MODEL BASED FUZZY LOGIC CONTROL

by

John Lygeros, Datta N. Godbole, and Charles P. Coleman

Memorandum No. UCB/ERL M94/60

26 August 1994

MODEL BASED FUZZY LOGIC CONTROL

by

John Lygeros, Datta N. Godbole, and Charles P. Coleman

Memorandum No. UCB/ERL M94/60

26 August 1994

MODEL BASED FUZZY LOGIC CONTROL

by

John Lygeros, Datta N. Godbole, and Charles P. Coleman

Memorandum No. UCB/ERL M94/60

26 August 1994

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

MODEL BASED FUZZY LOGIC CONTROL

by

John Lygeros, Datta N. Godbole, and Charles P. Coleman

Memorandum No. UCB/ERL M94/60

26 August 1994

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Model Based Fuzzy Logic Control

John Lygeros, Datta N. Godbole and Charles P. Coleman*

Intelligent Machines and Robotics Laboratory
University of California, Berkeley
Berkeley, CA 94720
lygeros, godbole, charles @ robotics.eecs.berkeley.edu

Abstract

In this paper, we introduce a general formalism for generating a fuzzy plant model. We provide a description of fuzzification, inference rules, and defuzzification in terms of mappings from the set of real numbers to the fuzzy set domain. We demonstrate the use of this formalism by developing a general fuzzy logic rule based plant model for a stable, discrete-time, linear, time-invariant system, with rational coefficients. We prove that the fuzzy plant model exactly duplicates the behavior of the discrete-time linear system. A worked example and simulation results are also presented. Finally, we outline a method for control design using fuzzy rule base plant models. Possible application of this method for regulation and tracking is considered.

*Research supported in part by ARO under grant number DAAL03-91G-0171 and DAAL03-92G-0124.

1 Introduction

1.1 Motivation

Fuzzy Logic Controllers

In the twenty years that have passed since the publication of the landmark papers [1] and [3], there has been an explosion in the development of the theory and application of fuzzy logic to control systems [4]–[18]. In most cases the rule base of the fuzzy logic controller is specified by considering the desired closed loop behavior of the system. In such cases, the rule base model of the plant is not explicitly given (or needed). Assumptions about the rule base of the plant are, however, implicitly contained in the rule base of the controller.

A Methodology for Model Based Fuzzy Logic Control

An approach to the design of fuzzy logic control systems is to derive explicit rule base models of plants, and then to perform rule based control synthesis. Control synthesis consists of finding the appropriate input(s) to affect desired closed loop behavior.

This approach would allow, for a fixed plant, the development of many rule based controllers for different control objectives. It would circumvent having to specify totally new desired closed loop rule bases for each new control objective. This methodology has the advantage that plant modeling and control synthesis are all done in the rule base domain.

1.2 Outline of Paper

Fuzzy Plant Models

In Section 2.1 we introduce a formalism for generating a general fuzzy plant model. We provide a description of fuzzification, inference rules, and defuzzification in terms of mappings from the set of real numbers to the fuzzy set domain.

In Section 2.2 we demonstrate the use of this formalism and develop a general fuzzy logic rule based plant model of a stable, discrete-time, linear, time-invariant systems, with rational coefficients. We prove that the fuzzy model plant exactly duplicates the behavior of the discrete-time linear system. A worked example and simulation results are also presented. This effort is a small step towards our long term goal to investigate the modeling capabilities of fuzzy logic control, and to compare and contrast fuzzy logic and conventional control techniques.

Model Based Fuzzy Logic Control Synthesis

In Section 3 we outline a method for control design using fuzzy rule base plant models of the form given in Section 2.1. We outline a method for regulation and tracking. We hope that this method will lead us to a control design methodology completely contained in the fuzzy rule base domain.

2 Modeling for Fuzzy Logic Controller design

Modeling is an integral part of control design as it provides a way of formally proving properties of the closed loop system. Of course the validity of the proof strongly depends on how accurately the model describes the actual system behavior. Though this is not an issue for systems where a model can be derived straight from the laws of physics, for most systems the inaccuracy of the model can cause problems in the controller performance. To overcome this drawback techniques such as adaptive and robust control were developed.

Most conventional control design methods are based on some form of explicit plant models (e.g. we need a transfer function model to design a PID controller or a state space model to design Kalman filters etc). Fuzzy logic control on the other hand has mostly been based on implicit plant models. The reason for this is probably that most fuzzy controllers are designed using the engineering intuition of the designer and/or expert advice, both of which draw on an implicit model of the plant behavior derived from common sense and experience. The problem with such an approach is that it does not lend itself to formal closed loop performance analysis.

Our approach uses an explicit model for design of fuzzy logic controllers. To simplify the problem of model analysis and controller synthesis, we use a rule based fuzzy model of the plant, not unlike the models currently used for fuzzy feedback control.

2.1 Model Elements

2.1.1 Relevant Quantities

As with almost all modeling problems the first step is to identify all the relevant quantities whose interaction the model will specify. In most fuzzy logic designs that appear in the literature the only relevant quantities considered are the inputs and outputs of the system. Our formalism will be slightly different and allow in addition internal state variables. The reason behind this is that such a "state space" description is more general and will hopefully allow us to tackle problems such as controllability and observability which are difficult to address from an input-output point of view.

We will assume that all input, output and state variables of the real plant take values in $\Re$ and, as usual, we will denote the state by $x \in \Re^n$ the input by $u \in \Re^m$ and the output by $y \in \Re^l$.

2.1.2 Fuzzification

Once all the relevant quantities have been identified a formalism for expressing them in terms of fuzzy sets is needed. This is done by a process known as *fuzzification.* First the designer chooses a (finite) number n_i of *fuzzy sets* for each quantity w_i (which in this case can denote an input, an output or a state variable). Secondly he assigns labels to them. It is customary to use linguistic labels that convey some characteristic of the particular set (e.g. negative, small, hot etc.).

Associated with each fuzzy set is a *membership function.* These functions can be thought of as maps F_j^i from the real numbers (where the quantities w_i live) to the unit interval $[0, 1]$. They determine to what extend the label associated with the fuzzy set j characterizes

the current value of the quantity w_i. Small values of $F_j^i(w_i)$ indicate that w_i has little to do with set j while large values indicate that j is a good description of w_i. Under this formalism fuzzification of the quantity w_i becomes equivalent to applying all the maps $F_j^i, j = 1, \ldots, n_i$ to it. We can write this symbolically as:

$$\begin{array}{rcl} F^i : \Re & \longrightarrow & I^{n_i} \equiv [0,1] \times \ldots \times [0,1] \\ w_i & \longmapsto & \begin{bmatrix} p_1^i \\ \vdots \\ p_{n_i}^i \end{bmatrix} = \begin{bmatrix} F_1^i(w_i) \\ \vdots \\ F_{n_i}^i(w_i) \end{bmatrix} \end{array} \tag{1}$$

Hence a *fuzzy vector* is associated to each quantity. It should be noted that, unlike probability distributions, fuzzy vectors need not satisfy the property $\sum_{j=1}^{n_i} p_j^i = 1$. It is not even necessary for the range of all F_j^j to be a subset of the unit interval. We assume this is the case here as, without loss of generality, we can always scale all the functions by a fixed number so that all the ranges lie in $[0,1]$.

2.1.3 Firing or Inference Rules

The firing rules form the heart of a fuzzy model. They code the dynamics as they determine the fuzzy values of the outputs (and the states if a state-space description is used) in the immediate future given the current fuzzy values of the inputs (and the states). As in Section 2.1.1 we assume that the system involves n states $x_1, \ldots, x_n$, m inputs $u_1, \ldots, u_m$ and l outputs $y_1, \ldots, y_l$. Assume that all the inputs and state variables pass through a fuzzification procedure and let $x_i^F \in I^{a_i}$ and $u_j^F \in I^{b_j}$ denote the corresponding fuzzy values. Also assume that fuzzy sets are also chosen for the outputs $y_k^F \in I^{c_k}$. Then the firing rules can be viewed as mappings:

$$FR_i^x : I^{a_1} \times \ldots \times I^{a_n} \times I^{b_1} \times \ldots \times I^{b_m} \longrightarrow I^{d_i} \; i = 1, \ldots, n \tag{2}$$

$$FR_k^y : I^{a_1} \times \ldots \times I^{a_n} \times I^{b_1} \times \ldots \times I^{b_m} \longrightarrow I^{c_k} \; k = 1, \ldots, l \tag{3}$$

from fuzzy values of the inputs and states to fuzzy values of states and outputs. Note that the fuzzy description of the states coming out of the firing rules (element of I^{d_i}) need not be the same as the one going in (element of I^{a_i}), though it generally will be.

2.1.4 Defuzzification

The last part of a fuzzy model is the defuzzification procedure. It provides a way of producing information in the real domain given information in the fuzzy domain. The most direct way of thinking of defuzzification in this scenario is by a set of maps, one for each quantity that will get defuzzified.

$$DF_i : I^{d_i} \longrightarrow \Re \tag{4}$$

Here I^{d_i} stands for the fuzzy representation of the quantity in question, be it input, output or state. In this sense the defuzzification process can be thought of as the inverse of the fuzzification process defined in Section 2.1.2, even though mathematically speaking the maps F and DF need not be (and typically will not be) inverses of one another. In fact neither of the maps needs to be invertible and the domains and ranges of DF and F need not coincide.

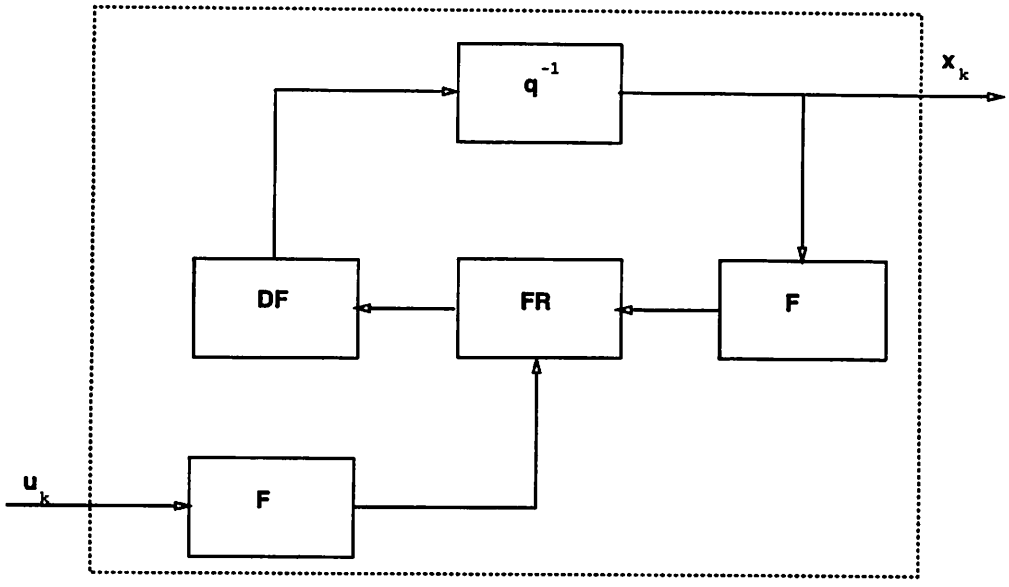

Figure 1: Block Diagram of a Discrete Time Fuzzy Model

Once all of the above elements (fuzzification, firing rules and defuzzification) have been specified the fuzzy model can be assembled as in figure 1. Fuzzy plant models such as this can be obtained in a number of different ways. The most common way (and one of the reasons that made fuzzy logic popular) is by polling the opinion of an expert or by intuitive knowledge of the designer about the plant dynamics. Usually this will result in a rule base and crude idea about the fuzzy sets that are needed. Further tuning of the membership functions and rules will then have to be carried out based on open loop experiments. Another way such a model can be obtained is by converting a conventional model to the fuzzy framework. This method has the advantage that it brings the fuzzy design closer to the conventional designs and therefore may lead to a formalism capable of analytical comparison between the two. This is the approach we will follow in this paper. Finally work has also been done in identifying fuzzy models directly from input-output data of the plant. It should be noted that, depending on the problem requirements, there may be a need for many iterations of the above techniques before a satisfactory model is obtained.

An interesting feature of the fuzzy model described above is its similarity to conventional discrete time models. This similarity becomes apparent when the block diagram of a fuzzy model (figure 1) is compared to that of a conventional, linear, discrete time model (figure 2). We will try to exploit this similarity in the next section, where we attempt to produce a fuzzy model that will duplicate the performance of a conventional linear model.

2.2 Fuzzy Models for Linear Systems

Consider the standard representation for a discrete time linear system with the additional assumption that all the matrices have rational entries:

$$x^{k+1} = \hat{A}x^k + \hat{B}u^k \tag{5}$$

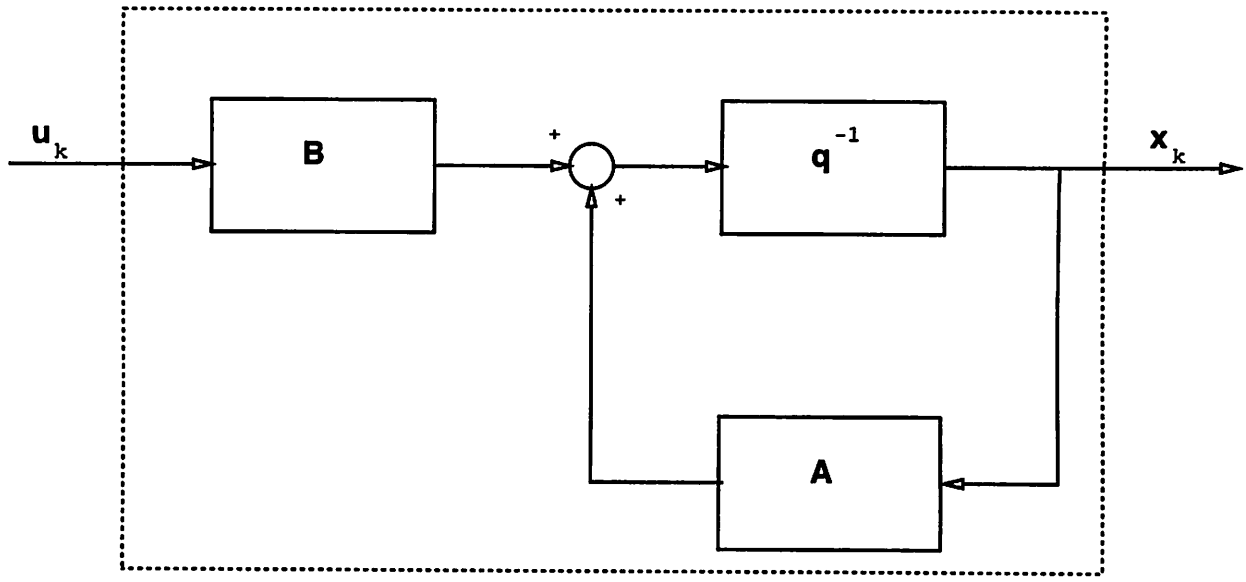

Figure 2: Block Diagram of a Conventional Discrete Time Linear Model

where

$$\begin{aligned} x : \mathcal{Z}^+ &\longrightarrow \Re^n \\ k &\longmapsto x^k = \begin{bmatrix} x_1^k \\ \vdots \\ x_n^k \end{bmatrix} \\ u : \mathcal{Z}^+ &\longrightarrow \Re^m \\ k &\longmapsto u^k = \begin{bmatrix} u_1^k \\ \vdots \\ u_m^k \end{bmatrix} \end{aligned}$$

$$\begin{aligned} \hat{A} &\in \mathcal{Q}^{n \times n} \\ \hat{B} &\in \mathcal{Q}^{n \times m} \end{aligned}$$

As all the entries of A and B are rational numbers, there exists a positive integer $N \in \mathcal{N}$ such that:

$$x_{k+1} = \frac{1}{N}(Ax_k + Bu_k) \tag{6}$$

where

$$A = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \vdots & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \in \mathcal{Z}^{n \times n}$$

$$B = \begin{bmatrix} b_{11} & \dots & b_{1n} \\ \vdots & \vdots & \vdots \\ b_{n1} & \dots & b_{nn} \end{bmatrix} \in \mathcal{Z}^{n \times n}$$

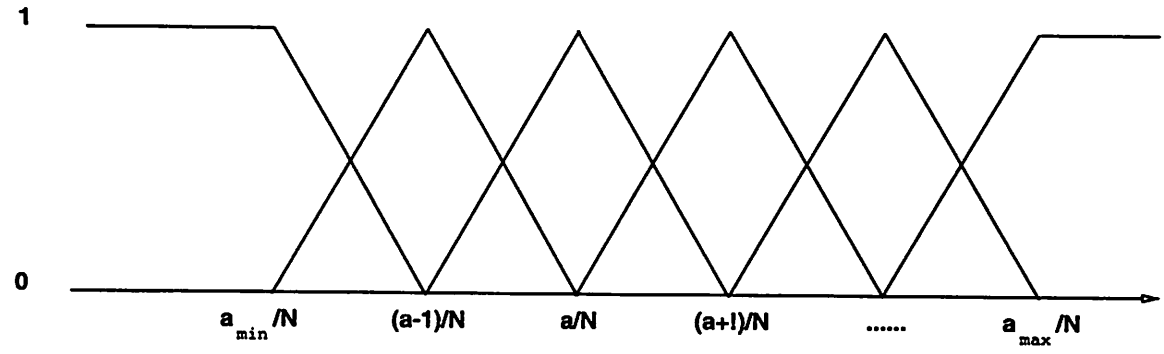

Figure 3: Membership Functions

In order to get an exact fuzzy model with a finite number of membership functions we will restrict x and u to bounded sets. More formally we assume that:

$$x_k \in \mathcal{U} \subset \Re^n \tag{7}$$
$$u_k \in \mathcal{V} \subset \Re^m \tag{8}$$

where $\mathcal{U}$ and $\mathcal{V}$ are open and their closures $\bar{\mathcal{U}}$ and $\bar{\mathcal{V}}$ are compact in the standard metric topologies. It should be noted that these assumptions are rather restrictive as they essentially exclude all unstable linear systems. However physical considerations impose constraints that lead to saturation of the states and inputs of most real systems. Hence in most cases where a linear model is applicable its validity is restricted to a bounded set anyway; outside this set saturation (a nonlinear effect) sets in and dominates the system behavior. We will now specify the elements described in the previous section for a fuzzy model that is capable of duplicating the behavior of 5 under assumptions 7 and 8. Whenever assumptions 7 and 8 are violated our fuzzy model will display the saturation behavior discussed above.

2.2.1 Fuzzification

For each of the states and inputs we will chose a collection of fuzzy sets with membership functions of the form shown in figure 3. In order to obtain consistent firing rules we would like to be able to carry out some form of algebraic calculations using the fuzzy sets. With this in mind we assign indices from the set $\mathcal{Z}$ of integers to the fuzzy sets in a natural way, i.e. α is assigned to the set that is centered at α/N. We will only be allowing a finite number of membership functions for each quantity. Thus the fuzzy set indices for x_i will belong to a set $\{a_{i_{min}}, \ldots, a_{i_{max}}\}$ and the indices for u_j will belong to a set $\{b_{j_{min}}, \ldots, b_{j_{max}}\}$. To make sure that all values of interest are covered we choose these sets so that:

$$\begin{aligned} \mathcal{U} &\subset \left(\frac{a_{1_{min}}}{N}, \frac{a_{1_{max}}}{N}\right) \times \ldots \times \left(\frac{a_{n_{min}}}{N}, \frac{a_{n_{max}}}{N}\right) \\ \mathcal{V} &\subset \left(\frac{b_{1_{min}}}{N}, \frac{b_{1_{max}}}{N}\right) \times \ldots \times \left(\frac{b_{m_{min}}}{N}, \frac{b_{m_{max}}}{N}\right) \end{aligned} \tag{9}$$

Note that this is only a necessary condition on the values of $a_{i_{min}}, a_{i_{max}}, b_{j_{min}}$ and $b_{j_{max}}$. A sufficient condition for exact performance duplication will be given in Section 2.2.2. Once the

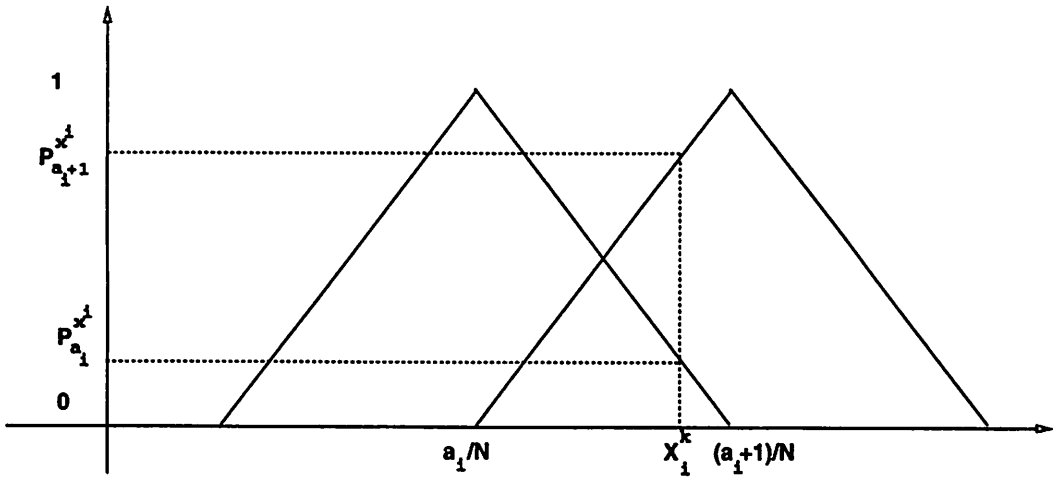

Figure 4: Determining the Output of Fuzzification

membership functions have been chosen we can calculate the fuzzy vector corresponding to a given value of a quantity. Let $x_i^F = [p^{x^i}_{a_{i_{min}}} \dots p^{x^i}_{a_{i_{max}}}]^T$ denote the fuzzy vector associated with the state x_i^k. Then intuition and a little geometry (figure 4) reveal that if:

$$x_i^k \in \left[\frac{\alpha_i}{N}, \frac{\alpha_i + 1}{N}\right)$$

then:

$$p_l^{x^i} = \begin{cases} \alpha_i + 1 - Nx_i^k & l = \alpha_i \\ Nx_i^k - \alpha_i & l = \alpha_i + 1 \\ 0 & \text{else} \end{cases}$$

A similar procedure is also used to fuzzify the inputs u_j^k. The calculations are essentially the same. If:

$$u_j^k \in \left[\frac{\beta_j}{N}, \frac{\beta_j + 1}{N}\right)$$

then:

$$p_l^{u^j} = \begin{cases} \beta_j + 1 - Nu_j^k & l = \beta_j \\ Nu_j^k - \beta_j & l = \beta_j + 1 \\ 0 & \text{else} \end{cases}$$

The above description is essentially a definition of a collection of fuzzification maps, F_i^x and F_j^u for the states and inputs. There are two very important characteristics of these maps that play a crucial role on the development of the model:

1. The map from the real to the fuzzy value is injective provided that the quantities lie in the sets $\mathcal{U}$ and $\mathcal{V}$. Therefore no information is lost when we fuzzify, i.e. given the resulting fuzzy value we can extract the real value that produced it unambiguously. This property is undesirable strictly speaking as it defeats one of the main reasons behind all linguistic and abstract descriptions (such as fuzzy logic), namely that information is condensed by the abstraction. On the other hand the fuzzification process presented here *does* allow some abstraction. For example a controller can now be designed that

deals explicitly only with the fuzzy sets (e.g. as in Section 3). Moreover, the injectivity property is necessary if exact duplication of the linear system performance is required.

2. The choice of rules is such that we can carry out algebra using their indices the same way we do for integers. This property will be crucial in order to design consistent firing rules.

2.2.2 Firing Rules

Recall that by assumption the indices of the membership functions and the entries of both A and B are integers (elements of $\mathcal{Z}$). Hence, because of the fact that $\mathcal{Z}$ is closed under addition and multiplication, we can carry out algebraic calculations involving membership function indices, the entries of A and B and operations $+$ and $\times$ and regard the result as the index of a membership function of a state x_i^{k+1}, provided it lies in the set $\{a_{i_{min}}, \ldots, a_{i_{max}}\}$. Using this observation we can consistently define the firing rules that produce the fuzzy value of the i^{th} state at the next time instant as mappings FR_i^x. Let $q_i = a_{i_{max}} - a_{1_{min}}$ and $r_j = b_{j_{max}} - b_{j_{min}}$

$$FR_i^x : I^{q_1} \times \ldots \times I^{q_n} \times I^{r_1} \times \ldots \times I^{r_m} \longrightarrow I^{q_i} \tag{10}$$

We define the output of these maps to have zeros in all the entries, except the ones with indices:

$$\sum_{l=1}^{n} a_{il}\nu_l + \sum_{l=1}^{m} b_{il}\mu_l \tag{11}$$

where ν_l takes values in the set of membership function indices of x_l^k that have nonzero entries and μ_l takes values in the same sets for u_l^k. In our framework these sets contain only two elements each, namely $\nu_l \in \{\alpha_l, \alpha_l + 1\}$ and $\mu_l \in \{\beta_l, \beta_l + 1\}$ in the notation of the previous section. The values corresponding to the indices in 11 are:

$$(\prod_{l=1}^{n} p_{\nu_l}^{x^l})(\prod_{l=1}^{m} p_{\mu_l}^{u^l}) \tag{12}$$

If for some choices of ν_l and μ_l the sum in 11 is greater than $a_{i_{max}}$ or less than $a_{i_{min}}$ we will set FR_i^x equal to $a_{i_{max}}$ and $a_{i_{min}}$ respectively. However if we would like to exactly reproduce the behavior of the linear model this kind of saturation effect is undesirable. We will therefore choose $a_{i_{min}}, a_{i_{max}}$ to guarantee that, if assumption 7 holds, the fuzzy sets after the firing rules will not be influenced by saturation. This will give sufficient conditions on the values of $a_{i_{min}}$ and $a_{i_{max}}$ for exact performance duplication.

Recall that, by assumption, the set $\bar{\mathcal{U}}$, the closure of $\mathcal{U}$ is compact in the standard metric topology of $\Re^n$, hence we can find integers $n_{i_{min}}$ and $n_{i_{max}}$ such that:

$$\mathcal{U} \subset \left(\frac{n_{1_{min}}}{N}, \frac{n_{1_{max}}}{N}\right) \times \ldots \times \left(\frac{n_{n_{min}}}{N}, \frac{n_{n_{max}}}{N}\right) \tag{13}$$

Then the following can be shown:

Lemma 1 *Under assumptions 7 and 8, if we choose:*

$$a_{i_{min}} = Nn_{i_{min}} - \left(\sum_{l=1}^{n}|a_{il}| + \sum_{l=1}^{m}|b_{il}|\right) \tag{14}$$

$$a_{i_{max}} = Nn_{i_{max}} + \left(\sum_{l=1}^{n}|a_{il}| + \sum_{l=1}^{m}|b_{il}|\right) \tag{15}$$

we can guarantee that saturation will not be observed in the above fuzzy rules.

Proof: Clearly this choice satisfies the necessary condition 9, so we need not worry about that. Consider the single state autonomous system:

$$x^{k+1} = \frac{a}{N}x^k$$

where $a \in \mathcal{Z}$. A little algebra reveals that:

$$\frac{\alpha}{N} \leq x^k \leq \frac{\alpha+1}{N} \Rightarrow \begin{cases} ax^k - \frac{|a|}{N} \leq \frac{a\alpha}{N} \leq ax^k + \frac{|a|}{N} \\ ax^k - \frac{|a|}{N} \leq \frac{a(\alpha+1)}{N} \leq ax^k + \frac{|a|}{N} \end{cases}$$

or, in the previous notation:

$$ax^k - \frac{|a|}{N} \leq \frac{\nu_1}{N} \leq ax^k + \frac{|a|}{N} \tag{16}$$

Assume that a relation like 16 is valid for each state x_i^k of a $n-1$ state m input system, that is:

$$\begin{aligned} & \sum_{l=1}^{n-1} a_{il}x_l^k + \sum_{l=1}^{m} b_{il}u_l^k - \frac{1}{N}\left(\sum_{l=1}^{n-1}|a_{il}| + \sum_{l=1}^{m}|b_{il}|\right) \\ \leq & \frac{1}{N}\left(\sum_{l=1}^{n-1} a_{il}\nu_l + \sum_{l=1}^{m} b_{il}\mu_l\right) \\ \leq & \sum_{l=1}^{n-1} a_{il}x_l^k + \sum_{l=1}^{m} b_{il}u_l^k + \frac{1}{N}\left(\sum_{l=1}^{n-1}|a_{il}| + \sum_{l=1}^{m}|b_{il}|\right) \end{aligned} \tag{17}$$

Then we claim it will also have to be valid for an arbitrary n state m input system. Indeed, as for the single state system:

$$\frac{\alpha_n}{N} \leq x_n^k \leq \frac{\alpha_n+1}{N} \Rightarrow \begin{cases} a_{in}x_n^k - \frac{|a_{in}|}{N} \leq \frac{a_{in}\alpha_n}{N} \leq a_{in}x_n^k + \frac{|a_{in}|}{N} \\ a_{in}x_n^k - \frac{|a_{in}|}{N} \leq \frac{a_{in}(\alpha_n+1)}{N} \leq ax_n^k + \frac{|a_n|}{N} \end{cases}$$

Then adding the last two equations to equation 17 one at a time we obtain:

$$\begin{aligned} & \sum_{l=1}^{n} a_{il}x_l^k + \sum_{l=1}^{m} b_{il}u_l^k - \frac{1}{N}\left(\sum_{l=1}^{n}|a_{il}| + \sum_{l=1}^{m}|b_{il}|\right) \\ \leq & \frac{1}{N}\left(\sum_{l=1}^{n} a_{il}\nu_l + \sum_{l=1}^{m} b_{il}\mu_l\right) \\ \leq & \sum_{l=1}^{n} a_{il}x_l^k + \sum_{l=1}^{m} b_{il}u_l^k + \frac{1}{N}\left(\sum_{l=1}^{n}|a_{il}| + \sum_{l=1}^{m}|b_{il}|\right) \end{aligned} \tag{18}$$

A similar step can be carried out for the transition from a system of n states and $m-1$ inputs to a system of n states and m inputs. Hence, by induction, the bounds of 18 should hold for a general n state m input system, for all $n, m \in \mathcal{N}$. Substituting the value of x_i^{k+1} for the linear system trajectory:

$$\begin{aligned} & Nx_{i+1}^k - \frac{1}{N}\left(\sum_{l=1}^{n}|a_{il}| + \sum_{l=1}^{m}|b_{il}|\right) \\ \leq & \frac{1}{N}\left(\sum_{l=1}^{n} a_{il}\nu_l + \sum_{l=1}^{m} b_{il}\mu_l\right) \\ \leq & Nx_{i+1}^k + \frac{1}{N}\left(\sum_{l=1}^{n}|a_{il}| + \sum_{l=1}^{m}|b_{il}|\right) \end{aligned} \tag{19}$$

Given the bounds 13 on $\tilde{U}$ the above can be written as:

$$Nn_{i_{min}} - \left(\sum_{l=1}^{n}|a_{il}| + \sum_{l=1}^{m}|b_{il}|\right) \leq \sum_{l=1}^{n} a_{il}\nu_l + \sum_{l=1}^{m} b_{il}\mu_l \leq Nn_{i_{max}} + \left(\sum_{l=1}^{n}|a_{il}| + \sum_{l=1}^{m}|b_{il}|\right) \tag{20}$$

Hence choosing $a_{i_{min}}$ and $a_{i_{max}}$ as suggested in the statement of the lemma will guarantee that all the rules fired will be in the set $\{a_{i_{min}}, \ldots, a_{i_{max}}\}$ and therefore no saturation will take place.

The above expressions may seem complicated at the moment, but they will hopefully become clear when we illustrate them for a single state system in the next section and a three state example later on. It should be noted here that the number of nonzero entries in the fuzzy vector comming out of the firing rules does not have to be less than or equal to two, as is the case for the fuzzy vectors coming in. In fact there can be as many as 2^{n+m} nonzero entries. This suggests that the output of the firing rule FR_i^x is not a fuzzy vector that can be obtained by passing a real number through the fuzzifying function F_i^x and, in order to extract real information from it, a defuzzification process, like the one introduced in the next section, is needed. This is an illustration of the point raised in Section 2.1.4 that, strictly speaking, defuzzification is not the inverse of fuzzification.

2.2.3 Defuzzification

We will define a defuzzification process for each state x_k^i as a map DF_i^x that takes the fuzzy value of the state $x_i^F = [p_{a_{i_{min}}}^{x_i} \ldots p_{a_{i_{max}}}^{x_i}]^T$ and assigns to it the real number:

$$DF_i^x(x_i^F) = \frac{1}{N^2}\sum_{l=a_{i_{min}}}^{a_{i_{max}}} l p_l^{x_i} \tag{21}$$

2.2.4 Model Performance

Claim 1 *The fuzzy model described in Sections 2.2.1, 2.2.2 and 2.2.3 will exactly duplicate any trajectory generated by the discrete time linear system 5 provided that:*

1. *The input u^k remains in the set $\mathcal{V}$ for all k*

2. *The state x^k remains in the set $\mathcal{U}$ for all k*

3. *Both models start at the same initial condition*

Recall that Lemma 1 guarantees that, under assumption 2 above, saturation of the rules fired at each step can not occur. Hence this question will not come into the proof. The proof of this claim will hopefully also help clarify the fuzzy model formalism. We will first introduce a Lemma, whose proof follows the same outline as the proof of the claim.

Lemma 2 *The sum of all the entries of the fuzzy vector produced by passing values of the states and inputs fuzzified as described in 2.2.1 through the firing rules described in 2.2.2 is always equal to 1, i.e.:*

$$\sum_{\nu_l,\mu_l} (\prod_{l=1}^{n} p_{\nu_l}^{x^l})(\prod_{l=1}^{m} p_{\mu_l}^{u^l}) = 1 \tag{22}$$

where $\nu_l \in \{\alpha_l, \alpha_l + 1\}$ and $\mu_l \in \{\beta_l, \beta_l + 1\}$.

Proof: (of Lemma) The proof will be by induction on the dimensions n and m of the state and the input.

- if $n = m = 1$ equation 22 becomes:

$$\begin{aligned}
\sum_{\nu_l,\mu_l} (\prod_{l=1}^{1} p_{\nu_1}^{x^1})(\prod_{l=1}^{m} p_{\mu_1}^{u^1}) &= p_{\alpha_1}^{x^1} p_{\beta_1}^{u^1} + p_{\alpha_1+1}^{x^1} p_{\beta_1}^{u^1} + p_{\alpha_1}^{x^1} p_{\beta_1+1}^{u^1} + p_{\alpha_1+1}^{x^1} p_{\beta_1+1}^{u^1} \\
&= p_{\alpha_1}^{x^1}(p_{\beta_1}^{u^1} + p_{\beta_1+1}^{u^1}) + p_{\alpha_1+1}^{x^1}(p_{\beta_1}^{u^1} + p_{\beta_1+1}^{u^1}) \\
&= (p_{\alpha_1}^{x^1} + p_{\alpha_1+1}^{x^1})(p_{\beta_1}^{u^1} + p_{\beta_1+1}^{u^1}) \\
&= 1
\end{aligned}$$

- Assume that the Lemma holds for $n-1$ states and m inputs. Then it will also have to hold for n states and m inputs, as:

$$\begin{aligned}
\sum_{\nu_l,\mu_l} (\prod_{l=1}^{n} p_{\nu_l}^{x^l})(\prod_{l=1}^{m} p_{\mu_l}^{u^l}) &= \sum_{\nu_n} \left(\sum_{\nu_l,\mu_l} (\prod_{l=1}^{n-1} p_{\nu_l}^{x^l})(\prod_{l=1}^{m} p_{\mu_l}^{u^l}) \right) p_{\nu_n}^{x^n} \\
&= \left(\sum_{\nu_l,\mu_l} (\prod_{l=1}^{n-1} p_{\nu_l}^{x^l})(\prod_{l=1}^{m} p_{\mu_l}^{u^l}) \right) (\sum_{\nu_n} p_{\nu_n}^{x^n}) \\
&= \left(\sum_{\nu_l,\mu_l} (\prod_{l=1}^{n-1} p_{\nu_l}^{x^l})(\prod_{l=1}^{m} p_{\mu_l}^{u^l}) \right) (p_{\alpha_n}^{x^n} + p_{\alpha_n+1}^{x^n}) \\
&= \sum_{\nu_l,\mu_l} (\prod_{l=1}^{n-1} p_{\nu_l}^{x^l})(\prod_{l=1}^{m} p_{\mu_l}^{u^l}) \\
&= 1
\end{aligned}$$

  by assumption.

- It is easy to show in exactly the same way that if the Lemma holds for n states and $m-1$ inputs it will also hold for n states and m inputs.

- By induction the Lemma holds for all integers n and m, i.e. for any number of inputs and states.

Proof: (of Claim) The proof will again be by induction on the number n of states and the number m of inputs.

- If $n = m = 1$, the system becomes:

$$x^{k+1} = \frac{1}{N}(ax^k + bu^k) \in \Re$$

Given a pair of real values for x^k and u^k we apply the fuzzy model discussed above.
1. Fuzzification: Given the real value of the state at time k, x^k we can derive the corresponding fuzzy vector using the fuzzification procedure of 2.2.1. All the entries will be zero except two, say α and $\alpha + 1$, that will be equal to $p^x_\alpha = \alpha + 1 - Nx^k$ and $p^x_{\alpha+1} = Nx^k - \alpha$ respectively. Similarly the fuzzy vector corresponding to u^k will contain only two non zero entries, say $p^u_\beta = \beta + 1 - Nu^k$ and $p^u_{\beta+1} = Nu^k - \beta$.

2. Firing Rules: After passing these two vectors through the firing rules described in Section 2.2.2 a new fuzzy vector will be obtained. It will have at most four non zero entries:

$$\begin{aligned} a\alpha + b\beta &\rightsquigarrow p^x_\alpha p^u_\beta \\ a(\alpha+1) + b\beta &\rightsquigarrow p^x_{\alpha+1} p^u_\beta \\ a\alpha + b(\beta+1) &\rightsquigarrow p^x_\alpha p^u_{\beta+1} \\ a(\alpha+1) + b(\beta+1) &\rightsquigarrow p^x_{\alpha+1} p^u_{\beta+1} \end{aligned}$$

3. Defuzzification:

$$\begin{aligned} x^{k+1} &= \frac{1}{N^2}\left[(a\alpha + b\beta)p^x_\alpha p^u_\beta + (a(\alpha+1) + b\beta)p^x_{\alpha+1}p^u_\beta \right. \\ &+ \left. (a\alpha + b(\beta+1))p^x_\alpha p^u_{\beta+1} + (a(\alpha+1) + b(\beta+1))p^x_{\alpha+1}p^u_{\beta+1}\right] \\ &= \frac{1}{N^2}\left[(a\alpha + b\beta)(p^x_\alpha + p^x_{\alpha+1})(p^u_\beta + p^u_{\beta+1}) + ap^x_{\alpha+1}(p^u_\beta + p^u_{\beta+1}) + bp^u_{\beta+1}(p^x_\alpha + p^x_{\alpha+1})\right] \\ &= \frac{1}{N^2}\left[a(\alpha + p^x_{\alpha+1}) + b(\beta + p^u_{\beta+1})\right] \\ &= \frac{1}{N^2}\left[a(\alpha + Nx^k - \alpha) + b(\beta + Nu^k - \beta)\right] \\ &= \frac{1}{N}(ax^k + bu^k) \end{aligned}$$

So the claim indeed holds for the $n = m = 1$ case.

- Assume that the claim holds for systems with $n - 1$ states and m inputs. We will then show that it also has to hold for for systems with n states and m inputs. Consider what happens to the i^{th} state at time $k + 1$. Under the conventional discrete time model:

$$x^{k+1}_i = \frac{1}{N}\left(\sum_{l=1}^{n} a_{il}x^k_l + \sum_{l=1}^{m} b_{il}u^k_l\right)$$

Fuzzify as before and consider the output of the firing rules. The nonzero entries of the fuzzy vector corresponding to x_i^{k+1} will be:

$$\sum_{l=1}^{n-1} a_{il}\nu_l + \sum_{l=1}^{m} b_{il}\mu_l + a_{in}\alpha_n \;\rightsquigarrow\; \prod_{l=1}^{n-1} p_{\nu_l}^{x^l} \prod_{l=1}^{m} p_{\mu_l}^{u^l} p_{\alpha_n}^{x^n}$$

$$\sum_{l=1}^{n-1} a_{il}\nu_l + \sum_{l=1}^{m} b_{il}\mu_l + a_{in}(\alpha_n+1) \;\rightsquigarrow\; \prod_{l=1}^{n-1} p_{\nu_l}^{x^l} \prod_{l=1}^{m} p_{\mu_l}^{u^l} p_{\alpha_n+1}^{x^n}$$

Passing this fuzzy vector through the defuzzification procedure we obtain:

$$\begin{aligned}
x_i^{k+1} &= \frac{1}{N^2} \sum_{\nu_l,\mu_l} \left[\left(\sum_{l=1}^{n-1} a_{il}\nu_l + \sum_{l=1}^{m} b_{il}\mu_l + a_{in}\alpha_n \right) \prod_{l=1}^{n-1} p_{\nu_l}^{x^l} \prod_{l=1}^{m} p_{\mu_l}^{u^l} p_{\alpha_n}^{x^n} \right. \\
&+ \left. \left(\sum_{l=1}^{n-1} a_{il}\nu_l + \sum_{l=1}^{m} b_{il}\mu_l + a_{in}(\alpha_n+1) \right) \prod_{l=1}^{n-1} p_{\nu_l}^{x^l} \prod_{l=1}^{m} p_{\mu_l}^{u^l} p_{\alpha_n+1}^{x^n} \right] \\
&= \frac{1}{N^2} \sum_{\nu_l,\mu_l} \left[\left(\sum_{l=1}^{n-1} a_{il}\nu_l + \sum_{l=1}^{m} b_{il}\mu_l \right) \prod_{l=1}^{n-1} p_{\nu_l}^{x^l} \prod_{l=1}^{m} p_{\mu_l}^{u^l} (p_{\alpha_n}^{x^n} + p_{\alpha_n+1}^{x^n}) \right] \\
&+ \frac{1}{N^2} \left[\sum_{\nu_l,\mu_l} \prod_{l=1}^{n-1} p_{\nu_l}^{x^l} \prod_{l=1}^{m} p_{\mu_l}^{u^l} \right] a_{in}(\alpha_n p_{\alpha_n}^{x^n} + (\alpha_n+1) p_{\alpha_n+1}^{x^n})
\end{aligned}$$

Applying Lemma 2 the above expression simplifies to:

$$\begin{aligned}
x_i^{k+1} &= \frac{1}{N^2} \sum_{\nu_l,\mu_l} \left[\left(\sum_{l=1}^{n-1} a_{il}\nu_l + \sum_{l=1}^{m} b_{il}\mu_l \right) \prod_{l=1}^{n-1} p_{\nu_l}^{x^l} \prod_{l=1}^{m} p_{\mu_l}^{u^l} \right] \\
&+ \frac{1}{N^2} a_{in}(\alpha_n p_{\alpha_n}^{x^n} + (\alpha_n+1) p_{\alpha_n+1}^{x^n}) \\
&= \frac{1}{N^2} \sum_{\nu_l,\mu_l} \left[\left(\sum_{l=1}^{n-1} a_{il}\nu_l + \sum_{l=1}^{m} b_{il}\mu_l \right) \prod_{l=1}^{n-1} p_{\nu_l}^{x^l} \prod_{l=1}^{m} p_{\mu_l}^{u^l} \right] \\
&+ \frac{1}{N^2} a_{in}(\alpha_n (p_{\alpha_n}^{x^n} + p_{\alpha_n+1}^{x^n}) + p_{\alpha_n+1}^{x^n})
\end{aligned}$$

Under the assumption that the claim holds for systems with $n-1$ states and m inputs and substituting the expressions for $p_{\alpha_n+1}^{x^n}$ we obtain:

$$\begin{aligned}
x_i^{k+1} &= \frac{1}{N} \left(\sum_{l=1}^{n-1} a_{il} x_l^k + \sum_{l=1}^{m} b_{il} u_l^k \right) + \frac{1}{N^2} a_{in}(\alpha_n + N x_n^k - \alpha_n) \\
&= \frac{1}{N} \left(\sum_{l=1}^{n-1} a_{il} x_l^k + \sum_{l=1}^{m} b_{il} u_l^k \right) + \frac{1}{N} a_{in} x_n^k
\end{aligned}$$

Hence the expression obtained for x_i^{k+1} from the fuzzy model is identical to the one obtained from the conventional discrete time system.

- The above proof can be repeated to show that if the claim holds for systems with n states and $m-1$ inputs it will also have to hold for systems with n states and m inputs.
- By induction the Claim holds for all integers n and m, i.e. for any number of inputs and states.

2.3 Example

This technique for transforming conventional models to fuzzy was applied to the following discrete time linear system:

$$\begin{aligned} x^{k+1} &= \frac{1}{10}\begin{bmatrix} -9 & 10 & 5 \\ 0 & 4 & 14 \\ 0 & -6 & 4 \end{bmatrix} x^k + \frac{1}{10}\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u^k \\ y^k &= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} x^k \end{aligned} \tag{23}$$

The above system has one eigenvalue inside the unit circle ($\{-0.9\}$) and a pair on it ($\{0.4 + 0.916i, 0.4 - 0.916i\}$ hence it is stable. We will assume that all states and the input will lie in the interval $[-0.5, 0.5]$, i.e.

$$\begin{aligned} \mathcal{U} &= [-0.5, 0.5] \times [-0.5, 0.5] \times [-0.5, 0.5] \\ \mathcal{V} &= [-0.5, 0.5] \end{aligned}$$

In accordance to the procedure of Section 2.2.1 we partition all states and inputs to fuzzy sets of the form shown in figure 3. Lemma 1 indicates that for exact tracking of the above system one needs up to 148 such sets for x_1, 136 sets for x_2, 120 sets for x_3 and 10 sets for u. Recall that the first and last membership functions will not be triangular but extend to infinity. This process will lead to quite a few inference rules in the sense of conventional fuzzy control. Fortunately there is an algebraic way to code these rules and we need not state them explicitly.

If each x_i^k excites the two fuzzy sets a_i and $a_i + 1$ by $p_{a_i}^i$ and $p_{a_i+1}^i$ respectively and u_i^k excites b and $b+1$ by p_b^u and p_{b+1}^u eight rules will be fired for x_1^{k+1}:

$$\begin{aligned} -9a_1 + 10a_2 + 5a_3 &\rightsquigarrow p_{a_1}^1 p_{a_2}^2 p_{a_3}^3 \\ -9a_1 + 10a_2 + 5(a_3+1) &\rightsquigarrow p_{a_1}^1 p_{a_2}^2 p_{a_3+1}^3 \\ -9a_1 + 10(a_2+1) + 5a_3 &\rightsquigarrow p_{a_1}^1 p_{a_2+1}^2 p_{a_3}^3 \\ -9a_1 + 10(a_2+1) + 5(a_3+1) &\rightsquigarrow p_{a_1}^1 p_{a_2+1}^2 p_{a_3+1}^3 \\ -9(a_1+1) + 10a_2 + 5a_3 &\rightsquigarrow p_{a_1+1}^1 p_{a_2}^2 p_{a_3}^3 \\ -9(a_1+1) + 10a_2 + 5(a_3+1) &\rightsquigarrow p_{a_1+1}^1 p_{a_2}^2 p_{a_3+1}^3 \\ -9(a_1+1) + 10(a_2+1) + 5a_3 &\rightsquigarrow p_{a_1+1}^1 p_{a_2+1}^2 p_{a_3}^3 \\ -9(a_1+1) + 10(a_2+1) + 5(a_3+1) &\rightsquigarrow p_{a_1+1}^1 p_{a_2+1}^2 p_{a_3+1}^3 \end{aligned}$$

Defuzzifying to retrieve the real value of x_1^{k+1} consists of multiplying the above rules by the amount they are excited, adding and dividing by 100.

Similarly for x_2^{k+1} four rules will fire, namely:

$$\begin{aligned} 4a_2 + 14a_3 &\rightsquigarrow p_{a_2}^2 p_{a_3}^3 \\ 4a_2 + 14(a_3+1) &\rightsquigarrow p_{a_2}^2 p_{a_3+1}^3 \\ 4(a_2+1) + 14a_3 &\rightsquigarrow p_{a_2+1}^2 p_{a_3}^3 \\ 4(a_2+1) + 14(a_3+1) &\rightsquigarrow p_{a_2+1}^2 p_{a_3+1}^3 \end{aligned}$$

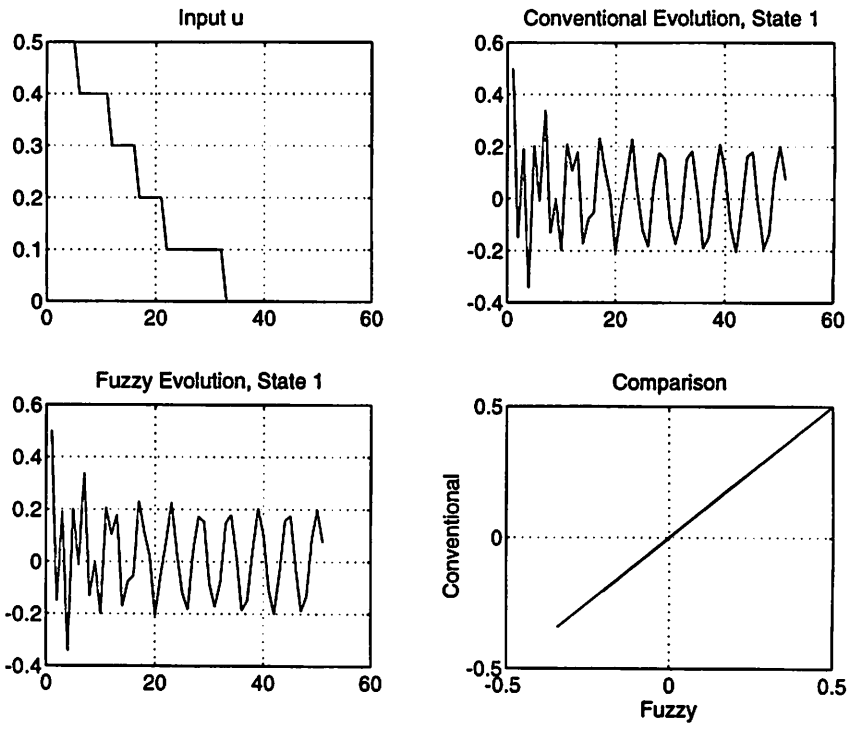

Figure 5: Model Comparison

Finally for x_3^{k+1} eight more rules will fire:

$$\begin{aligned}
-6a_2 + 4a_3 + b &\rightsquigarrow p_{a_2}^2 p_{a_3}^3 p_b^u \\
-6a_2 + 4a_3 + (b+1) &\rightsquigarrow p_{a_2}^2 p_{a_3}^3 p_{b+1}^u \\
-6a_2 + 4(a_3+1) + b &\rightsquigarrow p_{a_2}^2 p_{a_3+1}^3 p_b^u \\
-6a_2 + 4(a_3+1) + (b+1) &\rightsquigarrow p_{a_2}^2 p_{a_3+1}^3 p_{b+1}^u \\
-6(a_2+1) + 4a_3 + b &\rightsquigarrow p_{a_2+1}^2 p_{a_3}^3 p_b^u \\
-6(a_2+1) + 4a_3 + (b+1) &\rightsquigarrow p_{a_2+1}^2 p_{a_3}^3 p_{b+1}^u \\
-6(a_2+1) + 4(a_3+1) + b &\rightsquigarrow p_{a_2+1}^2 p_{a_3+1}^3 p_b^u \\
-6(a_2+1) + 4(a_3+1) + (b+1) &\rightsquigarrow p_{a_2+1}^2 p_{a_3+1}^3 p_{b+1}^u
\end{aligned}$$

The defuzzification process used for the other two states is similar to the one used for x_1^{k+1}.

The fuzzy model obtained thus was simulated along side the conventional linear system model from which it was derived. A typical trajectory for the output (first state) under both models, as well as the input used to obtain it are shown in figure 5. For comparison purposes a plot of one trajectory against the other is also given in the figure. The fact that all points in this graph lie on a straight line at 45° to the axes indicates that the trajectories generated by the two models are identical.

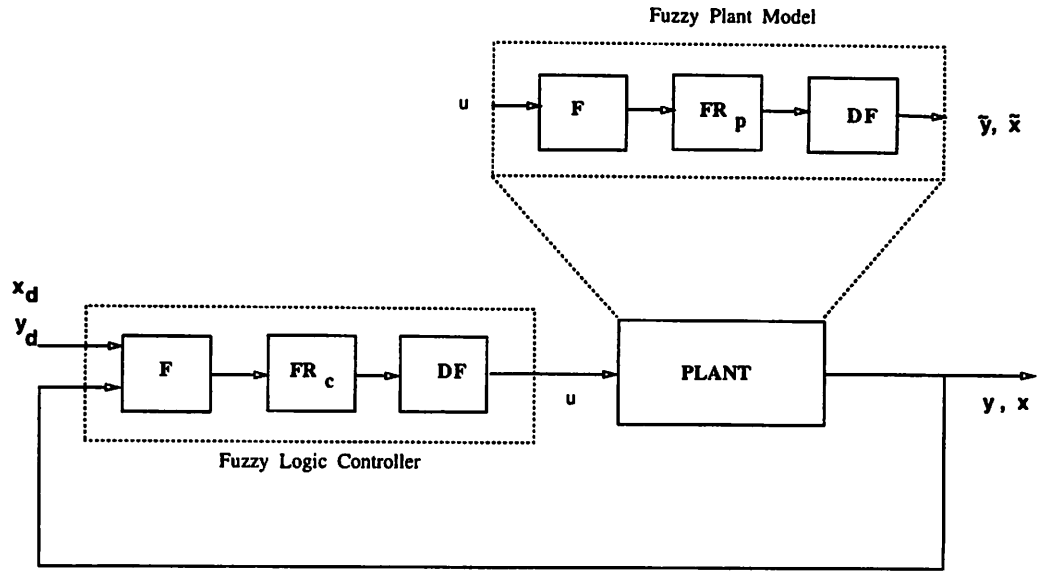

Figure 6: Block Diagram of the closed loop system

3 Fuzzy Logic Controller Design

A fuzzy plant model, like the one outlined in Section 2.1 will now be used to design a fuzzy logic controller. Figure 6 shows the block diagram of the closed loop system. As usual, the fuzzy logic controller consists of three parts; the fuzzification map F, the defuzzification map DF, and the controller rule base. In our method, the controller uses the same fuzzification and defuzzification method as in the plant model, that is the fuzzy sets of u, x and y are identical in the model and the controller. In addition the reference inputs x_d, y_d are also fuzzified using the plant fuzzy sets of x and y respectively. In this report we assume that the internal state variables, if present in the model, can be measured directly and therefore are available to the controller. The design of the rule base is presented in the next section. This rule base along with the F and DF maps described above will result in a complete specification of the fuzzy logic controller.

3.1 Design of controller rule base

The rule base of the plant model can be described as a map from input and state fuzzy sets to state and output fuzzy sets, as in equations 2 and 3). Suppose each internal state x_i is divided into n_{x_i} fuzzy sets, each input u_j is divided into n_{u_j} fuzzy sets and each output y_k is divided into n_{y_k} fuzzy sets. Then the plant rule base will have $(\prod_{i=1}^{n} n_{x_i}\ \prod_{k=1}^{l} n_{y_k})$ discrete states[1]. Each state is a $(n+l)$ tuple with each each coordinate being choice of a possible fuzzy set for each variable. Thus this is a conservative approach as the plant constraints between x and y will rule certain states to be impossible. In every state there are $\prod_{j=1}^{m} n_{u_j}$ choices of inputs. Thus input fuzzy sets can be regarded as events (choosing an input value that belongs to a

[1]If the output does not explicitly depend on inputs, as in the example of 23, then this number will be reduced to $\prod_{i=1}^{n} n_{x_i}$

particular fuzzy set), that define the directed edges (transitions) between states.

3.1.1 Rule base as a discrete event dynamical system

We can regard the plant rule base as a finite state abstraction of the plant behavior. We can represent this as a finite automaton given by:

$$G = (Q, \Sigma, \delta, q_0, F) \tag{24}$$

where,

$$\begin{aligned} Q &= \text{Finite Set of states of the plant model rule base} \\ \Sigma &= \text{Finite set of events (inputs)} \\ \delta &= \text{State Transition Function} \\ \delta &: Q \times \Sigma \rightarrow Q \\ q_0 &= \text{Set of initial states} \\ F &= \text{Set of final states} \end{aligned}$$

The transition function δ may not be be defined on the entire domain. For this example, the sets q_0 and F will contain only one state each.

We can now use this representation to design controller rule base for *regulation*. Given the plant automaton, the task is to design a rule based controller which will produce acceptable closed loop behavior. We can cast this as a discrete event supervisor design problem (see [19] for details) in the following way: all the events we have considered[2] so far (choice of inputs), are controllable in terms of Ramadge-Wonham theory of [19]. (i.e. you can block/disable them at any given time). Given an initial state of the plant automaton, the problem of control design (regulation) reduces to finding a sequence of inputs that will take the state of plant automaton from the initial state, q_0, to a final state, F. This is equivalent to finding $[(\prod_{j=1}^{m} n_{u_j}) - 1]$ events (control choices) at every state which should be blocked by the controller. By using Ramadge-Wonham theory we can check the existence of a supervisor to accomplish the control task. Algorithms exists in the literature [20, 21] to design supervisors for discrete event dynamical systems.

3.2 Regulation

For regulation, we are interested in finding out the sequence of inputs which will take the system from the initial state to the final state. The final state may represent the set point. We present two ways of designing controller rule base:

- Construction of controller rule base using **backward projection**:
  This is an iterative algorithm. We start the method with $k = 1$ and $B_0 = \text{F}$. The k^{th} iteration of this algorithm will be given by

  1. Check if $q_0 \in B_{k-1}$. If yes, stop. The algorithm has terminated successfully.

[2]The null event ϵ is not a meaningful event in this scenario

2. Find all the states such that proper choice of input will trigger a transition of the plant automaton to one of the states in the collection B_{k-1}.

$$B_k = \{q_k | \exists (u_{q_k} \in \Sigma) \; st. \; \delta(q_k, u_{q_k}) \in B_{k-1}\}$$

There might be more than one u_{q_k} choices for certain q_k because either two choices of input lead to two different states in B_{k-1} or they lead to same state in B_{k-1}. In this case, we have to chose only one of the possible inputs. This choice might depend on other design criterion (eg. control magnitude). The controller rule base will associate the input u_{q_k} with the state q_k.

3. Check if $B_k \subseteq B_{k-1}$? If yes, stop. There is no path from the initial state to the final state.
4. Replace k by $k+1$. Go back to step 1 and continue

This algorithm is guarenteed to terminate in a finite number of steps. It will terminate successfully if the desired final state is reachable from the initial state. It will also find the path which visits minimum number of states. For example, in the figure 7, the path $q_3 \rightarrow q_5 \rightarrow F$ will be discarded in favor of the direct transition $q_3 \rightarrow F$. The input sequence that will take plant of figure 7 state from q_0 to F is given by $(u1, u2, u3, u4)$.

The state and control pairs derived at every iteration in the above algorithm constitute the rule base for the fuzzy logic controller.

To remove the dependence on desired final state, one can select $(y - y_d)$ and $(x - x_d)$ as the feedback variable. Then the final plant state becomes independent of the desired set point.

- Automated search using verification software:
  If the number of states of the plant automaton is large, then the method described above will not be efficient. If the initial state q_0 of the plant automaton is fixed, then we can use COSPAN to get a control trace that will take the plant state from q_0 to F. COSPAN is a software program [22], that is used for verification of regular languages. In this case, we can enter the 'uncontrolled' plant state machine in COSPAN, define the task to be 'never visit the final state'. If there is a path from the initial state to the final state, then the run will fail and COSPAN will return the entire trace of events that failed the task. This will provide one trace of inputs to take the plant from the initial state to the final state. Note however that in this case there is no guarantee that the returned path will be the shortest route to the final state.

3.3 Tracking

Tracking is a dynamic phenomenon. We can cast this as a regulation problem if the desired trajectory can be expressed as an output of a reference fuzzy model using the same membership functions of the plant model. This may not always be possible however. We consider the one step finite state machine model, at every instant of time, for designing tracking controllers. In general, the tracking controllers need some on line calculations as opposed to the regulation controllers.

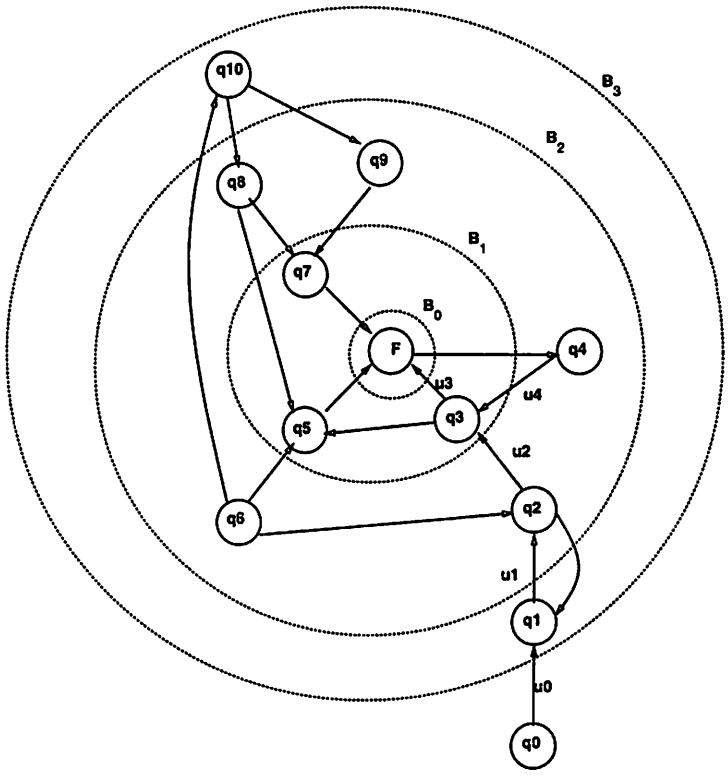

Figure 7: Backward Projection Algorithm

We will define a distance function on the discrete state space. For a given state of the plant automaton, there are at most $\prod_{j=1}^{m} n_{u_j}$ choices of next states (corresponding to each choice of the input). Thus at every step, the distance between the desired next state and the possible next states can be calculated. Choose the input, at this step, which makes the next state closest to the desired next state. Thus the controller has to perform $\prod_{j=1}^{m} n_{u_j}$ distance calculations and solve a discrete minimization problem at each step. Note that this method will result in exact tracking if it is achievable.

Many distance measures can be defined on the discrete space of fuzzy sets. A very simple choice is the following:

We associate the center $C_{x_i^j}$ of the membership function to the fuzzy set j of the variable x_i. Then the distance between two fuzzy sets of the same variable is given by $d(x_i^j - x_i^k) = |C_{x_i^j} - C_{x_i^k}|$. A discrete state of the plant automaton has $(n + l)$ coordinates. The distance between two discrete states will be defined as the Euclidean norm of the $(n + 1)$ dimensional real vector whose entries correspond to distance between each coordinate of the two states.

Clearly a proof of the claim that the controllers designed using this distance function is still needed.

3.4 Analysis

The performance of the closed loop discrete event system can be verified by automatic verification tools, such as COSPAN. In addition, we can add uncontrollable events, corresponding to disturbances, to the plant model. These events disturb the state of the plant and the controller should be able to bring it back to the desired state. We can use COSPAN to check that in an infinite run the controlled plant visits the desired state infinitely often.

The ultimate task of this research is to produce a technique for deriving proofs of performance of the closed loop system consisting the fuzzy plant model and the fuzzy controller discussed in this paper. In the future, we would like to be able to show that the controller rule base designed as in section 3 and verified as above will result in desired closed loop behavior.

4 Conclusions

In this paper we have presented a general framework for the the creation of fuzzy rule base plant models. In addition, we have synthesized a fuzzy rule base plant that exactly simulates a discrete-time, linear, time-invariant system with rational coefficients under certain assumptions on bounds on the state and input. Hopefully, our fuzzy plant model, will allow us to compare fuzzy rule base control with conventional control techniques. Further, we have presented a general formalism for regulation and tracking control synthesis of fuzzy rule base plants.

In the future we plan to use the fuzzy rule base plant model described in Section 2.2 and the control synthesis method presented in Section 3 to verify the closed loop performance of a fuzzy system. In addition, we hope to compare the performance of the fuzzy logic controller developed this way with conventional controllers.

5 Acknowledgments

The authors would like to thank Professor S. Shankar Sastry for his gracious support and steady encouragement.

References

[1] E. H. Mamdani, S. Assilian, "An Experiment in Linguistic synthesis with a Fuzzy Logic Controller", *International Journal of Man-Machine Studies*, Vol. 7, No. 1., pp 1–13, 1975.

[2] L. A. Zadeh, "Fuzzy logic and its application to approximate reasoning", *Inform. Process.*, Vol. 74, pp. 591–594, 1973.

[3] L. A. Zadeh, "Fuzzy logic and approximate reasoning", *Synthese*, Vol. 30, pp. 407–428, 1975.

[4] M. Sugeno, Editor, *Industrial Applications of Fuzzy Control*, North-Holland, Amsterdam, Holland, 1985.

[5] R. M Tong, "An Annotated Bibliography of Fuzzy Control", in *Industrial Applications of Fuzzy Control*, M. Sugeno, Ed., North-Holland, Amsterdam, Holland, 1985, pp249–269.

[6] H. Berenji, "Fuzzy Logic Controllers", in *An Introduction to Fuzzy Logic Applications in Intelligent Systems*, R. R. Yager and L. A. Zadeh, Eds., Kluwer Academic Publishers, Boston, MA, USA, 1992, pp 69–96.

[7] R. R. Yager and L. A. Zadeh, Editors. *An Introduction to Fuzzy Logic Applications in Intelligent Systems*, Kluwer Academic Publishers, Boston, MA, USA, 1992.

[8] W. Pedrycz. *Fuzzy Control and Fuzzy Systems*, 2nd extended ed. John Wiley & Sons, New York, NY, USA, 1993.

[9] R. R. Yager, "A Useful Bibliography for Fuzzy Sets", *Fuzzy Set and Possibility Theory : Recent Developments*, R. R. Yager, Ed., Pergamon, New York, NY, USA, 1982, pp 615–633.

[10] R. R. Yager, Editor, *Fuzzy Set and Possibility Theory : Recent Developments*, Pergamon, New York, NY, USA, 1982.

[11] H.-J. Zimmermann. *Fuzzy Set Theory and Its Applications.* Kluwer-Nijhoff Publishing, Hingham, MA, USA, 1984

[12] A. Kandel, *Fuzzy mathematical techniques with applications*, Addison-Wesley, Reading, MA, USA, 1986.

[13] V. Novák, *Fuzzy Sets and Their Applications*, Adam Hilger, Bristol, UK, 1989.

[14] H.-J. Zimmermann. *Fuzzy Set Theory and Its Applications*, 2nd rev. ed., Kluwer Academic Publishers, Boston, MA, USA, 1991

[15] T. Terano, K. Asai, M. Sugeno. *Fuzzy Systems Theory and Its Applications.* Academic Press, Boston, MA, USA, 1992.

[16] D. Driankov, *An Introduction to Fuzzy Control*, Springer-Verlag, Berlin, DGR, 1993.

[17] L.-X. Wang, *Adaptive Fuzzy Systems and Control: Design and Stability Analysis*, Prentice Hall, Englewood Cliffs, NJ, USA, 1994.

[18] D. McNeill, P. Freiberger, *Fuzzy Logic*, Simon & Schuster, New York, NY, USA, 1993.

[19] P. J. G. Ramadge and W. M. Wonham, "The control of discrete event dynamical systems," *Proceedings of IEEE*, vol. Vol.77, no. 1, pp. 81–98, 1989.

[20] A. Deshpande and P. Varaiya, "Control of Discrete Event Systems in Temporal Logic",(preprint).

[21] Y. J. Wei and P. E. Caines, "The Logical Complexity of Control Problem Formulated in COCOLOG," in *IEEE Control and Decision Conference*, pp. 2303–2308, 1993.

[22] Z. Har'El and R. Kurshan, *Cospan User's Guide.* AT&T Bell Laboratories, 1987.