed to the second inference engine. Using pairs of fuzzy values of individual circuit groups, the second inference engine combines the two fuzzy sets and produces a fuzzy set describing the availability of that circuit pair. The defuzzification module now uses the fuzzy route quality as obtained from the second inference engine and determines, for each traffic stream, the route quality and the best path is selected for all individual traffic relations.

The fuzzy adaptive routing was simulated in a model of the French long distance telephone network. The results show that although fuzzy adaptive routing is robust and efficient other methods do outperform it slightly in terms of percentage of network loss.

In [11] the authors construct a fuzzy traffic controller for ATM networks. The controller is a fuzzy implementation of the two threshold congestion controller and the equivalent capacity admission control method.

The equivalent capacity admission control method works by assigning to each connection an amount of bandwidth called the equivalent or effective bandwidth. Each connection is treated as if it requires this amount of bandwidth throughout its duration. The actual bandwidth utilized varies between some minimum bitrate and the peak rate of the connection.The concept of effective bandwidth simplifies Call Admission Control (CAC). The CAC mechanism calculates an effective bandwidth for each connection request. This effective bandwidth is added to the sum of the bandwidths of existing connections utilizing the same link(s). If the result is less than the total capacity of the link(s) the connection is allowed. Obviously, if the result exceeds the total capacity of the link(s), the connection cannot be allowed.

The ATM traffic controller is based on the following model of an ATM network. Input traffic is categorized into two types, real-time (type 1) and non real-time (type 2). Examples of real-time traffic include video and voice, and data is an example of non real-time traffic. The ATM traffic for the two traffic types are first stored in separate pre-buffers in the Customer Premises Equipment(CPE). Let the size of the pre-buffer for type i traffic be K_i $(i = 1, 2)$. As for transmission capacity, a portion, C_r of capacity is reserved for type 1 traffic and the remaining $1 - C_r$ is reserved for type 2 traffic. If either traffic type does not use the total capacity reserved for it, the unused portion can be used for the other type.

The various modules and their operation can be described as follows. The Performance Measures Estimator measures the system performance variables, namely:

$$\begin{aligned} q &= \text{queue length} \\ \triangle q &= \text{queue length change rate} \\ p_l &= \text{cell loss probability} \end{aligned}$$

These variables are defined and measured for each traffic type separately and their values are fed to the Fuzzy Congestion Controller which produces an output, y. A positive value of y indicates that the system is relatively free from congestion and a negative value that the system is congested. The value of y is used to modify the rate at which cells of the two traffic types are transmitted to avoid or relieve congestion.

The Fuzzy Bandwidth Predictor predicts the equivalent capacity C_e for a new call from the traffic parameters specified in the traffic contract.

The Network Resource Estimator is responsible for the accounting of system resources. For every new call accepted, the call's equivalent capacity, C_e is subtracted from C_a. Conversely, for every connection that is released, the connection's equivalent capacity is added to C_a.

The results of simulations run by the authors indicates that the fuzzy admission controller improves system utilization by 11% while the performance of the fuzzy congestion controller is 4% better than the conventional two-threshold congestion method.

Another application of Fuzzy Control in ATM networks is in developing policing mechanisms for this type of network.

When a user requests a new connection the Network Management System checks the available resources to determine if the requirements of the connection can be met. If enough resources are available, a decision is made to allow the connection. This process is called Connection Acceptance Control (CAC). The Network Management System will also reserve the necessary resources for the accepted call. The new call will only be accepted if the Quality of Service of the existing as well as the new call can be guaranteed.

Another network management function, Usage Parameter Control (UPC), is required to ensure that each source conforms to its negotiated parameters.

A major problem in defining an efficient policing mechanism is identifying the traffic parameters that best characterize the behaviour of the source. The difficulty comes from the fact that different sources have different statistical properties as they range across different services. Also, one needs to define parameters that can be monitored during the call. The two traffic parameters that are enforced by UPC are the Peak Cell Rate (PCR) and the Mean Cell Rate (MCR). Enforcing the PCR is not difficult. However, enforcing the MCR is problematic, since short term statistical fluctuations are allowed as long as the source respects the average value negotiated, λ_n in the long term.

Most of the control mechanisms are window based. In these mechanisms a constant upper bound is set on the number of cells that can be accepted in a fixed time interval, T. This upper bound is called the window. Examples of window based mechanisms are the Jumping Window and the Exponentially Weighted Moving Average [10]. Both of these schemes seem to be unable to cope efficiently with the conflicting requirements of an ideal policer - that is, a low false alarm probability and high responsiveness. A false alarm can be explained as follows. The policing mechanism should police the average rate, i.e. a source is allowed to exceed it's negotiated rate parameter at times, as long as the average rate is respected. Suppose now that a particular source exceeds its negotiated rate for a period of time. Suppose the policing mechanism cuts off the source while the excessive traffic from the source did not affect the Quality of Service of any other connection. We refer to this as a false alarm.

The authors of [10] constructed a fuzzy policer and compared its performance

with that of the conventional policing mechanisms. As stated above the target of the fuzzy policer is to make a generic source respect its negotiated MCR, λ_n. The inputs to the fuzzy controller used are: the average number of cell arrivals per window since the start of the connection, A_{oi}, and the number of cell arrivals in the last window, A_i. The first gives an indication of the long term behaviour of the source while the second indicates current behaviour. A third parameter, the value of N_i in the current window is used to indicate the degree of tolerance the mechanism has over the source. Thus, the control mechanism grants credit to a source that in the past has respected the parameter negotiated by increasing its value for N_i in the current window up to a maximum possible value for N_i, provided that it continues with the non-violating behaviour. Conversely, every time a source violates its negotiated parameter N_i for the source will be decreased. The output of the fuzzy controller is $\triangle N_{i+1}$, the change to be made to the threshold N_i in the next window.

The input domains of discourse are partitioned by three fuzzy sets each, Low, Medium and High, while seven fuzzy sets partitions the output domain of discourse, namely ranging from Negative Big through Zero to Positive Big.

The performance of the fuzzy mechanism has been evaluated through several simulations and compared with some of the more popular policing systems like the EWMA. The results indicate that the performance of the fuzzy policer is much better than that of the conventional policing systems.

Intelligent Networks

The Intelligent Network was invented by Bellcore Labs during the 1970's. The fundamental principle underlying the IN is the separation of switching and control functions in the network. The aim of this separation is facilitating the creation of new services and minimizing the time taken for the development and deployment of such new services.

The two entities of an IN that will concern us here are the SCP (Service Control Point) and the SSP (Service Switching Point).

The SCP is a centralized non-switching node connected to the switches via the common channel signalling system. It contains service specific software and subscriber data. Examples of SCP based services include user authentication, call number translation and alternative forms of billing. The SSP is a switch with enhancements to the call control functionality of the PSTN switch.

The SCP communicates with the SSP, allowing it to establish or release connections. When an SSP receives an IN service type call it transmits the request to the SCP. The SCP validates the call and performs a number of service related functions like caller authentication. If these functions are successful the SCP will command the SSP to establish a connection to the call party. The SCP also sends a packet back to the requesting SSP to acknowledge receipt of the service request.

A typical service setup scenario will involve five to ten messages between the SSP and SCP. A number of SSP's will be connected to a single SCP. IN congestion control schemes attempt to control the rate at which new service requests are transmitted to the SCP in order to avoid or relieve congestion at the SCP.

The Model

The developed algorithm models the operation of a single SCP connected to four SSP's. The arrival times of packets at the SSP's are randomly generated. No more than one packet arrives at an SSP and each arriving packet represents a request for a service. This service request is enqueued at the SSP request queue. As soon as it is possible to, this first packet in the SSP request queue is processed. The processing speed of the request queue is set at 7ms per packet. After the processing of a packet at the SSP is completed it is sent to the SCP. At the SCP it is again placed in a queue where it awaits processing by the SCP. After being processed by the SCP an acknowledgement packet is sent to the SSP that sent the request. This packet is placed in the SSP acknowledgement queue where it awaits processing. In the SSP the processing of acknowledgements takes precedence over the processing of new requests. Acknowledgement packets are processed at a rate of 3ms per packet.

The processing time of the SCP has two components, a fixed value of 10 ms per packet and a randomly generated component. The random number comes from an exponential distribution with mean equal to one. These two components are added to yield the processing time for each packet. Figure 5.1. below illustrates this simplified network model.

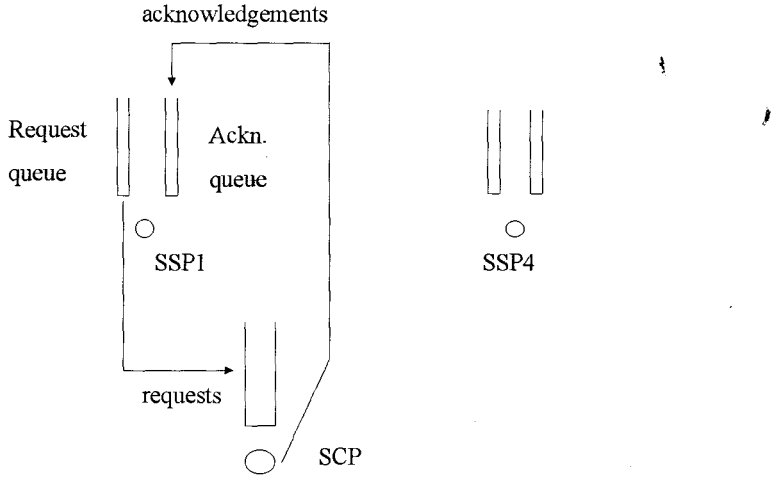


Figure 5.1: Simple IN model used in the simulation

Some IN cogestion control schemes

IN congestion control schemes are usually either rate based or window based schemes. We will focus on the window based schemes.

The Static Window Mechanism

The operation of the static window based scheme is outlined in the following. Each SSP connected to the congestion avoiding SCP is assigned a window size. The window size of an SSP is the maximum number of unacknowledged requests that an SSP is allowed to have at any point in time. That means that if the number of outstanding requests equals the window size the SSP is not allowed to transmit any more requests until some acknowledgements have been received. Hence each SSP has two variables, out, the number of outstanding service requests and win, the window size. On transmission of a service request to the SCP out is incremented by one and on receipt of an acknowledgement from the SCP out is decremented by one. As the name indicates the window size is not changed during the operation of the network.

The Adaptive Window Mechanism

In the adaptive window scheme the window size of an SSP is continually changed. The value of the window size varies between winmin and winmax. An SSP keeps track of any of its requests that are lost or dropped at the SCP due to the SCP queue being full. In addition to the two variables used in the static case the adaptive window scheme uses a counter, c, which varies between cmin and cmax. Every request that is dropped decreases the out variable by one and decreases the win variable by one (unless the window size is already at its minimum value) . Every acknowledgement that is received increases c by one. When c reaches a maximum value, indicating that number of successfully transmitted requests, the window size is increased by one and c is reset to zero (cmin).

The Fuzzy Adaptive Window Mechanism

One can also devise a fuzzy scheme to adapt the window size of an SSP. This was done and its performance compared with those of the above two schemes. The fuzzy controller uses the round trip delay (rtd) of requests as its input and its output is the window size of the SSP. The rtd of a request is the difference between the arrival time of an acknowledgement and the time at which the request was transmitted to the SCP.

The System, the variables and the operation of the algorithms

The simplified IN is represented by the following system variables

1. The number of packets in each of the request and acknowledgement queues for each SSP.,

$$qa = \text{number of packets in ack. queue}$$
$$qr = \text{number of packets in req. queue}$$

2. The number of packets in the SCP queue, represented by *Qlength.*

3. Q is a vector representing the SCP queue. An i (for $1 \leq i \leq 4$) occurring at some position in Q represents a request from SSP i.

4. Q max is the maximum length of the SCP queue, while qr max is the corresponding limit on the lengths of SSP request queues. For the simulation Q max=100 and qr max=5.

5. Transmission times were set at 10ms for both directions of transmission.

6. The states of each SSP and the SCP. These can be either busy or idle and control the operation of the respective SSP or the SCP. For example, when an SSP starts processing a request or acknowledgement, the state of the SSP is switched to busy (or 1) and the 'end of the busy cycle' (ebc) time is set. At this time the state is switched back to idle and the SSP is ready to process the next acknowledgement if $qr > 0$. If $qa = 0$ and $qr > 0$ it processes a request. When the state is busy nothing can be processed.

The events that change the state of the system are:

1. Arrival of new requests or acknowledgements at an SSP. This increases either qa or qr for the SSP.

2. Processing of a request or an acknowledgement by an SSP. This switches the state to busy and sets the ebc time.

3. Transmission of a request from an SSP to the SCP. Either qa or qr (as appropriate) reduces by one.

4. Arrival of packets at the SCP. If the SCP queue is not full this increases the *Qlength* by the number of arriving packets and slots the numbers of the requesting SSP's into the next available positions in Q.

5. Processing of a packet at the SCP switches its state to busy and sets the time for the end of its busy cycle.

6. Transmission of an acknowledgement from the SCP Q decreases the *Qlength* by one and removes the first packet in Q from the vector.

The variables listed above are the variables required to describe the status of the system at any point in time. However in order to measure system performance for the various control schemes other variables had to be added. These are all listed with their initial values in the copies of the programs in the appendix. The i in SSP(i).qr refers to qr for the i-th SSP. Similarly for the rest of the variables.

The algorithm is a discrete event simulation. It updates the values of the variables at discrete points in time (at every 1ms point).

The Fuzzy Controller

The fuzzy controller has one input, rtd (round trip delay) and one output, window size. The rtd ranges from 0 ms to 1200 ms. Window size ranges in the set $\{1; 2; ...; 30\}$. Each domain is partitioned by seven fuzzy sets with linguistic labels Very Small, Small, Small Medium, Medium, Large Medium, Large, Very Large.

These are illustrated below:

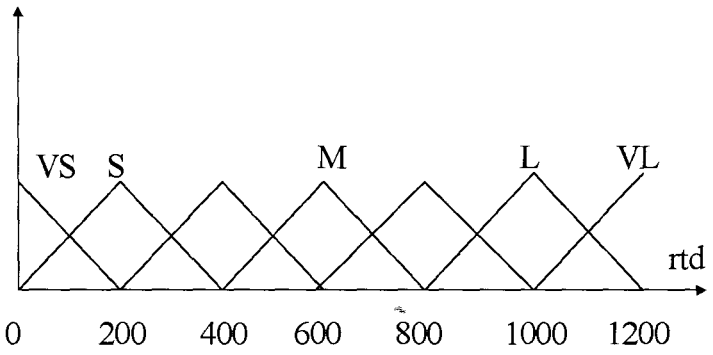


Figure 5.2: Fuzzy sets on round trip delay

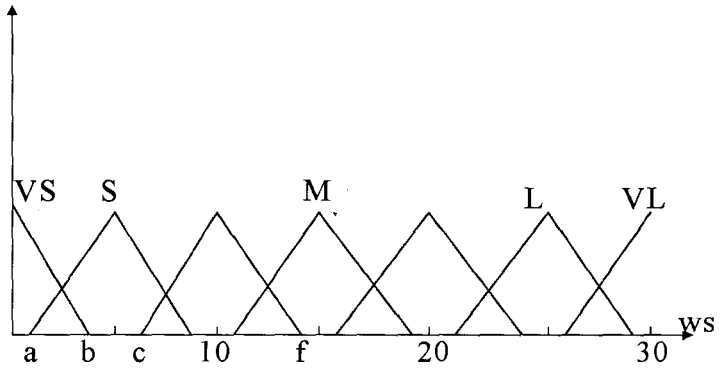


Figure 5.3: Fuzzy sets on window size

We require the window size to be smaller for a larger round trip delay and vice versa so the fuzzy controller is driven by the following rules:

R_1 : if rtd is VS then ws is VL

window size	40	80	120	160	200	240	280	320
COA	27	27	26	26	26	24	23	23
MOM	28	28	25	25	25	25	25	20
MAX	29	28	25	24	24	23	23	18

Table 5.1: Output for Defuzzification mechanisms

R_2 : if rtd is S then ws is L
R_3 : if rtd is SM then ws is LM
R_4 : if rtd is M then ws is M
R_5 : if rtd is LM then ws is SM
R_6 : if rtd is L then ws is S
R_7 : if rtd is VL then ws is VS

We constructed three controllers, each with a different defuzzification mechanism. The defuzzification mechanisms were the Centre Of Area, the Mean of Maxima and the Maximum criteria respectively. Then we compared their outputs over a range of round trip delay values. The results are shown in the table 5.1.

As can be seen the output of the COA defuzzification yields a slightly smoother transition from one value of round trip delay to the next. Although not shown in the table, this pattern is displayed over the rest of the range of values for round trip delay. For this reason the COA defuzzification mechanism was used in all further work.

The Genetic Algorithm

The letters a, b, c, d, e, f, g, h, i, j, k, l parameterize the start- and endpoints of the supports of the fuzzy sets. Each list of twelve parameter values defines a fuzzy controller. Each parameter has a range as indicated below:

$$
\begin{aligned}
a &\in \{0;1;2\} \\
b &\in \{3;4;5\} \\
c &\in \{5;6;7\} \\
d &\in \{8;9;10\} \\
e &\in \{10;11;12\} \\
f &\in \{13;14;15\} \\
g &\in \{15;16;17\} \\
h &\in \{18;19;20\} \\
i &\in \{20;21;22\} \\
j &\in \{23;24;25\}
\end{aligned}
$$

$$k \in \{25; 26; 27\}$$
$$l \in \{28; 29; 30\}$$

The controller was designed in this way so that a GA could be applied to it in order to determine a list of parameters for an optimal controller. The Centre Of Area defuzzification method was used.

The following steps outline the operation of the GA:

1 The algorithm starts with a randomly chosen initial population of different strings.

2 For each string the network simulation is run three times and an average fitness value is calculated for the string. The fitness value of a string is equal to the number of packets processed by the SCP during the runtime of the simulation.

3 A number of strings with highest fitness values are promoted directly to the following generation. A second number of strings are chosen randomly to which the crossover and mutation operators are applied. The crossed over and mutated strings are added to the following generation. New strings are compared to old ones to ensure that no repetitions occur.

4 The algorithm now iterates steps 2 and 3 until the fifth generation strings are evaluated and the string with highest fitness value is chosen. This string was then used in all following simulations.

Simulations

The aim of the simulations is to determine which congestion control mechanism optimizes the throughput of the SCP. Before the simulations were run to obtain the performance indices for the different congestion controllers, we optimized the performance of the static and fuzzy controllers.

First we determined at which value of window size the static window mechanism yielded a maximum value for SCP throughput. The values of window size were chosen from the set $\{12; 14; ...; 32\}$. For each value in this set the network simulation was run three times and an average value for SCP throughput calculated. Maximum throughput occurred at a window size of 20.

With the fuzzy controlled algorithm it turned out to be impractical to calculate a value of window size for each new value of round trip delay. Instead, we let an SSP accept a number of acknowledgements, after which the fuzzy controller uses the final value of round trip delay to calculate a new value of window size. We refer to this number as the firing cycle of the fuzzy controller. Using a range of values for the firing cycle, we determined that optimum performance occurred

at a firing cycle of 10. Results for these two optimization steps are included in Appendix B.

The throughput of the SCP is simply the number of requests going through the SCP from the start to the end of the simulation. One timestep in the simulation represents 1 ms. The simulation runs for 75 s, i.e. 75000 iterations. During the simulation the algorithm keeps track of the number of packets through the SCP in each second. We are interested in the operation of the SCP under overload conditions. Thus the arrival rates of requests at the SSP's were chosen such that the SCP is overloaded. The simulation was run with two different inputs. For the first input the rate of arrival of requests at the SSP's (and hence at the SCP) starts at twenty packets per second. It then increases in steps until it reaches a maximum value of 40 packets per second. It then tapers off until it reaches a constant 35 packets per second, which stays the same until the end of the experiment. Since there are four SSP's overload starts when the rate of arrival at the SSP's is greater than 25. The throughput of the SCP is calculated as

$$SCP.throughput = \sum_{t=1}^{75} SCP.caps(t)$$

where $SCP.caps(t)$ is the number of call attemps that the SCP processes in second t. *SCP.throughput* is also used as the fitness of a string during the operation of the genetic algorithm. For the second run of the simulation the arrival rates of requests at the different SSP's were independently and randomly chosen from the set of values:

$$\{35; 36...; 40\}$$

For each input the simulation was run ten times for each mechanism. This yielded ten values of throughput for each of the three mechanisms. For each mechanism we then calculated the average throughput and a 95% confidence interval for the true value of the mean using the Student's t distribution. The results of the two runs of the simulation can be seen in the tables in Appendix 2.

Chapter 6

Discussion and Conclusion

We now turn our attention to the two tables of results in Appendix B. These summarize the results for the various simulations of the network. The last three rows of the tables give some statistics for the data. We now use the values for the averages and the last entry in each column to calculate the 95% confidence interval for the true value of the mean (using a Student's t-distribution). These ranges are shown in the Table 6.1 below.

In both cases the Adaptive Window Mechanism outperforms the other two schemes. In one case the fuzzy scheme performs slightly better than the Static Window Mechanism. However, the differences are very small and while the fuzzy controller does not perform better than the other two mechanisms, its performance is very close to the performance of the other two schemes.

Future work will probably also include an investigation of rate based schemes and a comparison of these with the schemes discussed above. In some of these schemes the SCP sends an explicit rate control signal to the SSP to inform it of its allowed rate. These schemes have the advantage that the SSP receives the congestion information much quicker than in the above discussed cases. It is quite possible that this will increase the system performance.

Final Remarks

In this work we have attempted to do the following:

1. Provide an introduction to the mathematical background forming the basis for fuzzy model development. Specifically the two most commonly used fuzzy modeling tools, the Mamdani and Sugeno Controllers were discussed

	95% Conf. Int.	95% Conf. Int.
Static Window	6804-6814	6836-6844
Adaptive Window	6806-6816	6837-6848
Fuzzy Window	6803-6811	6838-6846

Table 6.1: Summary of results in Tables B.2 and B.3

in the second chapter. The Sugeno controller was basically just introduced while the Mamdani Controller was discussed in much more detail. The inference process was considered and mathematical structures developed to support the modeling process.

2. Provide some indication of theoretical questions that have been investigated regarding the modeling capabilities of fuzzy systems. This is done (as said before) in an attempt to change the attitude that the reasoning in fuzzy systems is vague and not rigorous. The idea is to provide a set of simple and rigorous structures on which to develop the processes and reasoning schemes employed in fuzzy modeling.

3. Introduce the reader to some of the systematic tools used to design fuzzy controllers, including fuzzy equivalence relations, fuzzy relational equations, fuzzy clustering and an ad hoc method referred to as Wang and Mendel's method. Of course each of these areas has developed independently of fuzzy control. The focus here was on providing a set of procedures for starting with a set of data for the system and ending up with a controller for the system.

4. Discuss the use of genetic algorithms for designing controllers and optimizing existing controllers. Here again the focus was on the practical aspects of the area.

5. Indicate some of the previous research work done in applying fuzzy controllers to telecommunication networks. While only a few examples were considered many more exist and most show that there is good reason for investigating the use of fuzzy solutions to problems in network management.

6. Produce a simple fuzzy controller to illustrate the possible use of fuzzy control in congestion control in a telecommunications network. It is hoped that we have shown that further investigations are justified.

Appendices

The appendices contain the following:

1. The first appendix contains the proofs of the statements 3.27 and 3.28.

2. The second appendix contains four tables summarizing the results for the experimental runs of the various mehanisms.

3. The third appendix contains the following:

4. The code for the simulation using the static window.

5. The code for the simulation using the adaptive window.
6. The code for the simulation using the fuzzy window.
7. The code for the fuzzy controller
8. The code for the genetic algorithm
9. A graph showing the variation with time of arrival rates of packets at the SSP's for a sample input.
10. A graph showing the number of packets processed by the SCP per second for the sample input.
11. A list and a table giving the final values of variables in the simulation for the above sample run of the simulation.

Bibliography

[1] Abe, S., Neural Networks and Fuzzy Systems: Theory and Applications, Kluwer Academic Publishers, 1997.

[2] Bonde, A. and Ghosh, S., A comparative study of Fuzzy versus Fixed thresholds for robust Queue Management in cell-switching networks, IEEE/ACM Transactions on Networking, vol 2, no 4, August 1994, pp337-344.

[3] Braae, M., and Rutherford, D., Fuzzy Relations in a Control setting, Kybernetes, vol 7, 1978, pp185-188.

[4] Braae M. and Rutherford D.A., Selection of parameters for a fuzzy logic controller, Fuzzy Sets and Systems, 1979, pp185-199.

[5] Buckley, J., Universal Fuzzy Controllers, Automatica, vol 28, no 6, 1992, pp1245-1248.

[6] Buckley, J., Theory of the fuzzy controller: An introduction, Fuzzy Sets and Systems, vol 51, 1992, pp249-258.

[7] Buckley, J., Sugeno type controllers are universal controllers, Fuzzy Sets and Systems, vol53, 1993, pp299-303.

[8] Buckley, J., Numerical relationships between neural networks, continuous functions and fuzzy systems, Fuzzy Sets and Systems, vol 60, 1993, pp1-8.

[9] Castro, J.L., Fuzzy Logic Controllers are Universal Approximators, IEEE Transactions on Systems, Man and Cybernetics, vol 25, no 4, April 1995, pp629-635.

[10] Catania, V., Ficili, G., Palazzo, S., and Panno, D., A Comparative Analysis of Fuzzy versus Conventional Policing Mechanisms for ATM Networks., IEEE/ACM Transactions on Networking, vol. 4, no. 3, June 1996, pp449-459.

[11] Cheng, R., Chang, C., Design of a Fuzzy Traffic Controller for ATM Networks, IEEE/ACM Transactions on Networking, vol 4, no.3, June 1996, pp460-469.

[12] Chemouil, P., Khalfet, J., Lebourges, M., A Fuzzy Control Approach for Adaptive Traffic Routing., IEEE Communications Magazine, July 1995, pp70-76.

[13] Chen, J., Xi, Y., Zhang, Z., A clustering algorithm for fuzzy model identification, Fuzzy Sets and Systems, vol 98, 1998, pp319-329 .

[14] Cooper, M., Vidal, J., Genetic design of Fuzzy Controllers: The Cart and Jointed-Pole problem., Second International Conference on Fuzzy Theory and Technology, Durham, NC, 1993, pp1332-1337.

[15] Cox, E., Solving Problems with Fuzzy Logic, AI Expert, March 1992, pp28-32.

[16] Czogala, E., Hirota, K., Probabilistic sets: Fuzzy and stochastic approach to decision, control and recognition processes, Verlag, 1986, pp73-83.

[17] De Silva, C., Intelligent Control: Fuzzy Logic Applications, CRC Press, 1995 .

[18] Forrest S., Genetic Algorithms: Principles of Natural Selection Applied to Computation, Science, vol. 261, August 1993, pp872-877.

[19] Gaines, B., Foundations of fuzzy reasoning, Int. J. Man-Machine Studies, vol 8,1976, pp623-668.

[20] Goldberg, D., Genetic and Evolutionary Algorithms come of age., Communications of the ACM, vol. 37, no.3, March1994, pp113-119.

[21] Harris, C., Advances in Intelligent Control, Taylor and Francis Ltd., 1994.

[22] Herrera, F., Lozano, M.,Verdegay, J., Tuning Fuzzy Logic Controllers by Genetic Algorithms, International Journal of Approximate Reasoning, vol 12, 1995, pp301-315.

[23] Jantzen, J., "Fuzzy Control," Technical University of Denmark: Electric Power Eng. Dept.,1991, pp41-61.

[24] Karr, C., Genetic Algorithms for Fuzzy Controllers, AI Expert, Feb.1991, pp26-32.

[25] Karr, C., Applying Genetics to Fuzzy Logic, AI Expert, March 1991, pp38-43.

[26] Karr, C. and Gentry, E., Fuzzy Control of pH using Genetic Algorithms., IEEE Transactions on Fuzzy Systems, vol 1,no. 1, Feb.1993, pp46-53.

[27] Kickert, W., and Van Naute Lemke, H., Application of a Fuzzy Controller in a Warm Water Plant, Automatica, vol 12, 1976, pp301-308.

[28] Kickert, W., and Mamdani, E., Analysis of a Fuzzy Logic Controller, Fuzzy Sets and Systems,

vol 1,1978, pp29-44.

[29] Klawonn, F. and Kruse, R., Equality relations as a basis for fuzzy control, Fuzzy Sets and Systems, vol 54, 1993, pp147-156.

[30] Klawonn, F., Fuzzy sets and vague environments., Fuzzy Sets and Systems, vol66, 1994, pp207-221.

[31] Klawonn, F., Kinzel J., Kruse R., Modifications of Genetic Algorithms for Designing and Optimizing Fuzzy Controllers, IEEE International Conference on Fuzzy Systems, 1996, pp1-6.

[32] Klawonn F. and Kruse R, Constructing a fuzzy controller from data, Fuzzy Sets and Systems, vol 85, 1997, pp237-255.

[33] Kroszynski U., Zhou J., Fuzzy Clustering, Principles, Methods and Examples, at http://www.ifs.dtu.dk/people/homepage/urikroszynski/fuzzyintro.htm.

[34] Lee, C., Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Part 1 and Part 2, IEEE Transactions on Systems, Man and Cybernetics, vol 20, no.2, March/April 1990, pp404-435.

[35] Liska, J. and Melsheimer, S., Complete Design of Fuzzy Logic Systems Using Genetic Algorithms, IEEE International Conference on Fuzzy Systems, 1994, pp1377-1382.

[36] Magdalena, L. and Monasterio-Huelin, F., A Fuzzy Logic Controller with learning through evolution of its Knowledge Base., International Journal of Approximate Reasoning, Oct., 1996, pp335-358.

[37] Mamdani, E., Application of fuzzy algorithms for control of a simple dynamic plant, Proc. IEE, vol 121, no12, Dec. 1974, pp1585-1588.

[38] Mamdani, E.H. and Assilian, S., An experiment in Linguistic Synthesis with a Fuzzy Logic Controller, Int. J. Man-Machine Studies, vol 7, 1975, pp1-13.

[39] Mamdani, E.H., Advances in the linguistic synthesis of fuzzy controllers, Int. J. Man-Machine Studies, vol 8,1976, pp669-678.

[40] Mamdani, E. H., Applications of Fuzzy Logic to Approximate Reasoning using Linguistic Synthesis, IEEE Transactions on Computers, vol.c-26, no.12, December 1977, pp1182-1191.

[41] Wang, L. and Mendel J. M., Generating Fuzzy Rules by Learning from Examples., IEEE Transactions on Systems, Man and Cybernetics, vol 22, no 6, Dec.92, pp1414-1427.

[42] Nelles, O., Fischer, M., Muller, B., Fuzzy Rule extraction by a Genetic Algorithm and constrained non-linear optimization of membership functions, IEEE International Conference on Fuzzy Systems, 1996, pp213-219.

[43] Nguyen, H., Sugeno, M., Tong, R., Yager, R., Theoretical Aspects of Fuzzy Control, John Wiley and Sons, 1995.

[44] Omidyar, C. and Pujolle, G., Introduction to Flow and Congestion Control, IEEE Communications Magazine, November 1996, pp30-32.

[45] Pedrycz, W. and Czogala, E., On identification in fuzzy systems and its applications in control problems, Fuzzy Sets and Systems, vol 6, 1981, pp153-167.

[46] Pedrycz, W., An approach to the analysis of fuzzy systems., Int. J. Control, vol 34, 1981, pp403-421.

[47] Pedrydcz, W., Fuzzy Control and Fuzzy Systems, John Wiley and Sons, 1993.

[48] Pham, D.T. and Karaboga, D., Optimum design of Fuzzy Logic Controllers using Genetic Algorithms., J. Syst Engineering, 1991, pp114-118.

[49] Pham, X. and Betts, R., Congestion Control for Intelligent Networks, Computer Networks and ISDN Systems, vol 26, 1994, pp511-524.

[50] Qi, X., Chin, T., Genetic algorithm based fuzzy controller for higher oredr systems, Fuzzy Sets and Systems, vol 91, 1997, pp279-284.

[51] Scheffer, M., Kunicki, J., Fuzzy Adaptive Traffic Enforcement for ATM Networks., Regional International Teletraffic Conference, South Africa, 1995, pp1-4.

[52] Self, K., Designing with Fuzzy Logic, IEEE Spectrum, Nov.1990, pp42-45.

[53] Sugeno, M. and Nishida, M., Fuzzy Control of Model Car, Fuzzy Sets and Systems, vol 16, 1985, pp103-113.

[54] Sugeno, M., Murofushi, T., Mori, T., Tatematsu, T., Tanaka, J., Fuzzy Algorithmic Control of Model Car by oral instructions, in Fuzzy Sets and Systems, vol 32, 1989, pp207-219.

[55] Takagi, H. and Lee, M.A., Integrating design stages of fuzzy systems using Genetic Algorithms, FuzzIEEE 93, pp1-6.

[56] Tan, G., Hu, X., On designing Fuzzy Controllers using Genetic Algorithms, IEEE International Conference on Fuzzy Systems, 1996, pp905-911.

[57] Tong, R.M., A Control Engineering Review of Fuzzy Systems, Automatica, vol 3, 1977, pppp559-569.

[58] Tong, R M., Synthesis of fuzzy models for industrial processes-some recent results, in Int. Journal General Systems, vol 4, 1978, pp143-162.

[59] Von Altrock, C., Fuzzy Logic and Neurofuzzy Applications Explained, Prentice Hall, 1995, pp1845-1851.

[60] Wang Li-Xin., Fuzzy systems are universal approximators, Proceedings of the IEEE International Conference on Fuzzy Systems, San Diego, 1992, pp1163-1169.

[61] Winston, W., Operations Research: Applications and Algorithms, Duxbury Press, 1994.

[62] Winter, G., Periaux, J., Galan, M., Cuesta, P., Genetic Algorithms in Engineering and Computer Science, John Wiley and Sons, 1995

[63] Yager, R., Filev, D., Essentials of Fuzzy Modeling and Control, John Wiley and Sons, 1994.

[64] Yoshinari, Y., Pedrycz, W., Hirota, K., Construction of fuzzy models through clustering techniques., Fuzzy Sets and Systems, vol 54, 1993, pp157-165.

[65] Zadeh, L.A., Fuzzy Sets, Information and Control, vol 8, 1965, pp338-353.

[66] Zadeh, L A, Outline of a new approach to the analysis of complex systems and decision processes, IEEE Transactions on Systems, Man, and Cybernetics, January 1973, pp28-44.

[67] Zadeh, L.A., Making computers think like people, IEEE Spectrum, Aug. 1984, pp26-32.

[68] Zadeh, L.A., The Calculus of fuzzy if-then rules, AI Expert, March 1992, pp23-27.

[69] Zimmerman, H., Fuzzy Set Theory and its Applications, Boston: Kluwer Academic Publishers, 1991.

Appendix A

1. We start by proving that a necessary condition for the function in 3.26 to have a minimum is given by the formula in 3.27 :

   Proof. Note that

   $$J(c) = \sum_{k=1}^{n}\sum_{i=1}^{c}(\mu_{ik})^w(d_{ik})^2$$

   has a trivial minimum at zero. To avoid this trivial minimum we modify the function above to:

   $$J'(c) = \sum_{k=1}^{n}\sum_{i=1}^{c}(\mu_{ik})^w(d_{ik})^2 - \lambda(1 - \sum_{i=1}^{c}\mu_{ik})$$

   For a minimum we take the derivative of the modified objective function and set it equal to zero:

   $$\frac{\partial J'}{\partial \mu_{jk}} = w\mu_{jk}^{w-1}d^2(v_j, x_k) - \lambda = 0 \text{ and} \tag{7.1}$$

   $$\frac{\partial J'}{\partial \lambda} = 1 - \sum_{k=1}^{c}\mu_{ik} = 0 \tag{7.2}$$

   Now from 3.32 get

   $$w\mu_{jk}^{w-1}d^2(v_j, x_k) - \lambda = 0 \text{ hence} \tag{7.3}$$

   $$\mu_{jk}^{w-1} = \frac{\lambda}{wd_{jk}^2} \text{ hence}$$

   $$\mu_{jk} = \left(\frac{\lambda}{wd_{jk}^2}\right)^{\frac{1}{w-1}} \tag{7.4}$$

   From 7.1 get:

   $$1 = \sum_{i=1}^{c}\mu_{ik}$$

   $$= \left(\sum_{i=1}^{c}\frac{1}{(wd_{ik}^2)^{\frac{1}{w-1}}}\right)\lambda^{\frac{1}{w-1}} \text{ using 7.2 hence}$$

$$\lambda^{\frac{1}{w-1}} = \frac{1}{\left(\sum_{i=1}^{c} \frac{1}{(wd_{ik}^2)^{\frac{1}{w-1}}}\right)} \text{ hence}$$

$$\lambda = \left(\frac{1}{\sum_{i=1}^{c} \frac{1}{(wd_{ik}^2)^{\frac{1}{w-1}}}}\right)^{w-1} \text{ hence}$$

$$\mu_{jk} = \left(\frac{\left(\frac{1}{\sum_{i=1}^{c} \frac{1}{\left(wd_{ik}^2\right)^{\frac{1}{w-1}}}}\right)^{w-1}}{wd_{jk}^2}\right)^{\frac{1}{w-1}}$$

$$= \frac{1}{\sum_{i=1}^{c} \left(\frac{d_{jk}^2}{d_{ik}^2}\right)^{\frac{1}{w-1}}} \tag{7.5}$$

Another necessary condition for a local minimum is

2. Next, 3.27 is proved below:

Proof. For a local minimum we need

$$\frac{\partial}{\partial v_i} \sum_{j=1}^{n} \sum_{l=1}^{c} \mu_{l,j}^{w} \|x_j - v_l\|^2 = 0 \tag{7.6}$$

Now

$$\begin{aligned}
& \frac{\partial}{\partial v_i} \sum_{j=1}^{n} \sum_{l=1}^{c} \mu_{lj}^{w} \|x_j - v_l\|^2 \\
= & \sum_{j=1}^{n} \mu_{ij}^{w} \frac{\partial}{\partial v_i} \|x_j - v_i\|^2 \\
= & \sum_{j=1}^{n} \mu_{ij}^{w} \lim_{t \to 0} \frac{\|x_j - (v_i + t\xi)\|^2 - \|x_j - v_i\|^2}{t}; \; t \in \mathbf{R} \; ; \; \xi \in \mathbf{R}^m \\
= & \sum_{j=1}^{n} \mu_{ij}^{w} \lim_{t \to 0} \frac{1}{t} \left[\left((x_j - v_i) - t\xi \right)^T \left((x_j - v_i) - t\xi \right) - (x_j - v_i)^T (x_j - v_i) \right] \\
= & \sum_{j=1}^{n} \mu_{ij}^{w} \lim_{t \to 0} \frac{-2t \, (x_j - v_i)^T \xi^T \xi}{t} \\
= & -2 \sum_{j=1}^{n} \mu_{ij}^{w} \, (x_j - v_i)^T \xi
\end{aligned}$$

Finally, get:

$$\begin{aligned} \sum_{j=1}^{n} \mu_{ij}^{w} \left(x_j - v_i \right) &= 0 \\ \Leftrightarrow \quad v_i &= \frac{\sum_{j=1}^{n} \mu_{ij}^{w} x_j}{\sum_{j=1}^{n} \mu_{ij}^{w}} \end{aligned}$$

which completes the proof.

Appendix B

The three tables in this appendix contain the results from the two simulation runs. Table 8.1 shows the throughput of the SCP for different values of the window size in the Static Window mechanism. Clearly the highest throughput occurred for a window size of 20.

Tables 8.2 and 8.3 give the throughput for the different runs of the three mechanisms for the two inputs:

Window Size	Throughput
12	2684
14	2690
16	2685
18	2687
20	2690
22	2687
24	2689
26	2619
28	2454
30	2310
32	2177

Table 8.1: Optimizing Window Size for the Static Window

Firing Cycle	Throughput
5	1822
10	1824
15	1815
20	1816
25	1815
30	1817
35	1819

Table 8.2: Optimizing Firing Cycle

Run	Static Window	Adaptive Window	Fuzzy Window
1	6803	6811	6814
2	6816	6803	6800
3	6802	6817	6801
4	6813	6825	6810
5	6818	6813	6800
6	6800	6805	6806
7	6801	6807	6811
8	6809	6815	6805
9	6814	6810	6814
10	6813	6807	6806
Average	6809	6811	6807
Std. Dev.	6,8	6,5	5,4
Conf. Int.	6804-6814	6806-6816	6803-6811

Table 8.3: Results for the three mechanisms for input 1

Run	Static Window	Adaptive Window	Fuzzy Window
1	6840	6838	6841
2	6849	6850	6833
3	6833	6843	6833
4	6845	6836	6849
5	6833	6845	6849
6	6841	6834	6849
7	6835	6833	6841
8	6841	6857	6838
9	6837	6844	6841
10	6845	6851	6845
Average	6840	6843	6842
Std. Dev.	5,4	7,9	6,1
Conf. Int.	6836-6844	6837-6848	6838-6846

Table 8.4: Results for the three mechanisms for input 2

Appendix C

This appendix contains copies of the software developed. The following programs and graphs are included:

1. The simulation using the static window mechanism;
2. The simulation using the adaptive window mechanism;
3. The simulation using the fuzzy window;
4. The fuzzy controller;
5. The genetic algorithm;
6. A graph illustrating the variation of arrival rates of packets at the SSP's
7. A graph illustrating the output from the simulation using the static window;
8. The final values of some of the variables for the above run of the simulation.

The graph in 9.2plots the number of packets processed by the SCP in every second for the 25 seconds of the runtime of the simulation. The system performance is summarized by the following list and 9.1 of final values of variables.

$$\begin{aligned} SCP\ throughput &= 2277 \\ Q\ length &= 78 \end{aligned}$$

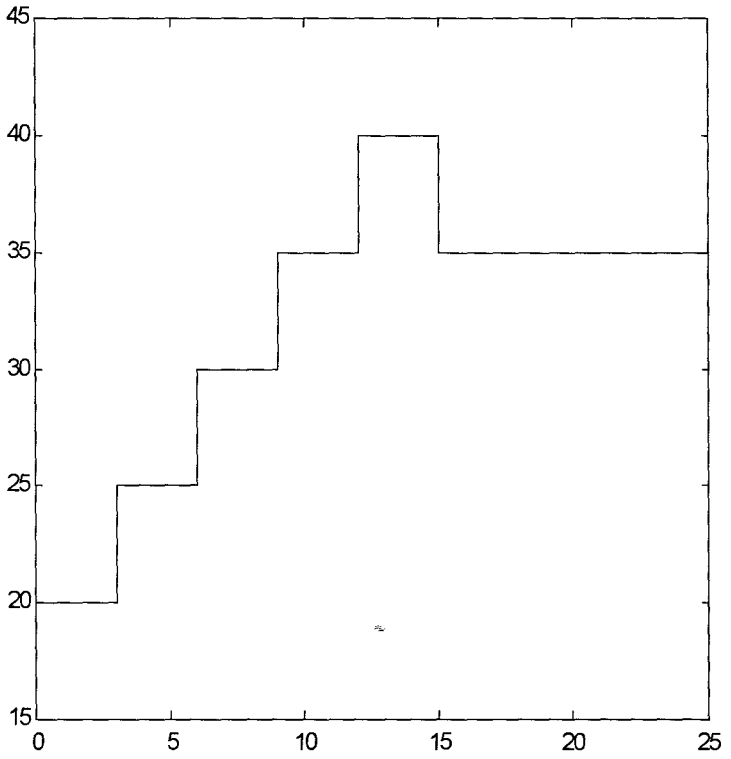


Figure 9.1: Variation of arrivals for the sample input

	SSP(1)	SSP(2)	SSP(3)	SSP(4)
Q dropped	0	0	0	0
Req. queue	0	0	0	0
Ack. queue	0	0	0	0
Dropped	220	221	221	222
Outstanding	18	17	17	17
Number of Ack. Packets	561	562	562	561
State	0	0	0	0
Arrival time	25003	25013	25012	25021

Table 9.1: Final values of variables for the sample run above

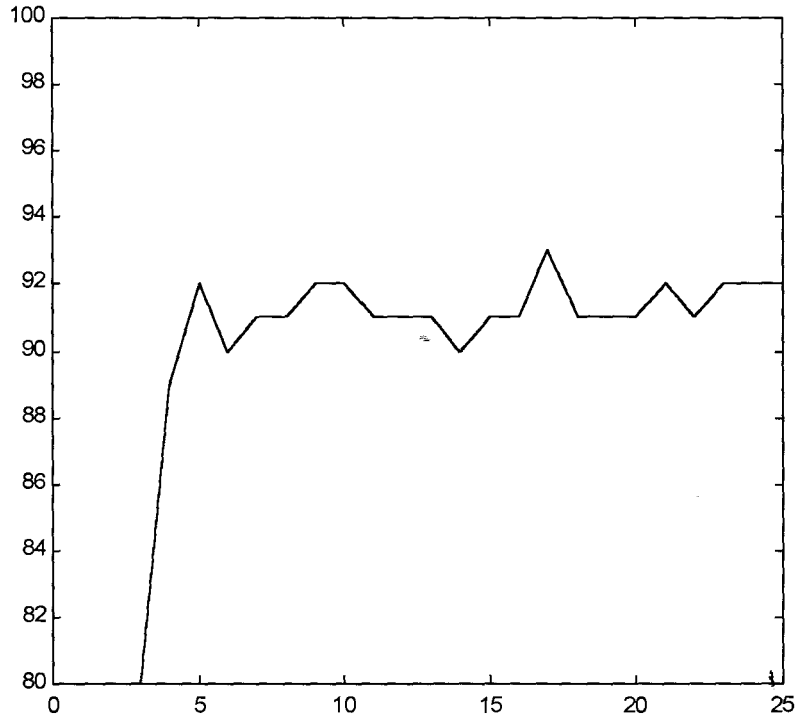


Figure 9.2: Processing of packets at the SCP

```
            %network simulation with static window
for i=1:4
   ebc(i)=0;
   state(i)=0;
   qa(i)=0;
   qr(i)=0;
   at(i)=0;
   iat(i)=0;
   count(i)=0;
   qrtt(i)=0;
   dropped(i)=0;
   SSP(i).qaat=[0 0];
   SSP(i).Qat=[];
   SSP(i).arr=[];
   arrived(i)=0;
   win(i)=20;
   pack_off(i)=0;
   out(i)=0;
   SSP(i).ttimes=[];
   qatt(i)=0;
   nap(i)=0;
   arrival(i)=0;
end
Q=[];                       SCP.state=0;
Qlength=0;                  Qmax=100;
eff=[];
SCP.eff=[];
Qtimer=[];                  SCP.ebc=0;
SCP.tt=0;                   SCP.counter=0;
SCP_off_load=[];            Qdropped=0;
time=[];                    SCP.caps=0;
qrmax=5;

for t=1:75000
% SSP's modules
% module 1 - state control
for i=1:4
   if ebc(i)==t
      state(i)=0;
   end
end

% module 2 - ack que arrivals
for i=1:4
   if SSP(i).qaat(1)==t
      qa(i)=qa(i)+1;
      l=find(SSP(i).ttimes==SSP(i).qaat(2));
```

```
      SSP(i).ttimes(1)=[];
      out(i)=size(SSP(i).ttimes,2);
   end
end

%module 3 - ack que transmissions
for i=1:4
   if qatt(i)==t
      qa(i)=qa(i)-1;
      nap(i)=nap(i)+1;
   end
end

%module 4 - ack que processing
for i=1:4
   if qa(i)>0 & state(i)==0
      state(i)=1;
      ebc(i)=t+3;
      qatt(i)=t+3;
   end
end

% module 5 - req que arrivals
for i=1:4
   if at(i)==t
      arrival(i)=arrival(i) + 1;
      if  qrmax > qr(i)
         qr(i) = qr(i) + 1;
      else
         dropped(i) = dropped(i) + 1;
      end
   end
end

for i=1:4

   if t==1
      pps=35;
   end

      iat(i)=floor((1000/(pps+1))*rand(1,1));
      if iat(i)==0
         at(i)=t+1;
      else
         at(i)=t+iat(i);
      end
```

```
   if t==at(i)
      SSP(i).arr(size(SSP(i).arr,2)+1)=t;
      if arrived(i)<pps
         n=floor(t/10000);
         m=floor((t-n*10000)/1000);
         trem=n*10000+(m+1)*1000-t;
         iat(i)=floor((trem/(pps-arrived(i)+1))*rand(1,1));
         if iat(i)==0
            at(i)=t+1;
         else
            at(i)=t+iat(i);
         end
      end
   end
   if rem(t,1000)==0
      iat(i)=floor((1000/(pps+1))*rand(1,1));
      if iat(i)==0
         at(i)=t+1;
      else
         at(i)=t+iat(i);
      end
      arrived(i)=0;
   end
end


% module 7 - request que transmissions
for i=1:4
   if qrtt(i)==t & win(i) > out(i)
      qr(i)=qr(i) - 1;
      SSP(i).Qat(size(SSP(i).Qat,2)+1)=t+10;
      count(i)=count(i)+1;
      SSP(i).ttimes(size(SSP(i).ttimes,2)+1)=t;
      out(i)=size(SSP(i).ttimes,2);
   elseif qrtt(i)==t & ~(win(i) > out(i))
      qrtt(i) = qrtt(i) + 1;
   end
end


% timeout check
if t>1200
   for i=1:4
      l=find(t-SSP(i).ttimes>1200);
      if isempty(l)==0
         for j=1:size(l,2)
```

```
            SSP(i).ttimes(l(j))=[];
         end
      end
      out(i)=size(SSP(i).ttimes,2);
   end
end

% module 8 - req que processing
for i=1:4
   if state(i)==0 & qr(i)>0
      state(i)=1;
      ebc(i)=t+7;
      qrtt(i)=t+7;
   end
end

% module 9 - SCP state control
if SCP.ebc==t
   SCP.state=0;
end

% module 10 - arrivals at SCP
for i=1:4
   if size(SSP(i).Qat,2)>0 & SSP(i).Qat(1)==t
      if size(Q,2)<Qmax
         Q(size(Q,2)+1)=i;
         Qtimer(size(Qtimer,2)+1)=t-10;
      elseif size(Q,2)==Qmax
         Qdropped=Qdropped+1;
      end
         SSP(i).Qat(1)=[];
   end
end


% module 11 - SCP transmission
if SCP.tt==t
   SSP(Q(1)).qaat(1)=t+10;
   SSP(Q(1)).qaat(2)=Qtimer(1);
   Q(1)=[];
   Qtimer(1)=[];
   SCP.counter=SCP.counter+1;
end


% module 12 - SCP processing
if SCP.state==0 & size(Q,2)>0
```

```
    SCP.state=1;
    % rand number generator
    r=rand(1,1);
    x=-log(r);
    if x-floor(x)<.5
    x=floor(x);
    else
    x=ceil(x);
    end
    SCP.ebc=t+10+x;
    SCP.tt=t+10+x;
end
for j=1:75
    time(j)=j;
end
if rem(t,1000)==0
        Dropped(t/1000)=dropped(1);
        dropped(1)=0;
        Arrivals(t/1000)=arrival(1);
        arrival(1)=0;
        SCP_off_load(t/1000)=sum(count);
        count=zeros(1,4);
        SCP_dropped(t/1000)=Qdropped;
        Qdropped=0;
    Qlength(t/1000)=size(Q,2);
    SCP.caps(t/1000)=SCP.counter;
    SCP.counter=0;
    pps=round(35+5*rand(1,1));

end
end
figure(1)
plot(time,SCP_dropped);axis([0 75 0 50]);xlabel('time in
seconds');ylabel('Dropped at SCP');
figure(2)
plot(time,Dropped);axis([0 75 0 100]);xlabel('time in
seconds');ylabel('Dropped at SSP(1)');
figure(3)
plot(time,SCP.caps);axis([0 75 80 100]);xlabel('time in
seconds');ylabel('Processed at SCP');
figure(4)
plot(time,Qlength);axis([0 75 60 100]);xlabel('time in
seconds');ylabel('Qlength');
figure(5)
plot(time,Arrivals);axis([0 75 20 50]);xlabel('time in
seconds');ylabel('Arrivals at SSP(1)');
```

```
E=sum(SCP.caps)*(1-
(sum(SCP_dropped)/(sum(SCP_dropped)+sum(SCP.caps))));
disp(['Static case :']);
disp(['time='num2str(t) ':']);
disp(['Dropped at SCP='num2str(sum(SCP_dropped))]);
disp(['Dropped at SSP(1)='num2str(sum(Dropped))]);
disp(['Processed by SCP='num2str(sum(SCP.caps))]);
disp(['Arrivals at SSP(1)='num2str(sum(Arrivals))]);
disp(['Efficiency='num2str(E)]);
```

```
          %network simulation with adaptive window
for i=1:4
   ebc(i)=0;
   state(i)=0;
   qa(i)=0;
   qr(i)=0;
   at(i)=1;
   count(i)=0;
   qrtt(i)=0;
   dropped(i)=0;
   SSP(i).qaat=[0 0];
   SSP(i).Qat=[];
   pack_off(i)=0;
   win(i)=1;
   out(i)=0;
   SSP(i).ttimes=[];
   qatt(i)=0;
   nopack(i)=1;
   nap(i)=0;
   c(i)=0;
   arrival(i)=0;
   arrived(i)=0;
end
Q=[];                      SCP.state=0;
Qlength=0;                 Qmax=100;
winmin=1;                  eff=[];
winmax=26;                 SCP.eff=[];
Qtimer=[];                 SCP.ebc=0;
SCP.tt=0;                  SCP.counter=0;
SCP_off_load=[];           Qdropped=0;
time=[];                   SCP.caps=0;
qrmax=5;
cmax=4;
```

```
for t=1:75000
% SSP's modules
% module 1 - state control
for i=1:4
   if ebc(i)==t
      state(i)=0;
   end
end

% module 2 - ack que arrivals
for i=1:4
   if SSP(i).qaat(1)==t
      qa(i)=qa(i)+1;
      l=find(SSP(i).ttimes==SSP(i).qaat(2));
      SSP(i).ttimes(l)=[];
      out(i)=size(SSP(i).ttimes,2);
      c(i)=c(i)+1;
      if c(i)>cmax
         c(i)=0;
         if win(i)<winmax
            win(i)=win(i)+1;
         end
      end
   end
end

%module 3 - ack que transmissions
for i=1:4
   if qatt(i)==t
      qa(i)=qa(i)-1;
      nap(i)=nap(i)+1;
   end
end
%module 4 - ack que processing
for i=1:4
   if qa(i)>0 & state(i)==0
      state(i)=1;
      ebc(i)=t+3;
      qatt(i)=t+3;
   end
end

% module 5 - req que arrivals
for i=1:4
   if at(i)==t

      if  qrmax > qr(i)
```

```
            qr(i) = qr(i) + 1;
        else
            dropped(i) = dropped(i) + 1;
        end
    end
end

% module 6 - packet generator
for i=1:4

    if t==1
        pps=35;
        iat(i)=floor((1000/(pps+1))*rand(1,1));
        if iat(i)==0
            at(i)=t+1;
        else
            at(i)=t+iat(i);
        end
    end

    if t==at(i)
        arrived(i)=arrived(i)+1;
        if arrived(i)<pps
            n=floor(t/10000);
            m=floor((t-n*10000)/1000);
            trem=n*10000+(m+1)*1000-t;
            iat(i)=floor((trem/(pps-arrived(i)+1))*rand(1,1));
            if iat(i)==0
                at(i)=t+1;
            else
                at(i)=t+iat(i);
            end
        end
    end
    if rem(t,1000)==0
        iat(i)=floor((1000/(pps+1))*rand(1,1));
        if iat(i)==0
            at(i)=t+1;
        else
            at(i)=t+iat(i);
        end
        arrived(i)=0;
    end
end

% module 7 - request que transmissions
for i=1:4
```

```
   if qrtt(i)==t & win(i) > out(i)
      qr(i)=qr(i) - 1;
      SSP(i).Qat(size(SSP(i).Qat,2)+1)=t+10;
      count(i)=count(i)+1;
      SSP(i).ttimes(size(SSP(i).ttimes,2)+1)=t;
      out(i)=size(SSP(i).ttimes,2);
   elseif qrtt(i)==t & ~(win(i) > out(i))
      qrtt(i) = qrtt(i) + 1;
   end
end

% timeout check
if t>1200
   for i=1:4
      l=find(t-SSP(i).ttimes>1200);
      if isempty(l)==0
         for j=1:size(l,2)
            SSP(i).ttimes(l(j))=[];
         end
         if ~(win(i)==winmin)
            win(i)=win(i)-1;
         end
      end
      out(i)=size(SSP(i).ttimes,2);
   end
end

% module 8 - req que processing
for i=1:4
   if state(i)==0 & qr(i)>0
      state(i)=1;
      ebc(i)=t+7;
      qrtt(i)=t+7;
   end
end

% module 9 - SCP state control
if SCP.ebc==t
   SCP.state=0;
end

% module 10 - arrivals at SCP
for i=1:4
   if size(SSP(i).Qat,2)>0 & SSP(i).Qat(1)==t
      if size(Q,2)<Qmax
         Q(size(Q,2)+1)=i;
         Qtimer(size(Qtimer,2)+1)=t-10;
```

```
      elseif size(Q,2)==Qmax
         Qdropped=Qdropped+1;
      end
         SSP(i).Qat(1)=[];
   end
end


% module 11 - SCP transmission
if SCP.tt==t
   SSP(Q(1)).qaat(1)=t+10;
   SSP(Q(1)).qaat(2)=Qtimer(1);
   Q(1)=[];
   Qtimer(1)=[];
   SCP.counter=SCP.counter+1;
end


% module 12 - SCP processing
if SCP.state==0 & size(Q,2)>0
   SCP.state=1;
   % rand number generator
   r=rand(1,1);
   x=-log(r);
   if x-floor(x)<.5
   x=floor(x);
   else
   x=ceil(x);
   end
   SCP.ebc=t+10+x;
   SCP.tt=t+10+x;
end
for j=1:75
   time(j)=j;
end
if rem(t,1000)==0
      Dropped(t/1000)=dropped(1);
      dropped(1)=0;
      Arrivals(t/1000)=arrival(1);
      arrival(1)=0;
      SCP_off_load(t/1000)=sum(count);
      count=zeros(1,4);
      SCP_dropped(t/1000)=Qdropped;
      Qdropped=0;
   Qlength(t/1000)=size(Q,2);
   SCP.caps(t/1000)=SCP.counter;
   SCP.counter=0;
```

```
   pps=round(35+5*rand(1,1));

end

end
figure(1)
plot(time,SCP_dropped);axis([0 75 0 50]);xlabel('time in
seconds');ylabel('Dropped at SCP');
figure(2)
plot(time,Dropped);axis([0 75 0 80]);xlabel('time in
seconds');ylabel('Dropped at SSP(1)');
figure(3)
plot(time,SCP.caps);axis([0 75 80 100]);xlabel('time in
seconds');ylabel('Processed at SCP');
figure(4)
plot(time,Qlength);axis([0 75 0 100]);xlabel('time in
seconds');ylabel('Qlength');
figure(5)
plot(time,Arrivals);axis([0 75 0 80]);xlabel('time in
seconds');ylabel('Arrivals at SSP(1)');
E=sum(SCP.caps)*(1-
(sum(SCP_dropped)/(sum(SCP_dropped)+sum(SCP.caps))));
disp(['Adaptive case :']);
disp(['time='num2str(t) ':']);
disp(['Dropped at SCP='num2str(sum(SCP_dropped))]);
disp(['Dropped at SSP(1)='num2str(sum(Dropped))]);
disp(['Processed by SCP='num2str(sum(SCP.caps))]);
disp(['Arrivals at SSP(1)='num2str(sum(Arrivals))]);
disp(['Efficiency='num2str(E)]);


            %network simulation with fuzzy window
for i=1:4
   ebc(i)=0;
   state(i)=0;
   qa(i)=0;
   qr(i)=0;
   at(i)=1;
   count(i)=0;
   qrtt(i)=0;
   dropped(i)=0;
   SSP(i).qaat=[0 0];
   SSP(i).Qat=[];
   SSP(i).arr=[];
   pack_off(i)=0;
   win(i)=1;
   out(i)=0;
```

```
   rtdl(i)=0;
   SSP(i).ttimes=[];
   qatt(i)=0;
   nopack(i)=1;
   nap(i)=0;
   fuzzcount(i)=0;
   arrival(i)=0;
   arrived(i)=0;
end
Q=[];                          SCP.state=0;
Qlength=0;                     Qmax=100;
Qtimer=[];                     SCP.ebc=0;
SCP.tt=0;                      SCP.counter=0;
SCP_off_load=[];               Qdropped=0;
time=[];                       SCP.caps=0;
qrmax=5;                       fuzzcountmax=10;
cmax=4;                        E=0;

for t=1:75000
% SSP's modules
% module 1 - state control
for i=1:4
   if ebc(i)==t
      state(i)=0;
   end
end

% module 2 - ack que arrivals
for i=1:4
   if SSP(i).qaat(1)==t
      qa(i)=qa(i)+1;
      l=find(SSP(i).ttimes==SSP(i).qaat(2));
      SSP(i).ttimes(l)=[];
      out(i)=size(SSP(i).ttimes,2);
      fuzzcount(i)=fuzzcount(i)+1;
      if fuzzcount(i)==fuzzcountmax
         fuzzcount(i)=0;
         rtd(i)=t-SSP(i).qaat(2);
         win(i)=fuzzy7coaga(rtd(i));
      end
   end
end

%module 3 - ack que transmissions
for i=1:4
   if qatt(i)==t
      qa(i)=qa(i)-1;
```

```
      nap(i)=nap(i)+1;
   end
end
%module 4 - ack que processing
for i=1:4
   if qa(i)>0 & state(i)==0
      state(i)=1;
      ebc(i)=t+3;
      qatt(i)=t+3;
   end
end

% module 5 - req que arrivals
for i=1:4
   if at(i)==t
      arrival(i)=arrival(i) + 1;
      if  qrmax > qr(i)
         qr(i) = qr(i) + 1;
      else
         dropped(i) = dropped(i) + 1;
      end
   end
end

% module 6 - packet generator
for i=1:4

   if t==1
      pps=35;
      iat(i)=floor((1000/(pps+1))*rand(1,1));
      if iat(i)==0
         at(i)=t+1;
      else
         at(i)=t+iat(i);
      end
   end

   if t==at(i)
      arrived(i)=arrived(i)+1;
      SSP(i).arr(size(SSP(i).arr,2)+1)=t;
      if arrived(i)<pps
         n=floor(t/10000);
         m=floor((t-n*10000)/1000);
         trem=n*10000+(m+1)*1000-t;
         iat(i)=floor((trem/(pps-arrived(i)+1))*rand(1,1));
         if iat(i)==0
            at(i)=t+1;
```

```
            else
               at(i)=t+iat(i);
            end
         end
      end
      if rem(t,1000)==0
         iat(i)=floor((1000/(pps+1))*rand(1,1));
         if iat(i)==0
            at(i)=t+1;
         else
            at(i)=t+iat(i);
         end
         arrived(i)=0;
      end
   end

   % module 7 - request que transmissions
   for i=1:4
      if qrtt(i)==t & win(i) > out(i)
         qr(i)=qr(i) - 1;
         SSP(i).Qat(size(SSP(i).Qat,2)+1)=t+10;
         count(i)=count(i)+1;
         SSP(i).ttimes(size(SSP(i).ttimes,2)+1)=t;
         out(i)=size(SSP(i).ttimes,2);
      elseif qrtt(i)==t & ~(win(i) > out(i))
         qrtt(i) = qrtt(i) + 1;
      end
   end

   % timeout check
   if t>1200
      for i=1:4
         l=find(t-SSP(i).ttimes>1200);
         if isempty(l)==0
            for j=1:size(l,2)
               SSP(i).ttimes(l(j))=[];
            end
            if ~(win(i)==winmin)
               win(i)=win(i)-1;
            end
         end
         out(i)=size(SSP(i).ttimes,2);
      end
   end

   % module 8 - req que processing
   for i=1:4
```

```
   if state(i)==0 & qr(i)>0
      state(i)=1;
      ebc(i)=t+7;
      qrtt(i)=t+7;
   end
end

% module 9 - SCP state control
if SCP.ebc==t
   SCP.state=0;
end

% module 10 - arrivals at SCP
for i=1:4
   if size(SSP(i).Qat,2)>0 & SSP(i).Qat(1)==t
      if size(Q,2)<Qmax
         Q(size(Q,2)+1)=i;
         Qtimer(size(Qtimer,2)+1)=t-10;
      elseif size(Q,2)==Qmax
         Qdropped=Qdropped+1;
      end
         SSP(i).Qat(1)=[];
   end
end


% module 11 - SCP transmission
if SCP.tt==t
   SSP(Q(1)).qaat(1)=t+10;
   SSP(Q(1)).qaat(2)=Qtimer(1);
   Q(1)=[];
   Qtimer(1)=[];
   SCP.counter=SCP.counter+1;
end


% module 12 - SCP processing
if SCP.state==0 & size(Q,2)>0
   SCP.state=1;
   % rand number generator
   r=rand(1,1);
   x=-log(r);
   if x-floor(x)<.5
   x=floor(x);
   else
   x=ceil(x);
   end
```

```
   SCP.ebc=t+10+x;
   SCP.tt=t+10+x;
end
for j=1:75
   time(j)=j;
end
if rem(t,1000)==0
      Dropped(t/1000)=dropped(1);
      dropped(1)=0;
      Arrivals(t/1000)=arrival(1);
      arrival(1)=0;
      SCP_off_load(t/1000)=sum(count);
      count=zeros(1,4);
      SCP_dropped(t/1000)=Qdropped;
      Qdropped=0;
      Qlength(t/1000)=size(Q,2);
      SCP.caps(t/1000)=SCP.counter;
      SCP.counter=0;
      pps=round(35+5*rand(1,1));

end
end
E=sum(SCP.caps)*(1-
(sum(SCP_dropped)/(sum(SCP_dropped)+sum(SCP.caps))));
figure(1)
plot(time,SCP_dropped);axis([0 75 0 50]);xlabel('time in
seconds');ylabel('Dropped at SCP');
figure(2)
plot(time,Dropped);axis([0 75 0 50]);xlabel('time in
seconds');ylabel('Dropped at SSP(1)');
figure(3)
plot(time,SCP.caps);axis([0 75 80 100]);xlabel('time in
seconds');ylabel('Processed at SCP');
figure(4)
plot(time,Qlength);axis([0 75 80 100]);xlabel('time in
seconds');ylabel('Qlength');
figure(5)
plot(time,Arrivals);axis([0 75 40 60]);xlabel('time in
seconds');ylabel('Arrivals at SSP(1)');
disp(['time='num2str(t) ':']);
disp(['Dropped at SCP='num2str(sum(SCP_dropped))]);
disp(['Dropped at SSP(1)='num2str(sum(Dropped))]);
disp(['Processed by SCP='num2str(sum(SCP.caps))]);
disp(['Arrivals at SSP(1)='num2str(sum(Arrivals))]);
disp(['Efficiency='num2str(E)]);
```

```
                        %fuzzy Controller
function ws=fuzzy7coaga(rtd)

a=1;b=4;c=6;d=8;e=12;f=14;g=15;h=20;ip=22;j=24;kp=26;lp=30;

x=sym('x');
   f1=-(1/200)*rtd+1;   y1_2=(-5*b)/(a-b-5);
   f2=(1/200)*rtd;      y3_4=(d*(10-c)+c*(d-5))/(d-c+5);
   f3=-(1/200)*rtd+2;   y5_6=(f*(15-e)+e*(f-10))/(f-e+5);
   f4=(1/200)*rtd-1;    y7_8=(h*(20-g)+g*(h-15))/(h-g+5);
   f5=-(1/200)*rtd+3;   y9_10=(j*(25-ip)+ip*(j-20))/(j-
ip+5);
   f6=(1/200)*rtd-2;    y11_12=(lp*(30-kp)+kp*(lp-25))/(lp-
kp+5);
   f7=-(1/200)*rtd+4;
   f8=(1/200)*rtd-3;
   f9=-(1/200)*rtd+5;
   f10=(1/200)*rtd-4;
   f11=-(1/200)*rtd+6;
   f12=(1/200)*rtd-5;

   g1=-(1/b)*x+1;
   g2=(x-a)/(5-a);
   g3=(d-x)/(d-5);
   g4=(x-c)/(10-c);
   g5=(f-x)/(f-10);
   g6=(x-e)/(15-e);
   g7=(h-x)/(h-15);
   g8=(x-g)/(20-g);
   g9=(j-x)/(j-20);
   g10=(x-ip)/(25-ip);
   g11=(lp-x)/(lp-25);
   g12=(x-kp)/(30-kp);

if rtd>=0 & rtd<200
         z2_10=f2*(25-ip)+ip;z3_10=f3*(25-ip)+ip;
         z4_10=f4*(25-ip)+ip;z1_11=-f1*(lp-25)+lp;
         z2_12=f2*(30-kp)+kp;z1_12=f1*(30-kp)+kp;z2_11=-
f2*(lp-25)+lp;
if f1>=f2 & f2<=(y11_12-kp)/(30-kp)
       ws=(int(g10*x,x,ip,z2_10)+int(f2*x,x,z2_10,z2_12)+
       int(g12*x,x,z2_12,z1_12)+int(f1*x,x,z1_12,30))/(int(
       g10,x,ip,z2_10)+int(f2,x,z2_10,z2_12)+int(g12,x,z2_1
       2,z1_12)
       +int(f1,x,z1_12,30));
```

```
        ws=round(double(ws));
elseif f1>(y11_12-kp)/(30-kp) & f2>(y11_12-kp)/(30-kp)
int(g11*x,x,z2_11,y11_12)+int(g12*x,x,y11_12,z1_12)+int(f1*
x,x,z1_12,30))/(int(g10,x,ip,z2_10)+int(f2,x,z2_10,z2_11)+i
nt(g11,x,z2_11,y11_12)+int(g12,x,y11_12,z1_12)+int(f1,x,z1_
12,30));
        ws=round(double(ws));
   elseif f2>f1 &  f1<=(y11_12-kp)/(30-kp)

ws=(int(g10*x,x,ip,z2_10)+int(f2*x,x,z2_10,z2_11)+int(g11*x
,x,z2_11,z1_11)+int(f1*x,x,z1_11,30))/(int(g10,x,ip,z2_10)+
int(f2,x,z2_10,z2_11)+int(g11,x,z2_11,z1_11)+int(f1,x,z1_11
,30));
        ws=round(double(ws));
   end
end

if rtd>=200 & rtd<400
   z4_8=f4*(20-g)+g;z4_10=f4*(25-ip)+ip;z3_10=f3*(25-
ip)+ip;z3_11=-f3*(lp-25)+lp;
   z4_9=-f4*(j-20)+j;z3_9=-f3*(j-20)+j;
   if f3>=f4 & f4<=(y9_10-ip)/(25-ip)

ws=(int(g8*x,x,g,z4_8)+int(f4*x,x,z4_8,z4_10)+int(g10*x,x,z
4_10,z3_10)+int(f3*x,x,z3_10,z3_11)+int(g11*x,x,z3_11,lp))/
(int(g8,x,g,z4_8)+int(f4,x,z4_8,z4_10)+int(g10,x,z4_10,z3_1
0)+int(f3,x,z3_10,z3_11)+int(g11,x,z3_11,lp));
        ws=round(double(ws));
    elseif f4>(y9_10-ip)/(25-ip) & f3>(y9_10-ip)/(25-ip)

ws=(int(g8*x,x,g,z4_8)+int(f4*x,x,z4_8,z4_9)+int(g9*x,x,z4_
9,y9_10)+int(g10*x,x,y9_10,z3_10)+int(g11*x,x,z3_10,lp))/(i
nt(g8,x,g,z4_8)+int(f4,x,z4_8,z4_9)+int(g9,x,z4_9,y9_10)+in
t(g10,x,y9_10,z3_10)+int(g11,x,z3_10,lp));
        ws=round(double(ws));
    elseif f4>=f3 & f3<=(y9_10-ip)/(25-ip)

ws=(int(g8*x,x,g,z4_8)+int(f4*x,x,z4_8,z4_9)+int(g9*x,x,z4_
9,z3_9)+int(f3*x,x,z3_9,z3_11)+int(g11*x,x,z3_11,lp))/(int(
g8,x,g,z4_8)+int(f4,x,z4_8,z4_9)+int(g9,x,z4_9,z3_9)+int(f3
,x,z3_9,z3_11)+int(g11,x,z3_11,lp));
        ws=round(double(ws));
    end
end

if rtd>=400 & rtd<600
```

```
   z6_6=f6*(15-e)+e;z6_8=f6*(20-g)+g;z5_8=f5*(20-
g)+g;z5_9=-f5*(j-20)+j;
   z6_7=-f6*(h-15)+h;z5_7=-f5*(h-15)+h;
      if f5>=f6 & f6<=(y7_8-g)/(20-g)

ws=(int(g6*x,x,e,z6_6)+int(f6*x,x,z6_6,z6_8)+int(g8*x,x,z6_
8,z5_8)+int(f5*x,x,z5_8,z5_9)+int(g9*x,x,z5_9,j))/(int(g6,x
,e,z6_6)+int(f6,x,z6_6,z6_8)+int(g8,x,z6_8,z5_8)+int(f5,x,z
5_8,z5_9)+int(g9,x,z5_9,j));
         ws=round(double(ws));
      elseif f6>(y7_8-g)/(20-g) & f5>(y7_8-g)/(20-g)

ws=(int(g6*x,x,e,z6_6)+int(f6*x,x,z6_6,z6_7)+int(g7*x,x,z6_
7,y7_8)+int(g8*x,x,y7_8,z5_8)+int(f5*x,x,z5_8,z5_9)+int(g9*
x,x,z5_9,j))/(int(g6,x,e,z6_6)+int(f6,x,z6_6,z6_7)+int(g7,x
,z6_7,y7_8)+int(g8,x,y7_8,z5_8)+int(f5,x,z5_8,z5_9)+int(g9,
x,z5_9,j));
         ws=round(double(ws));
      elseif f5<=f6 & f5<=(y7_8-g)/(20-g)

ws=(int(g6*x,x,e,z6_6)+int(f6*x,x,z6_6,z6_7)+int(g7*x,x,z6_
7,z5_7)+int(f5*x,x,z5_7,z5_9)+int(g9*x,x,z5_9,j))/(int(g6,x
,e,z6_6)+int(f6,x,z6_6,z6_7)+int(g7,x,z6_7,z5_7)+int(f5,x,z
5_7,z5_9)+int(g9,x,z5_9,j));
         ws=round(double(ws));
      end
end

if rtd>=600 & rtd<800
   z8_4=f8*(10-c)+c;z8_5=-f8*(f-10)+f;z7_6=f7*(15-
e)+e;z7_7=-f7*(h-15)+h;
   z8_6=f8*(15-e)+e;z7_5=-f7*(f-10)+f;
   if f7>(y5_6-e)/(15-e) & f8>(y5_6-e)/(15-e)

ws=(int(g4*x,x,c,z8_4)+int(f8*x,x,z8_4,z8_5)+int(g5*x,x,z8_
5,y5_6)+int(f6*x,x,y5_6,z7_6)+int(f7*x,x,z7_6,z7_7)+int(g7*
x,x,z7_7,h))/(int(g4,x,c,z8_4)+int(f8,x,z8_4,z8_5)+int(g5,x
,z8_5,y5_6)+int(f6,x,y5_6,z7_6)+int(f7,x,z7_6,z7_7)+int(g7,
x,z7_7,h));
      ws=round(double(ws));
   elseif f7>=f8 & f8<=(y5_6-e)/(15-e)

ws=(int(g4*x,x,c,z8_4)+int(f8*x,x,z8_4,z8_6)+int(g6*x,x,z8_
6,z7_6)+int(f7*x,x,z7_6,z7_7)+int(g7*x,x,z7_7,h))/(int(g4,x
,c,z8_4)+int(f8,x,z8_4,z8_6)+int(g6,x,z8_6,z7_6)+int(f7,x,z
7_6,z7_7)+int(g7,x,z7_7,h));
      ws=round(double(ws));
```

```
   elseif f8>f7 & f7<=(y5_6-e)/(15-e)

ws=(int(g4*x,x,c,z8_4)+int(f8*x,x,z8_4,z8_5)+int(g5*x,x,z8_
5,z7_5)+int(f7*x,x,z7_5,z7_7)+int(g7*x,x,z7_7,h))/(int(g4,x
,c,z8_4)+int(f8,x,z8_4,z8_5)+int(g5,x,z8_5,z7_5)+int(f7,x,z
7_5,z7_7)+int(g7,x,z7_7,h));
      ws=round(double(ws));
   end
end

if rtd>=800 & rtd<1000
   z10_2=f10*(5-a)+a;z10_3=-f10*(d-5)+d;z9_4=f9*(10-
c)+c;z9_5=-f9*(f-10)+f;
   z10_4=f10*(10-c)+c;z9_3=-f9*(d-5)+d;
   if f9>(y3_4-c)/(10-c) & f10>(y3_4-c)/(10-c)

ws=(int(g2*x,x,a,z10_2)+int(f10*x,x,z10_2,z10_3)+int(g3*x,x
,z10_3,y3_4)+int(g4*x,x,y3_4,z9_4)+int(f9*x,x,z9_4,z9_5)+in
t(g5*x,x,z9_5,f))/(int(g2,x,a,z10_2)+int(f10,x,z10_2,z10_3)
+int(g3,x,z10_3,y3_4)+int(g4,x,y3_4,z9_4)+int(f9,x,z9_4,z9_
5)+int(g5,x,z9_5,f));
      ws=round(double(ws));
   elseif f9>=f10 & f10<=(y3_4-c)/(10-c)

ws=(int(g2*x,x,a,z10_2)+int(f10*x,x,z10_2,z10_4)+int(g4*x,x
,z10_4,z9_4)+int(f9*x,x,z9_4,z9_5)+int(g5*x,x,z9_5,f))/(int
(g2,x,a,z10_2)+int(f10,x,z10_2,z10_4)+int(g4,x,z10_4,z9_4)+
int(f9,x,z9_4,z9_5)+int(g5,x,z9_5,f));
      ws=round(double(ws));
   elseif f10>f9 & f9<=(y3_4-c)/(10-c)

ws=(int(g2*x,x,a,z10_2)+int(f10*x,x,z10_2,z10_3)+int(g3*x,x
,z10_3,z9_3)+int(f9*x,x,z9_3,z9_5)+int(g5*x,x,z9_5,f))/(int
(g2,x,a,z10_2)+int(f10,x,z10_2,z10_3)+int(g3,x,z10_3,z9_3)+
int(f9,x,z9_3,z9_5)+int(g5,x,z9_5,f));
      ws=round(double(ws));
   end
end

if rtd>=1000 & rtd<=1200
   z12_1=-b*(f12-1);z11_2=f11*(5-a)+a;z11_3=-f11*(d-
5)+d;z11_1=-b*(f11-1);
   z12_2=f12*(5-a)+a;
   if f11>(y1_2-a)/(5-a) & f12>(y1_2-a)/(5-a)

ws=(int(f12*x,x,0,z12_1)+int(g1*x,x,z12_1,y1_2)+int(g2*x,x,
y1_2,z11_2)+int(f11*x,x,z11_2,z11_3)+int(g3*x,x,z11_3,d))/(
```

```
int(f12,x,0,z12_1)+int(g1,x,z12_1,y1_2)+int(g2,x,y1_2,z11_2
)+int(f11,x,z11_2,z11_3)+int(g3,x,z11_3,d));
      ws=round(double(ws));
   elseif f11<=f12 & f11<=(y1_2-a)/(5-a)

ws=(int(f12*x,x,0,z12_1)+int(g1*x,x,z12_1,z11_1)+int(f11*x,
x,z11_1,z11_3)+int(g3*x,x,z11_3,d))/(int(f12,x,0,z12_1)+int
(g1,x,z12_1,z11_1)+int(f11,x,z11_1,z11_3)+int(g3,x,z11_3,d)
);
      ws=round(double(ws));
   elseif f12<=f11 & f12<=(y1_2-a)/(5-a)

ws=(int(f12*x,x,0,z12_2)+int(g2*x,x,z12_2,z11_2)+int(f11*x,
x,z11_2,z11_3)+int(g3*x,x,z11_3,d))/(int(f12,x,0,z12_2)+int
(g2,x,z12_2,z11_2)+int(f11,x,z11_2,z11_3)+int(g3,x,z11_3,d)
);
      ws=round(double(ws));
   end
end
```

```
                    %Genetic Algorithm 2
global E choice q parameter
el_sel=[10 10 4 2];
mut=[20 15 8 4];
cross=[20 15 8 4];
choose;
parameter1=choice;
for r=2:60
   choose;
   parameter1(r,:)=choice;
   t=1;
   while t<r
         if isequal(parameter1(t,:),parameter1(r,:))==1
            choose;
            t=1;
         else
            t=t+1;
         end
           parameter1(r,:)=choice;
   end
end
parameter=parameter1;
for q=1:60;
   for s=1:3
      netsimandfuzzwin22;
```

```
      Eff_123(s)=E;
   end
   AveEff(q)=sum(Eff_123)/3;
   parameter1(q,13)=AveEff(q);
   parameter(q,13)=E;
end
minimum=find(parameter(:,13)==min(parameter(:,13)));
% Cycles
disp(['Generation 1 done']);
for v=1:4
   el_selection=[];
   nat_selection=[];
   crossover=[];
   mutation=[];

%   Select sel(v) strings from parameter
%   Elitist promotion (selection)
maximum=[];
while size(maximum,2)<el_sel(v)
   w=[];
   w=find(parameter(:,13)==max(parameter(:,13)));
   maximum=[maximum w'];
   for ind=1:size(w,1)
      parameter(w(ind),13)=0;
   end
end
ind=0;
for ind=1:el_sel(v)
   el_selection(ind,:)=parameter(maximum(ind),:);
end
if size(maximum,2)>el_sel(v)
   for ind=1:size(maximum,2)-el_sel(v)

nat_selection(ind,:)=parameter(maximum(ind+el_sel(v)),:);
   end
end

%Choose more strings to complete nat_selection
if cross(v)+mut(v)-size(nat_selection,1)>0
maximum=[];
w=[];
while size(maximum,1)<cross(v)+mut(v)-size(nat_selection,1)
   w=find(parameter(:,13)==max(parameter(:,13)));
   maximum=[maximum w'];
   for ind=1:size(w,1)
      parameter(w(ind),13)=0;
   end
```

```
end
end
ind=0;
for ind=1:size(maximum,2)

nat_selection(size(nat_selection,1)+ind,:)=parameter(maximu
m(ind),:);
end
if size(nat_selection,1)>cross(v)+mut(v)
   for ind=1:size(parameter1prime1,1)-(cross(v)+mut(v))
      nat_selection(cross(v)+mut(v)+ind,:)=[];
   end
end
parameter=[];

%  Apply crossover and mutation to nat_selection

%crossover operator

while size(crossover,1)<cross(v)
   int1=round(1+(size(nat_selection,1)-1)*rand(1,1));
   int2=round(1+(size(nat_selection,1)-1)*rand(1,1));
   while isequal(int2,int1)==1
      int2=round(1+(size(nat_selection,1)-1)*rand(1,1));
   end
   crosspnt=round(2+9*rand(1,1));
   crossed1=nat_selection(int1,:);
   crossed2=nat_selection(int2,:);
   temp=crossed1(1,crosspt:13);
   crossed1(1,crosspt:13)=crossed2(1,crosspt:13);
   crossed2(1,crosspt:13)=temp;
   vect1=find(parameter1(:,1:12)==crossed1(1,1:12));
   if vect1==[]
      crossover(size(crossover,1)+1,:)=crossed1(1,:);
      parameter1(size(parameter1,1)+1,:)=crossed1(1,:);
   end
   vect2=find(parameter1(:,1:12)==crossed2(1,1:12));
   if vect2==[]
      crossover(size(crossover,1)+1,:)=crossed2(1,:);
      parameter1(size(parameter1,1)+1,:)=crossed2(1,:);
   end
end
if size(crossover,1)>cross(v)
   for ind=1:size(cross1,1)-cross(v)
      crossover(50+ind,:)=[];
   end
end
```

```
%  mutation operator
while size(mutate,1)<mut(v)
   int3=round(1+(size(nat_selection,1)-1)*rand(1,1));
   mutpt=round(1+11*rand(1,1));
   mutate=nat_selection(int3,:);
   if rem(mutpt+1,2)==0
      min1=(floor(mutpt/2))*5;
      max1=min1+2;
   elseif rem(mutpt,2)==0
      min1=floor((mutpt-1)/2)*5+3;
      max1=min1+2;
   end
   if mutate(mutpt)==min1
      mutate(mutpt)=min1+1;
   elseif mutate(mutpt)==max1
      mutate(mutpt)=max1-1;
   else
      mutval=round(rand(1,1));
      if mutval==0
         mutate(mutpt)=min1;
      else
         mutate(mutpt)=max1;
      end
   end
   vect3=find(parameter1(:,1:12)==mutate);
   if vect3==[]
      mutation(size(mutation,1)+1,:)=mutate;
      parameter1(size(parameter1,1)+1,:)=mutate;
   end
end

%  Make new parameter

parameter=el_selection;
ind=0;
for ind=1:size(crossover,1)
   parameter(size(parameter,1)+1,:)=crossover(ind,:);
end
ind=0;
for ind=1:size(mutation,1)
   parameter(size(parameter,1)+1,:)=mutation(ind,:);
end

%  Run controllers, record fitness values

for q=1:size(parameter,1)
```

```
    for s=1:3
       netsimandfuzzwin22;
       Eff_123(s)=E;
    end
    AveEff(q)=sum(Eff_123)/3;
    parameter(q,13)=AveEff(q);
    parameter(q,13)=E;
end
disp([num2str(v)]);
end

maximum=[];
maximum=find(parameter(:,13)==max(parameter(:,13)));
disp(['optimal controller parameters
are'mat2str(parameter(maximum(1),:))]);
```