USC–SIPI REPORT #169

Generating Fuzzy Rules from Numerical Data, with Applications

by

Li-Xin Wang and Jerry M. Mendel

January 1991

Signal and Image Processing Institute
UNIVERSITY OF SOUTHERN CALIFORNIA
Department of Electrical Engineering-Systems
3740 McClintock Avenue, Room 404
Los Angeles, CA 90089-2564 U.S.A.

GENERATING FUZZY RULES FROM NUMERICAL DATA, WITH APPLICATIONS

Li-Xin Wang and Jerry M. Mendel

Abstract

In this report, a general method is developed to generate fuzzy rules from numerical data. This new method consists of five steps: Step 1 divides the input and output spaces of the given numerical data into fuzzy regions; Step 2 generates fuzzy rules from the given data; Step 3 assigns a degree to each of the generated rules for the purpose of resolving conflicts among the generated rules; Step 4 creates a combined Fuzzy-Associative-Memory (FAM) Bank based on both the generated rules and linguistic rules of human experts; and, Step 5 determines a mapping from input space to output space based on the combined FAM Bank using a defuzzifying procedure. The mapping is proved to be capable of approximating any non-linear function on a compact set to arbitrary accuracy. Applications to truck backer-upper control [1] and time series prediction [2] problems are presented. For the truck control problem, the performance of this new method is compared with a neural network controller and a pure limited-rule fuzzy controller; the new method shows the best performance. For the time series prediction problem, results are compared by using the new method, a neural network predictor, and an AR model predictor for real time series data and a chaotic time series.

1 INTRODUCTION

For most real-world control and signal processing problems, the information concerning design, evaluation, realization, etc., can be classified into two kinds: numerical information obtained from sensor measurements, and, linguistic information obtained from human experts. Most current intelligent control and signal processing approaches are heuristic in nature, i.e., they combine some standard control or signal processing methods with expert systems in an ad hoc way for a specific problem; simulations are then performed to show that the new approaches work well for the specific problem. This kind of approach has two weakpoints: (1) it is quite problem dependent, i.e., a method may work well for one problem but is not suited for another problem; and, (2) there is no common framework for modeling and representing different aspects of control or signal processing strategies, which makes theoretical analyses for these approaches very difficult. In this report, we propose a general method for combining both numerical and linguistic information into a common framework – a Fuzzy-Associative-Memory (FAM) Bank.

Suppose we have the following problem: there is a complex control system in which a human controller is an essential part; the environment facing this human controller is so complicated that no mathematical model exists for it, or, the mathematical model is strongly non-linear so that a design method does not exist. The task here is to design a control system to replace the human controller (Fig.1).

In order to design such a control system, we first need to see what information is available. We assume that there is no mathematical model, i.e., we consider a model-free design problem. Since there already is a human controller who is successfully controlling the system, there are two kinds of information available to us: (1) the experience of the human controller; and, (2) sampled input-output (state-control) pairs which are recorded from successful control by the human controller. The experience of the human controller is usually expressed as some linguistic "IF - THEN" rules which state in what situation(s) which action(s) should be taken. The sampled input-output pairs are some numerical data which give the specific values of the inputs and the corresponding successful outputs.

Each of the two kinds of information alone is usually incomplete. Although the system is successfully controlled by a human controller, some information will be lost when the human controller expresses his/her experience by linguistic rules. Consequently, linguistic rules alone are usually not enough for designing a successful control system. On the other hand, the information from sampled input-output data pairs is usually also not enough for a successful design, because the past operations usually cannot cover all the situations the control system will face. If expert linguistic rules and numerical data pairs are the only information we can get for such a control system design, the most interesting case for us is when the combination of these two kinds of information is sufficient for a successful design.

Fuzzy control is an effective approach to utilizing linguistic rules [3,4], whereas neural control is suited for using numerical data pairs (i.e., desired input-output pairs) [1,4]. Present fuzzy controllers only use linguistic rules, whereas present neural controllers only use numerical data pairs (recently, Kosko [14] proposed a method of generating fuzzy rules from numerical data using vector quantization). This leads to the following challenging question: "Is it possible to develop a general approach which combines both kinds of information into a common framework, and uses both information, simultaneously and cooperatively, to solve the control design or similar problems ?" In this paper, we develop such a general approach.

The key ideas of our new approach are to generate fuzzy rules from numerical data pairs, collect these fuzzy rules and the linguistic fuzzy rules into a common Fuzzy-Associative-Memory Bank (FAM Bank, [4]), and, finally, design a control or signal processing system based on this combined FAM Bank using a defuzzifying method.

In Section 2, we propose a five step procedure for generating fuzzy rules from numerical data pairs and show how to use these fuzzy rules to obtain a desired mapping. Step 1 divides the input and output spaces into fuzzy regions; Step 2 generates fuzzy rules from given desired input-output data pairs; Step 3 assigns a degree to each generated rule; Step 4 forms the combined FAM Bank; and, Step 5 presents the detailed defuzzifying procedure for obtaining a mapping based on the combined FAM Bank. In Section 3, we prove that the resulting mapping is capable of approximating any non-linear function on a compact set to arbitrary accuracy using the well-known Stone-Weierstrass Theorem in analysis [5]. In Section 4, we apply our new method to a truck backer-upper control problem [1,4]. We compare this new approach with pure neural and fuzzy approaches. The power of our new approach becomes apparant when it is used in the case where neither linguistic fuzzy rules nor input-output pairs are sufficient to successfully control the truck to a desired position, but the combination of both is sufficient. In Section 5, we show that our new method can be used for time-series prediction; and, a real time series (monthly mean temperature of St. Louis, Missouri) is used to test our new method against a neural network forecaster and an AR model forecaster. We also use our new method to predict a chaotic time series, and compare the results with those obtained using a neural network predictor. Conclusions and discussions are given in Section 6.

2 GENERATING FUZZY RULES FROM NUMERICAL DATA

Suppose we are given a set of desired input-output data pairs:

$$(x_1^{(1)}, x_2^{(1)}; y^{(1)}), (x_1^{(2)}, x_2^{(2)}; y^{(2)}), \cdots \tag{1}$$

where x_1 and x_2 are inputs, and y is the output. This simple two-input one-output case is chosen in order to emphasize and to clarify the basic ideas of our new approach; extensions to general multi-input multi-output cases are straightforward and will be discussed later in this section. The task here is to generate a set of fuzzy rules from the desired input-output pairs of (1), and use these fuzzy rules to determine a mapping $f : (x_1, x_2) \rightarrow y$.

Our new approach consists of the following five steps:

Step 1: Divide the Input and Output Spaces into Fuzzy Regions

Assume that the domain intervals of x_1, x_2 and y are $[x_1^-, x_1^+], [x_2^-, x_2^+]$ and $[y^-, y^+]$, respectively, where "domain interval" of a variable means that most probably this variable will lie in this interval (the values of a variable are allowed to lie outside its domain interval). Divide each domain interval into $2N + 1$ regions (N can be different for different variables, and the lengths of these regions can be equal or unequal), denoted by SN (Small N), ..., S1 (Small 1), CE (Center), B1 (Big 1), ..., BN (Big N), and assign each region a fuzzy membership function. Figure 2 shows an example where the domain interval of x_1 is divided into five regions (N=2), the domain region of x_2 is divided into seven regions (N=3), and the domain interval of y is divided into five regions (N=2). The shape of each membership function is triangular; one vertex lies at the center of the region and has membership value unity; the other two vertices lie at the centers of the two neighbouring regions, respectively, and have membership values equal to zero. Of course, other divisions of the domain regions and other shapes of membership functions are possible.

Step 2: Generate Fuzzy Rules from Given Data Pairs

First, *determine the degrees of given* $x_1^{(i)}, x_2^{(i)}$ *and* $y^{(i)}$ *in different regions.* For example, $x_1^{(1)}$ in Fig.2 has degree 0.8 in B1, degree 0.2 in B2, and zero degrees in all other regions. Similarly, $x_2^{(2)}$ in Fig.2 has degree 1 in CE, and zero degrees in all other regions.

Second, *assign a given* $x_1^{(i)}, x_2^{(i)}$ *or* $y^{(i)}$ *to the region with maximum degree.* For example, $x_1^{(1)}$ in Fig.2 is considered to be B1, and $x_2^{(2)}$ in Fig.1 is considered to be CE.

Finally, ***obtain one rule from one pair of desired input-output data***, e.g.,

$(x_1^{(1)}, x_2^{(1)}; y^{(1)}) \Rightarrow [x_1^{(1)}(0.8 \text{ in B1, max}), x_2^{(1)}(0.7 \text{ in S1, max}) ; y^{(1)}(0.9 \text{ in CE, max})] \Rightarrow$
Rule 1: IF x_1 is B1 and x_2 is S1, THEN y is CE;

$(x_1^{(2)}, x_2^{(2)}; y^{(2)}) \Rightarrow [x_1^{(2)}(0.6 \text{ in B1, max}), x_2^{(2)}(1 \text{ in CE, max}) ; y^{(2)}(0.7 \text{ in B1, max})] \Rightarrow$
Rule 2: IF x_1 is B1 and x_2 is CE, THEN y is B1.

The rules generated in this way are "and" rules, i.e., rules in which the conditions of the IF part must be met simultaneously in order for the result of the THEN part to occur. For the problems considered in this paper, i.e., generating fuzzy rules from numerical data, only "and" rules are required.

Step 3: Assign a Degree to Each Rule

Since there are usually lots of data pairs, and each data pair generates one rule, it is highly probable that there will be some conflicting rules, i.e., rules which have the same IF part but a different THEN part. One way to resolve this conflict is to assign a degree to each rule generated from data pairs, and accept only the rule from a conflict group that has maximum degree. In this way not only is the conflict problem resolved, but also the number of rules is greatly reduced.

We use the following product strategy to assign a degree to each rule: for the rule: "IF x_1 is A and x_2 is B, THEN y is C", the *degree of this rule*, denoted by $D(Rule)$, is defined as

$$D(Rule) = m_A(x_1) m_B(x_2) m_C(y). \tag{2}$$

As examples, Rule 1 has degree

$$\begin{aligned} D(Rule1) &= m_{B1}(x_1) m_{S1}(x_2) m_{CE}(y) \\ &= 0.8 \times 0.7 \times 0.9 = 0.504; \end{aligned} \tag{3}$$

and Rule 2 has degree

$$\begin{aligned} D(Rule2) &= m_{B1}(x_1) m_{CE}(x_2) m_{B1}(y) \\ &= 0.6 \times 1 \times 0.7 = 0.42. \end{aligned} \tag{4}$$

In practice, we often have some a priori information about the data pairs. For example, if we let an expert check given data pairs, the expert may suggest that some are very useful and crucial, but others are very unlikely and may be caused just by measurement errors. We can therefore assign a degree to each data pair which represents our belief of its usefulness. In this sense, the data pairs constitute a fuzzy set, i.e., the fuzzy set is defined as the useful measurements; a data pair belongs to this set to a degree assigned by a human expert.

Suppose the data pair $(x_1^{(1)}, x_2^{(1)}; y^{(1)})$ has degree $m^{(1)}$, then we redefine the degree of Rule 1 as

$$D(Rule1) = m_{B1}(x_1)m_{S1}(x_2)m_{CE}(y)m^{(1)}, \tag{5}$$

i.e., the degree of a rule is defined as the product of the degrees of its components and the degree of the data pair which generates this rule. By using this strategy, we are now in a totally fuzzy environment in that everything has a degree. This is important in practical applications, because real numerical data have different reliabilities, e.g. some real data can be very bad ("wild data"). For good data we assign higher degrees, and for bad data we assign lower degrees. In this way, human experience about the data is used in a common base as other information. If one emphasizes objectivity and does not want a human to judge the numerical data, our strategy still works by setting all the degrees of the data pairs equal to unity.

Step 4: Create a Combined FAM Bank

The form of the FAM Bank is shown in Fig.3. We fill the boxes of the bank with fuzzy rules according to the following strategy: *a combined FAM Bank is assigned rules from either those generated from numerical data or linguistic rules (we assume that a linguistic rule also has a degree which is assigned by the human expert and reflects the expert's belief of the importance of the rule); if there is more than one rule in one box of the FAM Bank, use the rule that has maximum degree.* In this way, both numerical and linguistic information are codified into a common framework – the combined FAM Bank. If a linguistic rule is an "and" rule, it fills only one box of the FAM Bank; but, if a linguistic rule is an "or" rule (i.e., a rule for which the THEN part follows if any condition of the IF part is satisfied), it fills all the boxes in the rows or columns corresponding to the regions of the IF part. For example, suppose we have the linguistic rule: "IF x_1 is S1 *or* x_2 is CE, THEN y is B2" for the FAM Bank of Fig.3; then we fill the seven boxes in the column of S1 and the five boxes in the row of CE with B2. The degrees of all the B2's in these boxes equal the degree of this "or" rule.

Step 5: Determine A Mapping Based on the FAM Bank

We use the following defuzzification strategy to determine the output control y for given inputs (x_1, x_2): first, we represent all the fuzzy rules in the FAM Bank by membership function forms (Fig.4 shows an example for Rules 1 and 2); then, for given inputs (x_1, x_2), we combine the antecedents of the i'th fuzzy rule using *product* operations to determine the degree, $m_{O^i}^i$, of the output control corresponding to (x_1, x_2), i.e.,

$$m_{O^i}^i = m_{I_1^i}(x_1)m_{I_2^i}(x_2), \tag{6}$$

where O^i denotes the output region of Rule i, and I^i_j denotes the input region of Rule i for the j^{th} component, e.g., Rule 1 gives

$$m^1_{CE} = m_{B1}(x_1)m_{S1}(x_2); \tag{7}$$

finally, we use the following centroid defuzzification formula to determine the output control

$$y = \frac{\sum_{i=1}^{K} m^i_{O^i}\bar{y}^i}{\sum_{i=1}^{K} m^i_{O^i}} \tag{8}$$

where $\bar{y}^i$ denotes the center value of region O^i (the *center* of a fuzzy region is defined as the point which has the smallest absolute value among all the points at which the membership function for this region has membership value equal to one), and K is the number of fuzzy rules in the combined FAM Bank. Figure 4 shows an example of the procedure for Rules 1 and 2.

From Steps 1 to 5 we see that our new method is simple and straightforward in the sense that it is a one-pass build-up procedure that does not require time-consuming training; hence, it has the same advantage that the fuzzy approach has over the neural approach, as pointed out in [4], namely, it is simple and quick to construct.

This five step procedure can easily be extended to general multi-input multi-output cases. Steps 1 to 4 are independent of how many inputs and how many outputs there are. In Step 5, we only need to replace $m^i_{O^i}$ in Eq.(6) with $m^i_{O^i_j}$, where j denotes the j^{th} component of the output vector (O^i_j is the region of Rule i for the j^{th} output component; $m^i_{O^i_j}$ is the same for all j), and change Eq.(8) to

$$y_j = \frac{\sum_{i=1}^{K} m^i_{O^i_j}\bar{y}^i_j}{\sum_{i=1}^{K} m^i_{O^i_j}} \tag{9}$$

where $\bar{y}^i_j$ denotes the center of region O^i_j.

A problem with the above approach is that the memory for the FAM Bank grows as more and more training data become available, i.e., the present approach only considers how to generate rules, and, as a rule is generated it is stored in the FAM Bank forever if no new rules with higher degree overlap it. This is a *growing memory* problem. One way to overcome this problem is to truncate "old" rules, i.e., only store and use the rules generated from a fixed number of the most recent data. This growing memory problem remains a topic for further research.

If we view this five step procedure as a block, then the inputs to this block are "examples" (desired input-output data pairs) and expert rules (linguistic IF-THEN statements), and the output is a mapping from input space to output space. For control problems, the input space is the state of the plant to be controlled, and the output space is the control applied to the plant. For time-series prediction problems, the input and output spaces are subsequences of the time series such that the input subsequence precedes the output subsequence (details are given in Section 5). Our new method essentially "learns" from the "examples" and expert rules to obtain a mapping which, hopefully, has the "generalization" property that when new inputs are presented the mapping continues to give desired or successful outputs. Hence, our new method can be viewed as a very general *Model-Free Trainable Fuzzy System* for a wide range of control and signal processing problems, where: "Model-Free" means no mathematical model is required for the problem; "Trainable" means the system learns from "examples" and expert rules, and can adaptively change the mapping when new "examples" and expert rules are available; and, "Fuzzy" denotes the fuzziness introduced into the system by linguistic fuzzy rules, fuzziness of data, etc..

3 FUZZY SYSTEM AS A UNIVERSAL APPROXIMATOR

The five step procedure of the last section generates a fuzzy system, i.e., a mapping from input space to output space. Specifically, this mapping is represented by Eqs.(6) and (8) for the two-input one-output case. Using simplified notations, we rewrite Eqs.(6) and (8), for the general n-input one-output case, as

$$m^i = \Pi_{1\le j\le n}[m_j^i(x_j)], \tag{10}$$

$$f(\underline{x}) = \frac{\sum_{i=1}^{K} \bar{y}^i m^i}{\sum_{i=1}^{K} m^i} = \frac{\sum_{i=1}^{K} \bar{y}^i \Pi_{1\le j\le n}[m_j^i(x_j)]}{\sum_{i=1}^{K} \Pi_{1\le j\le n}[m_j^i(x_j)]}, \tag{11}$$

where m_j^i is the membership function of the i'th rule for the j'th component of the input vector, and $\bar{y}^i$ is the center value of the output region of the i'th rule. We will prove that this generated fuzzy system, i.e., Eq.(11), is a universal approximator from a compact set $Q \subset R^n$ to R, i.e., it can approximate any real continuous function defined on Q to any accuracy, where the compact set Q is defined as

$$Q = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n] \subset R^n. \tag{12}$$

For notational convenience, we represent Rule i ($i = 1, 2, ..., K$) in the FAM Bank as: "IF x_1 is RG_1^i, x_2 is RG_2^i, ... , x_n is RG_n^i, THEN y is RG_0^i," where RG_j^i ($j = 1, 2, ..., n$) denotes the region for the j'th input antecedent of Rule i (e.g., it can be $S2$, CE, $B1$, etc.), and RG_0^i denotes the output region of Rule i.

Let F be the family of functions of the form of Eq.(11) on the compact set Q. There are three factors which determine a member of F: (1) the definition of fuzzy regions, i.e., how to define and divide the domain intervals; (2) the specific form of membership functions m_j^i; and, (3) the specific statements of fuzzy rules in the FAM Bank. By fixing fuzzy regions, membership functions, and fuzzy rules, we obtain an element of F. If f_1 and f_2 are different elements of F, then at least one of the three factors for f_1 and f_2 must be different. In order to analyse the family F, we make the following assumptions for these three factors:

AS.1: The fuzzy regions for the input and output spaces can be arbitrarily defined.

AS.2: The membership functions m_j^i can be any continuous functions from $[a_j, b_j]$ to $[0, 1]$ for $j = 1, 2, ..., n$ (i.e., for inputs) and from $(-\infty, \infty)$ to $[0, 1]$ for $j = 0$ (i.e., for output y); however, $m_j^i(x_j) \neq 0$ for $x_j \in RG_j^i$, $i = 1, 2, ..., K, j = 0, 1, ..., n$, with $x_0 = y$. This constraint means that the membership value of an antecedent for a rule cannot equal zero if the actual input value of this antecedent falls into the required region of the rule.

AS.3: Any rule can be assigned to any box of the FAM Bank.

These assumptions are always satisfied in practice. Specifically, we have total freedom in defining fuzzy regions; we can choose any membership functions subject to the constraint of AS.2; and, we can assign any rule to any box of the FAM Bank.

To analyse the properties of the function family F, we must first establish that the mapping defined by Eq.(11) is well-defined, i.e., that for any input $\underline{x} \in Q$, Eq.(11) will generate an output $f(\underline{x}) \in R$. The following two lemmas give sufficient conditions for Eq.(11) to be well-defined.

LEMMA 1: If all the membership functions m_j^i are non-zero, and there is at least one rule in the FAM Bank, then the mapping defined by Eq.(11) from Q to R is well-defined.

Proofs of lemmas and theorems are given in Appendix A.

LEMMA 2: If every box in the FAM Bank has a rule associated with it, i.e., there are no empty boxes in the FAM Bank, then the mapping defined by Eq.(11) from Q to R is well-defined under AS.2.

In practice, the input space is usually high dimensional, whereas the given successful data pairs and expert rules are often quite limited; as a result, many boxes of the FAM Bank may be empty . However, it is possible to fill up these empty boxes, based on the given limited rules, using the method of Section 2. Specifically, Steps 1-4 are first used to generate a FAM Bank based on the limited data pairs and linguistic rules; then, the output for some typical input for which the box in the FAM Bank is empty is determined based on the limited FAM Bank ; finally, the range in which the output has the maximum degree is assigned to the empty box as a new rule. This can be an iterative procedure, i.e., when a new rule is generated, it and the existing rules are combined into the FAM Bank, which is then used to generate the next new rule. This procedure can be started from the empty boxes which are the nearest neighbors to the full boxes; in this way, the FAM Bank expands from existing rules until all the boxes are filled up. This procedure always works if we choose the non-zero regions of the membership functions to be large enough such that the values of the membership functions will not be zero for some points of their nearest neighbors. We will not study this procedure in detail in this report; we give the basic ideas of the procedure in order to show that the conditions of Lemma 2 can be satisfied.

Now we state the main result of this section.

THEOREM 1: If the mapping defined by Eq.(11) is well-defined, and if AS.1, AS.2, and AS.3 are true, then the mapping defined by Eq.(11) is capable of approximating any real continuous function over the compact set Q of Eq.(12) to arbitrary accuracy..

(The proof of Theorem 1 uses AS.4 and the definition of "active" rule which will be given later in this section; hence, we suggest the reader read the rest of this section before going to the proof. The rest of this section does not use the result of Theorem 1.)

Theorem 1 is an existence theorem showing that there exists a way of defining fuzzy regions, a way of choosing membership functions, and a way of assigning fuzzy rules to the boxes of the FAM Bank, such that the resulting mapping, Eq.(11), approximates an arbitrary nonlinear continuous mapping from Q to R to any accuracy. This Theorem is similar to the results of [6] and [7] which showed that a three-layer feedforward neural network is a universal approximator provided that there are sufficiently large numbers of hidden-layer neurons. Theorem 1 provides the theoretical basis for successful applications of our new method to many different practical problems.

In many applications of fuzzy systems (e.g., [3,4]), the membership functions are triangular. We now study some properties of the fuzzy systems which use the specific form of membership functions which are defined as:

<u>AS.4</u>: The membership function for any intermediate fuzzy region (i.e., not the smallest or the largest region) is a triangle whose vertices are at $(x, m) = (x_{-1}, 0)$, $(x_0, 1)$, and $(x_1, 0)$, where the x-axis denotes a coordinate of the input or output space, the m-axis denotes the corresponding membership value, x_0 denotes the center of the region, and x_{-1} (x_1) denotes the center of the left (right) region. See Fig.17 for an example. The membership functions for the smallest and largest regions are determined by the way shown in Fig.17.

If every box of the FAM Bank has a rule and $[a_j, b_j]$ is divided into r_j fuzzy regions $(j = 1, 2, ..., n)$, then there are $N = r_1 \times r_2 \times \cdots \times r_n$ rules in the FAM Bank. N can be a huge number if the r_j's and n are large; however, under AS.4 there will only be a small fraction of these rules which are actually activated for use in Eq.(11) for any given $\underline{x} \in Q$. This represents the power of triangular membership functions.

<u>Definition</u>: The i'th fuzzy rule in the FAM Bank is *active* for $\underline{x} \in Q$ if $m_j^i(x_j) \neq 0$ for all $j = 1, 2, ..., n$. Refering to Eq.(11), we see that only if a rule is active will it be used in Eq.(11).

<u>LEMMA 3</u>: Under AS.4, the following is true:
(1) There are at most 2^n active rules for any $\underline{x} \in Q$.
(2) If r components of $\underline{x} \in Q$ are at the centers of some fuzzy regions $(r = 0, 1, 2, ..., n)$, there are at most 2^{n-r} active rules at the $\underline{x}$ (recall that the center of a fuzzy region is defined as the point which has the smallest absolute value among all the points at which the membership function for this region has membership value equal to unity).
(3) If r components of $\underline{x} \in Q$ are at the centers of some fuzzy regions, and if q components of the $\underline{x}$ are smaller (greater) than the center values of the smallest (largest) regions of the corresponding components, then there are at most 2^{n-r-q} active rules at

the $\underline{x}$.

Lemma 5 is useful in practice. Although we may need a huge memory to store the FAM Bank, when we use the FAM Bank for a given input $\underline{x} \in Q$, only a relatively small number of rules are used. In practice, we may store the FAM Bank in a cheap external memory; when we have an input, we only take the active rules from the FAM Bank into the host computer.

4 APPLICATION TO TRUCK BACKER-UPPER CONTROL

Backing a truck to a loading dock is a difficult exercise for all but the most skilled truck drivers. It is a severely non-linear control problem for which no traditional control system design methods exist. In [1], Nguyen and Widrow develop a neural network controller for the truck backer-upper problem; and, in [4], Kong and Kosko propose a fuzzy control strategy for the same problem. The neural network controller of [1] only uses numerical data, and cannot utilize linguistic rules determined from expert drivers; on the other hand, the fuzzy controller of [4] only uses linguistic rules, and cannot utilize sampled data. Since the truck backer-upper control problem is a good example of the control system design problem discussed in the Introduction of this paper (i.e., replace a human controller by a machine), it is interesting to apply the approach developed in Section 2 to this problem. In order to distinguish these methods, we call the method of [4] the "fuzzy approach", the method of [1] the "neural approach", and our new method the "numerical-fuzzy approach"(see Fig.5).

The results of [4] demonstrated superior performance of the fuzzy controller over the neural controller; however, from Fig.5 we see that the fuzzy and neural controllers use different information to construct the control strategies. It is possible that the fuzzy rules used in [4] to construct the controller are more complete and contain more information than the numerical data used to construct the neural controller; hence, the comparison between the fuzzy and neural controllers, from a final control performance point of view, is somewhat unfair. If the linguistic fuzzy rules were incomplete, whereas the numerical information contained lots of very good data pairs, it is highly possible that the neural controller would outperform the fuzzy controller.

Our new numerical-fuzzy approach provides a fair basis for comparing fuzzy and neural controllers (the numerical-fuzzy approach can be viewed as a fuzzy approach in the sense that it differs from the pure fuzzy approach only in the way it obtains fuzzy rules) . From Fig.5 we see that we can provide the same desired input-output pairs to both the neural and numerical-fuzzy approaches; consequently, we can compare the final control performances of both controllers fairly since they both use the same information.

Example 1: In this example, we use the same set of desired input-output pairs to simulate neural and numerical-fuzzy controllers, and compare their final control performances.

Problem Statement of the Truck Backer-Upper Control

The simulated truck and loading zone are shown in Fig.6 [1,4]. The truck corresponds to the cab part of the neural truck in the Nguyen-Widrow [1] neural truck backer-upper system. The truck position is exactly determined by the three state variables ϕ, x, and y, where ϕ is the angle of the truck with the horizontal as shown in Fig.6. Control to the truck is the angle θ. Only backing up is considered. The truck moves backward by a fixed unit distance every stage. For simplicity, we assume enough clearance between the truck and the loading dock such that y does not have to be considered as an input (see [4] for discussions on this assumption). The task here is to design a control system, whose inputs are $\phi \in [-90^o, 270^o]$ and $x \in [0, 20]$, and whose output is $\theta \in [-40^o, 40^o]$, such that the final states will be $(x_f, \phi_f) = (10, 90^o)$.

Generating Desired Input-Output Pairs $(x, \phi; \theta)$

We do this by trial and error: at every stage (given ϕ and x) starting from an initial state, we determined a control θ based on common sense (i.e., our own experience of how to control the steering angle in the situation); after some trials, we chose the desired input-output pairs corresponding to the smoothest successful trajectory.

The following 14 initial states were used to generate desired input-output pairs: (x_0, ϕ_0^o) = (1,0), (1,90), (1,270); (7,0), (7,90), (7,180), (7,270); (13,0), (13,90), (13,180), (13,270); (19,90), (19,180), (19,270). Since we performed simulations, we needed to know the dynamics of the truck backer-upper procedure. We used the following approximate kinematics (see Appendix B) :

$$x(t+1) = x(t) + cos[\phi(t) + \theta(t)] + sin[\theta(t)]sin[\phi(t)] \tag{13}$$

$$y(t+1) = y(t) + sin[\phi(t) + \theta(t)] - sin[\theta(t)]cos[\phi(t)] \tag{14}$$

$$\phi(t+1) = \phi(t) - sin^{-1}[\frac{2sin(\theta(t))}{b}] \tag{15}$$

where b is the length of the truck. We assumed $b = 4$ in the simulations of this paper. Equations (13) to (15) were used to obtain the next state when the present state and control are given. Since y is not considered a state, only Eqs.(13) and (15) were used in the simulations. We wrote Eq.(14) here for the purpose of showing the complete dynamics of the truck. Observe, from Eqs.(13)-(15), that even this simplified dynamic model of the truck is severely non-linear. The 14 sequences of desired $(x, \phi; \theta)$ pairs are given in Tables 1-14.

Neural Control and Simulation Results

We used a two-input single-output three-layer back-propagation neural network for our control task, as shown in Fig.7; 20 hidden neurons were used, and a sigmoid non-linear function was used for each neuron. The output of the third-layer neuron represents the steering angle θ according to a uniform mapping from [0,1] to $[-40^o, 40^o]$, i.e., if the neuron output is $g(t)$, the corresponding output $\theta(t)$ is

$$\theta(t) = 80g(t) - 40. \tag{16}$$

In real simulations, we normalized $[-40^{\circ}, 40^{\circ}]$ into $[-1, 1]$. Similarly, the inputs to the neurons were also normalized into $[-1, 1]$.

Our neural network controller is different from the Nguyen-Widrow neural controller [1]. First, we have only one neural network which does the same work as the Truck Controller of the Nguyen-Widrow network; the Truck Emulator of the Nguyen-Widrow network is not needed in our task. Second, and more fundamentally, we trained our neural network using desired input-output (state-control) pairs, which were obtained from the past successful control history of the truck, whereas Nguyen and Widrow [1] connected their neural network stage by stage and trained these concatenated neural networks by back-propagating the error at the truck's final state through this long network chain (their detailed algorithm is different from the standard error back-propagation algorithm in order to meet the constraint that the neural networks at each stage perform the same transformation; for details see [1]). Hence, the training of our neural network is simpler than that of the Nguyen-Widrow network. Of course, we need to know some successful control trajectories (state-control pairs) starting from some typical initial states; this is not required in the Nguyen-Widrow neural network controller.

We trained the neural network of Fig.7 using the standard error back-propagation algorithm [9] for the 14 sequences of desired $(x, \phi; \theta)$ pairs of Tables 1-14 . We used the converged network to control the truck whose dynamics are approximately given by Eqs.(13)-(15). Three arbitrarily chosen initial states, (x_0, ϕ_0^o) = (3,-30), (10,220), and (13,30), were used to test the neural controller. The truck trajectories from the three initial states are shown in Fig.8. We see that the neural controller successfully controls the truck to the desired position starting from all three initial states.

Numerical-Fuzzy Control and Simulation Results

We used the five-step procedure of Section 2 to determine the control law $f : (x, \phi) \rightarrow \theta$, based on the 14 sequences of successful $(x, \phi; \theta)$ pairs of Tables 1-14. For this specific problem, we used membership functions shown in Fig.9, which are similar to those used in [4] for fuzzy control of the problem based only on linguistic rules. The fuzzy rules generated from the desired input-output pairs of Tables 1-14 and their corresponding degrees are shown in Tables 15-28, where Table 14+i shows the rules generated from the data in Table i (i=1,2, ..., 14). These tables (Tables 15-28) show the results of Steps 1-3 of our method developed in Section 2. The final FAM Bank generated from the rules in Tables 15-28 is shown in Fig.10 (this is the result of Step 4 of our method in Section 2; here we assume that no linguistic rules are available). We see from Fig.10 that there are no generated rules for some ranges of x and ϕ. This shows that the desired trajectories of Tables 1-14 do not cover all the possible cases; however, we will see that the rules in Fig.10 are sufficient for controlling the truck to the desired state starting from some given initial states.

Finally, Step 5 of our numerical-fuzzy method was used to control the truck from the three initial states, (x_0, ϕ_0^o) = (3,-30), (10,220), and (13,30), which are the same states used in the simulations of the neural controller. The final trajectories of the truck are shown in Fig.11.

Discussion

Comparing Figs.8 and 11 we see that they are almost exactly the same; hence, we conclude that the neural and numerical-fuzzy controllers have the same control performance for these cases. We simulated the neural and numerical-fuzzy controllers for other initial truck positions, and observed that the truck trajectories using these two controllers were also almost the same. This is not surprising because both controllers used the same information (i.e., the desired trajectories of Tables 1-14) to construct their control laws.

Neural and numerical-fuzzy controls can be viewed as two different ways to perform "generalization" on the given data, where "generalization" means that when a controller, which has already been "trained" to match some desired patterns, is given a new input pattern, it still gives correct output patterns. From our simulation results we see that the "generalization" ability of these two approaches is the same for this truck backer-upper control problem. Because neural network approaches need time-consuming iterative training, whereas the numerical-fuzzy approach is just a one-pass construction procedure, the numerical-fuzzy approach appears to be superior to the neural network approach.

Example 2: In this example we consider the situation where neither linguistic fuzzy rules alone nor desired input-output pairs alone are sufficient to successfully control the truck to the desired position, i.e., neither the usual fuzzy controller with limited fuzzy rules nor the usual neural controller can control the truck to the desired position, but a combination of linguistic fuzzy rules and fuzzy rules generated from the desired input-output data pairs is sufficient to successfully control the truck to the desired position.

We consider the case where the beginning part of the information comes from desired input-output pairs whereas the ending part of the information comes from linguistic rules (see Fig.12). To do this we used only the first three pairs of each of the 14 desired sequences in Tables 1-14, and generated fuzzy rules based only on these truncated pairs. The FAM Bank generated from these truncated data pairs is shown in Fig.13 which is the same as Fig.10 except for the three center boxes. The FAM Bank of linguistic rules for the ending part was chosen as in Fig.14 which is the same as the three center rules of Fig.10.

We simulated the following three cases in which we used the: (1) FAM Bank generated from only the truncated data pairs (i.e., Fig.13); (2) FAM Bank of selected linguistic rules (i.e., Fig.14); and, (3) FAM Bank which combined the FAM Banks of (1) and (2) (i.e., Figs.13 and 14). We see that for Case 3 the FAM Bank is the same as in Fig.10; hence, the truck trajectories for this case must be the same as those using the FAM Bank of Fig.10. For each of the cases, we simulated the system starting from the following three initial states: (x_0, ϕ_0^o) = (3,-30), (10,220), and (13,30). The resulting trajectories for cases (1), (2), and (3) for the three initial states are shown in Figs.15, 16 and 11, respectively.

We see very clearly from these figures that, for cases (1) and (2) the truck cannot be controlled to the desired position, whereas for case (3) we successfully controlled the truck to the desired position.

5 APPLICATION TO TIME-SERIES PREDICTION

Time-series prediction is a very important practical problem [2]. Applications of time-series prediction can be found in the areas of economic and business planning, inventory and production control, weather forecasting, signal processing, control, and lots of other fields. Let $z(k)$ $(k = 1, 2, 3, ...)$ be a time series. The problem of time-series prediction can be formulated as: given $z(k-m+1), z(k-m+2), ..., z(k)$, determine $z(k+l)$, where m and l are fixed positive integers; i.e., determine a mapping from $[z(k-m+1), z(k-m+2), ..., z(k)] \in R^m$ to $[z(k+l)] \in R$. In practice, past samples of $z(k)$ are usually available which are used to determine and test the mapping.

Traditional approaches to this problem use parametric models to represent the time series $z(k)$ [2]. For example, the following autoregressive (AR) model may be used:

$$z(k+1) = \sum_{i=1}^{m} a_i z(k-i+1) + v(k), \tag{17}$$

where $v(k)$ is a white noise sequnece. The parameters a_i $(i = 1, 2, ..., m)$ are estimated using the known values of $z(k)$ (substitute the known $z(k)$'s into Eq.(17) and ignore the white noise $v(k)$ to form a set of linear equations, and then solve these equations for a_i using least-squares; the solutions are the prediction-error estimates of a_i), then ignoring the white noise $v(k)$ and substituting the estimated a_i into Eq.(17), we obtain a forecasting model for $z(k)$. The main disadvantage of traditional methods is that a parametric model must be assumed at the very beginning; thus, if the model is not suited for the time series, the predictions will have large errors.

A feedforward neural network can also be used for this problem[10,11]. For example, we can use a three-layer feedforward neural network, which has m input neurons and one output neuron, to represent the mapping from $[z(k-m+1), z(k-m+2), ..., z(k)]$ to $[z(k+l)]$. The network is trained for the known $z(k)$'s, and then the converged network is used for the prediction. Specifically, assume that $z(1), z(2), ..., z(M)$ are given; then we form $M - m$ desired input-output pairs:

$$\begin{aligned}&[z(M-m), ..., z(M-1); z(M)]\\ &[z(M-m-1), ..., z(M-2); z(M-1)]\\ &...\\ &[z(1), ..., z(m); z(m+1)].\end{aligned} \tag{18}$$

We train the neural network to match these $M - m$ pattern pairs using the error back-propagation algorithm [9].

Our numerical-fuzzy method in Section 2 can also be used for this time series prediction problem. Similar to the neural network approach, we assume that $z(1), z(2), ..., z(M)$ are given, and we form the $M - m$ desired input-output pattern pairs in (18). Steps 1-4 of our numerical-fuzzy approach are used to generate a FAM Bank based on the pattern pairs (18); then this FAM Bank is used to forecast $z(M + l)$ for $l = 1, 2, ...$ using the defuzzifying procedure of Step 5 of our numerical-fuzzy method, where the inputs to the network are $z(M+l-m), z(M+l-m+1), ..., z(M+l-1)$.

Example 3: Now we use these three methods (numerical-fuzzy, neural network, and AR model) for a real time series, namely, the monthly mean temperature of St. Louis, Missouri, from Jan. 1945 to Dec. 1974 (taken from Appendix III of [12]), as shown in Table 29. We simulated the following four cases: (1) two-year data from Jan. 1968 to Dec. 1969 was used to construct the systems (i.e., the FAM Bank, neural network, and AR model), and the resulting systems were used to predict the five-year temperatures from Jan. 1970 to Dec. 1974; (2) five-year data from Jan. 1965 to Dec. 1969 was used to construct the systems, and the resulting systems were used to predict the temperatures from Jan. 1970 to Dec. 1974; (3) thirteen-year data from Jan. 1957 to Dec. 1969 was used to construct the systems, and the resulting systems were used to predict the temperatures from Jan. 1970 to Dec. 1974; and, (4) twenty-five-year data from Jan. 1945 to Dec.1969 was used to construct the systems, and the resulting systems were used to predict the temperatures from Jan. 1970 to Dec. 1974. We chose $m = 9$ and $l = 1$ for all the simulations, i.e., the prediction systems have 9-inputs and one-output. They predict the temperature of a month based on the temperatures of the previous nine months. 40 hidden-layer neurons were used for the neural network predictor. For the numerical-fuzzy predictor, the membership function for any temperature is shown in Fig.17.

Predicted temperatures using the numerical-fuzzy, neural network, and AR model approaches are shown in: Figs.18-20 for case (1), Figs.21-23 for case (2), Figs.24-26 for case (3), and Figs.27-29 for case (4), respectively. In all simulations, the inputs to the resulting systems were the *true temperatures* when we predicted the temperatures from Jan.1970 to Dec. 1974, so that the error for a temperature estimate did not propagate to later estimates.

From the above simulation results we observe that: (1) when very few data were used to construct the systems (i.e., case (1)), our new numerical-fuzzy approach showed similar performance as the AR(9) model method, and higher performance than the neural network approach; and, (2) when more data were used to construct the systems (i.e., cases (2)-(4)), the three methods showed similar performances. Better performance using the numerical-fuzzy approach can be obtained by altering Fig.17 to include more categories; this kind of simulation will be performed in Example 4.

Example 4: Next we apply our numerical-fuzzy approach to chaotic time-series prediction [10,13]. Chaotic time series are generated from deterministic nonlinear systems and are sufficiently complicated that they appear to be "random" time series; however, because there are underlying deterministic maps that generate the series, chaotic time series are not random time series. In [10], feedforward neural networks were used for chaotic time-series prediction, and were compared with conventional approaches, like Linear Predictive Method, Gabor Polynomial Method, etc.. The results showed that the neural network approach gave the best prediction, and the accuracy obtained using the neural network approach was orders of magnitude higher than that obtained using the conventional approaches. Here we use our numerical-fuzzy approach applied to the same chaotic time series in [10], and compare the results obtained with those obtained using the neural network approach.

The chaotic time series is generated from the following delay differential equation:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t). \tag{19}$$

When $\tau > 17$, Eq.(19) shows chaotic behavior. Higher values of τ yield higher dimensional chaos. In our simulation, we chose the series with $\tau = 30$. Figure 30 shows 1000 points of this chaotic series which we used to test both the numerical-fuzzy and neural approaches.

As in Example 3, we chose $m = 9$ and $l = 1$, i.e., nine point values in the series were used to predict the value of the next time point. The membership function for any point is shown in Fig.31 for the numerical-fuzzy predictor. 40 hidden-layer neurons were used for the neural network predictor. The first 700 points of the series were used as training data, and the final 300 points were used as test data. We simulated three cases: (1) 100 training data (from 601 to 700) were used to construct the FAM Bank and to train the neural network; (2) 200 training data (from 501 to 700) were used; and, (3) 700 training data (from 1 to 700) were used. Figures 32 and 33 show the results of the numerical-fuzzy and neural predictors respectively for case (1); Figs.34 and 35 show similar results for case (2); and, Figs. 36 and 37 show similar results for case (3). As in [10], the "past" data needed to perform prediction is obtained from observing the actual time series; thus, one makes a prediction and uses the actual values to make the next prediction. We see from Figs.32 to 37 that our new numerical-fuzzy predictor gave slightly better results than the neural network predictor.

One advantage of the numerical-fuzzy approach is that it is very easy to modify the FAM Bank as new data become available. Specifically, when a new data pair becomes available, we create a rule for this data pair and add the new rule to the FAM Bank; then, the updated (i.e., adapted) FAM Bank is used to predict the future values. By using this "adaptive" procedure we use all the available information to predict the next value of the series. We simulated this adaptive procedure for the chaotic series of Fig.30: we started with the FAM Bank generated by the data $x(1)$ to $x(700)$, made a prediction of $x(701)$, then used the true value of $x(701)$ to update the FAM Bank, and this updated FAM Bank was then used to predict $x(702)$. This adaptive procedure continued until $x(1000)$.Its results are shown in Fig.38. Comparing Figs.38 and 36 we see that we obtain only a slightly improved prediction.

Finally, we show that prediction can be greatly improved by dividing the "domain interval" into finer regions. We performed two simulations: one with the membership function shown in Fig.39, and the other with the membership function shown in Fig.40. We used the adaptive-FAM-Bank procedure for both simulations. The results are shown in Figs.41 and 42, where Fig.41 (42) shows the result corresponding to the membership function of Fig.39 (40). Comparing Figs.38, 41 and 42 we see very clearly that we obtain better and better results as the "domain interval" is divided finer and finer. Figure 42 shows that we obtained an almost perfect prediction when we divided the "domain interval" into 29 regions. Of course, the price paid for doing this is that we use a larger FAM Bank.

6 CONCLUSIONS AND DISCUSSIONS

In this report, we developed a general method to generate fuzzy rules from numerical data. This method can be used as a general way to combine both numerical and linguistic information into a common framework – a Fuzzy-Associative-Memory (FAM) Bank. This FAM Bank consists of two kinds of fuzzy rules: some obtained from experts, and others generated from measured numerical data using the method of this report. We proved that the generated fuzzy system is capable of approximating any non-linear function on a compact set to arbitrary accuracy. We applied our new method to a truck backer-upper control problem [1,4], and observed that: (1) for the same training set (i.e., the same given input-output pairs), the final control performance of our new method is almost the same as that of the pure neural network controller; and, (2) in the case where neither numerical data nor linguistic rules contain enough information, both the pure neural and pure fuzzy methods failed to control the truck to the desired position, but our new method succeeded. We also applied our new method to real and chaotic time-series prediction problems, and the results showed that our new method had the best performance compared to an AR model predictor and a neural network predictor.

The main features and advantages of the new method developed in this paper are: (1) it provides us with a general method to combine measured numerical information and human lingustic information into a common framework – a combined FAM Bank; this could be viewed as a first step to develop some *theoretically analyzable* control algorithms which use both numerical and linguistic information; (2) it is a simple and straightforward one-pass build-up procedure; hence, no time-consuming iterative training is required, so that it requires much less construction time than a comparable neural network; (3) there is lots of freedom in choosing the membership functions; this provides us with lots of flexibilities to design systems according to different requirements (this advantage was not utilized in this report, but will be fully explored in subsequent publications); and, (4) it can perform successful control for some cases where neither a pure neural network control nor a pure fuzzy control can.

It is interesting to compare our numerical-fuzzy approach with feedforward neural networks when no linguistic information is available. They are similar in that they are given the same information and asked to do the same task. All the information they are given is contained in some "examples" from which they are required to "learn" so as to give successful or correct outputs when new inputs are presented. Additionally, both approaches are model-free.

Our numerical-fuzzy approach has the following important advantages over feedforward neural networks: (1) the construction of a numerical-fuzzy controller is just a one-pass build-up procedure, whereas the construction of a feedforward neural network controller needs time-consuming iterative training; (2) the choice of the number of hidden-layer neurons of a feedforward neural network is very difficult and quite problem dependent; an inaccurate choice of the number of hidden-layer neurons may cause either too large a network or a network which cannot match all the desired patterns; our numerical-fuzzy approach has no such problem; (3) it is more general than feedforward neural networks, i.e., our numerical-fuzzy approach can use not only the desired input-output patterns (i.e., "examples") but also linguistic rules from human experts, whereas the feedforward neural networks can only utilize "examples"; (4) it is more flexible than feedforward neural networks, i.e., we can choose the membership functions in a wide variety of forms and divide the "domain interval" into different regions; and, (5) it is very easy to update the FAM Bank, hence it lends itself to adaptive control or time-series prediction procedures.

In summary, the numerical-fuzzy approach developed in this report can be applied to all problems which are suitable for feedforward neural networks. If the performance of the numerical-fuzzy approach is similar to that of a feedforward neural network approach, as for the truck backer-upper control problem, then the numerical-fuzzy approach is strongly recommended because of the above advantages.

7 REFERENCES

[1] D.Nguyen and B.Widrow, "The Truck Backer-Upper: An Example of Self-Learning in Neural Network," *IEEE Control Systems Magazine*, Vol.10, No.3, pp.18-23, 1990.

[2] G.E.P.Box and G.M.Jenkins, "Time Series Analysis: Forecasting and Control," Holden-Day Inc., 1976.

[3] Y.F.Li and C.C.Lan, "Development of Fuzzy Algorithms for Servo Systems," *IEEE Control Systems Magazine*, Vol.9, No.3, pp.65-72, 1989.

[4] S.G.Kong and B.Kosko, "Comparison of Fuzzy and Neural Truck Backer-Upper Control Systems," *Proc. IJCNN-90*, Vol.3, pp.349-358, June, 1990.

[5] W.Rudin, *Principles of Mathematical Analysis*, New York: McGraw-Hill, 1964.

[6] K.Hornik, M.Stinchcombe and H.White, "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, Vol.2, pp.359-366, 1989.

[7] G.Cybenko, "Approximations by Superpositions of a Sigmoidal Function," *Mathematics of Control, Singals, and Systems*, 1989.

[8] F.V.D.Rhee, H.R.Lemke and J.G.Dijkman, "Knowledge Based Fuzzy Control of Systems," *IEEE Trans. on Automatic Contr.*, Vol-35, No.2, pp.148-155, 1990.

[9] R.P.Lippman, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, pp.4-22, April, 1987.

[10] A.Lapedes and R.Farber, "Nonlinear Signal Processing Using Neural Networks: Prediction and System Modeling," *LA-UR-87-2662*, 1987.

[11] A.Khotanzad and J.H.Lu, "Non-parametric Prediction of AR Processes Using Neural Networks," *Proc. 1990 ICASSP*, pp.2551-2554, Albuquerque, April, 1990.

[12] S.L.Marple Jr., "*Digital Spectral Analysis with Applications*," Prentice-Hall, Inc., 1987.

[13] J.D.Scargle, "Modeling Chaotic and Random Processes," submitted to *Astrophysical Journal*, 1989.

[14] B.Kosko, *Neural Networks and Fuzzy Systems*, Prentice-Hall, 1991.

A APPENDIX A

Proof of Lemma 1: Under the condition of this lemma, there exists at least one i such that $\Pi_{1\leq j\leq n}[m_j^i(x_j)] \neq 0$ for any $\underline{x} \in Q$, hence Eq.(11) is well-defined. Q.E.D..

Proof of Lemma 2: Since $0 \leq m_j^i(x_j) \leq 1$, it is sufficient to prove that for any $\underline{x} \in Q$ there exist some i such that $\Pi_{1\leq j\leq n}[m_j^i(x_j)] \neq 0$. Since every box in the FAM Bank has a rule, for any $\underline{x} \in Q$ there must be a rule, say Rule i, such that $x_j \in RG_j^i$ for $j = 1, 2, ..., n$. By AS.2, $m_j^i(x_j) \neq 0$ for all $j = 1, 2, ..., n$, hence $\Pi_{1\leq j\leq n}[m_j^i(x_j)] \neq 0$. Q.E.D..

In order to prove Theorem 1 we need some definitions. A family F of real functions defined on a set E is an *algebra* if F is closed under addition, multiplication, and scalar multiplication. The family F *separates points on* E if for every $x, y \in E$, $x \neq y$, there exists a function $f \in F$ such that $f(x) \neq f(y)$. The family F *vanishes at no point of* E if for each $x \in E$ there exists $f \in F$ such that $f(x) \neq 0$. Our proof of Theorem 1 is based on the Stone-Weierstrass Theorem [5] which we state here for convenience of the reader.

Stone-Weierstrass Theorem: Let F be an algebra of real continuous functions on a compact set K. If F separates points on K and if F vanishes at no point on K, then the uniform closure B of F consists of all real continuous functions on K.

The uniform closure B of F is the union of F and its limit points; hence, if B consists of all real continuous functions on K, then F is capable of approximating any real continuous function on K to arbitrary accuracy.

Proof of Theorem 1: Let F be the family of well-defined functions of the form of Eq.(11) on the compact set Q under AS.1, AS.2, and AS.3. If we prove that F is an algebra of real continuous functions, F separates points on Q, and F vanishes at no point of Q, then the Stone-Weierstrass Theorem guarantees the conclusion of Theorem 1.

By AS.2, the $m_j^i(x_j)'s$ are assumed to be real continuous functions; hence, F is a family of real continuous functions. Let $f_1, f_2 \in F$, so that we can write them as

$$f_1(\underline{x}) = \frac{\sum_{i=1}^{K1} \bar{y1}^i \Pi_{1\leq j\leq n}[m1_j^i(x_j)]}{\sum_{i=1}^{K1} \Pi_{1\leq j\leq n}[m1_j^i(x_j)]}, \tag{A.1}$$

$$f_2(\underline{x}) = \frac{\sum_{i=1}^{K2} \bar{y2}^i \Pi_{1\leq j\leq n}[m2_j^i(x_j)]}{\sum_{i=1}^{K2} \Pi_{1\leq j\leq n}[m2_j^i(x_j)]}. \tag{A.2}$$

Now,

$$\begin{aligned} & f_1(\underline{x}) + f_2(\underline{x}) \\ = & \frac{\sum_{i1=1}^{K1}\sum_{i2=1}^{K2} \bar{y1}^{i1}\Pi_{1\le j\le n}[m1_j^{i1}(x_j)m2_j^{i2}(x_j)] + \sum_{i1=1}^{K1}\sum_{i2=1}^{K2} \bar{y2}^{i2}\Pi_{1\le j\le n}[m1_j^{i1}(x_j)m2_j^{i2}(x_j)]}{\sum_{i1=1}^{K1}\sum_{i2=1}^{K2} \Pi_{1\le j\le n}[m1_j^{i1}(x_j)m2_j^{i2}(x_j)]} \\ = & \frac{\sum_{i1=1}^{K1}\sum_{i2=1}^{K2}(\bar{y1}^{i1}+\bar{y2}^{i2})\Pi_{1\le j\le n}[m1_j^{i1}(x_j)m2_j^{i2}(x_j)]}{\sum_{i1=1}^{K1}\sum_{i2=1}^{K2} \Pi_{1\le j\le n}[m1_j^{i1}(x_j)m2_j^{i2}(x_j)]}. \end{aligned} \quad \text{(A.3)}$$

Define $m1_j^{i1}(x_j)m2_j^{i2}(x_j)$ as a new membership function of x_j, say $m_j^{i1,i2}(x_j)$, and define $\bar{y1}^{i1}+\bar{y2}^{i2}$ as the output center of a new rule, say $\bar{y}^{i1,i2}$; then, Eq.(A.3) is of the form of Eq.(11); hence, $f_1+f_2 \in F$. Similarly, $f_1(\underline{x})f_2(\underline{x})$ can be written as

$$f_1(\underline{x})f_2(\underline{x}) = \frac{\sum_{i1=1}^{K1}\sum_{i2=1}^{K2} \bar{y1}^{i1}\bar{y2}^{i2}\Pi_{1\le j\le n}[m1_j^{i1}(x_j)m2_j^{i2}(x_j)]}{\sum_{i1=1}^{K1}\sum_{i2=1}^{K2} \Pi_{1\le j\le n}[m1_j^{i1}(x_j)m2_j^{i2}(x_j)]}, \quad \text{(A.4)}$$

which is of the form of Eq.(11); hence, $f_1f_2 \in F$. Finally, for any $c \in R$,

$$cf_1(\underline{x}) = \frac{\sum_{i=1}^{K1} c\bar{y1}^{i}\Pi_{1\le j\le n}[m1_j^{i}(x_j)]}{\sum_{i=1}^{K1} \Pi_{1\le j\le n}[m1_j^{i}(x_j)]}. \quad \text{(A.5)}$$

Define $c\bar{y1}^i$ as new centers of output fuzzy regions, so that Eq.(A.5) is of the form of Eq.(11); hence, $cf_1(\underline{x}) \in F$. In summary, F is an algebra of real continuous functions.

Next, we prove that F separates points on Q. Let $\underline{x}, \underline{z} \in Q$ and $\underline{x} \neq \underline{z}$. We now construct $f \in F$ with $f(\underline{x}) \neq f(\underline{z})$. First, we define the fuzzy regions of the input space Q such that each element of $\underline{x}$ and $\underline{z}$ is at the center of a fuzzy region (recall that the center of a fuzzy region is defined as the point which has the smallest absolute value among all the points at which the membership function for this region has membership value equal to one; see Section 2 Step 5). Then, we choose the membership functions for the input space Q to be of the specific triangular form defined by AS.4 of Section 3. By such a choice of fuzzy regions and membership functions, we have $m_j^i(x_j) = m_j^l(z_j) = 1$ for each active Rule i at $\underline{x}$, and each active Rule l at $\underline{z}$, and all $j = 1, 2, ..., n$ (the definition of *active rule* is given in Section 3); additionally, there is one and only one active rule for $\underline{x}$ and one and only one active rule for $\underline{z}$, because Eq.(11) is well-defined (which guarantees that there is at least one active rule for $\underline{x}$ and at least one active rule for $\underline{z}$), and because only the membership functions for the regions with centers at the components of $\underline{x}$ or $\underline{z}$ are nonzero at $\underline{x}$ or $\underline{z}$, whereas all other membership functions are zero at $\underline{x}$ and $\underline{z}$ (which guarantees that there is at most one active rule for $\underline{x}$ and at most one active rule for $\underline{z}$). Since $\underline{x} \neq \underline{z}$, there must be at least one j such that $x_j \neq z_j$, hence the only active rule for $\underline{x}$ and the only active rule for $\underline{z}$ are at two different boxes of the FAM Bank. Since we are free to assign any rules to the boxes of the FAM Bank (AS.3), we just assign two different rules to these two boxes, and obtain the required $f \in F$ with $f(\underline{x}) = \bar{y}^i \neq \bar{y}^l = f(\underline{z})$ (see Eq.(11)), where $\bar{y}^i$ ($\bar{y}^l$) is the center of the output region of the active rule for $\underline{x}$ ($\underline{z}$).

Finally, we prove that F vanishes at no point of Q. By AS.1 and AS.2, we can make all the $\bar{y}^i > 0$. Since Eq.(11) is well-defined, there exists at least one i such that $\Pi_{1\leq j\leq n}[m_j^i(x_j)] \neq 0$ for any $\underline{x} \in Q$. Since Eq.(11) is a weighted sum of positive $\bar{y}^i$'s with some nonzero weights, the result is also positive, i.e., we obtain $f \in F$ such that $f(\underline{x}) \neq 0$ (in fact, $f(\underline{x}) > 0$) for any $\underline{x} \in Q$. Q.E.D..

Proof of Lemma 3: For arbitrary $\underline{x} \in Q$ and fixed j, there are at most two m_j^i's which are nonzero at x_j under AS.4. Since a rule, say Rule i, is active only when $m_j^i(x_j) \neq 0$ for all $j = 1, 2, ..., n$, there are at most 2^n active rules for any $\underline{x} \in Q$; this proves (1). If r components of $\underline{x} \in Q$ are at the centers of some fuzzy regions, there is only one m_j^i which is nonzero at each of these r components (in fact, these m_j^i's are equal to unity at these r components), and for each of the other $n-r$ components there are at most two nonzero m_j^i's, hence the total number of active rules is at most 2^{n-r}; this proves (2). If r components of $\underline{x} \in Q$ are at the centers of some fuzzy regions and q components of the $\underline{x}$ are smaller (or greater) than the center values of the corresponding smallest (or the corresponding largest) fuzzy regions, then there is only one nonzero m_j^i for each of these $r+q$ components, the other $n-r-q$ components have two nonzero m_j^i's associated with each of them, hence the total number of active rules is at most 2^{n-r-q}; this proves (3). Q.E.D..

B APPENDIX B

Since we are only interested in the movement of the central rear point of the truck and the angle of the truck to horizontal, we can simplify the truck to a bar with length equal to b. We obtain the kinematic equations of the bar according to the following basic principle of object kinematics: the movement of a point on an object equals the sum of the movement of the mass center of the object and the movement of the point around the mass center. The details are shown in Fig.B1. From the figure we have

$$x^m(t) = x(t) + rcos[\phi(t) + \theta(t)] \tag{B.1}$$

$$y^m(t) = y(t) + rsin[\phi(t) + \theta(t)] \tag{B.2}$$

$$\phi^m(t) = \phi(t) \tag{B.3}$$

where $(x^m(t), y^m(t), \phi^m(t))$ denote the states of the bar after the movement of the mass center by a distance r. The final state $(x(t+1), y(t+1), \phi(t+1))$ after the rotation of the mass-center-moved bar is

$$x(t+1) = x^m(t) + [rsin(\theta(t))]sin(\phi(t)) \tag{B.4}$$

$$y(t+1) = y^m(t) - [rsin(\theta(t))]cos(\phi(t)) \tag{B.5}$$

$$\phi(t+1) = \phi^m(t) - sin^{-1}[\frac{rsin(\theta(t))}{b/2}]. \tag{B.6}$$

Substituting Eqs.(B.1)-(B.3) into Eqs.(B.4)-(B.6) and setting $r = 1$, we obtain Eqs.(13)-(15). There are some approximations in obtaining Eqs.(B.4)-(B.6), as shown in Fig.B1.

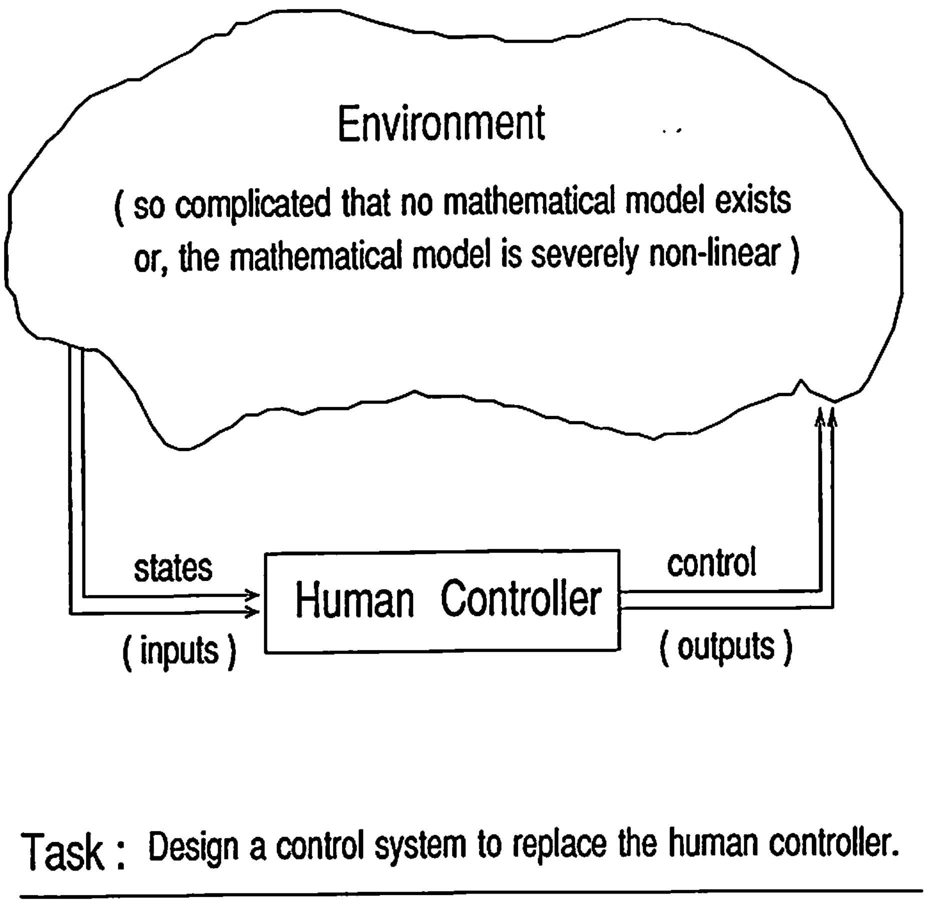

Environment
(so complicated that no mathematical model exists
or, the mathematical model is severely non-linear)
states
(inputs)
Human Controller
control
(outputs)
Task : Design a control system to replace the human controller.

Figure 1: A practical problem: design a control system to replace the human controller.

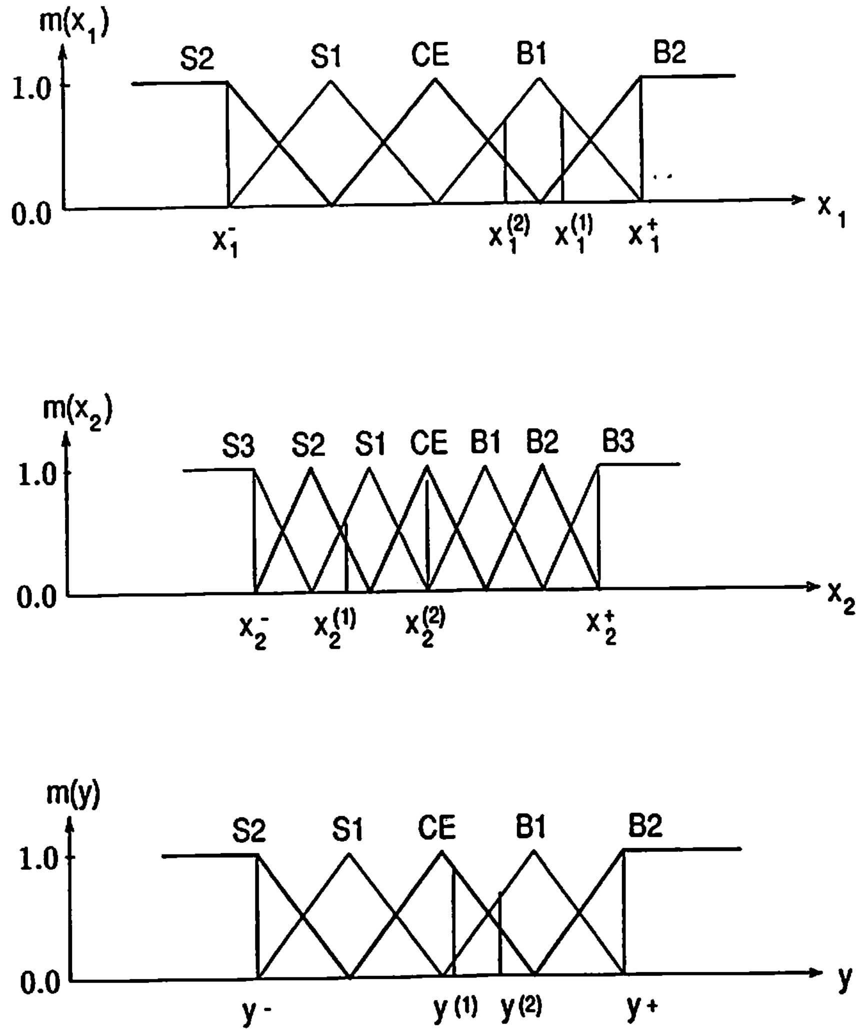


Figure 2: Divisions of the input and output spaces into fuzzy regions and the corresponding membership functions.

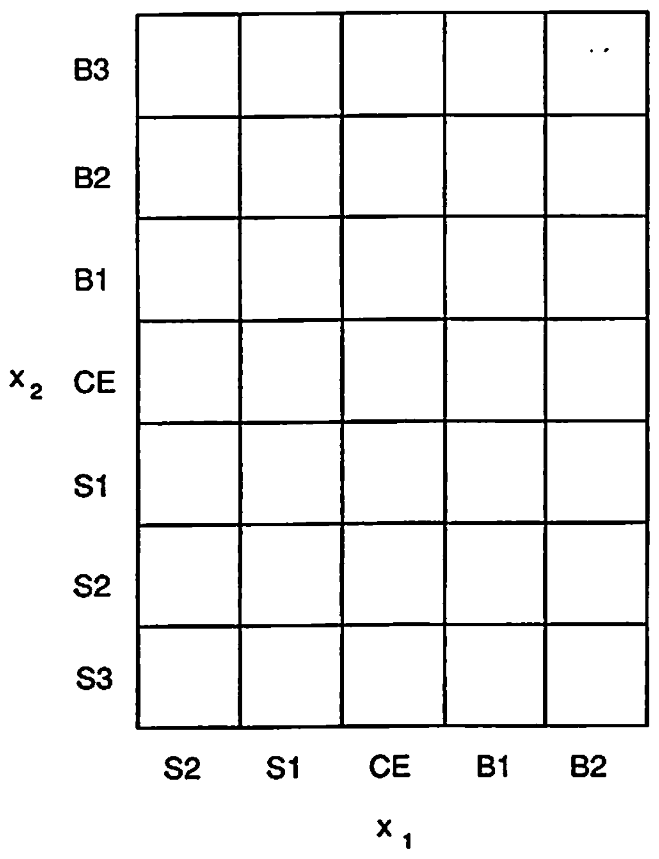
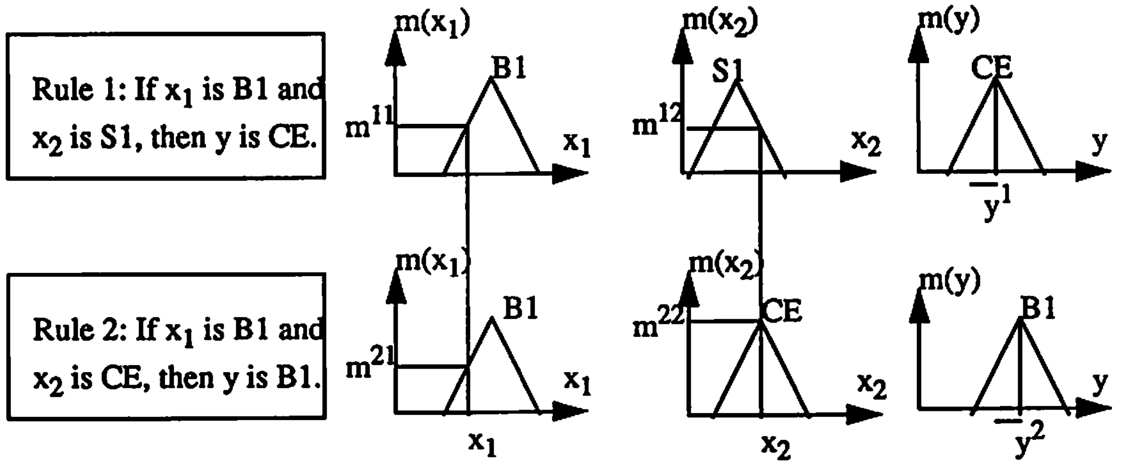


Figure 3: The form of a FAM Bank.


Rule 1: If x1 is B1 and x2 is S1, then y is CE.
Rule 2: If x1 is B1 and x2 is CE, then y is B1.
m(x1)
B1
m11
x1
m(x2)
S1
m12
x2
m(y)
CE
y
y1
m(x1)
B1
m21
x1
x1
m(x2)
CE
m22
x2
x2
m(y)
B1
y
y2

$$y = \frac{m_{CE}^{1}\,\bar{y}^{1} + m_{B1}^{2}\,\bar{y}^{2}}{m_{CE}^{1} + m_{B1}^{2}}$$

$$m_{CE}^{1} = m^{11} m^{12}$$

$$m_{B1}^{2} = m^{21} m^{22}$$

Figure 4: Determine control strategy from FAM Bank using product fuzzy inference with centroid defuzzification.

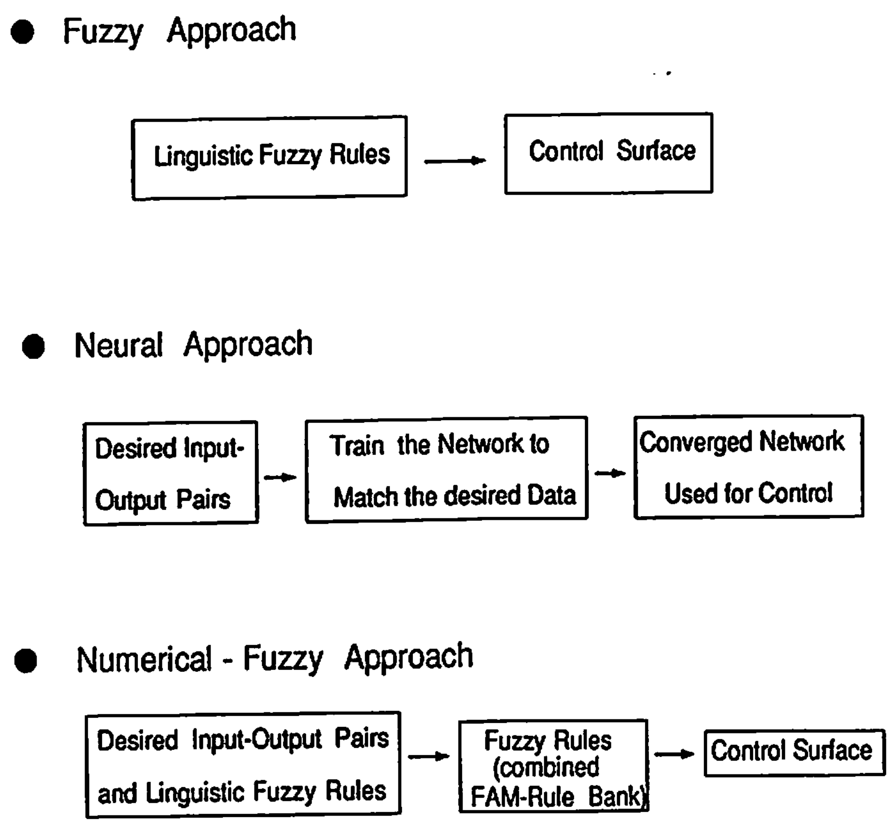

Fuzzy Approach
Linguistic Fuzzy Rules
Control Surface
Neural Approach
Desired Input-Output Pairs
Train the Network to Match the desired Data
Converged Network Used for Control
Numerical - Fuzzy Approach
Desired Input-Output Pairs and Linguistic Fuzzy Rules
Fuzzy Rules (combined FAM-Rule Bank)
Control Surface

Figure 5: Key steps of fuzzy, neural, and numerical-fuzzy approaches to the truck back-upper control problem.

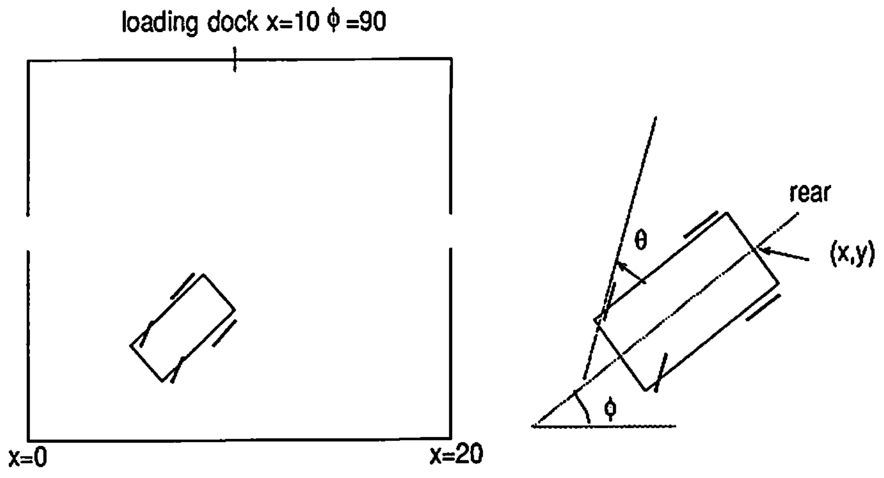


Figure 6: Diagram of simulated truck and loading zone.

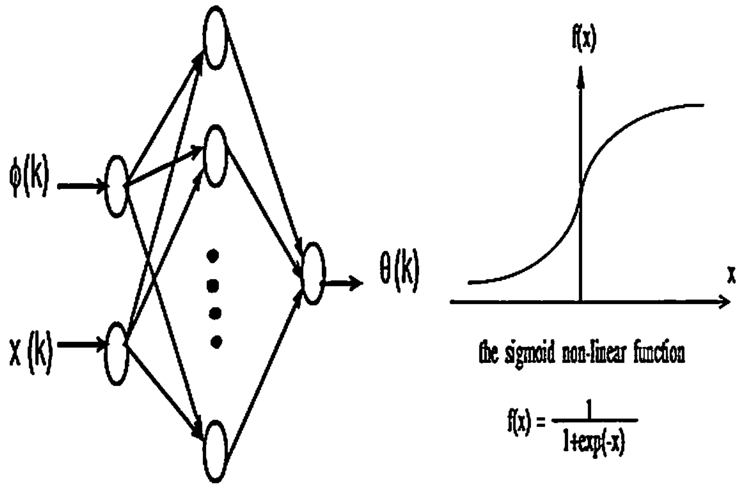


Figure 7: Neural controller for the truck backer-upper control problem.

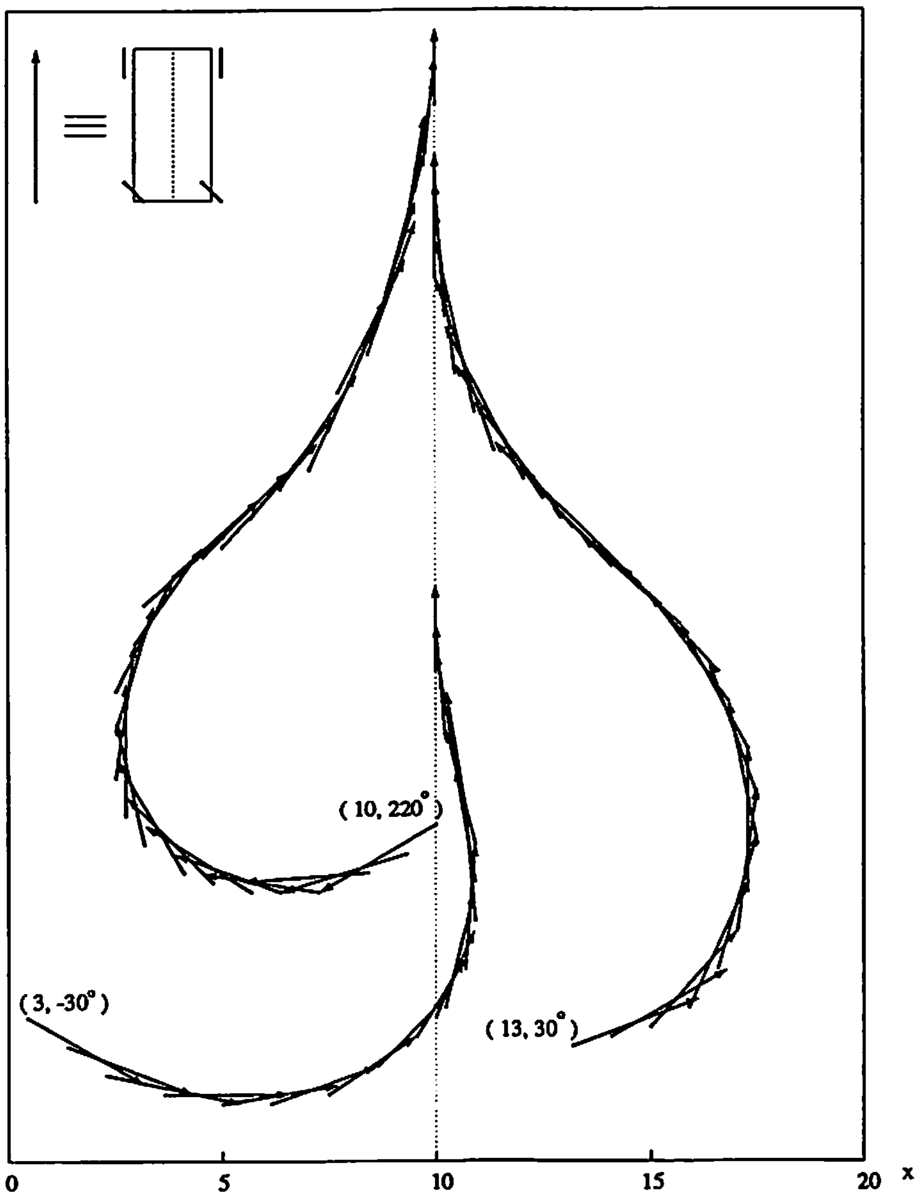
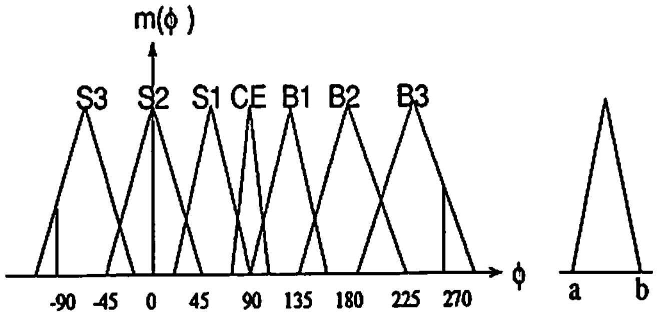
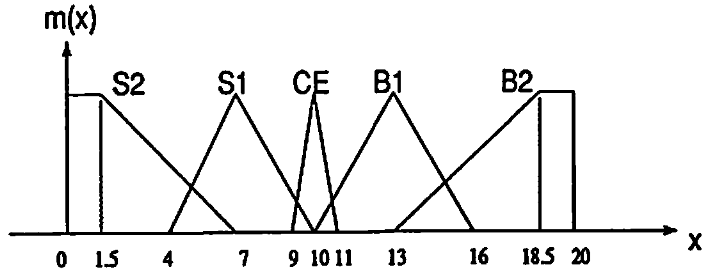


Figure 8: Truck trajectories using the neural controller.



	a	b
S3	-115	-15
S2	-45	45
S1	15	90
CE	80	100
B1	90	165
B2	135	225
B3	195	295



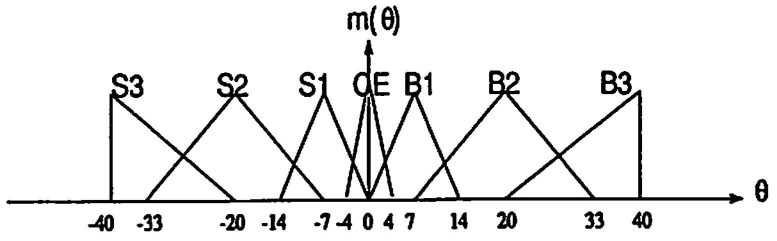


Figure 9: Fuzzy membership functions for the truck backer-upper control problem.

X

		S2	S1	CE	B1	B2
	S3	S2	S3			
	S2	S2	S3	S3	S3	
	S1	B1	S1	S2	S3	S2
ϕ	CE	B2	B2	CE	S2	S2
	B1	B2	B3	B2	B1	S1
	B2		B3	B3	B3	B2
	B3				B3	B2

Figure 10: The final FAM Bank generated from the numerical data in Tables 1-14.

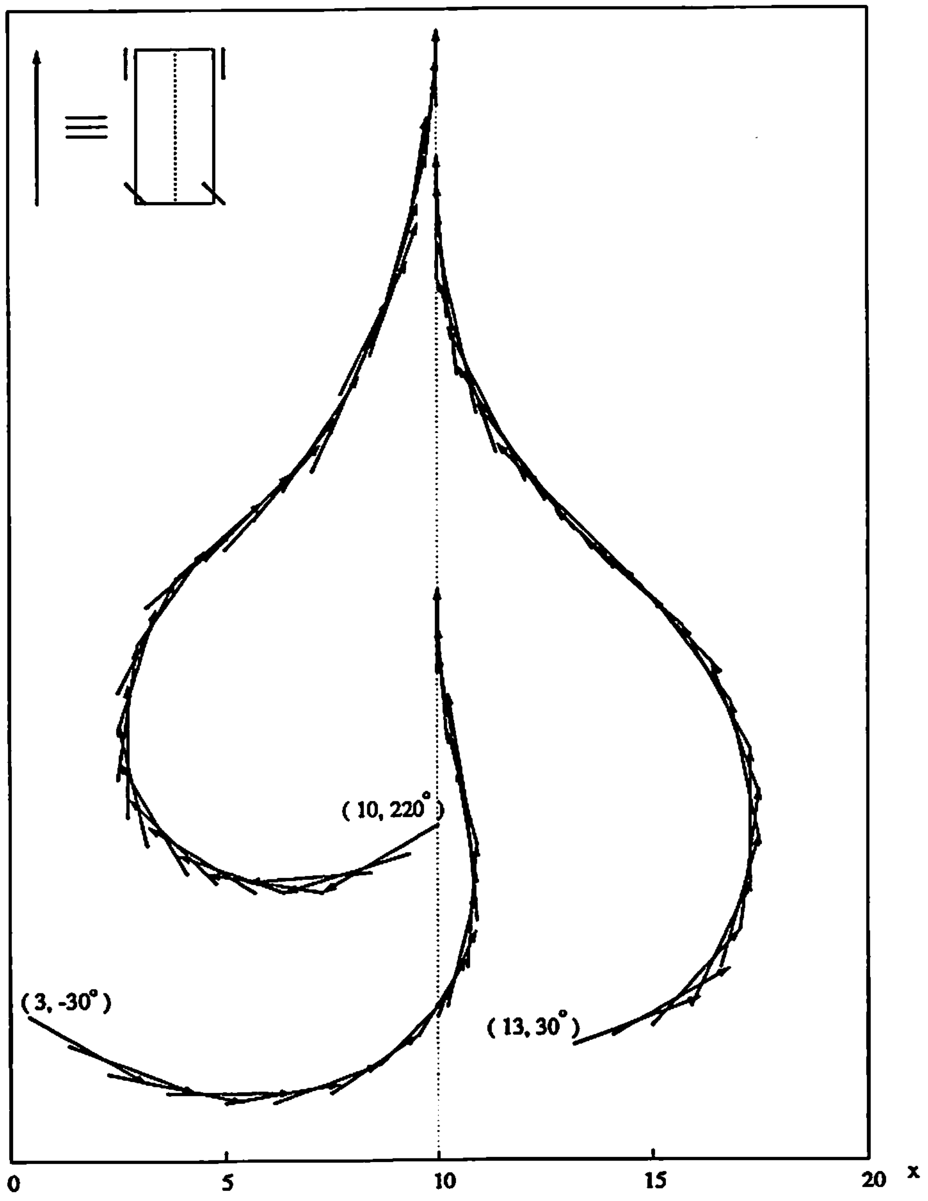


Figure 11: Truck trajectories using the numerical-fuzzy controller for Example 1. Also the truck trajectories using the combined FAM Banks for Example 2.

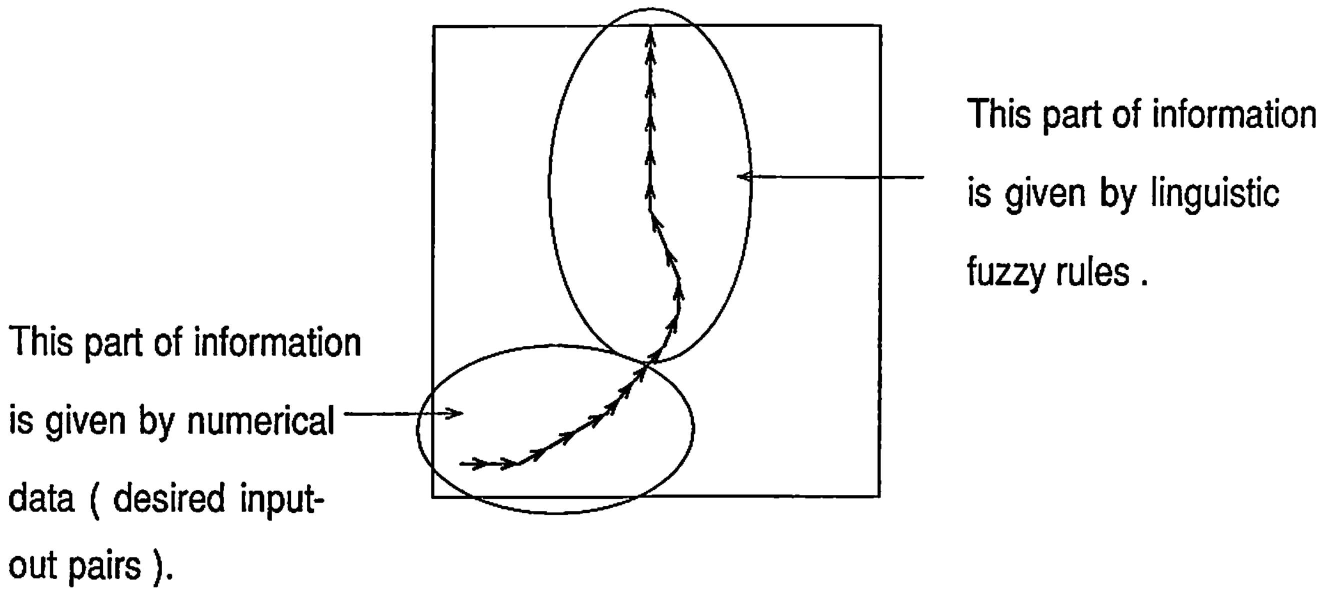

This part of information is given by linguistic fuzzy rules .
This part of information is given by numerical data (desired input-out pairs).

Figure 12: An example of truck back-upper control problem where the beginning part of information is given by desired input-output pairs while the ending part of information is given by linguistic rules.

		X				
		S2	S1	CE	B1	B2
	S3	S2	S3			
	S2	S2	S3	S3	S3	
	S1	B1	S1		S3	S2
ϕ	CE	B2	B2		S2	S2
	B1	B2	B3		B1	S1
	B2		B3	B3	B3	B2
	B3				B3	B2

Figure 13: The FAM Bank generated from the truncated numerical data which consist of only the first three pairs of each of the sequences in Tables 1-14.

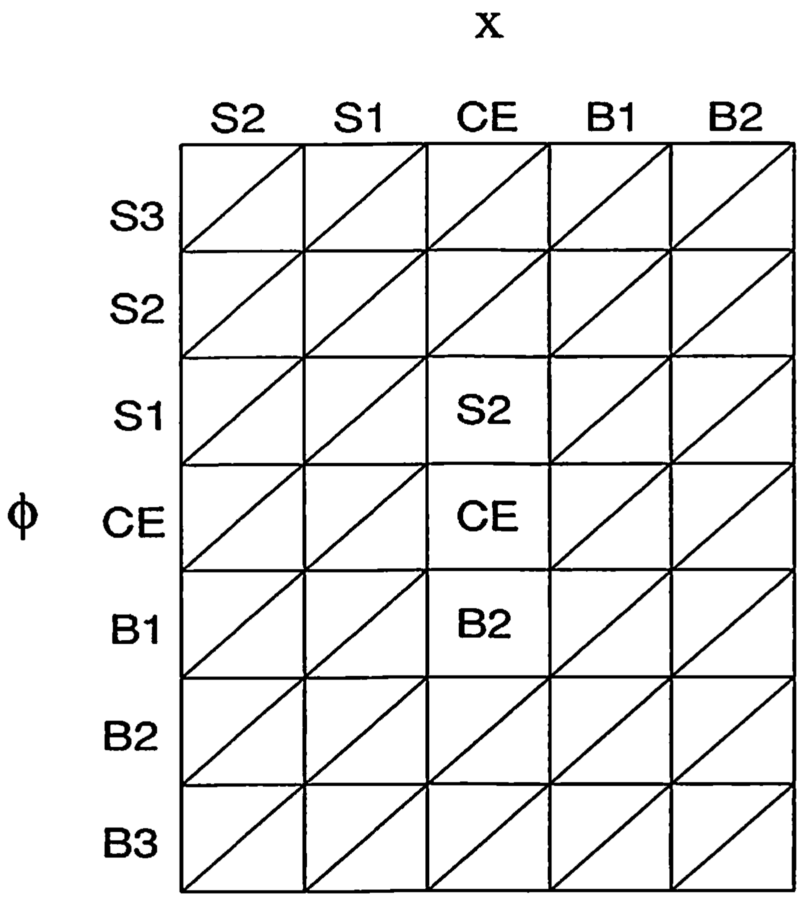


Figure 14: The FAM Bank of linguistic rules.

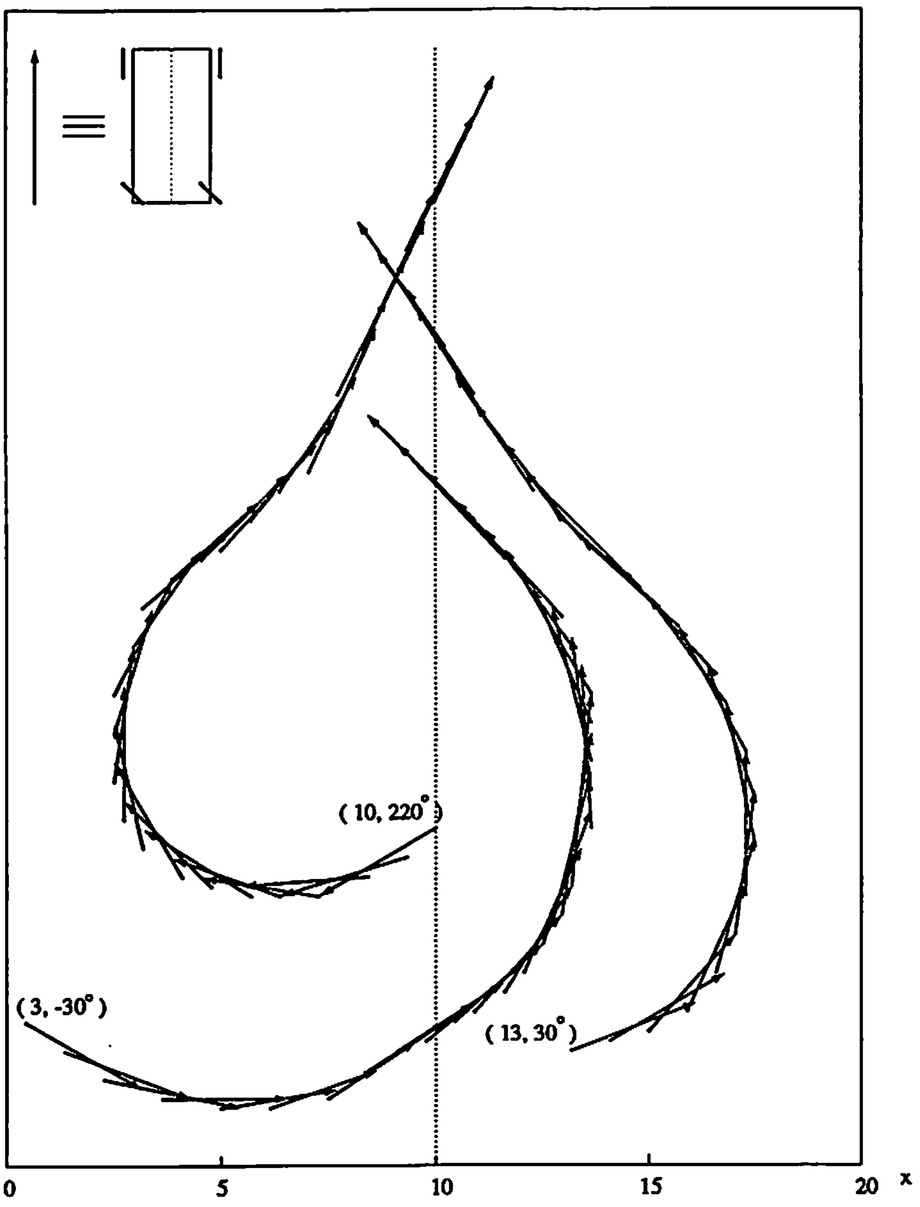


Figure 15: Truck trajectories using the FAM Bank generated from the truncated data pairs only.

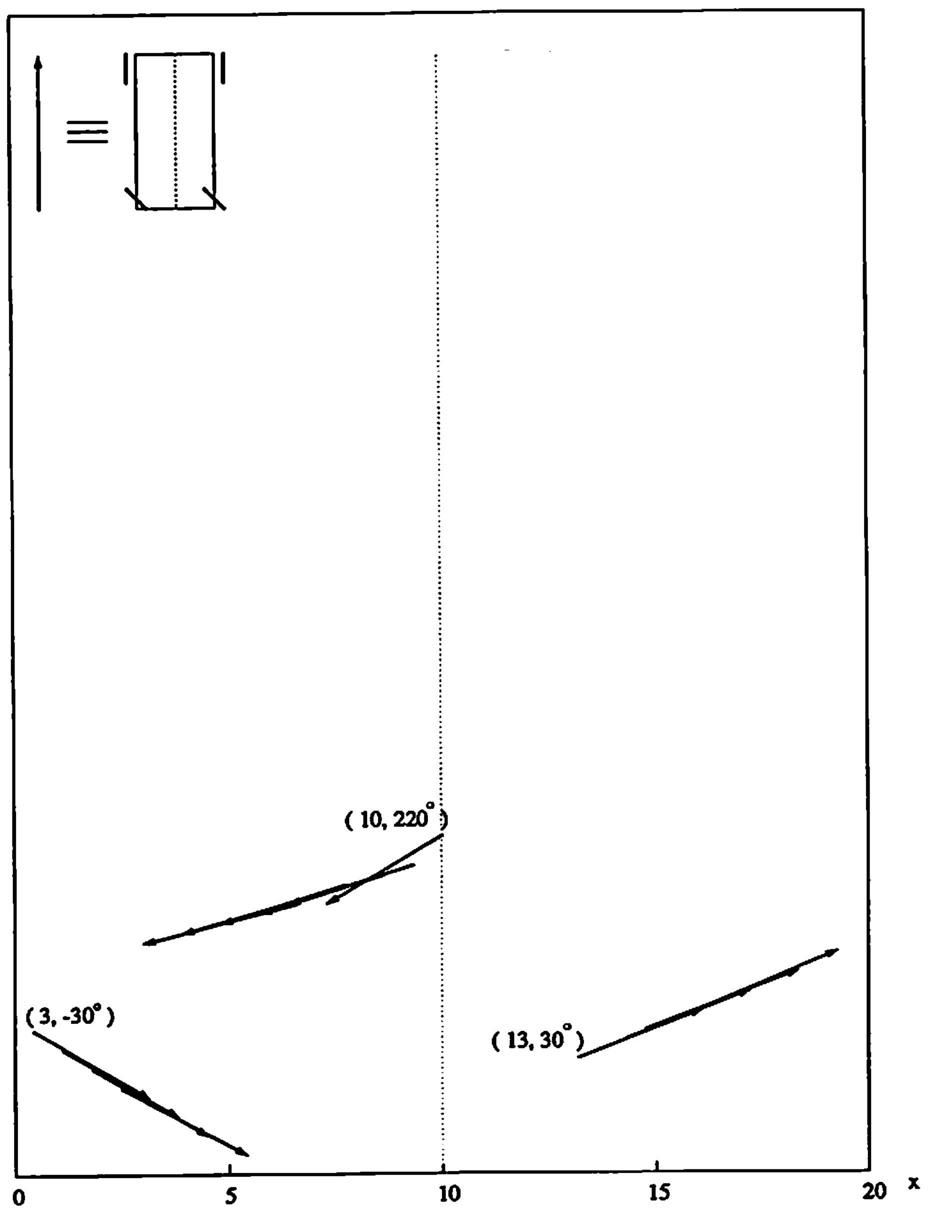


Figure 16: Truck trajectories using the FAM Bank of selected linguistic rules only.

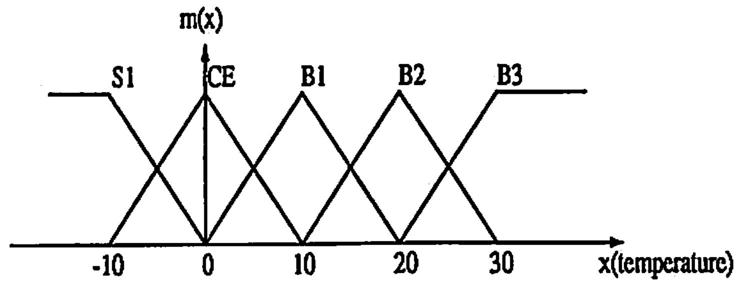


Figure 17: Membership function for the temperature prediction problem.

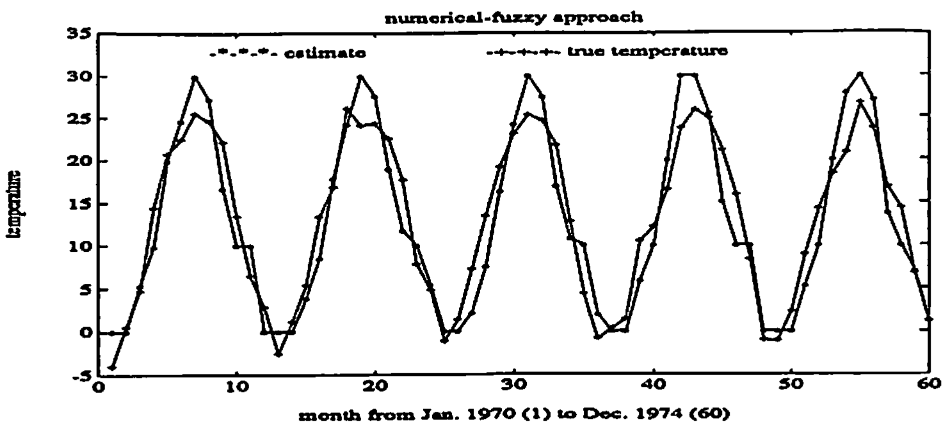


Figure 18: Temperature estimates of 1970-1974 using the numerical-fuzzy approach based on the data of 1968-1969.

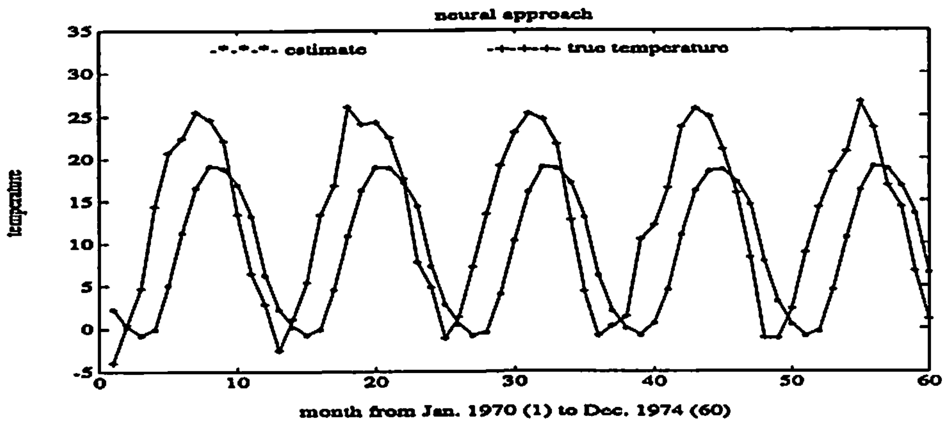


Figure 19: Temperature estimates of 1970-1974 using the neural approach based on the data of 1968-1969.

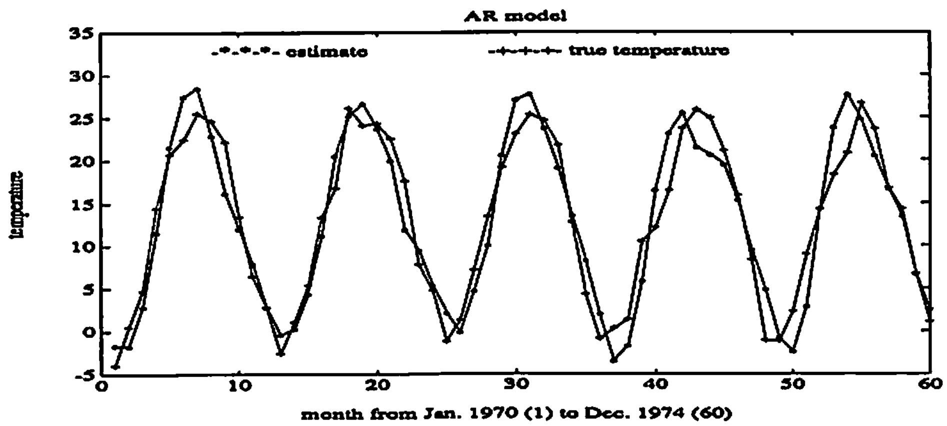


Figure 20: Temperature estimates of 1970-1974 using the AR model approach based on the data of 1968-1969.

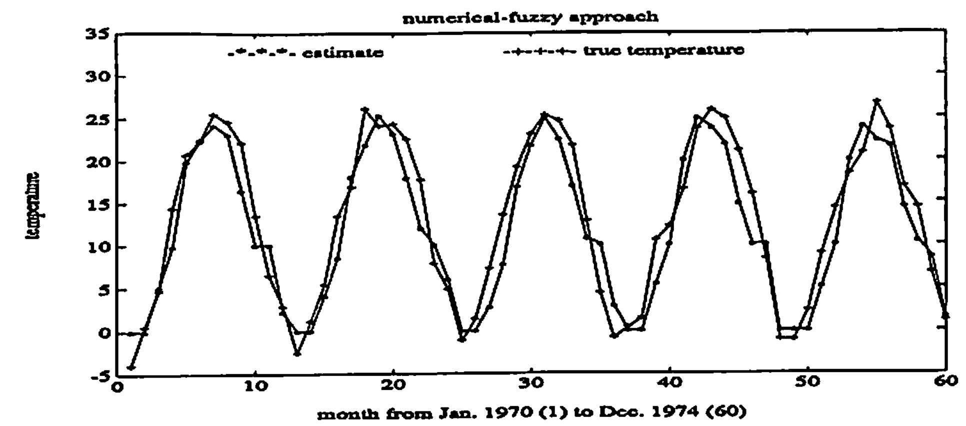


Figure 21: Temperature estimates of 1970-1974 using the numerical-fuzzy approach based on the data of 1965-1969.

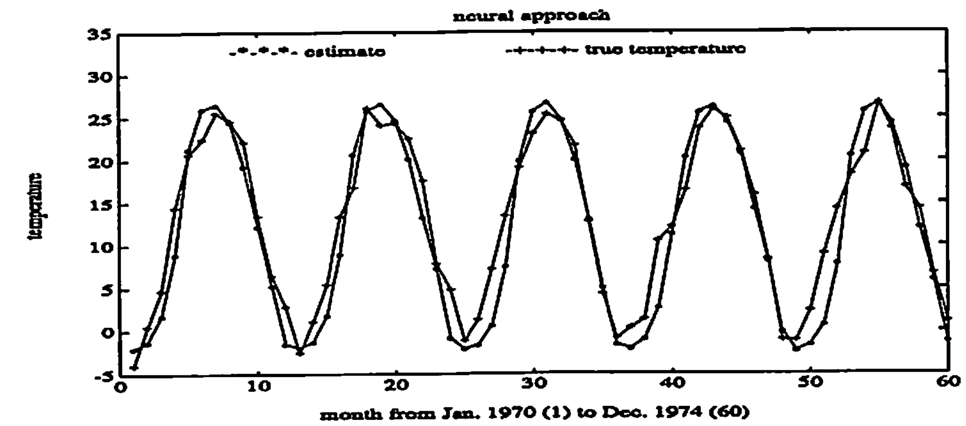


Figure 22: Temperature estimates of 1970-1974 using the neural approach based on the data of 1965-1969.

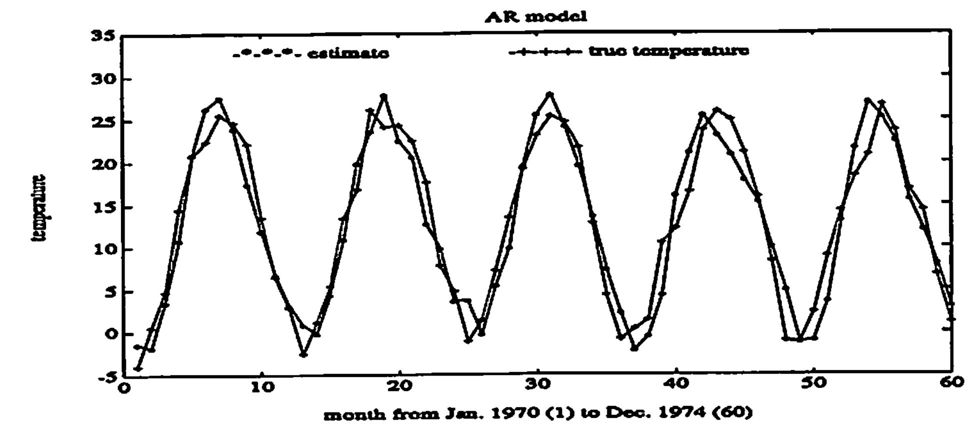


Figure 23: Temperature estimates of 1970-1974 using the AR model approach based on the data of 1965-1969.

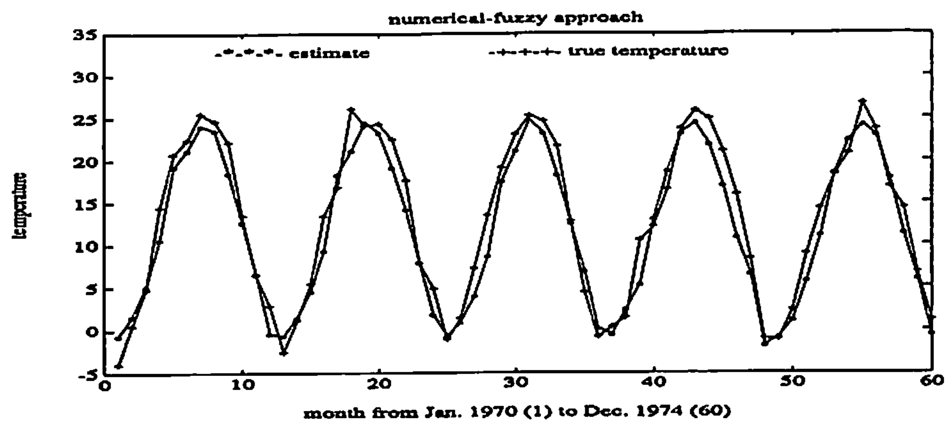


Figure 24: Temperature estimates of 1970-1974 using the numerical-fuzzy approach based on the data of 1957-1969.

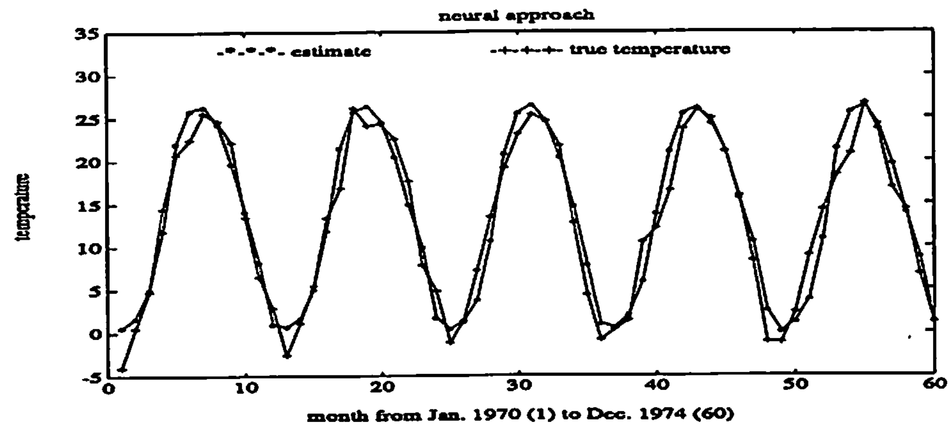


Figure 25: Temperature estimates of 1970-1974 using the neural approach based on the data of 1957-1969.

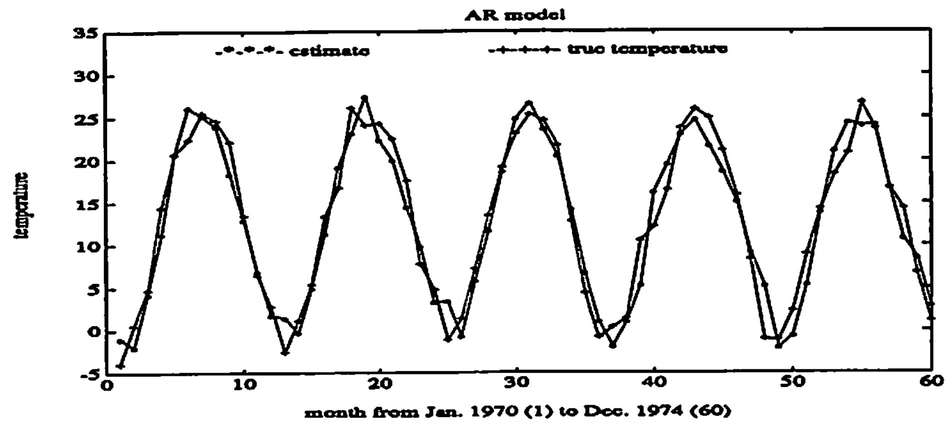


Figure 26: Temperature estimates of 1970-1974 using the AR model approach based on the data of 1957-1969.

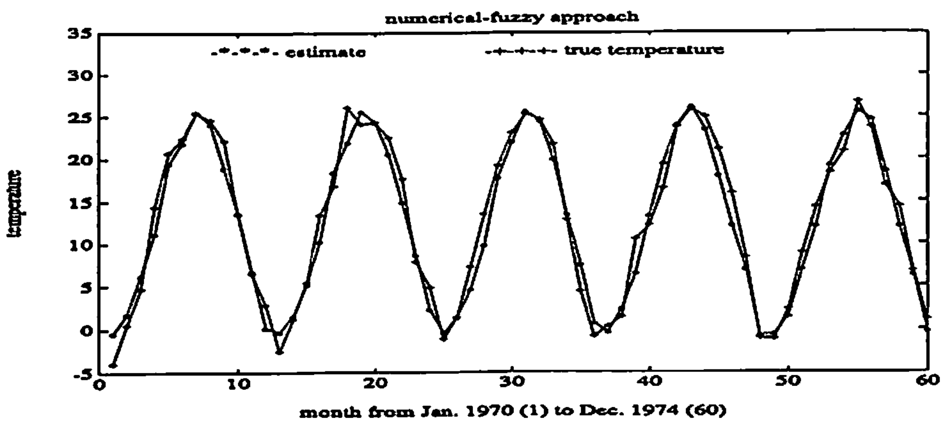


Figure 27: Temperature estimates of 1970-1974 using the numerical-fuzzy approach based on the data of 1945-1969.

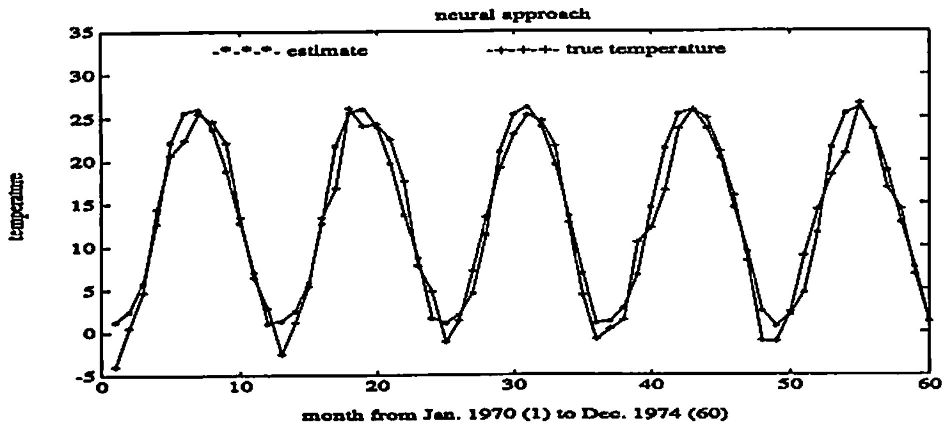


Figure 28: Temperature estimates of 1970-1974 using the neural approach based on the data of 1945-1969.

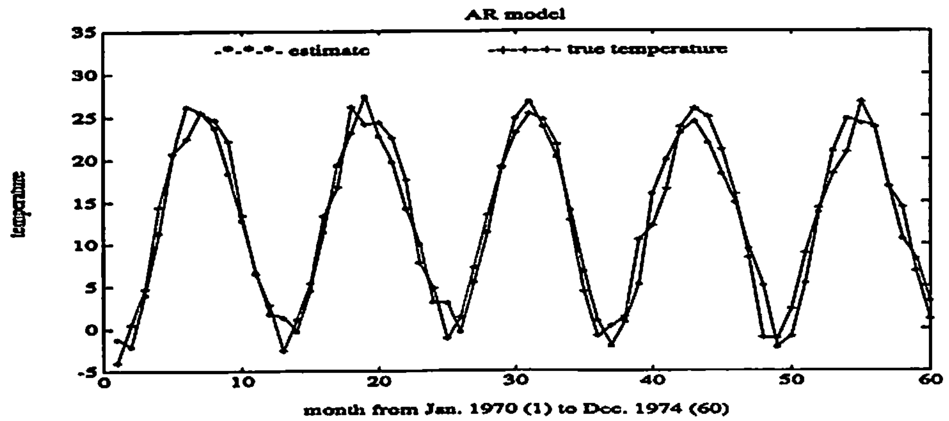


Figure 29: Temperature estimates of 1970-1974 using the AR model approach based on the data of 1945-1969.

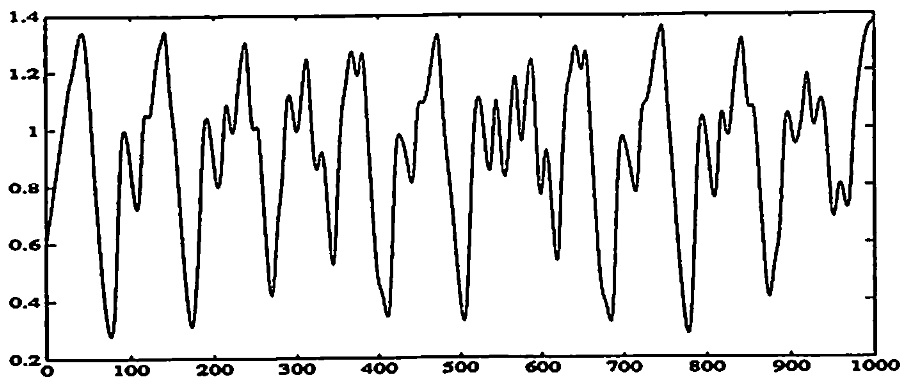


Figure 30: A chaotic time series.

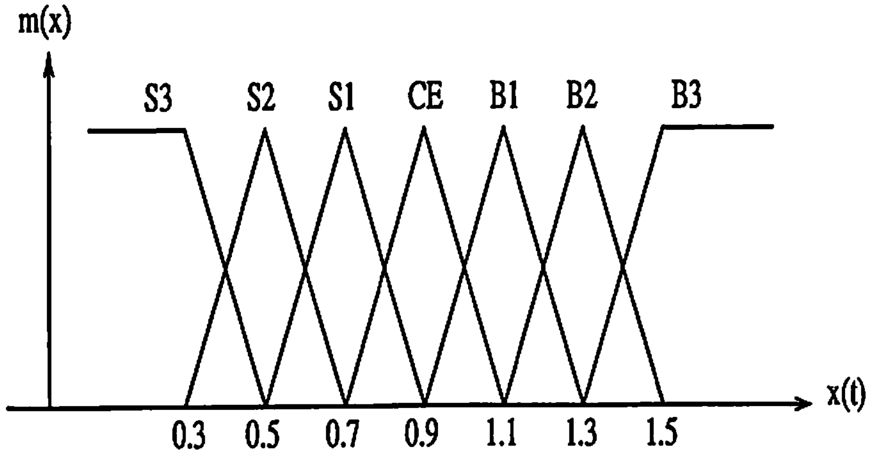


Figure 31: Membership function for the simulations of Figs.32 to 37.

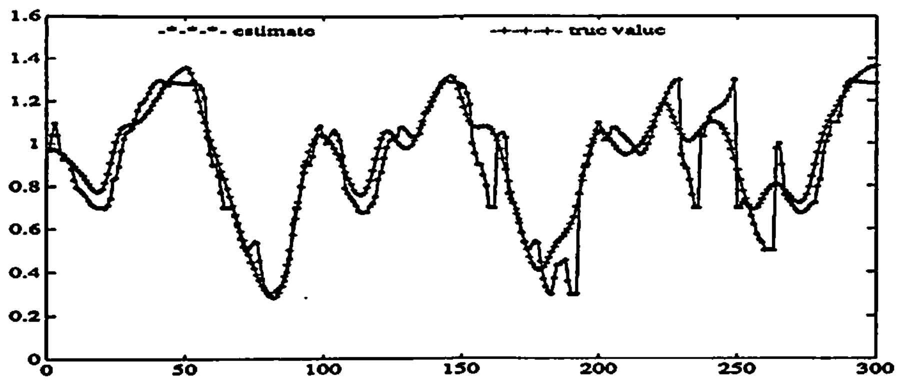


Figure 32: Prediction of the chaotic series from x(701) to x(1000) using the numerical-fuzzy predictor when 100 training data (from x(601) to x(700)) are used.

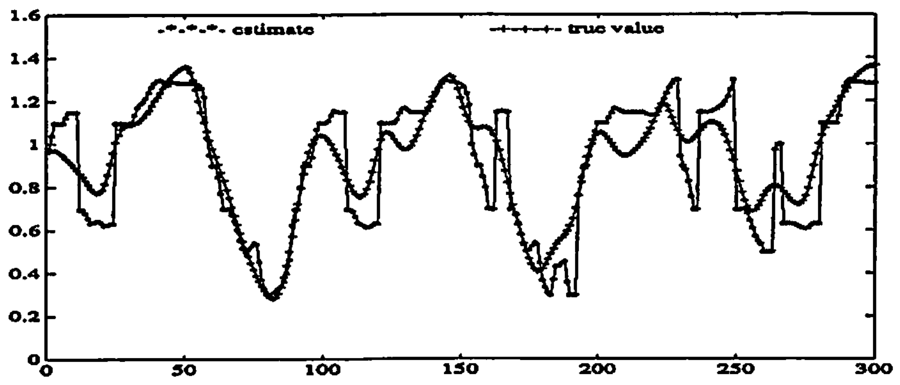


Figure 33: Prediction of the chaotic series from x(701) to x(1000) using the neural network predictor when 100 training data (from x(601) to x(700)) are used.

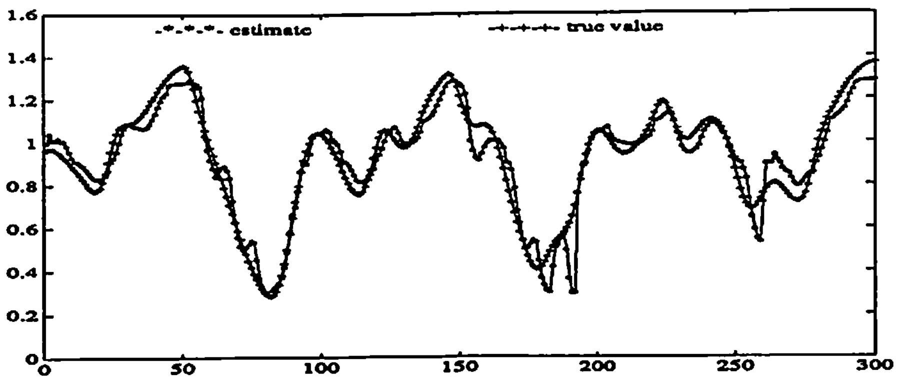


Figure 34: Prediction of the chaotic series from x(701) to x(1000) using the numerical-fuzzy predictor when 200 training data (from x(501) to x(700)) are used.

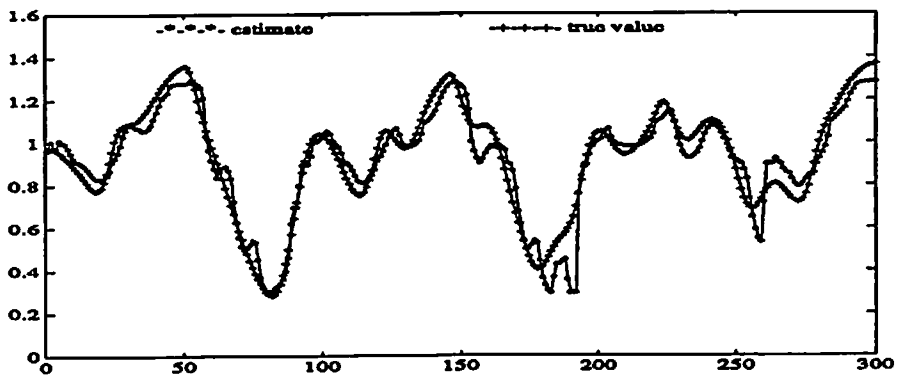


Figure 35: Prediction of the chaotic series from x(701) to x(1000) using the neural network predictor when 200 training data (from x(501) to x(700)) are used.

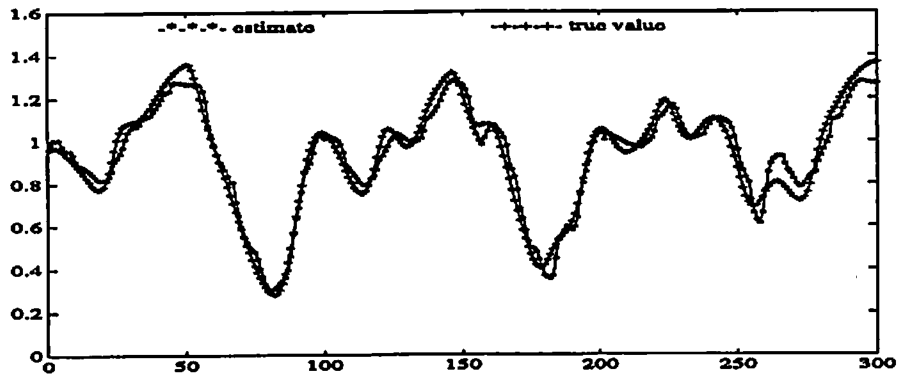


Figure 36: Prediction of the chaotic series from x(701) to x(1000) using the numerical-fuzzy predictor when 700 training data (from x(1) to x(700)) are used.

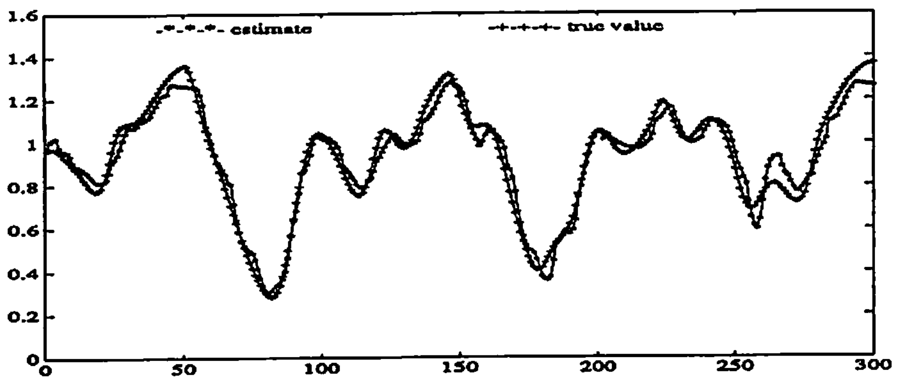


Figure 37: Prediction of the chaotic series from x(701) to x(1000) using the neural network predictor when 700 training data (from x(1) to x(700)) are used.

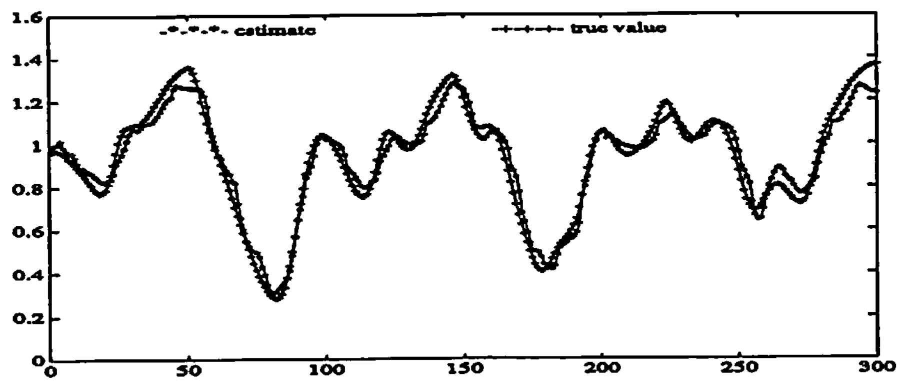


Figure 38: Prediction of the chaotic series from x(701) to x(1000) using the updating-FAM-Bank procedure.

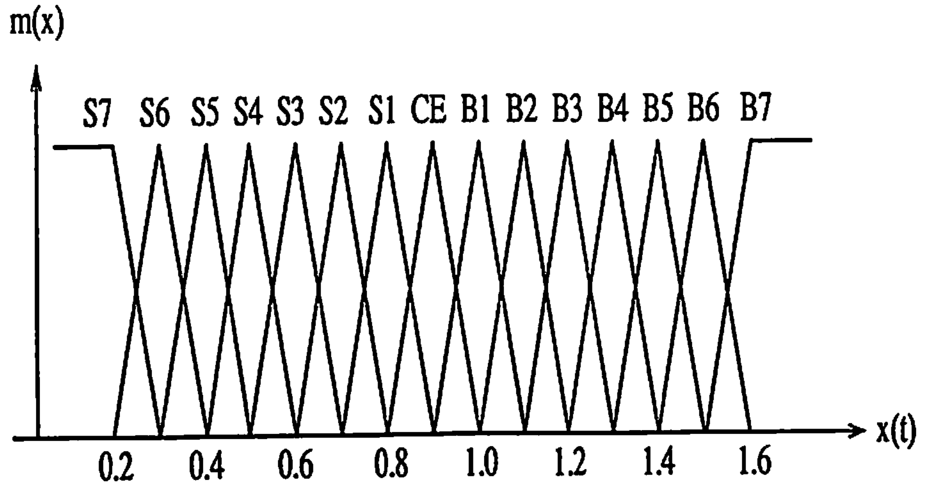


Figure 39: Membership function for the simulation of Fig.41.

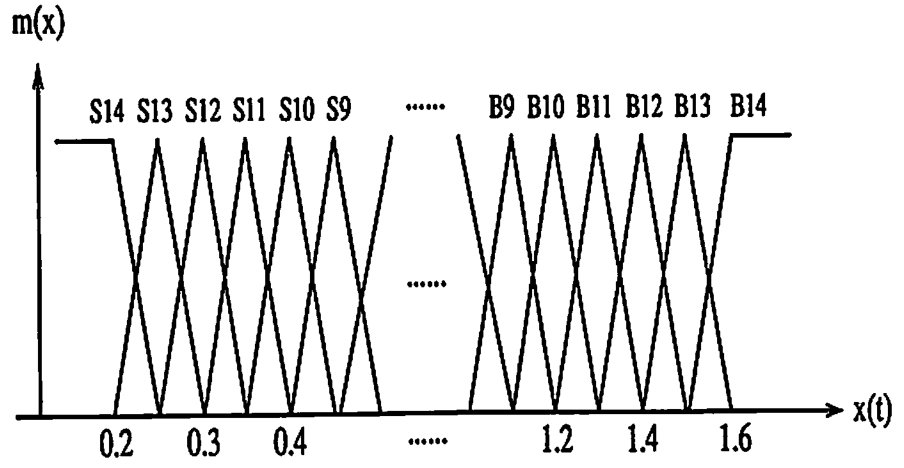


Figure 40: Membership function for the simulation of Fig.42.

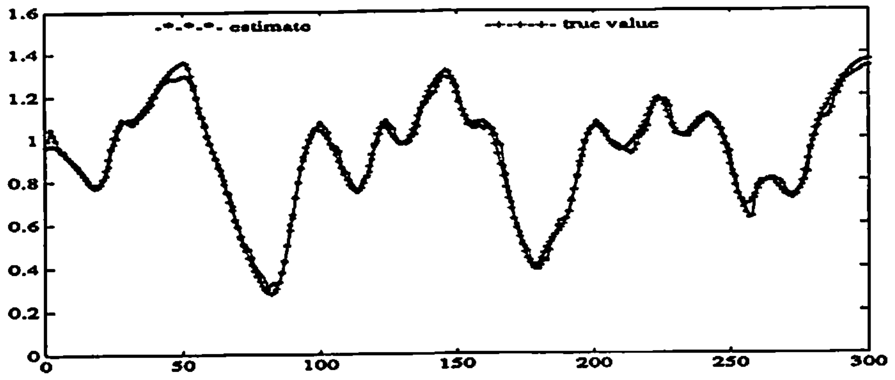


Figure 41: Prediction of the chaotic series from x(701) to x(1000) using the updating-FAM-Bank procedure with the membership function of Fig.38.

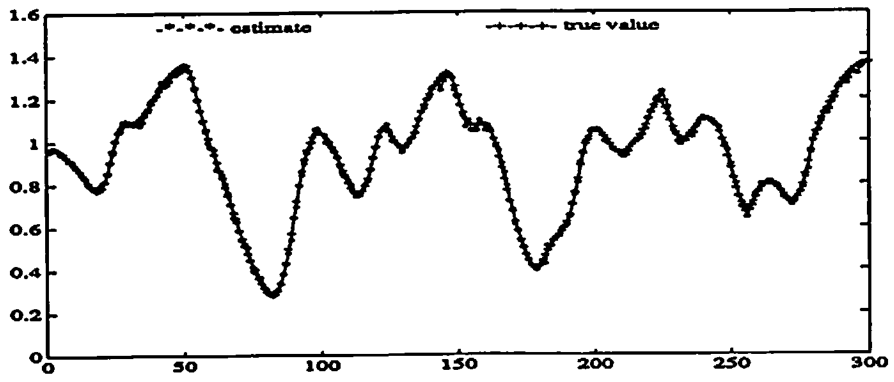


Figure 42: Prediction of the chaotic series from x(701) to x(1000) using the updating-FAM-Bank procedure with the membership function of Fig.39.

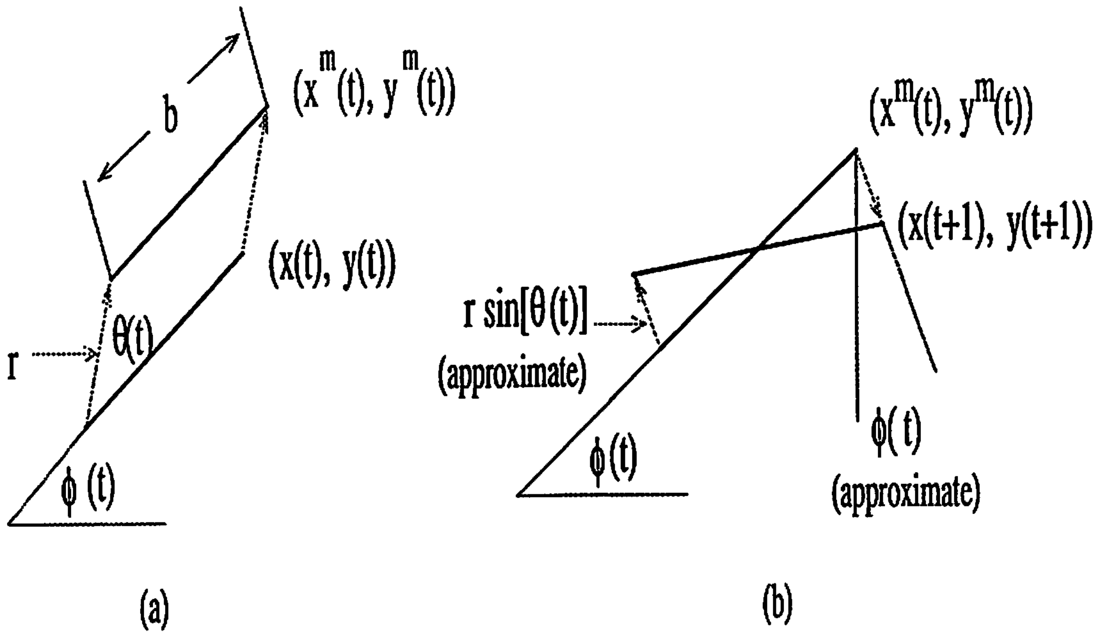


Figure B1. Approximate kinematics of the truck: (a) move the mass center by a distance r; and, (b) rotate the bar by an angle $\theta(t)$.

Table 1

Desired trajectory starting from $(x_0, \phi_0) = (1, 0^\circ)$

t	x	ϕ°	θ°
0	1.00	0.00	-19.00
1	1.95	9.37	-17.95
2	2.88	18.23	-16.90
3	3.79	26.59	-15.85
4	4.65	34.44	-14.80
5	5.45	41.78	-13.75
6	6.18	48.60	-12.70
7	7.48	54.91	-11.65
8	7.99	60.71	-10.60
9	8.72	65.99	-9.55
10	9.01	70.75	-8.50
11	9.28	74.98	-7.45
12	9.46	78.70	-6.40
13	9.59	81.90	-5.34
14	9.72	84.57	-4.30
15	9.81	86.72	-3.25
16	9.88	88.34	-2.20
17	9.91	89.44	0.00
18			
19			
20			

Table 2

Desired trajectory starting from $(x_0, \phi_0) = (1, 90^\circ)$

t	x	ϕ°	θ°
0	1.00	90.00	18.00
1	1.15	81.11	16.00
2	1.43	73.19	14.00
3	1.83	66.24	12.00
4	2.31	60.27	10.00
5	2.88	55.29	8.00
6	3.50	51.30	6.00
7	4.16	48.31	4.00
8	4.86	46.31	2.00
9	5.56	45.31	0.00
10	6.26	45.31	-2.00
11	6.95	46.31	-4.00
12	7.61	48.31	-6.00
13	8.23	51.30	-8.00
14	8.79	55.29	-10.00
15	9.28	60.27	-12.00
16	9.67	66.24	-14.00
17	9.95	73.19	-16.00
18	10.09	81.11	-18.00
19	10.09	90.00	0.00
20			

Table 3

Desired trajectory starting from $(x_0, \phi_0) = (1, 270°)$

t	x	$\phi°$	$\theta°$
0	1.00	-90.00	-30.0
1	1.22	-75.52	-29.0
2	1.64	-51.49	-28.0
3	2.24	-47.92	-27.0
4	2.98	-34.80	-26.0
5	3.82	-22.13	-25.0
6	4.72	-9.93	-24.0
7	5.64	1.80	-23.0
8	6.54	13.06	-22.0
9	7.39	23.86	-21.0
10	8.17	34.18	-20.0
11	8.85	44.04	-19.0
12	9.42	53.40	-18.0
13	9.66	62.29	-17.0
14	9.82	70.69	-16.0
15	10.08	78.62	-15.0
16	10.11	85.05	-14.0
17	10.13	90.02	0.0
18			
19			
20			

Table 4

Desired trajectory starting from $(x_0, \phi_0) = (7, 0°)$

t	x	$\phi°$	$\theta°$
0	7.00	0.00	-40.0
1	7.76	18.75	-38.5
2	8.51	36.88	-37.0
3	9.15	54.40	-35.5
4	9.62	71.28	-34.0
5	9.88	87.51	-3.0
6	9.93	89.90	0.0
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			

Table 5

Desired trajectory starting from $(x_0, \phi_0) = (7, 90°)$

t	x	$\phi°$	$\theta°$
0	7.00	90.00	12.00
1	7.10	84.03	10.06
2	7.29	79.02	8.12
3	7.55	74.97	6.18
4	7.86	71.89	4.24
5	8.21	69.77	2.30
6	8.57	68.62	0.34
7	8.84	68.44	-1.58
8	9.19	69.23	-3.52
9	9.42	70.99	-5.46
10	9.69	73.72	-7.40
11	9.81	77.45	-9.34
12	10.05	83.16	-11.28
13	10.15	89.67	0.00
14			
15			
16			
17			
18			
19			
20			

Table 6

Desired trajectory starting from $(x_0, \phi_0) = (7, 180°)$

t	x	$\phi°$	$\theta°$
0	7.00	180.00	40.00
1	6.23	161.25	36.80
2	5.48	143.82	33.60
3	4.80	127.76	30.40
4	4.27	113.10	27.20
5	3.93	99.89	24.00
6	3.77	88.16	20.80
7	3.80	77.93	17.60
8	4.00	69.23	14.40
9	4.35	62.09	11.20
10	4.81	56.52	8.00
11	5.35	52.53	4.80
12	5.96	50.12	1.60
13	6.60	49.32	-1.60
14	7.25	50.13	-4.80
15	7.89	52.53	-11.20
16	8.49	56.52	-14.40
17	9.04	62.09	-17.60
18	9.82	69.23	-20.80
19	10.02	78.12	-24.00
20	10.05	89.15	0.00

Table 7

Desired trajectory starting from $(x_0, \phi_0) = (7, 270^\circ)$

t	x	ϕ°	θ°
0	7.00	-90.00	-40.0
1	7.25	-71.25	-39.0
2	7.73	-52.91	-38.0
3	8.38	-34.98	-37.0
4	9.15	-17.47	-36.0
5	9.97	-0.38	-35.0
6	10.77	16.29	-34.0
7	11.48	32.53	-33.0
8	12.04	48.33	-32.0
9	12.42	63.70	-31.0
10	12.59	78.62	-30.0
11	12.54	93.10	-29.0
12	12.28	107.12	-28.0
13	11.77	120.70	0.0
14	11.28	120.70	15.0
15	10.90	113.27	13.0
16	10.62	106.81	11.0
17	10.32	101.33	9.0
18	10.20	96.84	7.0
19	10.14	93.35	5.0
20	10.09	90.35	0.0

Table 8

Desired trajectory starting from $(x_0, \phi_0) = (13, 0^\circ)$

t	x	ϕ°	θ°
0	13.00	0.00	-40.00
1	13.77	18.75	-36.80
2	14.52	36.18	-33.60
3	15.20	52.24	-30.40
4	15.73	66.90	-27.20
5	16.08	80.11	-24.00
6	16.23	91.84	-20.80
7	16.20	102.07	-17.60
8	16.00	110.77	-14.40
9	15.66	117.91	-11.20
10	15.20	123.48	-8.00
11	14.66	127.47	-4.80
12	14.05	129.87	-1.60
13	13.41	130.67	1.60
14	12.76	129.87	4.80
15	12.12	127.47	8.00
16	11.52	123.48	11.20
17	10.98	117.91	14.40
18	10.53	110.77	17.60
19	10.19	101.07	20.80
20	9.99	90.84	0.00

Table 9

Desired trajectory starting from $(x_0, \phi_0) = (13, 90^\circ)$

t	x	ϕ°	θ°
0	13.00	90.00	-12.00
1	12.90	95.97	-10.05
2	12.71	100.97	-8.10
3	12.45	105.01	-6.15
4	12.15	108.08	-4.20
5	11.80	110.18	-2.25
6	11.44	111.31	-0.30
7	11.08	111.46	1.65
8	10.72	110.63	3.60
9	10.40	108.83	5.55
10	10.13	106.06	7.50
11	9.98	102.31	9.45
12	9.89	97.50	11.40
13	9.81	90.94	0.00
14			
15			
16			
17			
18			
19			
20			

Table 10

Desired trajectory starting from $(x_0, \phi_0) = (13, 180^\circ)$

t	x	ϕ°	θ°
0	13.00	180.00	40.00
1	12.23	161.25	38.05
2	11.49	143.11	37.00
3	10.85	125.60	35.50
4	10.38	108.72	34.00
5	10.11	92.49	5.50
6	10.08	89.80	0.00
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			

Table 11

Desired trajectory starting from $(x_0, \phi_0) = (13, 270^\circ)$

t	x	ϕ°	θ°
0	13.00	270.00	40.00
1	12.75	251.26	39.00
2	12.28	232.91	38.00
3	11.62	214.09	37.00
4	10.85	197.47	36.00
5	10.03	180.38	35.00
6	9.23	163.71	34.00
7	8.53	147.48	33.00
8	7.96	131.67	32.00
9	7.58	116.31	31.00
10	7.41	101.38	30.00
11	7.71	86.91	29.00
12	8.22	72.88	28.00
13	8.71	59.29	0.00
14	9.09	59.29	-15.00
15	9.37	66.74	-13.00
16	9.58	73.19	-10.00
17	9.71	78.18	-8.00
18	9.81	81.92	-5.50
19	9.92	85.50	-3.50
20	9.97	89.23	0.00

Table 12

Desired trajectory starting from $(x_0, \phi_0) = (19, 90^\circ)$

t	x	ϕ°	θ°
0	19.00	90.00	-18.00
1	18.85	98.88	-16.00
2	18.57	106.81	-14.00
3	18.18	113.76	-12.00
4	17.69	119.72	-10.00
5	17.13	124.71	-8.00
6	16.51	128.70	-6.00
7	15.84	131.69	-4.00
8	15.16	133.69	-2.00
9	14.45	134.69	0.00
10	13.75	134.69	2.00
11	13.06	133.69	4.00
12	12.40	131.69	6.00
13	11.78	128.70	8.00
14	11.22	124.71	10.00
15	10.74	119.72	12.00
16	10.35	113.76	14.00
17	10.07	106.81	16.00
18	9.93	98.89	18.00
19	9.89	90.00	0.00
20			

Table 13

Desired trajectory starting from $(x_0, \phi_0) = (19, 180^\circ)$

t	x	ϕ°	θ°
0	19.00	180.00	19.00
1	18.05	170.63	17.95
2	17.12	161.77	16.90
3	16.21	153.41	15.85
4	15.35	145.56	14.80
5	14.55	138.22	13.75
6	13.83	131.40	12.70
7	13.18	125.09	11.65
8	12.62	119.29	10.60
9	12.14	114.01	9.55
10	11.74	109.26	8.50
11	11.42	105.01	7.45
12	11.16	101.30	6.40
13	10.96	98.10	5.45
14	10.73	95.43	4.50
15	10.53	93.28	3.55
16	10.37	91.66	2.50
17	10.11	90.56	0.00
18			
19			
20			

Table 14

Desired trajectory starting from $(x_0, \phi_0) = (19, 270^\circ)$

t	x	ϕ°	θ°
0	19.00	270.00	30.0
1	18.78	255.53	29.0
2	18.36	241.50	28.0
3	17.76	227.92	27.0
4	17.03	214.80	26.0
5	16.19	202.14	25.0
6	15.29	189.94	24.0
7	14.37	178.21	23.0
8	13.46	166.94	22.0
9	12.61	156.14	21.0
10	11.83	145.82	20.0
11	11.15	135.97	19.0
12	10.58	126.61	18.0
13	10.13	117.72	17.0
14	9.99	109.31	16.2
15	9.91	101.39	15.4
16	9.85	93.95	14.6
17	9.80	91.06	13.8
18			
19			
20			

Table 15

Fuzzy rules generated from the desired input-output pairs of Table 1, and the degrees of these rules.

Fuzzy rules for t=	IF		THEN	Degree
	x is	ϕ is	θ is	
0	S2	S2	S2	1.00
1	S2	S2	S2	0.92
2	S2	S2	S2	0.35
3	S2	S2	S2	0.12
4	S2	S2	S2	0.07
5	S1	S2	S1	0.08
6	S1	S1	S1	0.18
7	S1	S1	S1	0.52
8	S1	S1	S1	0.56
9	S1	S1	S1	0.60
10	CE	S1	S1	0.35
11	CE	S1	S1	0.21
12	CE	S1	CE	0.16
13	CE	CE	CE	0.32
14	CE	CE	CE	0.45
15	CE	CE	CE	0.54
16	CE	CE	CE	0.88
17	CE	CE	CE	0.92
18				
19				
20				

Table 16

Fuzzy rules generated from the desired input-output pairs of Table 2, and the degrees of these rules.

Fuzzy rules for t=	IF		THEN	Degree
	x is	ϕ is	θ is	
0	S2	CE	B2	0.91
1	S2	CE	B2	0.72
2	S2	S1	B2	0.31
3	S2	S1	B2	0.24
4	S2	S1	B1	0.21
5	S2	S1	B1	0.28
6	S2	S1	B1	0.44
7	S2	S1	B1	0.31
8	S1	S1	CE	0.28
9	S1	S2	CE	0.35
10	S1	S2	CE	0.28
11	S1	S1	S1	0.54
12	S1	S1	S1	0.72
13	S1	S1	S1	0.48
14	S1	S1	S1	0.35
15	CE	S1	S1	0.18
16	CE	S1	S2	0.18
17	CE	S1	S2	0.27
18	CE	CE	S2	0.48
19	CE	CE	CE	0.97
20				

Table 17

Fuzzy rules generated from the desired input-output pairs of Table 3, and the degrees of these rules.

Fuzzy rules for t=	IF		THEN	Degree
	x is	ϕ is	θ is	
0	S2	S3	S2	0.25
1	S2	S3	S2	0.40
2	S2	S3	S2	0.31
3	S2	S3	S2	0.28
4	S2	S2	S2	0.23
5	S2	S2	S2	0.26
6	S2	S2	S2	0.28
7	S1	S2	S2	0.41
8	S1	S2	S2	0.50
9	S1	S2	S2	0.46
10	S1	S2	S2	0.48
11	S1	S1	S2	0.47
12	CE	S1	S2	0.51
13	CE	S1	S2	0.36
14	CE	S1	S2	0.21
15	CE	CE	S1	0.30
16	CE	CE	S1	0.60
17	CE	CE	CE	0.97
18				
19				
20				

Table 18

Fuzzy rules generated from the desired input-output pairs of Table 4, and the degrees of these rules.

Fuzzy rules for t=	IF		THEN	Degree
	x is	ϕ is	θ is	
0	S1	S2	S3	1.00
1	S1	S2	S3	0.65
2	S1	S1	S3	0.39
3	CE	S1	S3	0.44
4	CE	S1	S3	0.38
5	CE	CE	CE	0.84
6	CE	CE	CE	0.97
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

Table 19

Fuzzy rules generated from the desired input-output pairs of Table 5, and the degrees of these rules.

Fuzzy rules for t=	IF		THEN	Degree
	x is	ϕ is	θ is	
0	S1	CE	B2	0.50
1	S1	CE	B2	0.42
2	S1	CE	B1	0.38
3	S1	S1	B1	0.54
4	S1	S1	B1	0.28
5	S1	S1	CE	0.31
6	S1	S1	CE	0.21
7	S1	S1	CE	0.19
8	CE	S1	CE	0.27
9	CE	S1	S1	0.25
10	CE	S1	S1	0.31
11	CE	S1	S1	0.36
12	CE	CE	S1	0.32
13	CE	CE	CE	0.96
14				
15				
16				
17				
18				
19				
20				

Table 20

Fuzzy rules generated from the desired input-output pairs of Table 6, and the degrees of these rules.

Fuzzy rules for t=	IF		THEN	Degree
	x is	ϕ is	θ is	
0	S1	B2	B3	1.00
1	S1	B2	B3	0.66
2	S1	B1	B3	0.30
3	S2	B1	B3	0.18
4	S2	B1	B2	0.27
5	S2	B1	B2	0.21
6	S2	CE	B2	0.53
7	S2	S1	B2	0.21
8	S2	S1	B2	0.18
9	S2	S1	B2	0.14
10	S2	S1	B1	0.18
11	S1	S1	B1	0.32
12	S1	S1	CE	0.41
13	S1	S1	CE	0.54
14	S1	S1	S1	0.42
15	S1	S1	S1	0.27
16	S1	S1	S2	0.20
17	S1	S1	S2	0.16
18	CE	S1	S2	0.50
19	CE	CE	S2	0.31
20	CE	CE	CE	0.97

Table 21

Fuzzy rules generated from the desired input-output pairs of Table 7, and the degrees of these rules.

Fuzzy rules for t=	IF		THEN	Degree
	x is	ϕ is	θ is	
0	S1	S3	S3	0.50
1	S1	S3	S3	0.61
2	S1	S3	S3	0.40
3	S1	S2	S3	0.27
4	CE	S2	S3	0.56
5	CE	S2	S3	0.80
6	CE	S2	S3	0.64
7	B1	S2	S3	0.18
8	B1	S1	S3	0.27
9	B1	S1	S3	0.23
10	B1	S1	S2	0.17
11	B1	CE	S2	0.27
12	B1	B1	S2	0.20
13	B1	B1	CE	0.32
14	B1	B1	B2	0.21
15	CE	B1	B2	0.20
16	CE	B1	B2	0.22
17	CE	B1	B1	0.32
18	CE	B1	B1	0.51
19	CE	CE	CE	0.50
20	CE	CE	CE	0.99

Table 22

Fuzzy rules generated from the desired input-output pairs of Table 8, and the degrees of these rules.

Fuzzy rules for t=	IF		THEN	Degree
	x is	ϕ is	θ is	
0	B1	S2	S3	1.00
1	B1	S2	S3	0.70
2	B1	S1	S3	0.15
3	B2	S1	S3	0.28
4	B2	S1	S2	0.31
5	B2	CE	S2	0.30
6	B2	CE	S2	0.54
7	B2	B1	S2	0.30
8	B2	B1	S1	0.15
9	B2	B1	S1	0.28
10	B2	B1	S1	0.41
11	B2	B1	S1	0.38
12	B1	B1	CE	0.45
13	B1	B1	CE	0.63
14	B1	B1	B1	0.55
15	B1	B1	B1	0.72
16	B1	B1	B1	0.33
17	CE	B1	B2	0.21
18	CE	B1	B2	0.32
19	CE	B1	B2	0.51
20	CE	CE	CE	0.99

Table 23

Fuzzy rules generated from the desired input-output pairs of Table 9, and the degrees of these rules.

Fuzzy rules for t=	IF		THEN	Degree
	x is	ϕ is	θ is	
0	B1	CE	S2	0.50
1	B1	CE	S1	0.31
2	B1	B1	S1	0.41
3	B1	B1	S1	0.32
4	B1	B1	S1	0.18
5	B1	B1	CE	0.22
6	B1	B1	CE	0.35
7	B1	B1	CE	0.28
8	CE	B1	B1	0.15
9	CE	B1	B1	0.21
10	CE	B1	B1	0.36
11	CE	B1	B1	0.28
12	CE	B1	B2	0.18
13	CE	CE	CE	0.92
14				
15				
16				
17				
18				
19				
20				

Table 24

Fuzzy rules generated from the desired input-output pairs of Table 10, and the degrees of these rules.

Fuzzy rules for t=	IF		THEN	Degree
	x is	ϕ is	θ is	
0	B1	B2	B3	1.00
1	B1	B2	B3	0.75
2	B1	B1	B3	0.19
3	CE	B1	B3	0.26
4	CE	B1	B3	0.41
5	CE	CE	B1	0.48
6	CE	CE	CE	0.96
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

Table 25

Fuzzy rules generated from the desired input-output pairs of Table 11, and the degrees of these rules.

Fuzzy rules for t=	IF		THEN	Degree
	x is	ϕ is	θ is	
0	B1	B3	B3	0.50
1	B1	B3	B3	0.72
2	B1	B3	B3	0.59
3	B1	B3	B3	0.41
4	CE	B2	B3	0.28
5	CE	B2	B3	0.72
6	CE	B2	B3	0.26
7	S1	B1	B3	0.23
8	S1	B1	B3	0.35
9	S1	B1	B3	0.27
10	S1	B1	B3	0.22
11	S1	CE	B2	0.49
12	S1	S1	B2	0.39
13	S1	S1	CE	0.61
14	CE	S1	S2	0.24
15	CE	S1	S2	0.28
16	CE	S1	S1	0.33
17	CE	S1	S1	0.33
18	CE	CE	B1	0.28
19	CE	CE	CE	0.56
20	CE	CE	CE	0.96

Table 26

Fuzzy rules generated from the desired input-output pairs of Table 12, and the degrees of these rules.

Fuzzy rules for t=	IF		THEN	Degree
	x is	ϕ is	θ is	
0	B2	CE	S2	0.81
1	B2	CE	S2	0.42
2	B2	B1	S2	0.25
3	B2	B1	S2	0.17
4	B2	B1	S1	0.47
5	B2	B1	S1	0.63
6	B2	B1	S1	0.43
7	B2	B1	S1	0.36
8	B2	B1	CE	0.25
9	B1	B1	CE	0.49
10	B1	B1	CE	0.35
11	B1	B1	B1	0.41
12	B1	B1	B1	0.49
13	B1	B1	B1	0.28
14	B1	B1	B1	0.16
15	CE	B1	B2	0.18
16	CE	B1	B2	0.27
17	CE	B1	B2	0.39
18	CE	CE	B2	0.41
19	CE	CE	CE	0.94
20				

Table 27

Fuzzy rules generated from the desired input-output pairs of Table 13, and the degrees of these rules.

Fuzzy rules for t=	IF		THEN	Degree
	x is	ϕ is	θ is	
0	B2	B2	B2	0.91
1	B2	B2	B2	0.72
2	B2	B2	B2	0.51
3	B2	B2	B2	0.26
4	B2	B1	B2	0.16
5	B1	B1	B2	0.23
6	B1	B1	B2	0.25
7	B1	B1	B2	0.21
8	B1	B1	B2	0.18
9	B1	B1	B1	0.20
10	B1	B1	B1	0.21
11	B1	B1	B1	0.22
12	B1	B1	B1	0.23
13	B1	CE	B1	0.25
14	CE	CE	B1	0.16
15	CE	CE	B1	0.32
16	CE	CE	CE	0.48
17	CE	CE	CE	0.97
18				
19				
20				

Table 28

Fuzzy rules generated from the desired input-output pairs of Table 14, and the degrees of these rules.

Fuzzy rules for t=	IF		THEN	Degree
	x is	ϕ is	θ is	
0	B2	B3	B3	0.50
1	B2	B3	B2	0.35
2	B2	B3	B2	0.48
3	B2	B3	B2	0.57
4	B2	B3	B2	0.44
5	B2	B2	B2	0.19
6	B2	B2	B2	0.22
7	B1	B2	B2	0.41
8	B1	B2	B2	0.42
9	B1	B2	B2	0.40
10	B1	B1	B2	0.48
11	B1	B1	B2	0.64
12	CE	B1	B2	0.51
13	CE	B1	B2	0.48
14	CE	B1	B2	0.40
15	CE	B1	B2	0.21
16	CE	CE	B2	0.41
17	CE	CE	CE	0.91
18				
19				
20				

Table 29: Monthly mean temperature of St. Louis, Missouri, from Jan. 1945 to Dec. 1974 (taken from Appendix III of [12]).

MONTHLY MEAN TEMPERATURE

YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
1945	-1.9	2.2	12.2	13.7	16.1	21.3	25.2	25.4	20.9	14.4	7.7	-2.0
1946	2.9	-2.1	2.7	13.1	17.7	22.8	24.9	29.7	22.2	19.3	5.2	3.7
1947	1.3	4.6	13.8	16.0	16.9	24.5	26.9	23.0	20.9	17.1	9.3	4.9
1948	-3.1	1.4	6.6	15.4	18.4	24.1	26.1	26.0	22.4	13.8	8.9	3.7
1949	0.6	2.7	6.7	12.3	20.6	25.3	27.0	25.5	18.4	15.9	9.4	4.9
1950	3.4	2.8	5.0	10.6	20.1	23.6	24.6	22.7	19.7	18.3	3.9	-1.4
1951	0.3	1.9	4.5	11.2	19.7	22.3	25.7	24.8	19.1	15.1	3.8	·0.9
1952	2.2	4.3	5.8	12.7	18.8	28.4	27.4	24.8	21.1	12.1	7.8	2.8
1953	1.8	4.4	8.2	11.2	19.6	27.5	27.4	25.7	22.4	16.7	8.7	3.2
1954	0.4	6.6	5.6	16.7	16.4	26.3	29.4	26.7	23.7	14.9	7.7	2.6
1955	0.6	2.2	7.2	17.2	20.2	21.9	28.6	26.8	23.1	14.6	5.4	0.6
1956	-1.6	2.7	7.3	11.9	19.9	24.6	25.4	26.3	21.6	18.1	7.2	3.6
1957	-2.8	4.7	6.3	13.6	19.0	24.2	26.7	26.3	20.2	12.7	6.4	4.9
1958	-0.4	-3.7	3.3	13.2	18.9	21.9	24.7	25.3	20.6	15.1	9.4	-0.9
1959	-3.2	2.1	7.6	13.8	20.9	24.7	25.6	27.1	21.8	13.9	3.9	4.1
1960	0.8	-0.7	-0.9	14.9	17.1	23.3	24.6	26.2	23.4	15.0	8.1	-0.6
1961	-2.1	2.2	7.5	10.0	14.6	21.2	24.5	24.1	21.7	15.1	6.7	0.0
1962	-4.1	2.6	4.2	11.8	22.2	23.3	24.3	24.2	19.1	16.1	7.2	-0.4
1963	-5.9	-1.9	9.3	14.6	17.7	24.1	25.2	23.8	20.3	19.4	7.6	-5.4
1964	1.5	1.3	5.7	14.8	20.9	23.8	25.8	24.7	20.7	12.2	8.5	-0.1
1965	-0.1	1.0	1.4	14.7	21.4	23.9	24.9	24.3	20.6	13.6	9.2	5.4
1966	-3.9	0.1	8.2	10.9	16.5	23.2	28.3	23.4	18.7	12.3	8.4	1.6
1967	1.5	-0.4	9.1	14.7	16.0	23.3	23.8	22.4	19.2	14.1	5.6	1.6
1968	-0.1	-1.6	7.8	13.1	16.6	25.0	25.3	25.1	19.3	13.9	6.4	0.2
1969	-1.6	1.8	3.2	13.8	18.7	22.5	26.9	25.0	20.7	13.4	6.3	0.1
1970	-4.0	0.5	4.7	14.4	20.7	22.4	25.5	24.6	22.1	13.4	6.4	2.8
1971	-2.6	1.1	5.4	13.3	16.8	26.1	24.1	24.3	22.5	17.6	7.8	4.8
1972	-1.2	1.3	7.2	13.4	19.1	23.0	25.3	24.6	21.7	12.7	4.3	-0.9
1973	0.3	1.4	10.5	12.1	16.5	23.7	25.9	24.9	21.1	15.9	8.3	-1.1
1974	-1.2	2.3	8.9	14.2	18.3	20.8	26.6	23.6	16.8	14.3	6.7	1.1