Intelligent identification and control for autonomous guided vehicles using adaptive fuzzy-based algorithms

C. J. Harris and C. G. Moore
Department of Aeronautics and Astronautics, Southampton University, Highfield, Southampton SO9 5NH, UK
(*Received October 1989*)

ABSTRACT

Fuzzy logic was first suggested as the mechanism by which humans drive cars. This paper addresses the use of fuzzy logic and algorithms towards the intelligent autonomous motion control of land vehicles. To cope with vehicle complexities, internal parametric changes, and with unpredictable environmental effects, the controllers that are presented, whilst heuristic in nature, are self-organizing or self-learning in that they generate automatically by observation an experiential rule base that models the vehicle, and via an appropriate performance index an optimal control rule base that is robust to large parametric changes. The methodology presented is applicable to any complex process which is too difficult to model or control using conventional methods, or which has relied on the experience of a human operator. An overview of fuzzy logic and static fuzzy logic control (akin to expert systems) is provided, together with illustrative examples.

INTRODUCTION

The earliest automatically guided vehicles were developed in Japan[1], the USA[2] and the EEC[3]. These were characterized by complex supervisory control systems with partial autonomy through on-board navigation. More recently[4], a wide range of special-purpose and generic *autonomous* guided vehicles (AGV) research programmes have been initiated (in the UK the MOD MARDI programme[5] (1988) includes tracked cross country and road vehicles; in the USA the DARPA programme[6]; in the EEC the Esprit II project (No. 2483) Panorama, includes a tracked mining vehicle, wheeled cross-country vehicle and a laboratory test bed). Of central importance to AGVs are the intelligent tasks of:

1. Multi-sensor data integration or fusion[7,8] to locate the vehicle, to represent or model its internal states and its environment (including obstacles) and to assess the current system situation state vector[9] (see also the EEC Esprit I project (No. 1560) SKIDS).
2. Planning and navigation.
3. Motion control[5,10,11].

The systems architecture of the majority of AGVs is hierarchical[12,13] with usually three levels of abstraction (as with command and control systems[9]). At the highest, most abstract (and least time-critical, precise or detailed) level, is the *planner*, which operates on a global mission to determine the connected subgoals or tasks to achieve an assigned objective. At the next level, the *navigator* utilizes a detailed plan or map to, say, evaluate an obstacle-free *local* path that optimizes a performance criterion (say, minimum fuel usage or minimum path, . . .) to produce motion and velocity trajectories. Given these trajectories, the pilot or motion controller must provide, in real time, optimal motion control avoiding obstacles not identified by the planning or navigation stages—requiring direct local feedback of 'visually' acquired proximity sensor data and appropriate very local path adaption. Vehicle motion control from current position and velocity is a two-point boundary-value problem with constrained states; by utilizing a

0952-1976/89/040267-19$2.00

hierarchical problem decomposition, the search space for optimization can be reduced and the resulting subproblem of motion control can then be defined over unconstrained subspaces. The procedure is then establishing obstacle-free subspaces and selecting those which provide, say, minimum time or effort trajectories, but since the vehicle and its environmental database are based upon models or representations that are incomplete, uncertain or fuzzy, then the associated rules or controls are equally fuzzy.

Humans use fuzzy-like algorithms when they drive or park a vehicle, search for an object or obstacle, etc., adopting procedures that are sufficiently flexible, robust, imprecise and intelligent that they can be adapted to slightly different situations (the principle of generalization) and have a capability to learn. An approximate solution is acceptable in that there are tolerances in the goals as well as in uncertainties associated with 'world' models. This is a natural top-down approach to AGVs since there is imperfect knowledge of (i) the environment (due to incompleteness and uncertainty of sensor data), (ii) the vehicle dynamics (e.g. variations in payload, velocity, frictional forces, road conditions), etc. and (iii) routes to be followed and some task objectives. Zadeh's principle of incompatibility[14] and fuzzy logic applies particularly to intelligent guidance and motion control of AGVs in unstructured environments, e.g. human drivers utilize experiential models, with inexact observations linguistically modelled as fuzzy sets (such as 'near', 'close' or 'very close') and implement vehicle control through the maximization of some objective (such as minimum time, or minimum fuel usage) via a set of adaptive fuzzy rules or algorithms. Fuzzy decision rule methods have been proposed for path determination of AGVs that progressively discover the environment[14], for car parking[15,16] and motion control[5,11,17] of AGVs.

Classical control is based upon (i) deriving from physical laws or identification algorithms a set of model equations, and (ii) generating a set of feedback control laws that ensure that these models behave as desired. To evaluate an AGV dynamic state equation it is necessary to evaluate the longitudinal and lateral dynamics. The vehicle's longitudinal dynamics can be represented[18] by the nonlinear equation:

$$m\dot{v} = F_1(I_g, w_e, T_p, R_s), \tag{1}$$

where m is vehicle mass, v vehicle velocity and $F_1(\cdot)$ is a complex function that includes gearing; inertial terms I_g for rotating elements; engine speed w_e; throttle position T_p; and resistance or loss terms R_s to represent drag, rolling resistance/braking, etc. Similarly the lateral dynamics can be represented by the two degree of freedom 'planar bicycle' model

$$\begin{aligned} mv\dot{y} &= F_2(U_f, f_r, v, c_f, \theta_v, \varepsilon, y), \\ I_z\ddot{\theta}_v &= F_3(U_f, f_r, v, c_f, \theta_v, \varepsilon, y), \end{aligned} \tag{2}$$

where $F_2(\cdot)$, $F_3(\cdot)$ are also nonlinear functions of the yaw angle θ_r; steering angle ε; side slip angle y; wheel circumferential and normal side forces U_f, c_f respectively; and f_r the coefficient of friction between wheels and surface. I_z is the vehicle moment of inertia about the vertical direction. The control inputs to (1) and (2) are acceleration a $(=\dot{v})$; and steering angle ε. The composite system (1), (2) is both nonlinear and nonstationary (e.g. parameters m, f); also variables c_f, U_f, ε satisfy other complex dynamical relationships[18]. However, these equations can be linearized and for a constant velocity vehicle on a smooth small radius of curvature road conventional Kalman filtering and pole placement control methods have been used[10]. It is, however, clear for an unstructured environment with substantial variations in environmental and vehicle states that the classical 'bottom up' control methods for AGVs are inappropriate, and a 'top down' experiential approach is currently the only practically feasible generic approach.

The layout of the paper is: first, an introduction and overview of fuzzy logic and algorithms; next, generation of fuzzy rule bases for system modelling and control, followed by a discussion of fuzzy controller synthesis and, finally, a description of self-organizing or learning fuzzy controllers.

FUZZY SETS, LOGIC, RELATIONSHIPS AND ALGORITHMS

Fuzzy sets

A fuzzy set is a more-general form of a classical set. In a classical set, a collection of objects, or points in a space, or attributes, are said to be elements of that set; a particular element of the universe of discourse is either in that set or outside it. However, in fuzzy-set theory, a fuzzy set A of a universe of discourse X $(A \subseteq X)$ may have elements $x \in X$ which *partially* belong to the set. The degree to which it belongs to the set A is characterized by the membership function $\mu_A(x)$ in the interval $[0, 1]$ that represents the *grade* of membership, with 1 representing full membership, 0.5 (etc.) partial

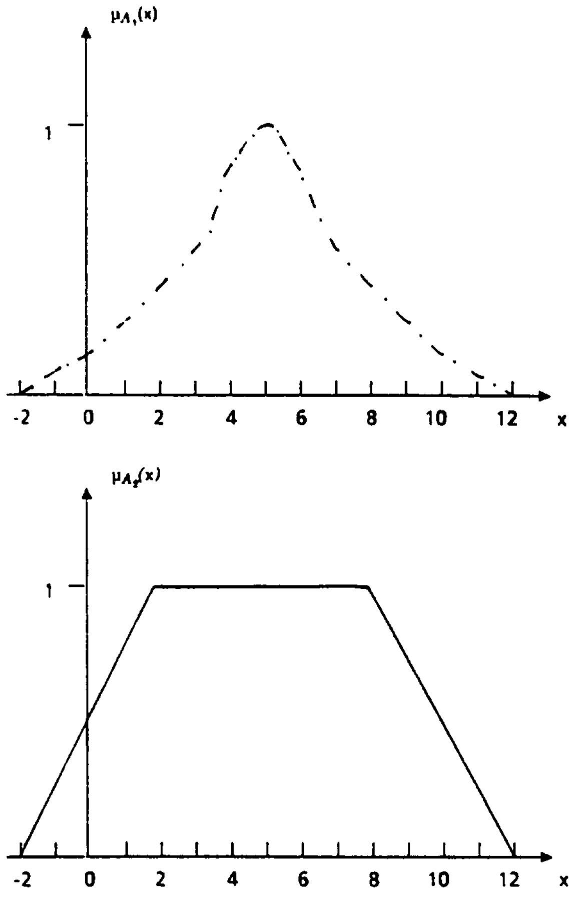

Figure 1 Membership functions for 'approximately 5'

membership and 0 no membership. $\mu_A(x)$ determines the degree or grade of membership of x of A. Continuous and discrete fuzzy sets are respectively defined by the sets:

$$A = (x, \mu_A(x)/x \in X), \tag{3}$$

$$A = \left\{ \sum_{k=1}^{m} \mu_A(x_k)/x_k \right\}. \tag{4}$$

Example 1

The fuzzy set 'approximately equal to 5' for x taking discrete values $(-2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)$; two possible membership functions are:

$$A_1 = \{0/-2, 0.1/-1, 0.17/0, 0.24/1, 0.36/2, 0.56/3, 0.84/4, 1/5, 0.83/6, 0.56/7, 0.36/8, 0.24/9, 0.17/10, 0.1/11, 0/12\},$$

$$A_2 = \{0/-2, 0.24/-1, 0.49/0, 0.74/1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8, 0.74/9, 0.49/10, 0.24/11, 0/12\},$$

which are illustrated in *Figure 1*. Note that the shape of the membership function is defined by the user; unfortunately fuzzy-set theory does not provide the means of selecting the most appropriate membership function; for most practical cases the results are not particularly sensitive to the actual shape.

Fuzzy logic

Example 2

Consider for illustration purposes three finite fuzzy sets of X defined by three linguistic qualifiers, positive big (PB), positive medium (PM) and positive small (PS) on the finite universe of real numbers $[0, 6]$. (See *Figure 2*.)

$$PB = \{0/0, 0/1, 0/2, 0.3/3, 0.7/4, 1/5, 1/6\}, \tag{5}$$

$$PM = \{0/0, 0.3/1, 0.7/2, 1/3, 0.7/4, 0.3/5, 0/6\}, \tag{5}$$

$$PS = \{1/0, 1/1, 0.7/2, 0.3/3, 0/4, 0/5, 0/6\}. \tag{5}$$

1. The union of two fuzzy sets A and B $(A, B \subseteq X)$; $A \cup B \equiv A + B$ has a membership function defined by

$$\mu_{(A+B)}(x) = \max[\mu_A(x), \mu_B(x)]. \tag{6}$$

This corresponds to the logic OR function. From (5) and (6) *Example 2* gives the membership function of PB OR PM (see *Figure 2*)

$$\begin{aligned}\mu(PB\,\text{OR}\,PM) = \{&\max[0, 0]/0, \max[0, 0.3]/1, \\ &\max[0, 0.7]/2, \max[0.3, 1]/3, \\ &\max[0.7, 0.7]/4, \max[1, 0.3]/5, \\ &\max[1, 0]/6\} \\ = \{&0/0, 0.3/1, 0.7/2, 1/3, 0.7/4, 1/5, \\ &1/6\}.\end{aligned}$$

2. The intersection of two sets A and B, $A \cap B$ is defined by

$$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)). \tag{7}$$

This corresponds to the logic AND function. From

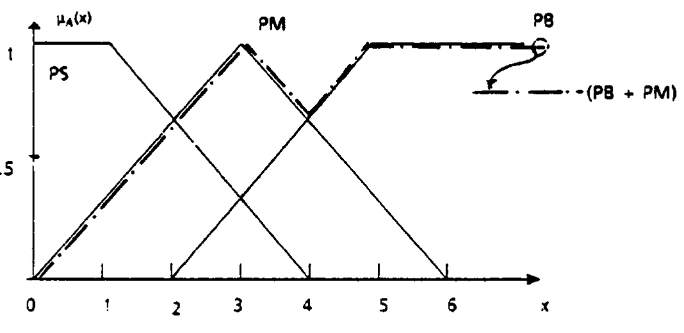

Figure 2 Membership functions for *PS*, *PM*, *PB* and (*PB* OR *PM*)

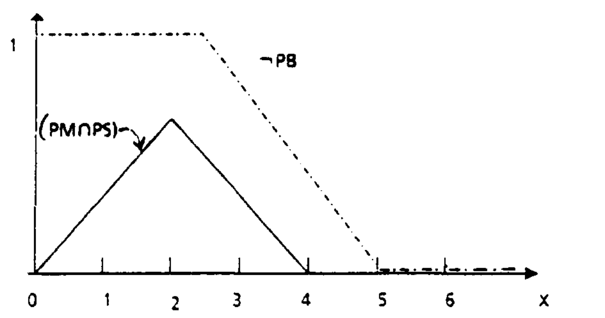

Figure 3 Membership functions for (*PM* AND *PS*) and for (¬*PB*)

(5) and (7) *Example 2* gives for the membership function (*PM* AND *PS*) (see *Figure 3*)

$$\begin{aligned}\mu(PB \text{ AND } PS) = \{&\min[0,1]/0, \min[0.3,1]/1,\\ &\min[0.7,0.7]/2, \min[1,0.3]/3,\\ &\min[0.7,0]/4, \min[0.3,0]/5,\\ &\min[0,0]/6\}\\ =\{&0/0, 0.3/1, 0.7/2, 0.3/3, 0/4,\\ &0/5, 0/6\}.\end{aligned}$$

3. The complement of a set A is defined by $\neg A$, with membership function

$$\mu_{\neg A}(x) = 1 - \mu_A(x), \qquad (8)$$

which corresponds to the logical negation NOT. For *Example 2*, NOT $PB \equiv \neg PB$ membership function is

$$\begin{aligned}\mu(\neg PB) = \{&(1-0)/0, (1-0)/1, (1-0)/2,\\ &(1-0.3)/3, (1-0.7)/4, (1-1)/5,\\ &(1-1)/6\}\\ =\{&1/0, 1/1, 1/2, 0.7/3, 0.3/4, 0/5, 0/6\},\end{aligned}$$

which is illustrated in *Figure 3*.

Fuzzy relations and algorithms

Control systems are essentially relationships or mappings between inputs and outputs of a controller. A fuzzy algorithm can be used to represent such a mapping as a set of situation-action pairs, between a fuzzy input variable U_i and the corresponding fuzzy output variable S_i defined over disparate universes of discourse X and Y respectively. Each situation-action pair is represented by a linguistic implication or conditional rule of the form

$$U_i \rightarrow S_i \quad \text{or} \quad IF\ U_i\ THEN\ S_i. \qquad (9)$$

This implied relation R_i is expressed in terms of the Cartesian product of the sets U_i and S_i, denoted by

$$R_i = U_i \times S_i. \qquad (10)$$

The membership function of which is given by

$$\begin{aligned}\mu_{R_i}(x,y) = \mu_{U_i \times S_i}(x,y) &= \min\{\mu_{U_i}(x), \mu_{S_i}(y)\}\\ &= \mu_{U_i}(x) \wedge \mu_{S_i}(y). \qquad (11)\end{aligned}$$

When U_i and S_i are defined by membership functions on finite discrete universes X and Y (i.e. $x_k \in X$: $k=1,\ldots,m$ and $y_l \in Y$; $l=1,\ldots,n$) then the relation R_i is given by the finite discrete relational matrix $\mu_{R_i}(k,l)$.

$$\mu_{R_i}(k,l) = \mu_{U_i}(x_k) \wedge \mu_{S_i}(y_l). \qquad (12)$$

Example 3

Consider the control rule:

R: IF (error is PB)
THEN (regulator position is reduced to PS)

for PB, PS defined in *Example 2*, hence from (12) and (5)

$$\mu_R(k,l) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & \cdot & \cdot\\ 0 & 0 & 0 & 0 & \cdot & \cdot & \cdot\\ 0.3 & 0.3 & 0.3 & 0.3 & \cdot & \cdot & \cdot\\ 0.7 & 0.7 & 0.7 & 0.3 & \cdot & \cdot & \cdot\\ 1 & 1 & 0.7 & 0.3 & \cdot & \cdot & \cdot\\ 1 & 1 & 0.7 & 0.3 & 0 & 0 & 0 \end{bmatrix}$$

A fuzzy algorithm is formed by collecting together several conditional rules, as given by (9), and combining them using the ELSE (union) operator:

$$IF\ U_1\ THEN\ S_1,\ ELSE\ IF\ U_2\ THEN\ S_2, \ldots,\ ELSE\ IF\ U_i\ THEN\ S_i.$$

If R is the relation describing the complete algorithm, then from (10)

$$\begin{aligned}R &= (U_1 \times S_1) \cup (U_2 \times S_2) \cup \cdots \cup (U_i \times S_i)\\ &= R_1 \cup R_2 \cup \cdots \cup R_i. \qquad (13)\end{aligned}$$

The corresponding membership function, for the case of finite discrete fuzzy sets, from (11) and (6) is

$$\begin{aligned}\mu_R(k,l) &= \max\{\mu_{R_1}(k,1), \mu_{R_2}(k,l), \ldots, \mu_{R_i}(k,l)\}\\ &= \max_i \{\mu_{U_i}(x_k) \wedge \mu_{S_i}(y_l)\}\\ &= \bigvee_i (\mu_{U_i}(x_k) \wedge \mu_{S_i}(y_l)). \qquad (14)\end{aligned}$$

The above can be readily extended to fuzzy relationships of multi-dimensional rules such as the case:

IF U_1 THEN IF F_1 THEN IF G_1 THEN IF H_1 THEN S_1 ELSE
IF U_2 THEN IF F_2 THEN IF G_2 THEN IF H_2 THEN S_2 . . . .

The relation describing the algorithm is now given by

$$R = \bigcup_i (U_i \times F_i \times G_i \times H_i \times S_i). \tag{15}$$

A typical controller has error E and change in error ΔE as inputs and control U as output. Rules are constructed from sets E_i, ΔE_i and U_i defined by fuzzy sets on the discrete universes, e.g. $e_j \in E; j = 1, \ldots, o$, $\Delta e_k \in \Delta E$; $k = 1, \ldots, m$ and $u_l \in U$; $l = 1, \ldots, n$. An example of a typical rule and the relation R of the overall controller are given by

R_i: *IF (error is E_i) AND (error change is ΔE_i) THEN (control is U_i).* (16a)

$$\mu_R(j, k, l) = \bigvee_i (\mu_{E_i}(e_j) \wedge \mu_{\Delta E_i}(\Delta e_k) \wedge \mu_{U_i}(u_l)). \tag{16b}$$

It is impractical to have a rule for every situation, therefore use is made of the compositional rule: given $R = A \times B$, the value B' which is inferred by R for any given A' is

$$B' = A' \circ R, \tag{17}$$

where $\circ$ denotes the compositional or max-min operator. The membership function for B' is given by

$$\mu_{B'}(b) = \max_a \min\{\mu_{A'}(a), \mu_R(a, b)\}. \tag{18}$$

Example 4

For the control rule or relational matrix of *Example 3* suppose that it is required to determine the change in regulator position if the positional error is 'about 3' (i.e. *PM* with membership function given by (5)). From this relational matrix R, equations (5) and (18), the regulator positional change membership function is

$$\mu_{Regulator\ position}(U_l) = \max_k \min\{\mu_{PM}(e_k), \mu_R(e_k, u_l)\}$$
$$= \{0.7/0, 0.7/1, 0.7/2, 0.3/3, 0/4, 0/5, 0/6\}.$$

Note that the control set has a membership function less than one, denoting incomplete knowledge of the situation—but at least a decision can be made! For practical purposes a deterministic value of regulator positional change is required, therefore the resultant membership function must be defuzzified.

Defuzzification

Given a fuzzy relation $R: U \rightarrow S$ formed from a fuzzy algorithm which describes the relationship between x and y and a particular measured value x_0 of x. x_0 is a non-fuzzy singleton, since its membership function is unity at $x = x_0$ and zero elsewhere. The composition of the non-fuzzy singleton set of x_0, with the fuzzy relation R, produces a fuzzy set S on Y. To produce a non-fuzzy singleton output y_0 as a consequence of a deterministic input x_0, the set S has to be defuzzified to a singular value y_0. Two methods of defuzzification are commonly used:

1. Mean of Maximum (MOM)[20] generates a value which corresponds to the maximum grade of membership in S; in the case when there is more than one maximum with the same magnitude, the algorithm generates a value which represents the mean of all local maxima in S, i.e.

$$y_0 = \sum_{j=1}^{J} \frac{y_j}{J}, \tag{19}$$

where $y_j = \max \mu_S(y)$; $J = |\{y\}|$.

2. Centre of Area (COA) method divides the first moment of area under the membership function into half, and the y value marking the dividing line is the defuzzified value of S; algorithmically this is expressed as:

$$y_0 = \frac{\sum_{k=1}^{m} y_k \mu_S(y_k)}{\sum_{k=1}^{m} \mu_S(y_k)}. \tag{20}$$

For *Example 4*, the regulator positional change would be $y_0 = 1.25$ by COA and $y_0 = 1$ by MOM defuzzification.

In control terms the MOM is analogous to a multi-level relay, whereas COA is analogous to the conventional PI controller.

FUZZY MODELLING AND CONTROL

From the properties of fuzzy sets, it has been shown that a set of *given* linguistic rules can be implemented

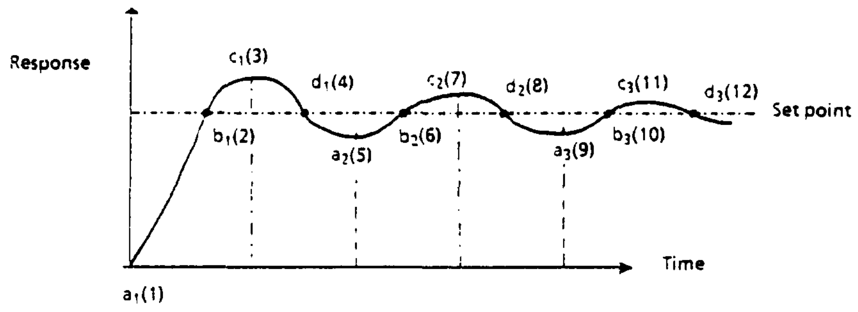

Figure 4 Open-loop response of a typical underdamped plant

through a fuzzy relation, and a single value deterministic output can be generated by defuzzification. The problem now is the generation of the set of rules that represents the system behaviour, and the sets on which they are defined. There are three stages to the construction of a fuzzy linguistic model of a system: (i) definition of the model class, (ii) rule production and (iii) assessment of model quality.

Definition of model class

Given a system's dynamic behaviour several characteristic parameters have to be determined[21]:

1. The rule dimension; those input/output variables which have a causal relationship, e.g. a system with two inputs (say, error and error change) and one output (say, control) is three-dimensional $(E, \Delta E \rightarrow U)$.
2. The range of the rule variables.
3. The number of quantization levels required on *each* variable range. The quantization spacing is selected on the basis of the smallest change in a variable that has to be detected.
4. The number of fuzzy sets (and associated membership function) for each range. Increasing the number of fuzzy sets leads to increased resolution, but greater sensitivity to measurement noise. A useful practical guide is the number of fuzzy sets on a variable's range is the smallest integer that satisfies (range)/(5 × standard deviation of noise).

Rule production

There are three methods of achieving a linguistic set of fuzzy rules that represents a system's input/output behaviour—verbalization, fuzzification and identification.

1. *Verbalization.* This is the most subjective method, and directly akin to expert systems rule acquisition, whereby a rule set is generated by interrogating an experienced operator through structured questions[22]. As our feedback controller example, consider the open-loop plant response of *Figure 4*, for which a controller is to be synthesized on the basis of the 'operator' observing zero crossing points (b_i, d_i) and maxima and minima points (a_i, c_i). and error change ΔE (current error, previous error) at these points. The variables $(E, \Delta E)$ can be, for example, represented by seven fuzzy linguistic qualifiers $\{NB, NM, NS, AZ, PS, PM, PB\}$, hence there are $7 \times 7 = 49$ possible control rules of the form of (16) of which the 'sample' points (a_i, b_i, c_i, d_i) provide 12 rules which are represented in the rule *Table 1*. The table *entries* denote the control output 'suggested' by an expert for the appropriate combinations of $(E, \Delta E)$ to produce a feedback controller with a fast response with minimum peak overshoot. The consequent output response is shown in *Figure 5* as trace *A*; an improved response, trace *B*, can be achieved by deriving six additional rules (numbers 14–19 in *Table 1*) based on the differences $(a_i - b_i)$ and $(c_i - d_i)$ for $i = 1, 2, 3$ in *Figure 4*. Even with this improved feedback behaviour there are potentially another 30 rules to

Table 1 Control rules for three-dimensional fuzzy controller. Linguistic definitions: $P \triangleq$ positive, $N \triangleq$ negative, $B \triangleq$ big, $M \triangleq$ medium, $S \triangleq$ small

ΔE / E	NB	NM	NS	AZ	PS	PM	PB
NB			NM(16)	NB(3)			
NM				NM(7)			
NS	PM(17)		AZ(19)	NS(11)			
Z	PB(4)	PM(8)	PS(12)	AZ(13)	NS(10)	NM(6)	NB(2)
PS				PS(9)	AZ(18)		NM(15)
PM				PM(5)			
PB				PB(1)	PM(14)		

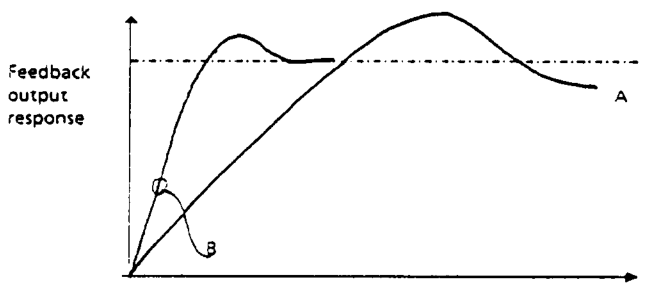

Figure 5 Verbalized feedback controlled response

be found, or alternatively there are 30 combinations if $(E, \Delta E)$ for which there is no control decision. To overcome this inadequacy of modelling, nearest neighbourhood type rules are implemented.

2. Fuzzification involves the formulation of a set of rules from a mathematical expression that represents the system. The advantage of this method is having obtained a set of rules, they can be readily validated against the known analytical model.

Example 3

In the context of vehicular motion, consider a breaking or decelerating vehicle with initial velocity v_0, and constant deceleration $-a$, then the stopping distance d is

$$d=\int_0^{t_s} v(t)\,dt, \qquad \text{but} \qquad v(t)=v_0+\int_0^{t_s} a\,dt$$
$$\text{hence} \qquad d=\frac{v_0^2}{2a}. \tag{21}$$

Thus (velocity, acceleration) are input variables that are mapped through (21) into (distance), i.e. there is a linguistic causal relationship

$$\langle\text{velocity, acceleration}\rangle \rightarrow \langle\text{distance}\rangle.$$

For acceleration control, this cause-effect relationship needs to be inverted to

$$\langle\text{velocity, distance}\rangle \rightarrow \langle\text{required acceleration}\rangle$$

(in mathematical terms $a=v_0^2/2d$). In general, the causal relationship R may not be isomorphic and invertible.

Example 6

Define the following linguistic qualifiers for velocity, distance and deceleration respectively; velocity = (very slow, slow, medium, medium fast, fast, very fast), distance = (almost zero, very close, close, medium, medium far, far, very far), deceleration = (almost zero, braking, hard braking, very hard braking); inversion of (21) leads to the braking rule *Table 2*.

In practice the rule table needs to be parameterized by environmental conditions such as (road surface, temperature, humidity, vehicle dynamics, . . .)—see section 'Self-organizing or learning fuzzy controllers' for adaptive fuzzy rule-based control.

3. *Identification* provides a system model for either the purpose of output prediction and simulation or for the design of a feedback controller. Fuzzy-set theory generates system models[5,24–36] that are descriptive and experiential, rather than the more classical integro-differential equation representations, and therefore do not rely on technical constraints (such as linearity or Lipschitz conditions) being satisfied. As with all identification or estimation algorithms, fuzzy-set modelling requires adequate and representative input/output signal pairs that provide a rich and completely dynamically stimulating data-base so that the consequent model, or estimated relational matrix R represents all, or nearly all, possible input/output situations.

Consider a causal and a time-invariant process, represented by a finite-dimensional fuzzy relation R that maps current states $S(t)$ into future states $S(t+T)$ (T is the sample period) for some input $U(t)$, i.e.

$$S(t)\times U(t) \xrightarrow{R} S(t+T). \tag{22}$$

The fuzzy relation or algorithm R is a collection of implication statements or rules of the form:

$$R_i\text{: } \mathit{IF}\, S_i(t)\, \mathit{AND}\, U_i(t) \;\; \mathit{THEN}\, S_i(t+T); \qquad i=1,2,\ldots,N. \tag{23a}$$

Table 2 Static braking fuzzy rule base

distance	velocity					
	VS	S	M	MF	F	VF
AZ	B	HB	HB	VHB	VHB	VHB
VC	B	B	HB	HB	VHB	VHB
C	AZ	B	B	HB	HB	VHB
M	AZ	B	B	B	HB	VHB
MF	AZ	B	B	B	HB	VHB
F	AZ	AZ	B	B	B	HB
VF	AZ	AZ	AZ	B	B	B

If $S_i(t)$, $U_i(t)$ and $S_i(t+T)$ are described by fuzzy sets with membership functions

$$\mu_{S_i(t)}, \mu_{U_i(t)}, \mu_{S_i(t+T)}$$

respectively, then by (7) the membership function for this particular rule R_i is

$$\mu_{R_i}(S(t), U(t), S(t+T)) = \min_i \{\mu_{S_i(t)}(S(t)), \mu_{U_i(t)}(U(t)), \mu_{S_i(t+T)}(S(t+T))\}. \tag{23b}$$

For practical applications, the fuzzy sets are described on finite discrete universes; in this case both R_i and the relational matrix R are finite discrete fuzzy relations. To construct R, collect all N rules, R_i, as R_1 OR R_2 OR R_3 OR ... OR R_N; hence from (6), the membership function of R is

$$\begin{aligned}\mu_R(S(t), U(t), S(t+T)) &= \max\{\mu_{R_i}(S(t), U(t), S(t+T))\}\\ &= \max_i \{\min\{\mu_{S_i(t)}(S(t)), \mu_{U_i(t)}(U(t)),\\ &\qquad \mu_{S_i(t+T)}(S(t+T))\}\}. \end{aligned} \tag{24}$$

The relational matrix R has to be evaluated from system input/output data, its dimensionality N being unknown *a priori*. Therefore consider every data set $(S_i(t), U_i(t), \ldots, S_i(t+T))$ as a possible rule, and use it to update the relational matrix R. However, it is necessary to identify and track systems with time-varying parameters—including those with catastrophic changes through system faults. Hence the relational matrix is dynamic—implying both learning and forgetting—through a forgetting operator D. At each update a new version, say, R' is computed from the previous version R through:

$$\begin{aligned}\mu_{R'}(S(t), U(t), S(t+T)) &= \max\{(D \times \mu_R(S(t), U(t), S(t+T))),\\ &\qquad \max_i \{\min\{\mu_{S_i(t)}(S(t)), \mu_{U_i(t)}(U(t)),\\ &\qquad \mu_{S_i(t+T)}(S(t+T))\}\}\}, \end{aligned} \tag{25}$$

where $D < 1$ is the forgetting factor causing old rules to decay slowly as new ones are added. Selection of D determines the speed of adaptation; the slower the rate the less susceptible the modelling process is to noise (simply by the averaging-out principle). The forgetting factor mechanism introduces a problem due to the uneven distribution of the input signal space over the relational matrix. If data is heavily biased to a particular region of R, then the forgetting factor will reduce all rules external to this region to zero. To avoid this, the updating procedure in (25) only applies for those $\{R_i\}$ for which our confidence in the data set

$$\{S_i(t), U_i(t), S_i(t+T)\}$$

being relevant to $\{R\}$ is greater than a prescribed threshold θ, i.e.

$$\min\{\mu_{S_i(t)}(S(t)), \mu_{U_i(t)}(U(t))\} > \theta. \tag{26}$$

Having derived a relational matrix R, the system rule base can be used with specific measured values of $S(t)$ and $U(t)$ (say, $S(t)'$ and $U(t)'$) to predict the corresponding output value $S(t+T)'$ through the composition operator

$$S(t+T)' = (U(t)' \times S(t)' \circ R) \tag{27}$$

or as a membership function

$$\mu_{S(t+T)'} S(t+T) = \max\{\min\{\mu_{S'(t)}(S(t)), \mu_{U'(t)}(U(t)), \mu_R(S(t), U(t), S(t+T))\}\} \tag{28}$$

which must be defuzzified (by maxima or centre of area method) to get the deterministic output prediction $S(t+T)'$.

Example 7

To evaluate the above fuzzy estimator/predictor, consider a single input/output system with second-order structure with model rules:

$$IF\, X_1(t)\, AND\, X_2(t)\, AND\, U(t)\, THEN\, X_3(t), \tag{29}$$

where X_1, X_2 and X_3 represent actual positional output, velocity and predicted acceleration respectively. Predicted acceleration X_3 is the constant value over one time period T, which takes the system from current state $S_t = (X_1(t), X_2(t))$ to a next state $S_{t+T} = (X_1(t+T), X_2(t+T))$. Having determined the rule structure, the discretization levels used in evaluating the fuzzy sets on each variable must be selected. Discretization of X_1 and U is linear over the expected input/output range $(-10, 10)$, to give uniform sensitivity across the range; whereas X_2 and X_3 used nonlinear discretization to give optimum sensitivity around zero, while retaining adequate range. Seven discretization levels were selected, hence for four variables (X_1, X_2, X_3, U) a $7 \times 7 \times 7 \times 7$ relational matrix is required! For the purposes of simulation an 'unknown' second-order linear system with damping ratio 0.2 and natural frequency 4.4 rad s^{-1} was selected together with modelling parameters of $D = 0.95$, $\theta = 0.4$ and a rate

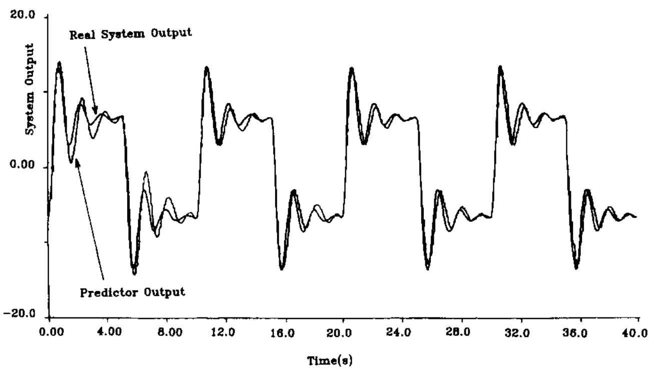

Figure 6 Fuzzy predictor output for unknown second-order plant ($\xi = 1.0$, $\omega_n = 4.4$ rad/sec)

of 20 Hz. The relational matrix R was evaluated from (25), (26) and (28) from an initially empty rule base; the one-step-ahead predictor (27) was driven by the estimated relational matrix R (essentially a dynamic expert-system rule base). *Figure 6* compares the real output response with the predictor output, for a square wave input; to demonstrate that the relational matrix has converged to a representative model rule set, the rule-updating process has been terminated after 20 seconds. Error convergence would have continued had not the rule base been stopped.

Model quality

There are several possible measures of model adequacy

1. The number of rules—the more rules required, the more complex the model.
2. Mean squared error between model predicted output and actual system output.
3. Normalized absolute error (normalized by dividing by the number of quantization rules).
4. Mean uncertainty—a set of fuzzy inputs to the model generates a fuzzy output set, if the membership function is 'peaky' then it is a 'fairly certain' output. Associated uncertainty can be expressed numerically by $1 - \max$ (membership grade). Usually a combination of the above criteria is used to evaluate a fuzzy model.

CONTROLLER SYNTHESIS

There are essentially three methods of synthesizing a rule-based controller

1. Verbalization (see also the section above, on rule production).
2. Inverse composition of the fuzzy relation of the system to be controlled and the fuzzy relation of the desirable system behaviour.
3. Inversion of the system rules and substitution of the rules of the desirable behaviour.

Verbalization

The vast majority of application studies have used this method to synthesize fuzzy logic controllers. The rules are derived by placing the controller in parallel with a human expert and learning or imitating the control actions for particular input/output situations. An example of a typical feedback two-term controller (essentially, equivalent to a classical PD controller) was given above. Practical applications include aircraft flight control[22], autopilots[27], warm water plants[30], PID servo controllers[11,21,28], and vehicle control[11,29].

Example 8: Case Study into automatic vehicle parking[16]

The purpose here is to design a controller that can, with adequate measurements, autonomously manoeuvre a vehicle into and park in a confined space. Three scenarios are considered:

1. Parallel forward parking.
2. Parallel reverse parking and
3. Slot or garage parking (see *Figure 7*).

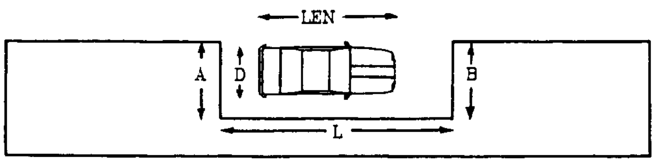

Figure 7a Parallel parking configuration

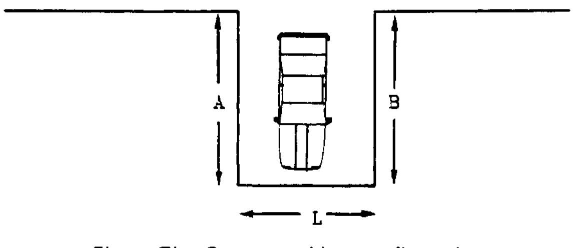

Figure 7b Garage parking configuration

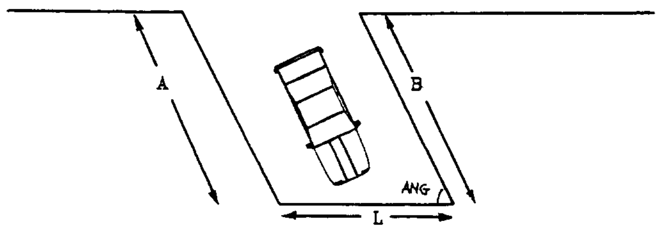

Figure 7c Garage at angle parking configuration

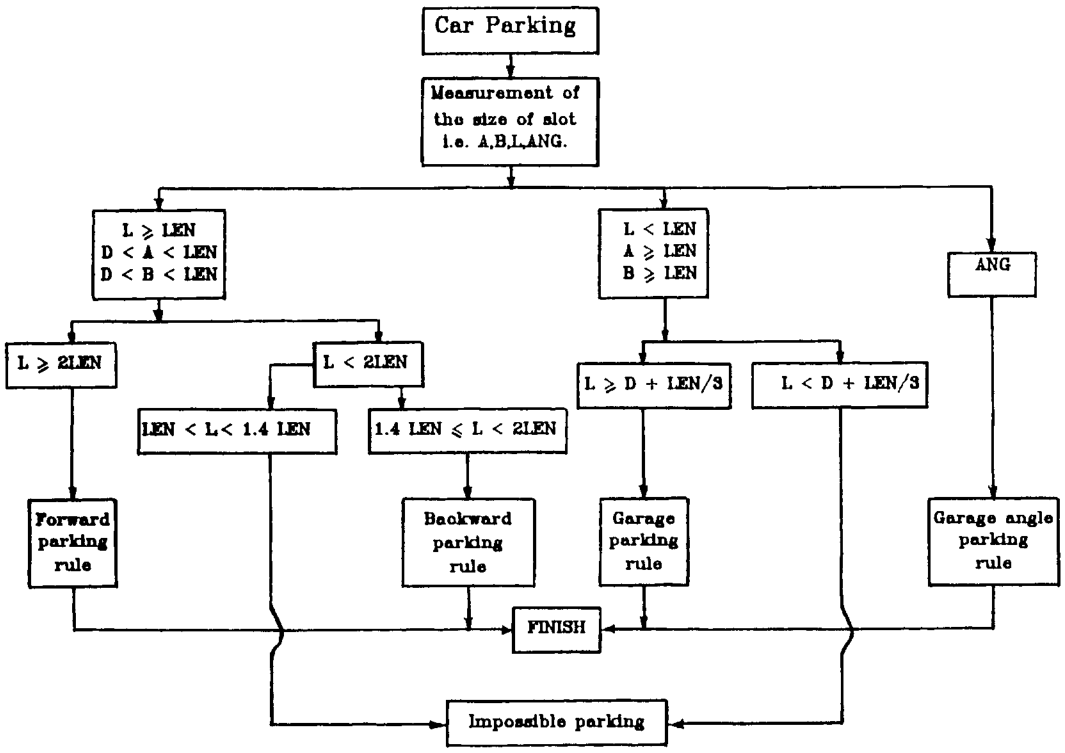

Figure 8 Parking-type logic

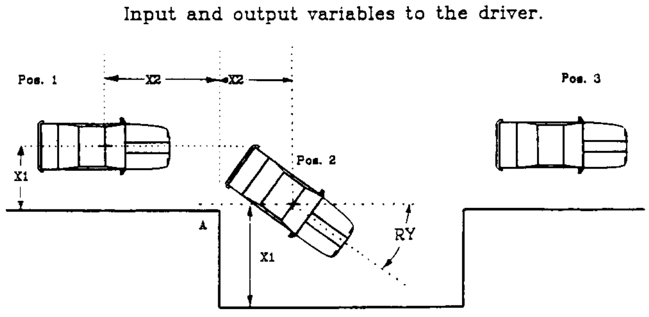

Figure 9 Parking vehicle measured variables

The control rules must also include a decision process that evaluates the potential parking slots; essential to this is the satisfaction (or otherwise) of a set of geometric constraints that define the various parking possibilities (see *Figure 8*). For forward parking, the vehicle's front wheel steering geometry requires a larger space than for reverse parking. For garage or slot parking an additional door-opening inequality must be satisfied to allow passengers to enter or leave the vehicle.

The possible control variables for fine positioning include steering angle and angular rate, and translational velocity. The vehicle's dynamics can be safely ignored during fine manoeuvres as the vehicle velocity is low, hence control variables selected are steering angle and translation velocity $\in (-c, 0, c)$ where c is a constant (typically <2.5 msec^{-1}). The input variables and vehicle location and direction (see *Figure 9*) are determined by two distance measures X_1 and X_2, where X_1 is vehicle distance from side wall/parked vehicles and X_2 is vehicle distance from parking slot entrance. The fuzzy sets for the input/output variables are shown in *Figure 10*. The control rule bases for the three scenarios are generated by modelling an expert driver, and then validating the resultant rule base by simulation. A typical rule base for forward parking is given in *Table 3*, note that the table has two output entries—one relating to required steering angle, the other to the translational velocity. A typical forward parking trajectory is shown in *Figure 11*; observe that the vehicle reverses to the back of the parking slot ready for the driving-out manoeuvre.

For reverse parking, the vehicle is driven forward past the parking slot, prior to being reverse parked; the iterated rule base is given in *Table 4*. A typical computer simulation is shown in *Figure 12*.

Similarly for garage or slot parking the rule base for the reversing mode is given in *Table 5*, with its associated simulation in *Figure 13*. More-complex fine positioning shapes can be decomposed into a set of primitives whose constituent elements are given above. Another fuzzy rule base is then required to interface the primitives.

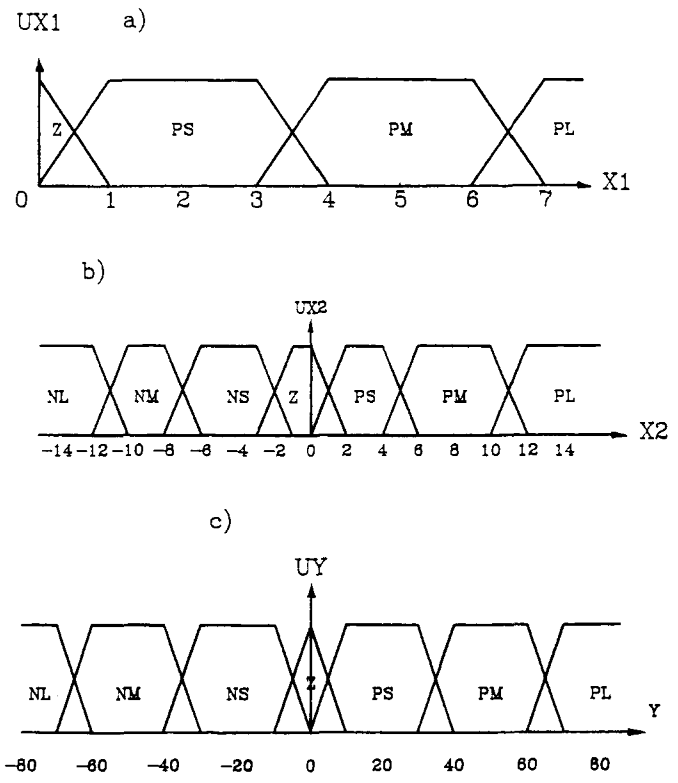

Figure 10 Membership functions for parking variables: (a) distance from parked car, (b) distance from entrance corner, (c) output angle

Table 3 Forward parking rule base

X_1 \ X_2	PB	PM	PS	AZ	NS	NM	NB
PB	PS / C	PS / C	AZ / C	PM / C	PM / C	PM / C	/ AZ
PM	NS / C	AZ / C	PS / C	PM / C	PM / C	PS / C	NS / -C
PS	NM / C	NB / C	PS / C	PS / C	PS / C	AZ / C	AZ / AZ

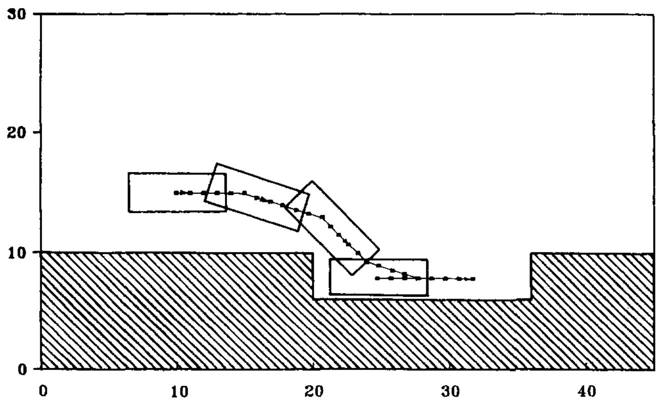

Figure 11 Forward parking trajectory

Table 4 Reverse parking rule base

X_1 \ X_2	PB	PM	PS	AZ	NS	NM	NB
PB	PM / -C	PS / -C	PS / -C	PM / -C	PB / -C	NM / -C	NS / -C
PM	AZ / -C	AZ / -C	AZ / -C	AZ / -C	PS / -C	NM / -C	NS / -C
PS	NS / -C	NS / -C	NS / -C	AZ / AZ	AZ / -C	NM / -C	NS / -C

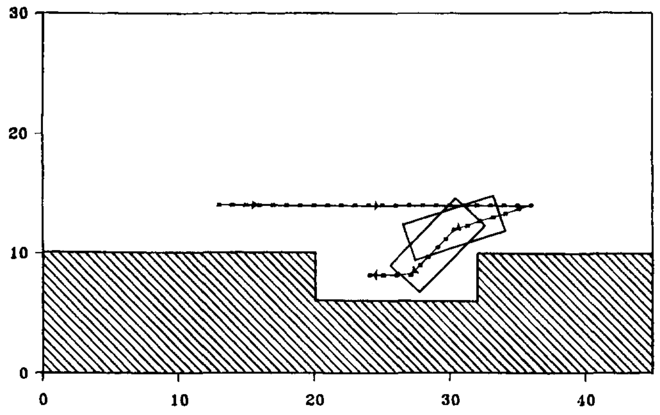

Figure 12 Reverse parking trajectory

Table 5 Reverse slot parking rule base

X_1 \ X_2	PB	PM	PS	AZ	NS	NM	NB
PB	PS / -C	PS / -C	AZ / -C	NS / -C	NS / -C	NM / AZ	NB / AZ
PM	AZ / -C	AZ / -C	AZ / -C	NS / -C	NB / AZ	NB / AZ	NB / AZ
PS	NS / AZ	NS / -C	AZ / -C	NM / -C			

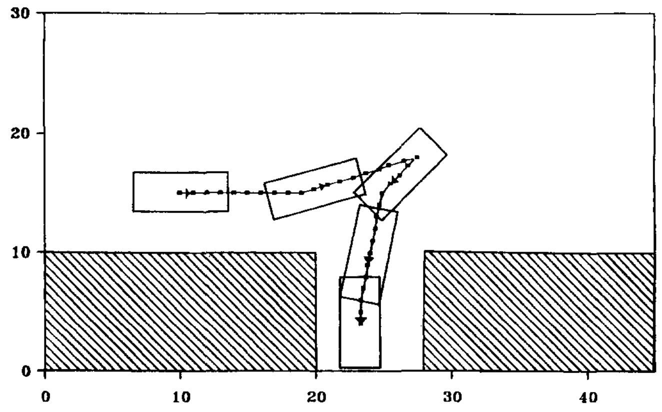

Figure 13 Reverse garage or slot parking

Inverse composition of the fuzzy relation of the system and fuzzy relation of desired system behaviour

Let the system be described by the discrete fuzzy state transition or mapping

$$S(t)\times U(t) \xrightarrow{R_S} S(t+T). \tag{30}$$

Given a desired system performance which infers a fuzzy set on $S'(t+T)$, from fuzzy sets $S'(t)$ and $G(t)$ (for G the set of system inputs demands) represented by the fuzzy relation R_P, i.e.

$$S'(t)\times G(t) \xrightarrow{R_P} S'(t+T). \tag{31}$$

The problem is to find a fuzzy controller R_C such that

$$S(t)\times G(t) \xrightarrow{R_C} U(t) \tag{32}$$

from which it is easy to see that

$$R_P = R_C \circ R_S. \tag{33}$$

Hence to evaluate R_C, it is necessary to generate an inverse composition operator ($\circ^{-1}$) so that

$$R_C = R_P \circ^{-1} R_S. \tag{34}$$

Unfortunately this inverse composition is difficult to achieve owing to the non-uniqueness of solution[31], also the resultant controller is a fuzzy relation rather than a rule base. An approximate compositional inversion can be achieved by a trial-and-error method[32]. The third fuzzy-logic controller method similarly produces a non-unique solution, but the inverse operation is produced linguistically.

Inverse of system rules and substitution of the rules of behaviour

This method[21] performs inverse composition directly on the algorithms (30), (31) by linguistically inverting the system algorithm (30) to produce a new algorithm R_I such that

$$S(t)\times S(t+T) \xrightarrow{R_I} U(t). \tag{35}$$

This is achieved by systematically searching the rules of the plant relational matrix R_S. This linguistic search produces three possible outcomes: (i) ideally, one fuzzy set on the universe U; (ii) no fuzzy sets on U (in this case the nearest-neighbour fuzzy set on U has to be selected); (iii) many fuzzy sets on U (in the case of a non-unique solution, the optimal set on U is required). This is generated by linguistically substituting the performance algorithm (31) into (35) to produce a new algorithm

$$(S(t)\times G(t) \xrightarrow{R_P} S(t+T))\times S(t) \xrightarrow{R_I} U(t)$$

or

$$(S(t)\times G(t)) \xrightarrow{R_{C'}} U(t) \tag{36}$$

and selecting those rules which minimize control effort.

SELF-ORGANIZING OR LEARNING FUZZY CONTROLLERS

The methods presented thus far have modelled a system as a set of linguistic causalities in the form of discrete fuzzy rules or algorithms, and by inversion of the causalities formulated a control rule base that optimizes some performance criteria. For many dynamical systems, such as AGVs there are substantial changes in system parameters or environment which must be reflected in turn in the model and in the consequent control rule base. To effect a self-adaptive or self-organizing rule-base controller[5,11,33,34] a fixed performance measure P is required. The performance relation P is composed of a static set of rules driven by, say, error $E(t)$ (between actual system behaviour) and error change $\Delta E(t)$; the control system designer specifies those combinations of $(E, \Delta E)\rightarrow U$ that lead to good or optimal system behaviour. Two approaches to performance index optimization are possible; to illustrate these methods, a second-order dynamic process (representing an AGV's yaw dynamics) is selected as the system to be controlled, but *without* any *a priori* knowledge by the controller. As with previous examples a language with seven linguistic qualifiers is used $L_{(\cdot)} = \{PB, PM, PS, AZ, NS, NM, NB\}_{(\cdot)}$.

Cost output performance link

The performance index *Table 6* entries are costs or penalties associated with combination of measured $(E, \Delta E)$ that are used to modify those rules which contributed to the poor system behaviour. The almost zero band represents a fuzzy set of ideal $(E, \Delta E)$ combinations, a control rule change is initiated if $(E, \Delta E)$ combination lies off the ideal band. The output of a rule is an input to the plant or system, hence the change (and direction) required to a rule output is determined by the performance table entry. Whilst the table indicates the direction and magnitude of rule change, it is still necessary to determine which control rule(s) contributed to

Table 6 Cost output performance rule table

E \ ΔE	NB	NM	NS	AZ	PS	PM	PB
NB	NB	NB	NB	PM	PS	AZ	AZ
NM	NB	PM	PM	PS	AZ	AZ	AZ
NS	NB	PM	PS	AZ	AZ	AZ	NS
AZ	PM	PS	AZ	AZ	AZ	NS	NM
PS	PS	AZ	AZ	AZ	NS	NM	NB
PM	AZ	AZ	AZ	NS	NM	NM	NB
PB	AZ	AZ	NS	NM	NB	NB	NB

poor performance, and those rules for which the system is most responsive. The controller evaluates all rules in parallel; applicable rules are selected by the inference property of the fuzzy relationship. Those rules that made the greatest contribution to the controller output at each sample are memorized; when a rule change is indicated by the performance table, the offending rule(s) can be readily identified. If a transportation delay of m-samples occurs between the process input/output, then the previous (m) set of rules cannot affect the current process output. In general ($m+k$) samples in the past can be changed where $k \geqslant 1$; however, all investigators[11,33,34] have used $k=1$ in practical studies. When there are several penalty-inducing rules, the amount of change may be the same for each rule, or could be weighted according to 'blame'.

As a rule-based controller can be described in two forms—as a fuzzy relation or as an algorithm—there are two basic methods of rule changing. When the rules are in algorithmic form (which has to be ultimately converted to a fuzzy relation) the consequent linguistic qualifier is simply changed. However, when rules are contained in a fuzzy relation R, a more-complex and computationally inefficient procedure for rule adaptation is necessary:

1. form a fuzzy relation R_1 of the old rule (which is to be changed)
2. form a fuzzy relation R_2 of the new rule
3. the rule change produces a new controller fuzzy relation R'

$$R' = (R \wedge \neg R_1) \vee R_2. \tag{37}$$

Example 9: Case Study into cost output performance-based self-organizing control of an AGV lateral dynamics

To demonstrate the self-organizing and robustness of an intelligent fuzzy-control algorithm, consider the control of an AGV lateral dynamics. For the purpose of illustration, assume that these yaw-type dynamics can be expressed as a second-order linear differential equation with constant coefficients (see 'Introduction').

$$\ddot{\theta}_v + 2\xi\omega_n\dot{\theta}_v + \omega_n^2\theta_v = K_1\omega_n^2\varepsilon, \tag{38}$$

where ξ is the damping ratio (nominally 0.5), ω_n is the undamped natural frequency (nominally $f_n = 10$ Hz). This model is selected purely for the purposes of a demonstrator process or plant database which can be readily validated; the proposed controller algorithm does not require process linearity or stationarity or *a priori* knowledge of internal plant dynamics/parameters. In practice the coefficients ξ, ω_n, K_1 are functions of the vehicle longitudinal velocity (assumed piecewise constant), wheel and axle forces, vehicle payload and wheel/road friction coefficient—all of which are assumed known (or can be estimated on-line) and constant over the control cycle.

The requirement is for the fuzzy controller to provide the 'fastest' transient response to a step input with 'little' or no overshoot, the control input is restricted to $10 > |U|$.

The fuzzy controller is a three-dimensional algorithm which infers the controller output (u) based upon output error (E) and error change (ΔE). The variable ranges of (U, E, ΔE) are selected for 13 quantization levels each. Also, seven fuzzy sets are defined on each range of error and error change, that is $7 * 7 = 49$ rules are required. The seven fuzzy sets could be defined on the linguistic qualifiers (*NB*, *NM*, *NS*, *AZ*, *PS*, *PM*, *PB*). The feedback controller (see *Figure 14*) can be implemented with the rules stored as an algorithm, with self-adaptive

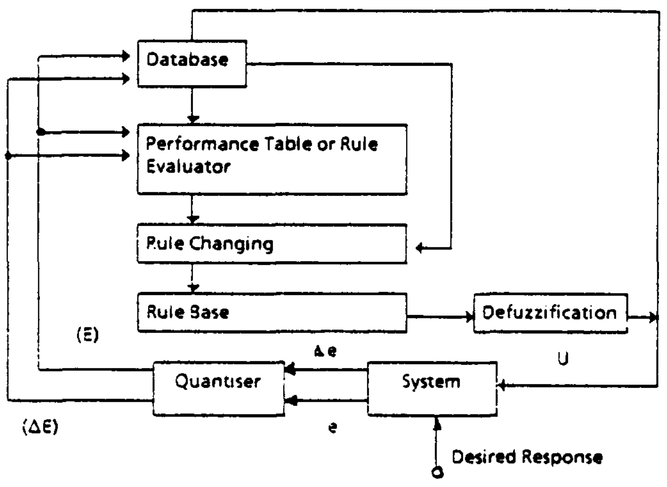

Figure 14 Self-organizing rule based fuzzy controller

rules. When compositional inference is used to infer an output from a fuzzy relation composed with non-fuzzy measurements (in this case E_0 and ΔE_0), then only those rules which are pertinent to the inputs have any effect upon the resulting fuzzy output set. A pertinent rule in this case is one for which the error and error change are *both* covering sets (defined as having a non-zero grade at the measured value). Hence for any error or error-change level, there will be one or two covering fuzzy sets; there are two covering sets in a level if they are overlapping at that level. That is, for any error, error-change measurement (quantized) combination (E_0, ΔE_0), there will be a maximum of *four* pertinent rules. Each of these rules can be implemented separately by forming a fuzzy relation of the rule and then performing a max-min composition with the quantized error and error-rate levels. The four fuzzy output sets can then be combined by maximization and the resulting fuzzy set is then de-fuzzified to give the non-fuzzy controller output. A typical rule is given by (16) for $i=1, 2, 3, 4$. Next the composition of this rule with the non-fuzzy singleton sets ($E_0, \Delta E_0$); the measured output set (U_0) generated by this rule has a membership function given by

$$\mu(U_0)=\min\{\mu_E(E_0), \mu_{\Delta E}(\Delta E_0), \mu_U(U)\}. \quad (39)$$

To implement this fuzzy controller it is necessary to perform:

1. One operation to evaluate $\min[\mu_E(E_0), \mu_{\Delta E}(\Delta E_0)]$
2. 13 operations to find U_0 for one rule
3. Each of the four rules can be processed in parallel
4. $13*3$ operations to combine the four pertinent rules by maximization to a single fuzzy output set
5. $13*2$ operations to find the first moment of area of the fuzzy output set
6. 13 operations to find the area of the fuzzy output set
7. 15 operations (typically) to find the non-fuzzy controller output by centre of area defuzzification
8. For self-adaptive control, one operation is required to check performance table and one operation to change rules (but these can be carried out in parallel).

This implies a total of 107 operations per sample period; assuming that one operation takes 10^{-6} sec of processor time, then 107×10^{-6} sec are required for completion of each fuzzy controller cycle task per sample. So even taking a sample rate of 100 times greater than the highest system resonant frequency, gives an upper plant cut-off frequency of 93 Hz. Digitization of error change, and error measurements is more likely to be a limiting factor than the plant highest resonant frequency.

Each fuzzy set can be represented as a number for storage—8-bit bytes are sufficient. The fuzzy-controller storage requirements are:

1. an array of $7*13$ bytes membership functions of the seven fuzzy sets
2. an array of 49 bytes to store the rules
3. 54 bytes directly associated with the rule processing
4. an array of $13*13$ bytes to store the performance table
5. an array of $10*3$ bytes to store the last 10 rules (typically) of most importance, and one byte for storing the rule reinforcement.

Therefore a total dedicated storage of 394 bytes (plus space for the programme) is required for implementing the three-dimensional fuzzy controller.

Returning to the demonstration example of (38), with $f_n=10$ Hz, a sample rate of 50 Hz is sufficient. For a minimum detectable yaw angular error of, say, 0.8% this should be just over one-half of a quantization level, hence error gain was selected as $>(0.008\times2)^{-1}$, i.e. about 70. Similarly, since the error-rate signal in the rule-based controller was found by taking the difference of two successive error signals, then the error change gain was set at $>2(0.008\times2)^{-1}$, i.e. about 150. Nineteen levels of quantization on each of the seven fuzzy spaces, numbered -9 to $+9$, were used. To achieve a self-adaptive controller, the performance *Table 6* for the 19 quantization levels in error, error change was utilized in the simulation.

For a second-order plant with damping ratio 0.5, $\omega_n=3$ rad/sec, the above rule-based controller was initially implemented *without* initial rules, after four control cycles for a positive unit step demand, and one with negative unit step demand, the rule base converged (i.e. no more rule changes were made) to those of *Table 7*.

Table 7 Converged rule base for $\xi=0.5$, $\omega_n=3$ rad/sec

ΔE / E	NB	PB	NS	AZ	PS	PM	PB
NB	PB	PB	PB	PB	PB	AZ	NB
NM	PB	PB	PB	PM	AZ	NS	NB
NS	PB	PB	PB	PM	AZ	NM	NB
AZ	PB	PB	PB	AZ	NB	NB	NB
PS	PB	PM	AZ	NM	NB	NB	NB
PM	PB	PS	AZ	NM	NB	NB	NB
PB	PB	AZ	NB	NB	NB	NB	NB

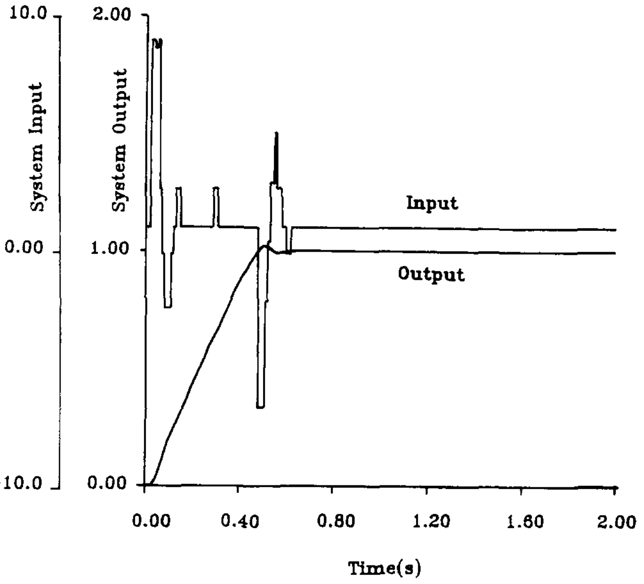

Figure 15 Step response; $\xi = 0.5$, $\omega_n = 3$ rad/sec

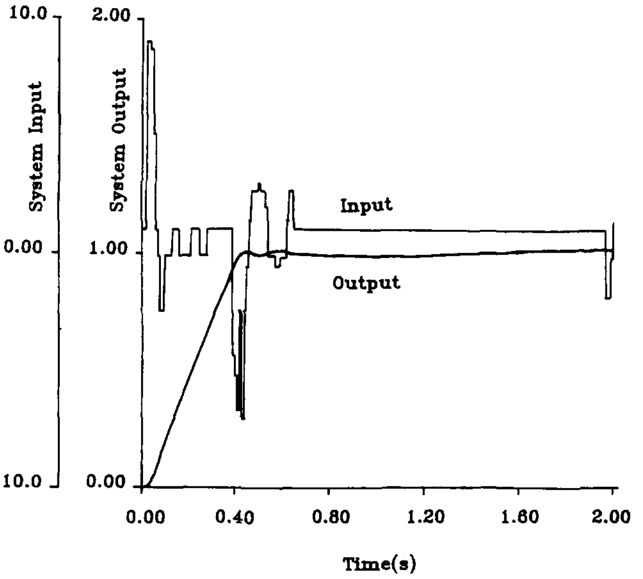

Figure 16 Step response; $\xi = 0$, $\omega_n = 3$ rad/sec

The closed-loop response is shown in *Figure 15*; this is essentially identical to an optimal conventionally designed proportional plus derivative controller response. To evaluate the robustness of the fuzzy-logic controller, the actual systems damping ratio was reduced to 0, and -1 (see *Figures 16* and *17* respectively) *without* this prior knowledge being provided to the controller database or rule base. In both cases the controller adapted its rule base well, and provided good transient response albeit with additional control effort. The comparable conventional two-term controller produced unstable responses in both cases.

Additional simulation tests[35] with additive noise, ramp inputs and sinusoidal inputs demonstrate similar robust and adaptive performance.

Next state performance based adaptive controllers

The majority of controllers are designed on the basis of a model or, in the case of fuzzy algorithms, the estimated or identified relational matrix R (see subsection 'Rule production', item 2.); i.e. find the control $U(t)$ that maps the plant state $S(t)$ into a desired state $S(t+T)$, whilst optimizing some performance relational matrix P, i.e.

$$S(t) \times S(t+T) \rightarrow U(t, P) \tag{40}$$

provided such an inverse relational matrix exists. The predictor/controller system structure is shown in *Figure 18*.

The performance relation P is composed of a static set of rules driven by, say, state errors $E(t)$ and error change $\Delta E(t)$; the rules are determined by the control-system designer. The output of the performance index is the required next state $S(t+T)$ to achieve the desired performance; this is used in (40) via the inverse relation R^{-1} to compute the desired control. Typical performance rules are of

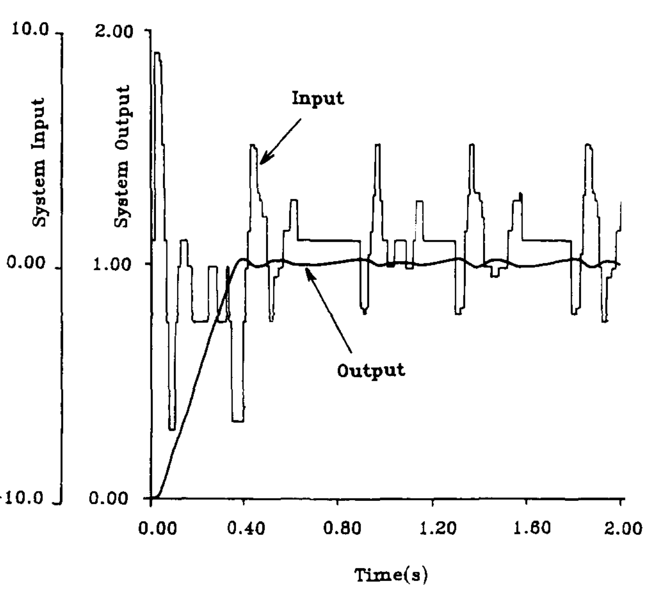

Figure 17 Step response; $\xi = -1$, $\omega_n = 3$ rad/sec

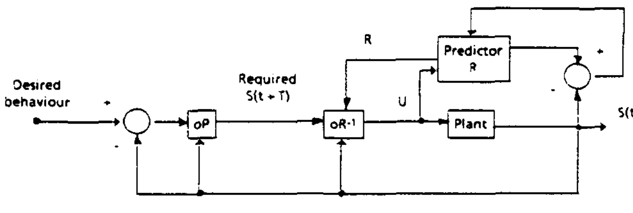

Figure 18 Self-organizing predictor/controller structure

Table 8 Next state performance rule base

E / ΔE	NB	NM	NS	AZ	PS	PM	PB
PB	PB	AZ	NB	NB	NB	NB	NB
PM	PB	PM	NM	NM	MB	NB	NB
PS	PB	PM	PM	NM	NM	NB	NB
AZ	PB	PM	PM	AZ	NM	NB	NB
NS	PB	PB	PM	PM	NM	NM	NB
NM	PB	PB	PM	PM	PM	NM	NB
NB	PB	PB	PB	PB	PB	AZ	NB

the form:

$$P_i\text{: } \textit{IF (error } E(t) \textit{ is } A_E\textit{) AND (plant state is } A_S\textit{) THEN (next state } S((t+T) \textit{ is } A_S\textit{))}, \tag{41}$$

where $A(\cdot)$ are linguistic qualifiers. The relational matrix P is constructed similarly to that of R_i in (22) and (23); the individual rules are then combined (as in (24)) to give the relational matrix P. Given non-fuzzy measurements of $E(t)$ and $S(t)$, $E_0(t)$ and $S_0(t)$ respectively, the fuzzy set $S_0(t+T)$ is computed from P by composition. The rules of P and the languages, $L(\cdot)$ (or sets of linguistic qualifiers) are selected by the designer to achieve some desired closed-loop system behaviour (for example, a step response with fast rise time and minimum peak overshoot).

Example 10: Continuation of Example 7

For the identified (second-order) model R of *Example 7*, the above performance rules are of the form:

$$P_i\text{: } \textit{IF (error } E(t) \textit{ is } A_E\textit{) AND (}\Delta E \textit{ is } A_E\textit{) THEN (}X_3 \textit{ is } A_{X_3}\textit{)}.$$

Table 8 illustrates a typical set of performance rules that lead to good closed-loop transient behaviour.

To evaluate the control[38] requires inversion of the model relational matrix R (defined by (22)); assuming that each rule of this form also implies its inverse rule given by

$$R_i^{-1}\text{: } \textit{IF } S_i(t) \textit{ AND } S_i(t+T) \textit{ THEN } U(t);$$

then R_i^{-1} can be evaluated from:

$$\mu_{R_i^{-1}}(S(t), S(t+T), U(t)) = \min\{\mu_{S_i(t)}(S(t)), \mu_{S_i(t+T)}(S(t+T)), \mu_{U_i(t)}(U(t))\}. \tag{42}$$

Comparing with (23), by equivalence, for particular values $S(t)$ and desired $S(t+T)$, say, $S_0(t)$ and $S_0(t+T)$, $U_0(t)$ is evaluated directly from R using composition.

$$\mu_{U_0(t)}(U(t)) = \max\{\min\{\mu_{S_0(t)}(S(t)), \mu_{S_0(t+T)}(S(t+T)), \mu_R(S(t), S(t+T), U(t))\}\}. \tag{43}$$

Previous use of this method[11] (and that in the previous section) for self-learning control has been highly successful, and requires little computational effort, and is readily implemented for real-time applications (such as an autopilot for large vehicle mass changes) via transputer arrays. Unfortunately this method requires an even distribution of rules within R; when partial or conflicting rules are present, the generated controls $U_0(t)'$ are biased towards recently updated values of $U(t)$. Since the controller also determines what is learnt by the model estimator, a coupling between control and identification emerges with the controller never 'teaching' its way out of partial rule situations.

An alternative numerical method of inverting R, that copes well with partial or conflicting rule situations (or data), utilizes the model predictor

$$S_0(t+T) = (U_0(t) \times S_0(t)) \circ R$$

to compute $S_0(t+T)'$ achieved at each of the discretization levels of $U_0(t)'$. *Figure 19* shows a typical plot of $S_0(t+T)'$ achieved for variable $U(t)$.

The control $U(t)$ is given by the intersection of $S_0(t+T)'$ with the desired state vector, through the following rules

1. If one cross-over point occurs, use linear interpolation between data points
2. If more than one zero or no cross-over occurs then use least-squares best-fit line to estimate a singular cross-over point
3. If no values of $S_0(t+T)'$ exist, then choose $U_0(t)'$ at random.

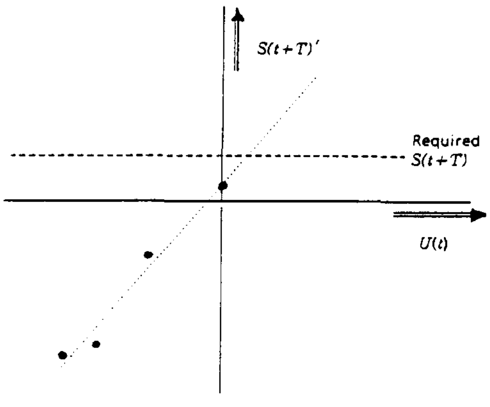

Figure 19 Typical plot of $S(t+T)'$ achieved for partial rule case

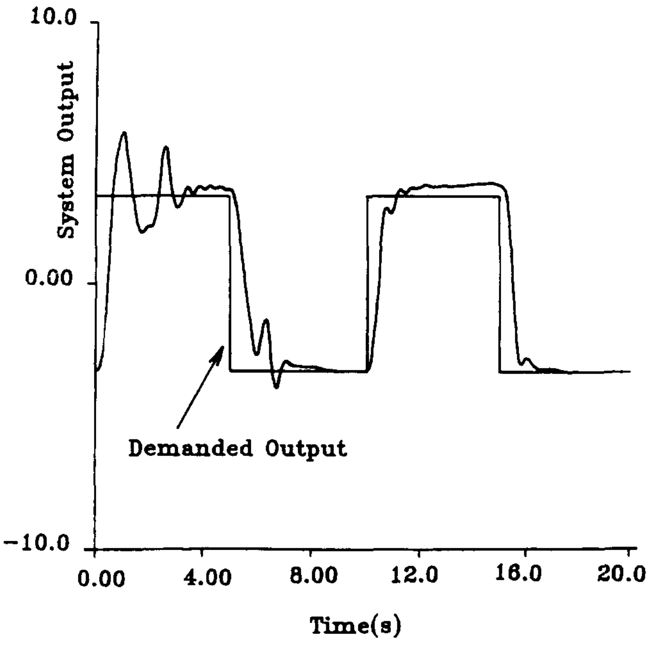

Figure 20a Learning controller response for, unknown plant; $\xi=0.2$, $\omega_n=4.4$ rad/sec

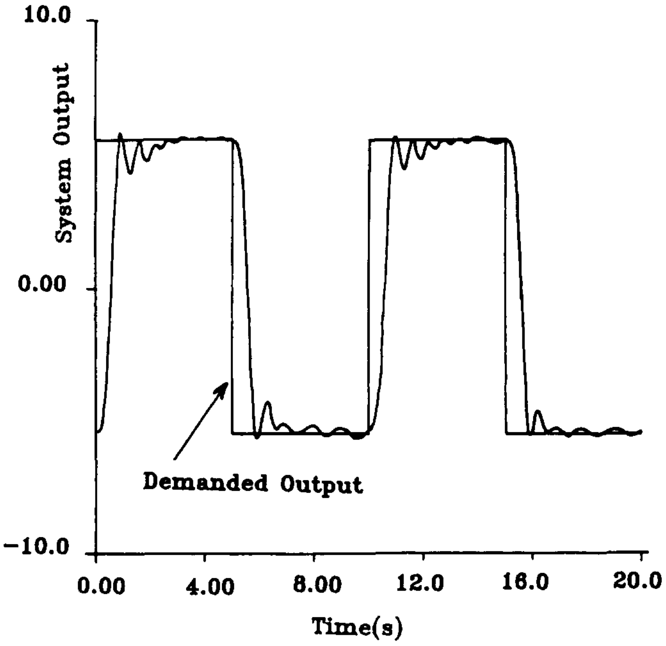

Figure 20b Learning control with sudden parametric change ($\xi=0$)

To demonstrate the above methodology consider again *Example 6* second-order unknown plant ($\xi=0.2$, $\omega_n=4.4$ rad sec^{-1}), and the performance index of *Table 8*. Although the controller output is digital in nature and instantaneous in magnitude step changes, we limit the speed at which it can change per sample to $|\Delta U(t)|=1.0$ to represent realistic or feasible actuator demands. The results of several simulation runs of simultaneous identification/control are shown in *Figure 20(a, b)* for variable square wave inputs; initially R is unknown. Fast identification and well-damped control is achieved in *Figure 20a*. To test the robustness of the method and ability to relearn, the actual plant's damping ratio was changed from 0.2 to 0.0. The control simulation is shown in *Figure 20b*. Note the relatively poor rise time in both cases is determined by the constraint $|\Delta U(t)|=1.0$. To evaluate the controller in the presence of nonlinearities, the backlash of width 1.0 (see *Figure 21*) was introduced in front of the linear plant. The simulation response is shown in *Figures 22(a, b)*. Again no prior knowledge of R was assumed, as is demonstrated in the initial transient behaviour of *Figure 22a*. Note in both cases that the learning and controlled response is amplitude-dependent owing to the presence of the nonlinearity in the control loop.

DISCUSSION

This paper has reviewed some applications of fuzzy logic to AGV motion control. Clearly, the ability to generate decisions or control rules without physical models, based entirely on experiential evidence, has substantial advantages for complex systems. Of even more significance is the ability of such systems to self-organize in response to changing system parameters. The initial computer simulations of self-adaptive or learning rule-based fuzzy-logic controllers indicate fast initial convergence of the rule base and an ability to cope with both large and unknown parametric variations and loop nonlinearities. The approach is somewhat

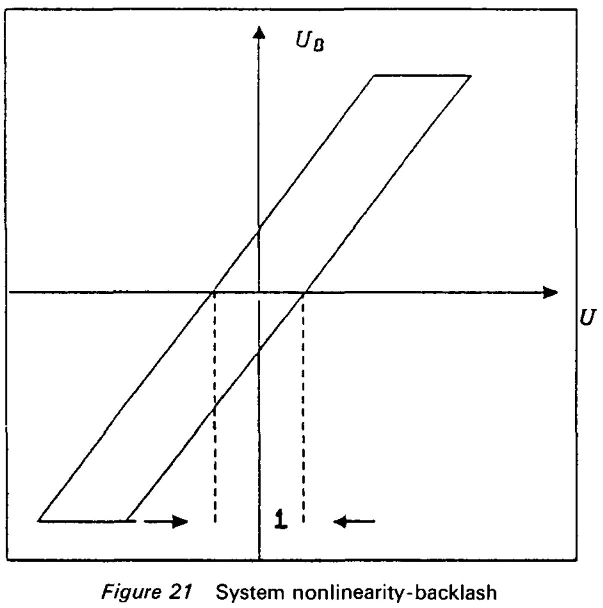

Figure 21 System nonlinearity-backlash

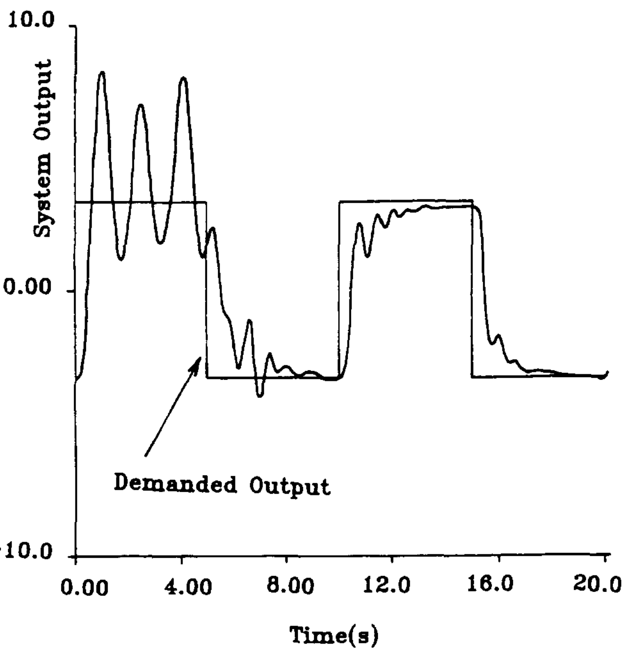

Figure 22a Learning control with nonlinearity, no initial plant knowledge

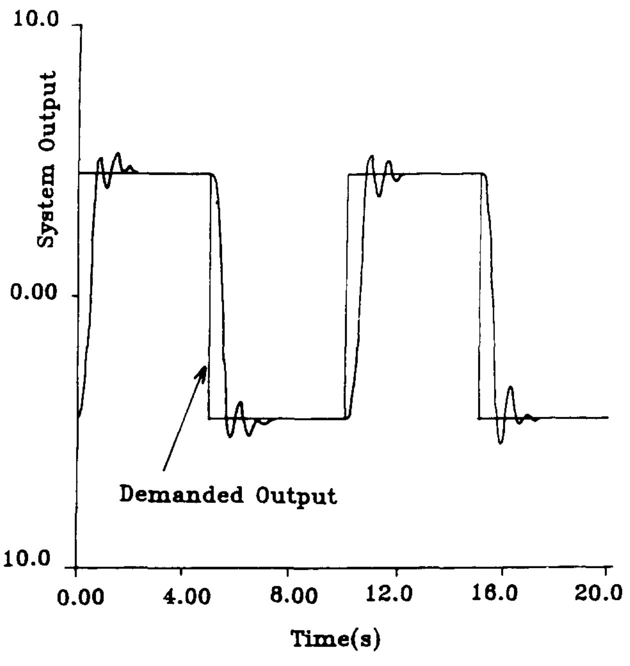

Figure 22b Learning control with some *a priori* knowledge

approximate, allowing small steady-state errors. This is due to large discretization ranges (or equivalent few linguistic quantifiers) selected in the implementation for the system variables. For the target application, Autonomous Guided Vehicles, the ability to adapt in real time to rapid system changes and system nonlinearities is considered more significant than small steady-state errors in position or velocity. However, these errors can be overcome by using a multi-level resolution control strategy; for global control use the above fast highly adaptive controller, for local control (say, zero steady-state errors) use the global control to initialize the local controller then switch over the local fine (small discretized levels) control.

The ability to continuously self-optimize the performance of the controller by adapting the production rules is a significant characteristic of the proposed fuzzy self-learning dual estimation/control algorithm. By utilizing this feature, it is envisaged that autonomous guided vehicles required to follow a path, traverse an obstacle or make turns (during docking), will be controlled whilst satisfying varying, asymmetric acceleration constraints.

ACKNOWLEDGEMENTS

The authors would like to thank the CEC, MOD, SERC and Lucas Aerospace for financial support for this work.

REFERENCES

1 Tsumara, T. Recent developments of AGV's in Japan. *Robotersysteme*, Vol. 2. Springer-Verlag, Berlin, pp. 91–97 (1986)
2 Nilsson, M. A mobile automaton: an application of AI techniques. *Proc. 1st J.C. AI, Washington, 1986*
3 Warnecke, H. J. and Linder, H. Trends in robotics in the European Community. *Proc. ICAR, Tokyo, 1985*, pp. 7–13
4 Rembold, V. and Levi, P. Sensors and control for autonomous vehicles. *Int. Conf. Intelligent Autonomous Vehicles, Amsterdam, 1986*, pp. 79–89
5 Harris, C. J. and Moore, C. G. Real time fuzzy based self-learning predictors and controllers. *12th IFAC World Congress, Tallinn, USSR, July 1990*
6 Linden, T. A., Marsh, J. P. and Dove, D. L. Architecture and early experience with planning for the ALV. *Proc. IEE Int. Conf. Robotics and Automation, San Francisco, 1986*
7 Harris, C. J. and Jackson, J. V. Techniques for intelligent multi-sensor data fusion for application to autonomously guided vehicles. *Proc. IMA Conf. Robotics, Loughborough, July 1989*. Oxford University Press, Oxford (1989)
8 Brady, M. J. (ed.). Special issue 'Sensor data fusion'. *Int. J. Robotics Research*, 7 (6) (1988)
9 Harris, C. J. *Application of Artificial Intelligence to Command and Control Systems*. Peter Peregrinus, Stevenage, UK (1988)
10 Dickmanns, E. D. and Zapp, A. Guiding land vehicles along roadways by computer vision. *Proc. AFCET, Congrès Automatique—The Tools for Tomorrow, Toulouse, 1985*, pp. 233–244
11 Harris, C. J. and Read, A. B. Knowledge based fuzzy motion control of autonomous vehicles. *1st IFAC Workshop on AI in Real Time Control, Swansea, 1988*, pp. 149–154
12 Brooks, R. A. A robust layered control system for mobile robots. *IEEE Trans. Robotics and Automation*, **RA-2** (1986)
13 Albus, J. S. *Systems description and design architecture for multiple autonomous undersea vehicles*. National Institute of Standards and Technology, NIST Report 1251 (1988)
14 Hogle, R. A. and Bonissone, P. P. A fuzzy algorithm for path selection in AGV navigation. *Proc. 23rd IEEE Decision and Control Conf., 1984*, pp. 898–900
15 Farreny, H. and Prade, H. Uncertainty handling and fuzzy logic control in navigation problems. *Int. Conf. Intelligent Autonomous Vehicles, Amsterdam, 1986*, pp. 217–225
16 Lo, C. *Navigation and control of an autonomous vehicle.*

Southampton University Aeronautics and Astronautics Department Project Report (1989)

17 Isik, C. and Meystel, A. M. Pilot level of a hierarchical controller for an unmanned mobile robot. *IEEE J. Robotics and Automation*, **4**, 241–255 (1988)

18 Mitschke, M. *Dynamik der Kraftfahrzeuge*. Springer-Verlag, Berlin (1982)

19 Zadeh, L. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Trans. Systems, Man and Cybernetics*, **SMC-1**, 28–44 (1973)

20 Kickert, W. and Mamdani, E. H. Analysis of a fuzzy logic controller. *Fuzzy Sets and Systems*, **1**, 29–44 (1978)

21 Braae, M. and Rutherford, D. A. Theoretical and linguistic aspects of the fuzzy logic controller. *Automatica*, **15**, 553–577 (1979)

22 Larkin, L. I. A fuzzy logic controller for aircraft flight control. In *Industrial Applications of Fuzzy Control* (M. Sugeno (ed.)). Elsevier Science Publishers, North-Holland (1985)

23 Sugeno, M. An introductory survey of fuzzy control. *Information Sciences*, **36**, 59–83 (1985)

24 Takagi, T. and Sugeno, M. Fuzzy identification of systems and its application to modelling and control. *IEEE Trans. Systems, Man and Cybernetics*, **SMC-15**, 116–132 (1985)

25 Pedryez, W. Applications of fuzzy relational equations for methods of reasoning in presence of fuzzy data. *Fuzzy Sets and Systems*, **16**, 163–175 (1985)

26 van der Rhee, F., van Nauta Lemke, H. R. and Dijkman, J. J. Applying fuzzy set theory to modelling processes. *10th IFAC World Congress, 1988*, Vol. 6, pp. 338–343

27 Sutton, R. and Towill, D. R. An introduction to the use of fuzzy sets in the implementation of control algorithms. *J. IERE*, **55**, 357–367 (1985)

28 Li, Y. F. and Lau, C. C. Application of fuzzy control for servo-systems. *IEEE Conf. on Robotics and Automation, 1988*, pp. 1511–1519

29 Sugeno, M. and Nishda, A. Fuzzy control of a model car. *Fuzzy Sets and Systems*, **16**, 100–113 (1985)

30 Kickert, W. and van Nauta Lemke, H. R. Application of a fuzzy controller in a warm water plant. *Automatica*, **12**, 301–318 (1976)

31 Sanchez, E. Compositions of fuzzy relations. In *Advances in Fuzzy Set Theory and Applications* (M. Gupta, K. Ragade and R. Vager (eds)). North-Holland, pp. 421–433 (1979)

32 Tong, R. M. and Efstathiou, J. Rule based decomposition of fuzzy relational models. *Proc. JACC, San Francisco, 1988*, Vol. 2

33 Scharf, E. M. and Mandic, N. J. The application of a multi-degree of freedom robot arm. In *Industrial Applications of Fuzzy Control* (M. Sugeno (ed.)). North-Holland, pp. 41–61 (1985)

34 Yamazaki, T. and Mamdani, E. H. On the performance of a rule-based self-organising controller. *IEE Conf. Applic. Adaptive and Multivariable Control, Hull, 1982*, pp. 50–55

35 Read, A. B. *Intelligent knowledge based fuzzy logic control.* Southampton University Department of Aeronautics and Astronautics Project Report (1988)