Monograph:
Linkens, D.A. and Hasnain , S.B. (1990) Self-Organising Fuzzy Logic Control and its Application to Muscle Relaxant Anaesthesia. Research Report. Acse Report 384 . Dept of Automatic Control and System Engineering. University of Sheffield

---

SELF-ORGANISING FUZZY LOGIC CONTROL AND ITS APPLICATION TO MUSCLE RELAXANT ANAESTHESIA

Linkens, D.A & Hasnain, S.B

Department of Control Engineering, University Of Sheffield

ABSTRACT

In this paper the authors describe the use of Self-Organising Fuzzy Logic Control (SOFLC) for automated drug delivery in muscle relaxant anaesthesia. The self-elicitation of a knowledge base is shown to be robust in the presence of model uncertainity, noise contamination, and parameter changes. Being computing intensive, SOFLC is considered for parallel implementation on transputers, both via use of Lisp interpreter and direct Occam coding. SOFLC in Occam code gave a fast implementation, which could be further speeded up using multiple transputers.

1. INTRODUCTION

Control theory has made remarkable progress from its tentative origins to the mathematically formalised discipline that it is today. Modern control theory has proved to be very successful in areas where the systems are well defined either deterministically or stochastically. Many systems involving man-machine interaction are characterised by a high degree of performance but they cannot be analysed to a comparable degree of accuracy. The human involved is often an adequate controller because he is able to construct in his mind a model of the process which is just accurate enough to carry out the task in hand. This model includes all the essential features of the process required by him to do the job. The control engineer does not always have the ability to extract the essential details from a process when he is replacing the human by an automatic controller. Considerable insight into such systems can be gained by modelling the method of human decision making in these situations. The human is also capable

of learning through experience which makes him less dependent on an accurate knowledge of his environment before he carries out a task. So, any attempt to automate the role of a human-being must, to some extent, model his powers of expression and reasoning as well as learning capabilities.

To support the translation of the more vague, non-numeric statements that might be made about such a control strategy a semi-quantitative calculus is required. It was against this background that Zadeh introduced and developed fuzzy set theory[Zadeh 1965] and approximate reasoning[Zadeh 1973]. The later paper introduced the concept of fuzzy linguistic variables and the fuzzy set, which seemed to provide a means of expressing linguistic rules in such a way that they can be combined into a coherent control strategy.

Since its introduction by Zadeh in 1965, fuzzy logic has been used successfully in a number of control applications. The first application of fuzzy set theory to the control of dynamic processes was reported by Mamdani and Assilian [1974]. They were concerned with the control of a small laboratory scale model of steam engine and boiler combination. The control problem was to regulate engine speed and boiler pressure. Despite the nonlinearity, noise and the strong coupling in the plant, they managed to get acceptable control, using a fuzzy logic controller. Kickert and Lemke [1976] applied fuzzy logic to design a controller for a laboratory scale warm water plant. The water tank was divided into several compartments. The aim was to control the temperature of water in one of the compartments by altering the flow rate through a heat exchanger contained in the tank. A secondary control task was to ensure fast response to step changes in outlet water temperature set point. The first experiment on an industrial plant with a fuzzy logic controller was undertaken by Rutherford and Carter [1976]. They developed a controller for the permeability at the Cleveland sinter plant, and showed that the fuzzy logic controller worked slightly better then the PI controller.

Tong applied fuzzy logic to a pressurised tank containing liquid [Tong 1976]. His problem was to regulate the total pressure and level of liquid into the tank by altering the rates of flow of the liquid into the tank and the pressurising air. Good control was achieved despite the nonlinearity and strong coupling,

however it was no better than that obtained by a controller designed using conventional techniques. Tong et al [1980] also examined the behaviour of an experimental fuzzy logic control algorithm on an Activated Sludge waste water treatment Process (ASP). He concluded that a fuzzy algorithm based on practical experiance can be made to work on this difficult process. The success obtained by Mamdani and Assilian in the control of a steam engine led them to study temperature control of a stirred vessel which formed the batch reactor process - a non-linear time varying gain and time delay process[Mamdani and Assilian 1975]. The results obtained showed that process can be controlled effectively using heuristic rules based on fuzzy control, but to achieve good control the fuzzy rules must be correctly formulated, to take account of time delays when they occur. In the same decade the techniques of fuzzy logic were applied by independent groups over a wide variety of processes: Ostergaard[1976] applied it successfully on a heat exchanger; Van Amerongen[1977] applied fuzzy sets to model the steering behaviour of ships. Larsen[1979] reported work to implement fuzzy logic control on a coal-fired wet process umax-cooler kiln. Sheridan and Skjoth [1983] attempted to replace kiln operators at the Durkee plant of the Oregon Portland cement company using fuzzy algorithms.

The major shortcoming of previous work is the slow response time of the fuzzy logic controller, due to the large calculations essential for the algorithm. This makes it unsuitable for dynamic operations. In this work an attempt is made to speed up the calculation time by splitting the fuzzy logic controller in different processes and running them in parallel on one or a combination of transputers. A novel technique to store the linguistic rules in Lisp (an AI language) and use it by the fuzzy logic controller on a transputer network is also presented.

The Self-Organising Fuzzy Logic Algorithm (SOFLC) initially presented by Procyk [1977] for a single input/output process provides an adaptive rule-learning capability to complement a fuzzy logic control strategy. Yamazaki and Mamdani [1982] examined the problems of poor settling time and occasional instability associated with SOFLC and proposed an improved version which overcomes these problems, by applying it on single input/output and multiinput multioutput processes. The main problem which restricted application of SOFLC over a wide area was the unclarity of the rule learning and storing procedure. It

was later modified by Daley [1984] and applied on a process of greater complexity and higher mathematical dimension i.e, attitude control of a space craft. Recently Larkin(1985) has applied SOFLC to aircraft control and Harris and Moore (1989) have applied it to the control of autonomous guided vehicles. Here a description is given of an attempt to control the non-linear muscle relaxant model of Pancuronium with SOFLC. The simulations were first performed on a sequential machine and then parallelism was introduced in the SOFLC to extend its implementation to a multi-processor system. The robustness of SOFLC is examined with consideration of parameter changes in the process model and inducing white noise to the process.

2. MUSCLE RELAXANT PROCESS MODELS

Muscle relaxant drugs are used by anaesthetists who, based on their own experience, determine the adequate dose in order to obtain a predefined degree of paralysis. However, anaesthetists sometimes fail to maintain a steady level of relaxation resulting often in a large consumption of drugs by the patients. A safe and fast method for designing closed-loop infusion systems is to replace the patient with a mathematical model.

The pharmacology of muscle relaxant drugs comprises two important parts, pharmacokinetics and pharmacodynamics. Pharmacokinetics concerns the absorption, distribution and excretion of drugs, whereas pharmacodynamics is a term employed in drug pharmacology to describe the relationship between drug concentration and the therapeutic effects of these drugs.

2.1 Pharmacokinetics

For the drug Pancuronium the body is considered to consist of two compartments between which reversible transfer of drugs can occur. The drug is injected into compartment 1 which represents the plasma volume and the perfused tissues such as heart, lung, lever etc., from which it is excreted into urine. The drug is exchanged between the 1st compartment and the connecting peripheral compartment. The plasma concentration of drug versus time is given by the bi-

exponential equation.

$$c(t) = A1e^{-\alpha 1t} + A2e^{-\alpha 2t} \tag{1}$$

where;

A1 and A2 are complex functions

$A1e^{-\alpha 1t}$ reflects the distribution phase

$A2e^{-\alpha 2t}$ reflects the elimination phase.

2.2 Pharmacodynamics

It is known that the interaction between the injected drug and the components of the body induce a certain therapeutic effect. The steady state plasma concentration of the drug is proportional to the amount of drug at the site of action. The plasma concentration of drug versus effect relationship can be described mathematically by a Hill equation[Whiting and Kelman 1980] of the form;

$$E=\frac{E_{\max} \cdot C^{\alpha}}{C^{\alpha} + C^{\alpha}_{50}} \tag{2}$$

where;

E represents the drug effect

$E_{\max}$ the maximum drug effect

C the drug concentration

C_{50} The drug concentration corresponding to a 50% effect.

Using previous pharmacological identification studies[Linkens et. al. 1982(a)], the drug Pancuronium can be modelled as a two-time-constant linear dynamic system with a dead-time in series with a non-linear part comprising a dead space and a limiting device or a Hill equation as illustrated in Fig 1. For the drug Atracurium a three compartment model is considered and the transfer

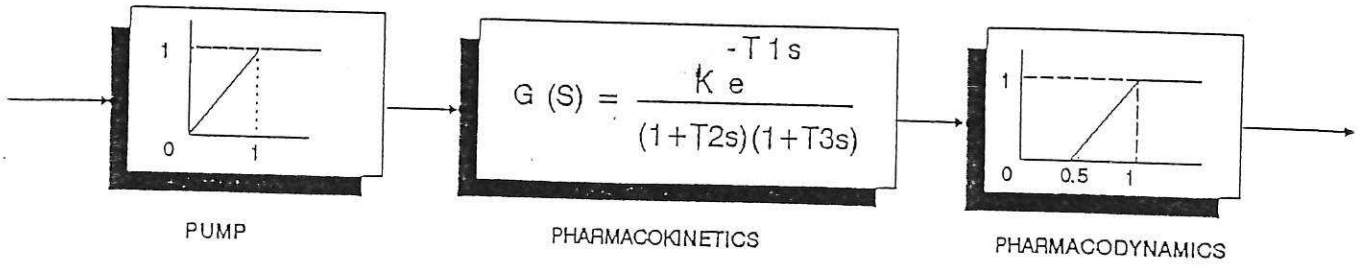

FIG 1: PANCURONIUM BROMIDE DRUG
MUSCLE RELAXANT MODEL

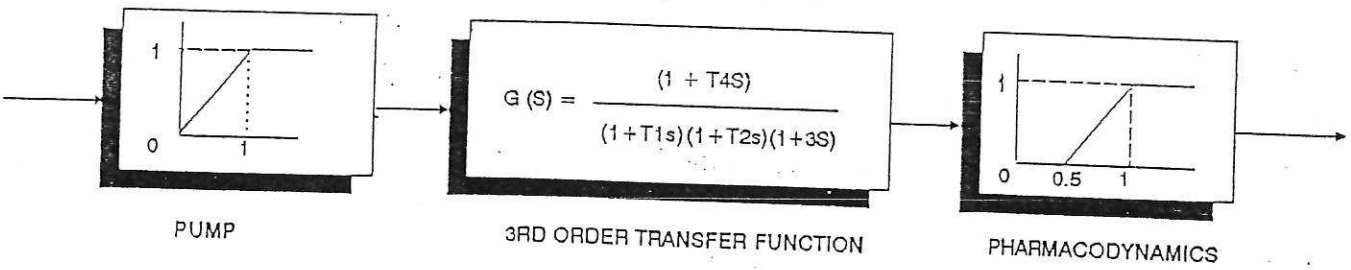

FIG 2: REPRESENTATION OF THE MUSCLE RELAXANT
SYSTEM RELATED TO ATRACURIUM DRUG

function is 3rd order as shown in Fig 2 [Linkens and Mahfouf 1989].

3. FUZZY LOGIC THEORY

In conventional set theory, there is a distinct difference between elements which belong to a set and those which do not. Thus, for example, if the space on which the sets are defined is the range of speed between 50 m.p.h and 100 m.p.h, then the non fuzzy set for the speed of the car greater than or equal to 70 m.p.h is given by Fig 3(a). The function μ is called the membership function of the set and takes values either 0 or 1. In this case μ is a step function with those speeds for which it is zero being outside the set, and those for which it is one being inside the set.

A similar fuzzy set might be "speed much greater then 70 m.p.h" and "speed about 70 m.p.h", shown respectively in Fig 3(b) and Fig 3(c). In both these cases $\mu(t)$ takes all the values on the closed interval (0,1) and indicates the degree of membership within the given range.

Fuzzy sets can be manipulated in the same way as non-fuzzy sets with operations of union, intersection and complementation being defined by simple arithmetic operations on the set of membership functions. The following example is used to define the fuzzy operations.

IF

A = { speed is much greater then 70 m.p.h)

B = { speed is about 70.m.p.h}

represented by Fig 3(b) and Fig 3(c),then

(i) UNION

The union of fuzzy sets A & B of the universe of discourse e is denoted by $A \cup B$ and the membership function is given by,

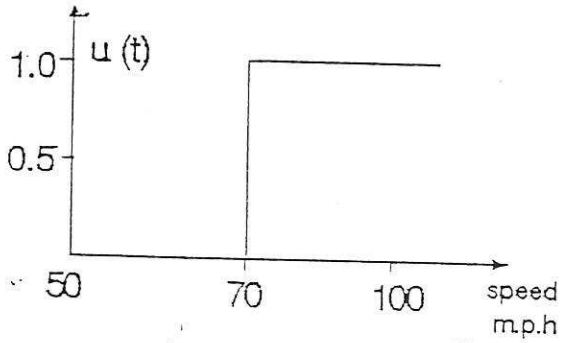

FIG 3(a): NON-FUZZY SET

(speed equal to and greater than 70 m.p.h)

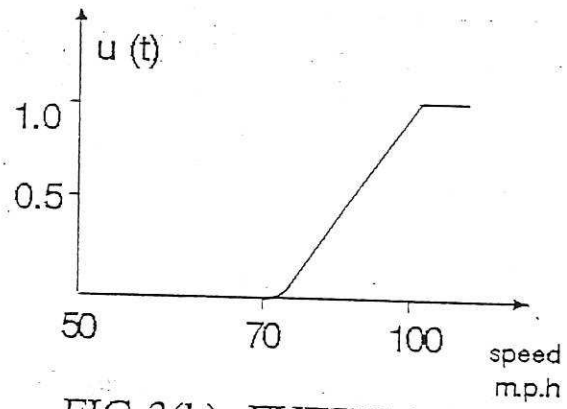

FIG 3(b): FUZZY SET

(speed much greater than 70 m.p.h)

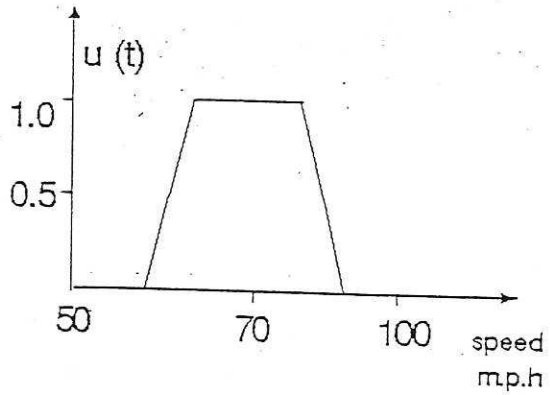

FIG 3(c): FUZZY SET

(speed about 70 m.p.h)

$$\mu_{A \cup B}(e) = \max [\, \mu_{A(e)} ; \mu_{B(e)} \,] \qquad e \in E \qquad (3)$$

This corresponds to the connective "OR" and is shown in Fig 4(a) for prior example.

(ii) INTERSECTION

The intersection of fuzzy sets A & B is denoted by $A \cap B$ with the membership function given by,

$$\mu_{A \cap B}(e) = \min [\, \mu_{A(e)} ; \mu_{B(e)} \,] \qquad e \in E \qquad (4)$$

This corresponds to the connective "AND" and is shown in Fig 4(b) for prior example.

(iii) COMPLEMENT

The complement of fuzzy set A is denoted by $\overline{A}$ with a membership function defined by,

$$\mu_{\overline{A}}(e) = 1.0 - \mu_A(e) \qquad (5)$$

This corresponds to the negation "NOT" and is shown in Fig 4(c) for prior example.

3.1 COMPOSITIONAL RULE OF INFERENCE

In general, several rules in the form of logical statements, are required to give adequate control. However it is impractical to have a rule for every possible situation and so one further step is required so that the collection of logical statements can be fully exploited. The composition rule of inference allows the rules to be used in situations which don't normally occur. This is an important feature of fuzzy controllers, since it enables, for example, an interpretation of the question " given the control shown above, what is the change in pressure of an accelerator pedal if the speed is much greater then 100.m.p.h?"

The membership value of each rule for a given controller input is calculated by fuzzy implication. Let us take the following example where the rule R_i is

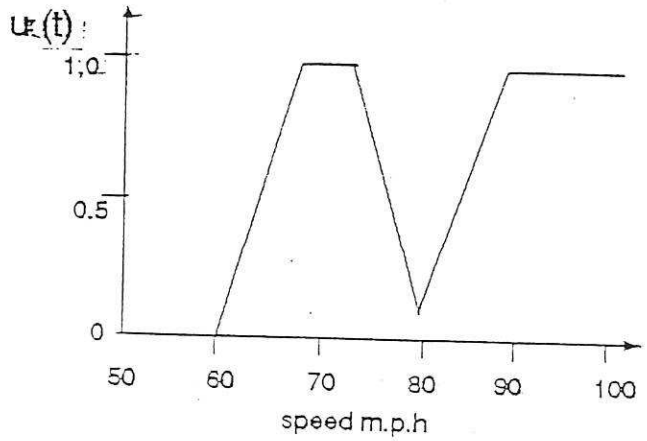

FIG 4(a): UNION OF FUZZY SETS

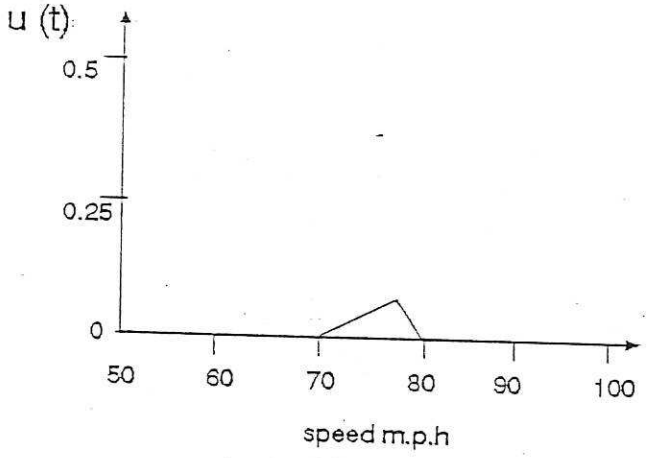

FIG 4(b): INTERSECTION OF FUZZY SETS

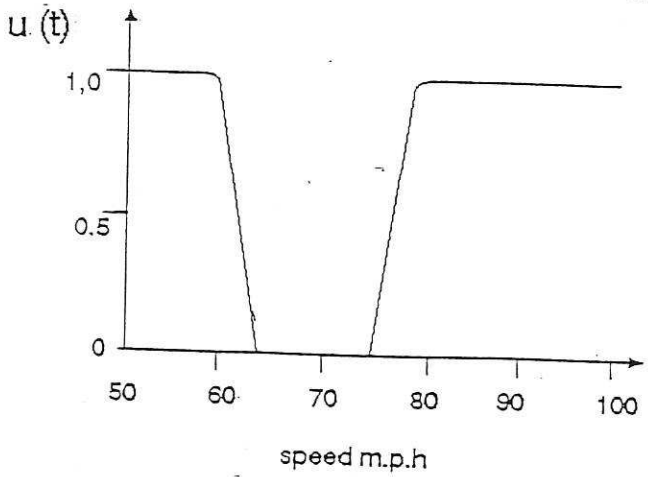

FIG 4(c): COMPLEMENT OF FUZZY SETS

expressed as,

$$R_i : If\ e\ is\ E_i\ then\ output\ is\ U_i$$

Fuzzy implication is expressed by,

$$R_i = E_i \times U_i \tag{6}$$

where

×: Fuzzy implication.

The controller output is inferred from the input and the control rule by compositional rule of inference.

$$U_i = R_i \circ E \tag{7}$$

where

E = Fuzzy set of the actual controller input

o = Composition

U_i = inferred Controller input

using the max-min rule.

$$\mu_{U_i}(u) = \max_e \min [\ \mu_E(e)\ ;\ \mu_{R_i}(e,u)\ ] \tag{8}$$

when there is more than one rule the maximum of the membership values of those rules is usually taken, which means ELSE in rules is interpreted as taking the maximum of the membership values.

$$\mu_U(u) = \sum_i \mu_{U_i}(u) = \max_i \max_e \min [\ \mu_E(e)\ ;\ \mu_{R_i}(e,u)\ ] \tag{9}$$

4 FUZZY LOGIC CONTROLLERS

4.1 SEQUENTIAL FUZZY LOGIC CONTROL

The simple fuzzy logic controller described here is designed to regulate the output of a process around a given set-point. The output at regular intervals is sampled and sent to the controller. The controller in Fig 5 shows the configuration in relation to a single-input/output process. In general there are two

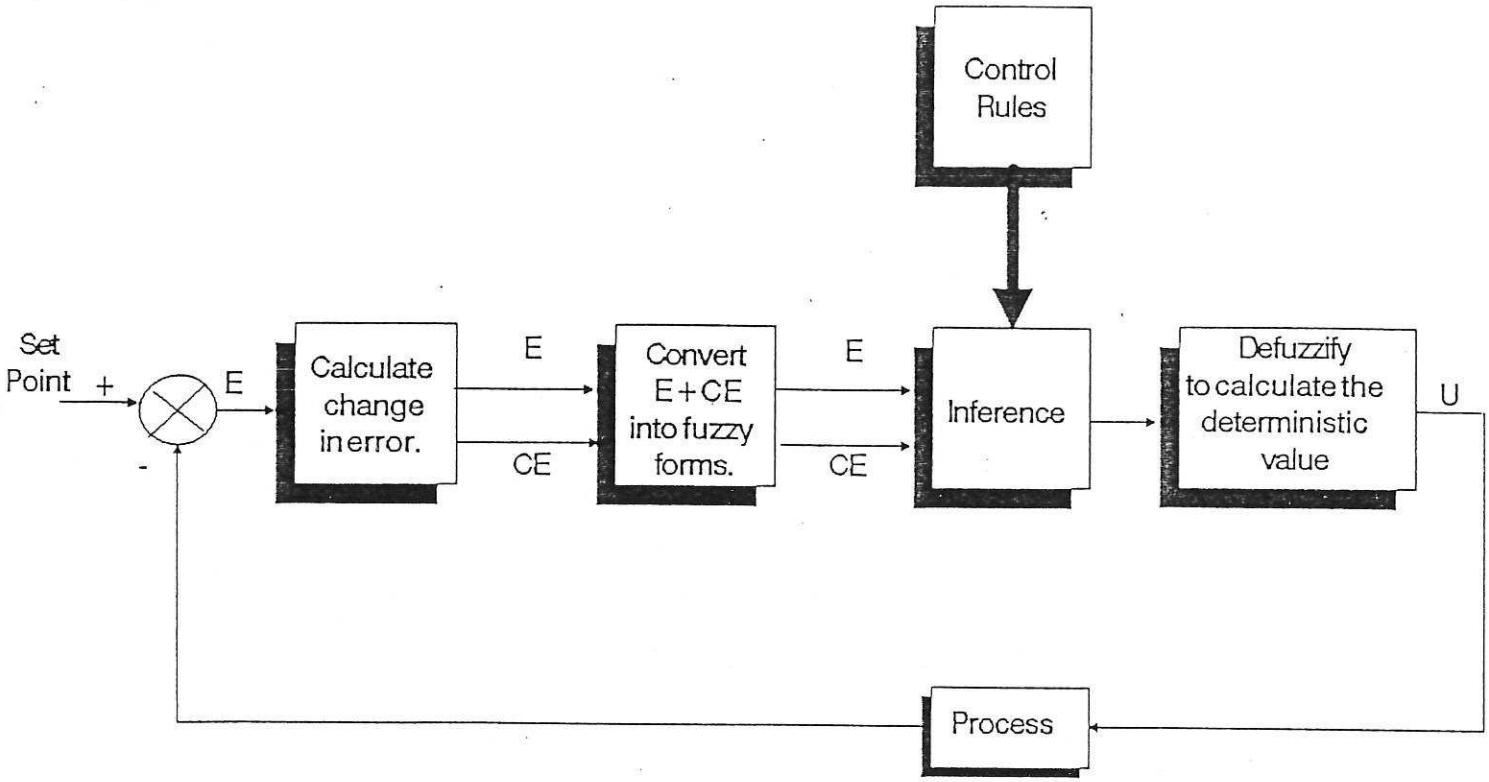

FIG 5: RULE-BASED FUZZY LOGIC CONTROLLER

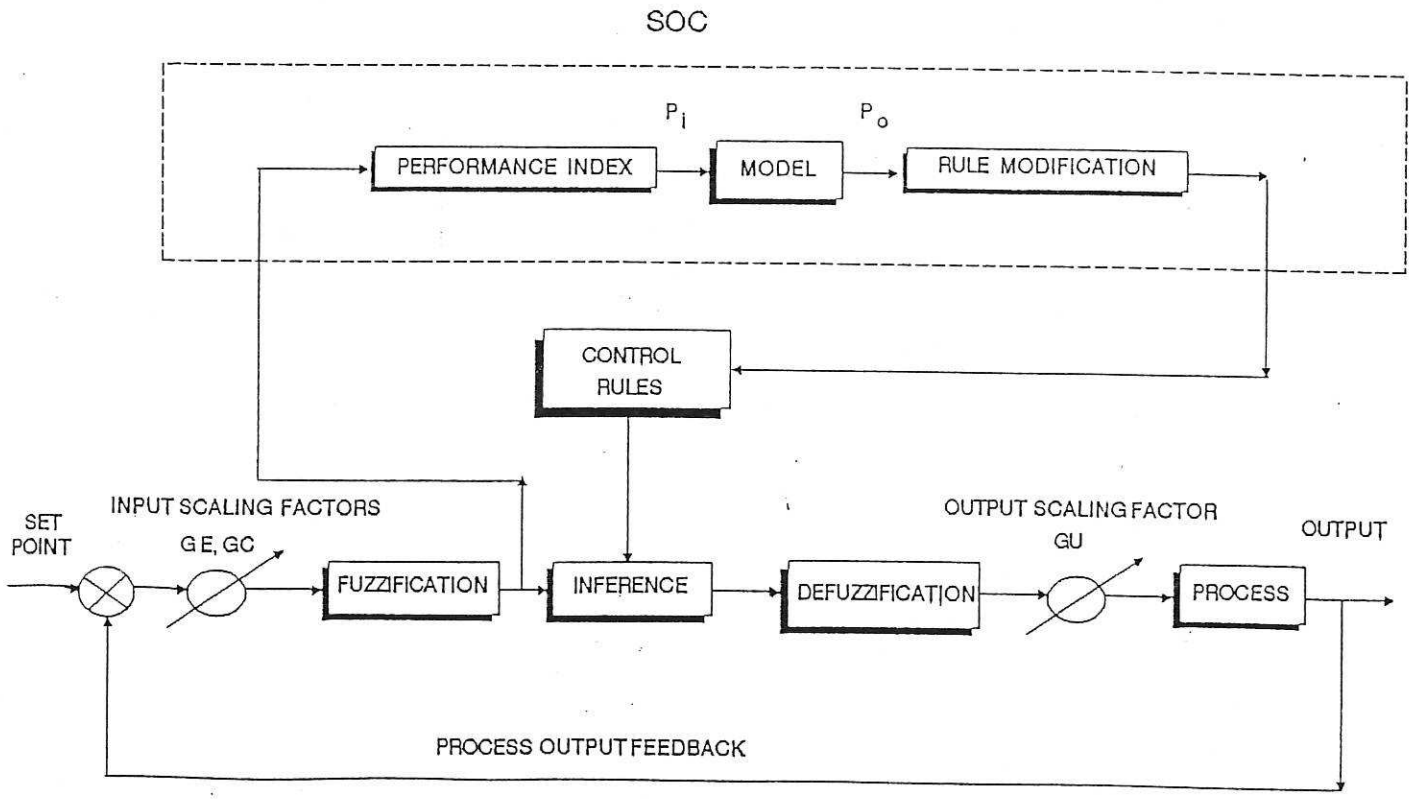

FIG 6: SELF-ORGANISING FUZZY LOGIC CONTROLLER

inputs and one output to the controller. One of the inputs is the process error E, the other input CE is the change in error, obtained by subtracting the error at the last sampling instant from the present one. Movements toward the set point are positive changes in error, and movements away are negative change in error. The control action U is the change in input to be applied to the process.

The linguistic synthesis of the fuzzy controller relates the state variables, E and CE to the action variable U, by means of the linguistic protocol. This linguistic protocol consists of a set of fuzzy rules which define individual control situations to form a fuzzy conditional sentence or algorithm. A section of such an algorithm might be,

If Error is ...

then If Change in Error is ...

then Output is...

Else

If Error is ...

then If Change in Error is ...

then Output is...

Each rule is a fuzzy conditional statement connecting the output to the input.

In the fuzzy controller the universe of discourse is discrete, thus membership vectors rather than membership functions are considered. The variables are therefore defined by Fuzzy subsets in the following manner.

$$\textit{Error} \quad E = (e,\ \mu_E(e)) \quad , \quad e \in E$$

$$\textit{Change in Error} \quad CE = (c,\ \mu_{CE}(c)) \quad , \quad c \in CE$$

$$\textit{Change in Input} \quad U = (u,\ \mu_U(u)) \quad , \quad u \in U$$

Error and its rate of change are first calculated and then converted into fuzzy variables after being scaled. Zadeh's rule of inference is used to infer the fuzzy output. The deterministic input is then calculated by defuzzifying the input

using the mean of maxima method..

The fuzzy sets are formed upon a discrete support universe of 14 elements of error; 13 elements of error rate and 15 elements of the controller output. Appropriate membership functions are assigned to each element of the support set. The resulting sets representing the linguistic terms are listed in Table 1.

Table 1: The Fuzzy set definitions

(a) ERROR

	-6	-5	-4	-3	-2	-1	-0	+0	+1	+2	+3	+4	+5	+6
PB	0	0	0	0	0	0	0	0	0	0	0.1	0.4	0.8	1.0
PM	0	0	0	0	0	0	0	0	0	0.2	0.7	1.0	0.7	0.2
PS	0	0	0	0	0	0	0	0.3	0.8	1.0	0.5	0.1	0	0
PO	0	0	0	0	0	0	0	1.0	0.6	0.1	0	0	0	0
NO	0	0	0	0	0.1	0.6	1.0	0	0	0	0	0	0	0
NS	0	0	0.1	0.5	1.0	0.8	0.3	0	0	0	0	0	0	0
NM	0.2	0.7	1.0	0.7	0.2	0	0	0	0	0	0	0	0	0
NB	1.0	0.8	0.4	0.1	0	0	0	0	0	0	0	0	0	0

(b) CHANGE IN ERROR

	-6	-5	-4	-3	-2	-1	-0	+1	+2	+3	+4	+5	+6
PB	0	0	0	0	0	0	0	0	0	0.1	0.4	0.8	1.0
PM	0	0	0	0	0	0	0	0	0.2	0.7	1.0	0.7	0.2
PS	0	0	0	0	0	0	0	0.9	1.0	0.7	0.2	0	0
ZO	0	0	0	0	0	0.5	1.0	0.5	0	0	0	0	0
NS	0	0	0.2	0.7	1.0	0.9	0	0	0	0	0	0	0
NM	0.2	0.7	1.0	0.7	0.2	0	0	0	0	0	0	0	0
NB	1.0	0.8	0.4	0.1	0	0	0	0	0	0	0	0	0

(c): OUTPUT

	-7	-6	-5	-4	-3	-2	-1	-0	+1	+2	+3	+4	+5	+6	+7
PB	0	0	0	0	0	0	0	0	0	0	0	0.1	0.4	0.8	1.0
PM	0	0	0	0	0	0	0	0	0	0.2	0.7	1.0	0.7	0.2	0
PS	0	0	0	0	0	0	0	0.4	1.0	0.8	0.4	0.1	0	0	0
Z	0	0	0	0	0	0	0.2	1.0	0.2	0	0	0	0	0	0
NS	0	0	0	0.1	0.4	0.8	1.0	0.4	0	0	0	0	0	0	0
NM	0	0.2	0.7	1.0	0.7	0.2	0	0	0	0	0	0	0	0	0
NB	1.0	0.8	0.4	0.1	0	0	0	0	0	0	0	0	0	0	0

The linguistic elements used are the same as those used in most applications and these items have the following meaning:

PB Positive Big	NO Negative Zero
PM Positive Medium	NS Negative Small
PS Positive Small	NM Negative Medium
PO Positive Zero	NB Negative Big

ZO Zero

The terms NO and PO are introduced to obtain finer control about the equilibrium state, NO being defined as values slightly below zero and PO terms slightly above zero.

4.2 PARALLEL FUZZY LOGIC CONTROL

The fuzzy control of a process is quite complex as indicated by Fig 5. Fuzzification, defuzzification and scanning of the control rules to find the appropriate one makes the system very slow. Parallel processing presents a possible solution. In order to experiment with a transputer-based system, an experimental process(muscle-relaxant drug model), the fuzzy control logic and the control rules were all coded in Occam and run on a single transputer. The results were encouraging, with considerable gain in speed. Trials showed that since the control rules are linguistic and Occam is not suitable for handling linguistic problems some alternative language should be used.

There are many problem-solving systems that are based on matching simple rules to given problems. They are often called rule-based expert systems, and sometimes they are called situation-action systems or production-rule systems. Lisp is a powerful language to implement rule-based systems with great potential to handle linguistic rules and commands. The available transputer Lisp system was originally written in 'C' and capable of running on a single transputer. In order to implement it as an interactive system which can advise the control algorithm on other transputers it has to be linked with Occam.

Inside the TDS(Transputer Development System) 'C' is considered as an alien language. The facilities provided for linking it with an Occam program results in slow communication to the knowledge base files required by the Lisp program. The 3L Parallel 'C' complier provides the necessary software needed to link 'C' programs with Occam outside the TDS. The Occam program is

compiled using a stand alone compiler and then linked to the 'C' compiled programs. A configuration file which explains the physical description of the links and processor descriptions connects the 'C' and Occam programs and produces an object file which can be executed on a transputer network.

It is worth mentioning at this point that the major limitation of the fuzzy algorithm was the off-line calculation of the rule map. Any change in the control rules were to be observed by the operator first, and then followed by the change in the fuzzy linguistic algorithm and calculation of the rule map, which is subsequently applied to the process. The whole procedure is repeatedly performed until an acceptable level of control is achieved. This leads us to the concept of a self-organising fuzzy logic controller.

4.3 SELF-ORGANISING FUZZY LOGIC CONTROL

The Self-Organising Fuzzy Logic Controller (SOFLC) is an extension of the simple fuzzy logic controller as shown in Fig 6. The SOFLC must be able to assess its performance in order to improve its control strategy. The basic functions of SOFLC can be summarised as follows

1. To issue appropriate control action whilst evaluating the performance.
2. To modify the control action based upon this evaluation.

The performance of the controller in relation to the process output is measured by the deviation of the control response from the desired one. The response of an output can be monitored by its error and change in error. The performance index takes the form of a decision maker which assigns a credit or reward based on the knowledge of error and change in error. This credit value is obtained by the fuzzy algorithm which linguistically defines the desired performance. The performance index matrix used for muscle relaxant control using Pancuronium is shown in Table 2.

Table 2: SOC PERFORMANCE INDEX MATRIX

ERROR	CHANGE IN ERROR						
	NB	NM	NS	Z	PS	PM	PB
NB	NB	NB	NB	NM	NM	NS	ZO
NM	NB	NB	NM	NM	NS	ZO	NS
NS	NB	NB	NS	NS	ZO	PS	PM
NO	NB	NM	NS	ZO	ZO	PM	PB
PO	NB	NM	ZO	ZO	PS	PM	PB
PS	NM	NS	ZO	PS	PS	PB	PB
PM	NS	ZO	PS	PM	PM	PB	PB
PB	ZO	PS	PM	PM	PB	PB	PB

If the input states are assumed to be fuzzy singletons i.e, fuzzy sets with membership function equal to zero every where except at the measured value where it equals 1, the linguistic performance rules can be transformed into a lookup table (Table 3) using the standard techniques of fuzzy calculus.

Table 3: PERFORMANCE INDEX LOOK UP TABLE

ERROR	CHANGE IN ERROR												
	-6	-5	-4	-3	-2	-1	-0	+1	+2	+3	+4	+5	+6
-6	-7.0	-6.5	-7.0	-5.0	-7.0	-7.0	-4.0	-4.0	-4.0	-2.5	-1.0	0.0	0.0
-5	-6.5	-6.5	-6.5	-5.0	-6.5	-6.5	-4.0	-4.0	-4.0	-2.5	-1.5	0.0	0.0
-4	-7.0	-6.5	-7.0	-5.0	-4.0	-4.0	-4.0	-1.0	-1.0	-2.5	0.0	1.5	1.0
-3	-6.50	-6.5	-4.0	-4.0	-4.0	-3.0	-2.5	-2.5	0.0	0.0	1.5	2.5	2.5
-2	-7.0	-6.5	-7.0	-4.0	-1.0	-1.0	-1.0	0.0	0.0	0.0	1.0	4.0	4.0
-1	-6.5	-6.5	-6.5	-4.0	-1.5	-1.5	-1.5	0.0	0.0	2.5	1.5	4.0	4.0
-0	-7.0	-6.5	-4.0	-3.5	-1.0	-1.0	0.0	0.0	0.0	2.5	4.0	6.5	7.0
+0	-7.0	-6.5	-4.0	-2.5	0.0	0.0	0.0	1.0	1.0	3.5	4.0	6.5	7.0
+1	-4.0	-4.0	-1.5	-2.5	0.0	0.0	1.5	1.5	1.5	4.0	6.5	6.5	6.5
+2	-4.0	-4.0	-1.0	0.0	0.0	0.0	1.0	1.0	1.0	4.0	7.0	6.5	7.0
+3	-2.5	-2.5	-1.5	0.0	0.0	2.5	2.5	3.0	4.0	4.0	4.0	6.5	6.5
+4	-1.0	-1.5	0.0	2.5	1.0	1.0	4.0	4.0	4.0	5.0	7.0	6.5	7.0
+5	0.0	0.0	1.5	2.5	4.0	4.0	4.0	6.5	6.5	5.0	6.5	6.5	6.5
+6	0.0	0.0	1.0	2.5	4.0	4.0	4.0	7.0	7.0	5.0	7.0	6.5	7.0

4.3.1 RULE MODIFICATION PROCEDURE

The rule modification procedure can be described as follows [Daley and Gill, 1986].

1. Let the input to the controller be $e(kT)$ and $ce(kT)$ and output of the controller be $u(kT)$

2. At some instant kT the process input reward is $p_i(kT)$

3. If the process input rT samples earlier contributed most to the present state,

4. Then the controller output due to the measurements $e(kT - rT)$ and $ce(kT - rT)$ should have been $u(kT - rT) + p_i(rT)$.

The rule modification can be approached by forming fuzzy sets around the above single values:

$$E(kT - rT) = F_z(e(kT - rT)) \tag{10}$$

$$CE(kT - rT) = F_z(ce(kT - rT)) \tag{11}$$

$$U(kT - rT) = F_z(u(kT - rT)) \tag{12}$$

where F_z is the fuzzification procedure.

The controller is modified by replacing the rules that most contributed to the process input rT samples earlier with the rule:

$$E\ (kT - rT) \longrightarrow CE(kT - rT) \longrightarrow U\ (kT - rT) \tag{13}$$

If the controller is empty then an initial rule must be created, further rules will be generated when required by the modification procedure. The initial rule will be formed by the fuzzification of the initial condition e^i,ce^i and p^i, and controller outputs u^i. This is done by forming a fuzzy kernel to provide a spread of values about the single support element, thus creating the following fuzzy sets:

$$E = (e - x), \mu_k(e - x) \tag{14}$$

$$CE = (ce - x), \mu_k(ce - x) \tag{15}$$

$$U = (u + P_i - x), \mu_k(u + P_i - x) \tag{16}$$

where

$$\begin{matrix} \mu_k(e - x) \\ \mu_k(ce - x) \\ \mu_k(u + P_i - x) \end{matrix} = \begin{cases} 1.0, & x = \pm 0 \\ 0.7, & x = \pm 1 \\ 0.2, & x = \pm 2 \\ 0.0, & 3 \leq x \leq -3 \end{cases}$$

The single elements and the input and output of the controller are stored for use by the modification procedure. To reduce computation time, Daley suggested storage of each element of the rules in 3×5 sub-locations of the rule matrix containing,

(i) A single value giving the number of non-zero membership values in the set

(ii) The desired support value of the set

(iii) The membership value of the set.

Thus if the present instant is kT and the modification is made to the controller output rT samples earlier, the rule to be included results from the fuzzification of

$$e(kT - rT), ce(kT - rT), u(kT - rT) + p_i(kT)$$

The output U is obtained from the rules by using a modification procedure as follows:

A control rule written as:

IF "E" THEN IF "CE" THEN "U"

is a fuzzy relation

R = E x CE x U

then,

$$\mu(e,ce,u) = \min \{ \mu_E(e), \mu_{CE}(ce), \mu_U(u) \} \tag{17}$$

If the measured fuzzy sets at some instant are $\hat{E}$ and $\hat{CE}$ then an implied output set $\hat{U}$ can be obtained by the above rule using the process of fuzzy composition.

$$\hat{U} = \{ \hat{E} \text{ o } [\hat{CE} \text{ o } (E \times CE \times U)] \} \tag{18}$$

which has a membership function given by

$$\mu_{\hat{U}}(u) = \max_e \min \{[\max_{ce} \min \{\mu_R(e,ce,u), \mu_{\hat{CE}}(ce)\}], \mu_{\hat{A}}(a)\} \tag{19}$$

For several control rules the output set U^o is characterised by the membership function;

$$\mu_{U^o}(u) = \max_{rules} \{\mu_{\hat{U}}(u)\} \tag{20}$$

and the deterministic control action is obtained from U^o by the 'mean of maxima' procedure.

5. SIMULATION STUDIES

(i) SIMPLE FUZZY LOGIC CONTROL

The fuzzy linguistic algorithm, for the muscle relaxant process of Pancuronium is written by studying the general behaviour of the process under different conditions, to provide fast convergence and high steady state accuracy(Table 4).

Table 4:SYSTEM STATE ACTION DIAGRAM

ERROR	CHANGE IN ERROR						
	NB	NM	NS	Z	PS	PM	PB
NB							
NM							
NS			PS				
NO			PS	PS	Z		
PO			PS	PS	PS		
PS			PB	PS			
PM			PB	PM			
PB			PB	PS			PB

The rules are written in the form of a matrix and interpreted as:

If "Error is NO" and "CE is NS" then controller output is "PS"

The non-linear muscle relaxant process model for Pancuronium was used to test the fuzzy logic controller. Both the process and controller were first implemented on a Perkin Elmer mini computer.

The theory of fuzzy logic calls for a large amount of computation i.e, multiplication of A x (B x C xD) matrices, where A is the number of rules given in the fuzzy linguistic algorithm (state action diagram Table 4). For the Pancuronium process 14 rules were selected initially. B, C and D are matrices representing the fuzzy sets error, change in error and controller output (Table 1). This induced a calculation of 14 x (14 x 13 x 15) matrices, to calculate the rule map. Two Fortran programs were written, one to calculate the rule map and the second to implement the fuzzy logic controller using the rule map. The process was simulated using PSI (an interactive simulation package [1980]) and connected to the Fortran program using a special communication routine written for

PSI called PSICON.MES[Linkens et. al. 1982(b)].

The parameters selected for the muscle relaxant transfer function

$$G(s) = \frac{Ke^{-T_1 s}}{(1 + T_2 s)(1 + T_3 s)} \tag{21}$$

were as follows.

$$T_1 = 1 \text{ min}; T_2 = 2 \text{ min}; T_3 = 20 \text{ min}$$

A sampling interval of 1 minute was used with a total experiment length of 200 minutes for each simulation. The initial fuzzy linguistic rules were written based on the required response of the process with step input. In order to get acceptable results these rules were modified by close observation during the subsequent runs. The final rule matrix gave an acceptable response for a constant set point.

The parallel version of Fuzzy Logic as described in section 4.2, was implemented on an array of transputers (Fig 7). The Lisp code containing the rules(knowledge) was loaded on the root transputer, and the process which was the muscle relaxant model(in this case) along with the fuzzy control algorithms was loaded on the slave transputers. Whenever a decision point is reached by any slave transputer it can gain advice from the Lisp system by communicating over the channels. An example of the Lisp-based fuzzy control applied to the muscle relaxant model is shown in Fig 8. The output response to a step command shows an adequate rise-time, but with some steady state offset which could be detrimental in certain cases.

5.2 SELF-ORGANISING FUZZY LOGIC CONTROL

The SOFLC algorithm was used next to control the muscle relaxant model for the drug Pancuronium. The simulations were first performed on a Sun Unix-based coding in Fortran, system and then repeated on a transputer system with a parallel version of the SOFLC, to compare the timing details.

Results are presented for time delay of T_1=1min, and time constants of T_2=1min and T_3=10min in the equation of the Pancuronium anaesthetic model (21).

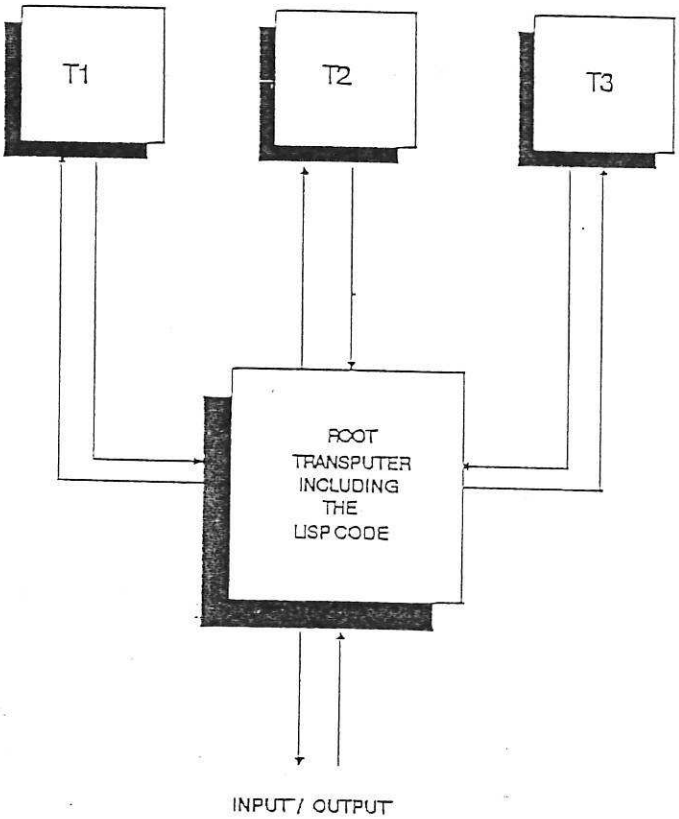

FIG 7: TRANSPUTER ARRANGEMENT FOR FUZZY CONTROL

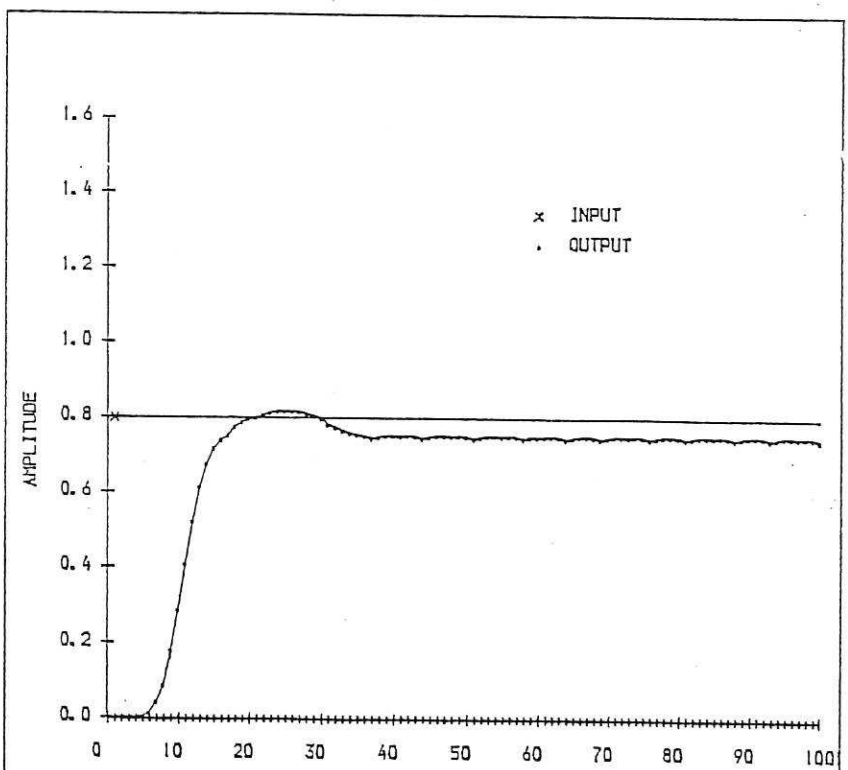

FIG 8: FUZZY CONTROL WITH LISP ON TRANSPUTER FOR PANCURONIUM DRUG, TIME CONSTANTS OF 2,20

The scaling factors were determined by trial and observation starting from an initial scaling factors, selected by the procedure described by Procyk [1977]. The simulation started with a completely empty controller and subsequently the entire control protocol was learned. Constant and varied set points were used (Fig 9, Fig 10)and convergence was indicated when the number of rule changes fell to zero.

The parallel implementation of fuzzy logic has been demonstrated in section 4.2. Here the SOFLC and the Pancuronium process were coded in Occam. A sample run with parameters of T_1 =1,T_2 =1 and T_3= 10 in equation 21 of the Pancuronium process was first performed on a T414 (without floating point hardware) transputer and then repeated on the T800 (with floating point hardware) transputer. The result of this simulation is presented in Fig 11. The timing details in Table 5 compares the execution time for the same controlled process on different machines, showing that the parallel implementation of SOFLC is ten times faster then on a sequential machine.

Table 5 :TIMING DETAILS	
Processor	**Fuzzy Logic S.O.C**
SUN Unix System	**51.00 sec**
T414 Transputer	**11.455 sec**
T800 Transputer	**5.119 sec**

5.3 ROBUSTNESS STUDIES

Critics of fuzzy logic control have argued that it should not be considered as an alternative algorithm to conventional techniques but only adopted when all else has failed. But application studies presented in sections 5.1 and 5.2 have shown that it is robust to process parameter changes. This is a desirable characteristic since in most control applications the accuracy of the plant model is not guaranteed, and plant operating characteristics may change randomly, making it essential to employ a robust controller.

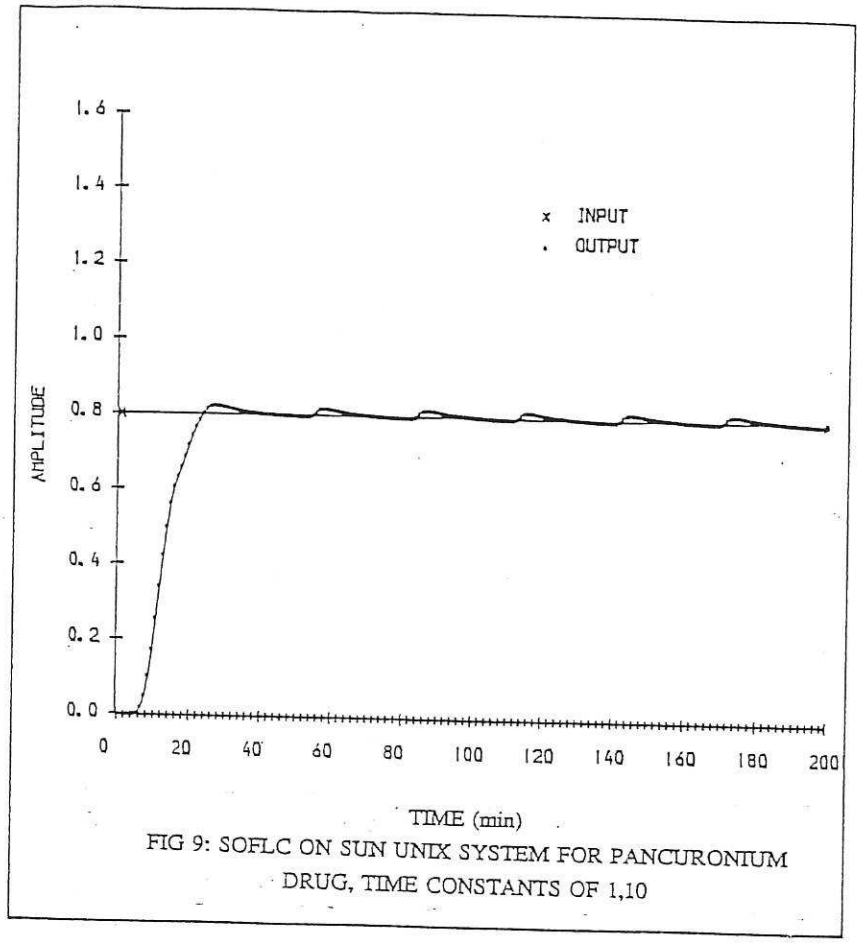

FIG 9: SOFLC ON SUN UNIX SYSTEM FOR PANCURONIUM DRUG, TIME CONSTANTS OF 1,10

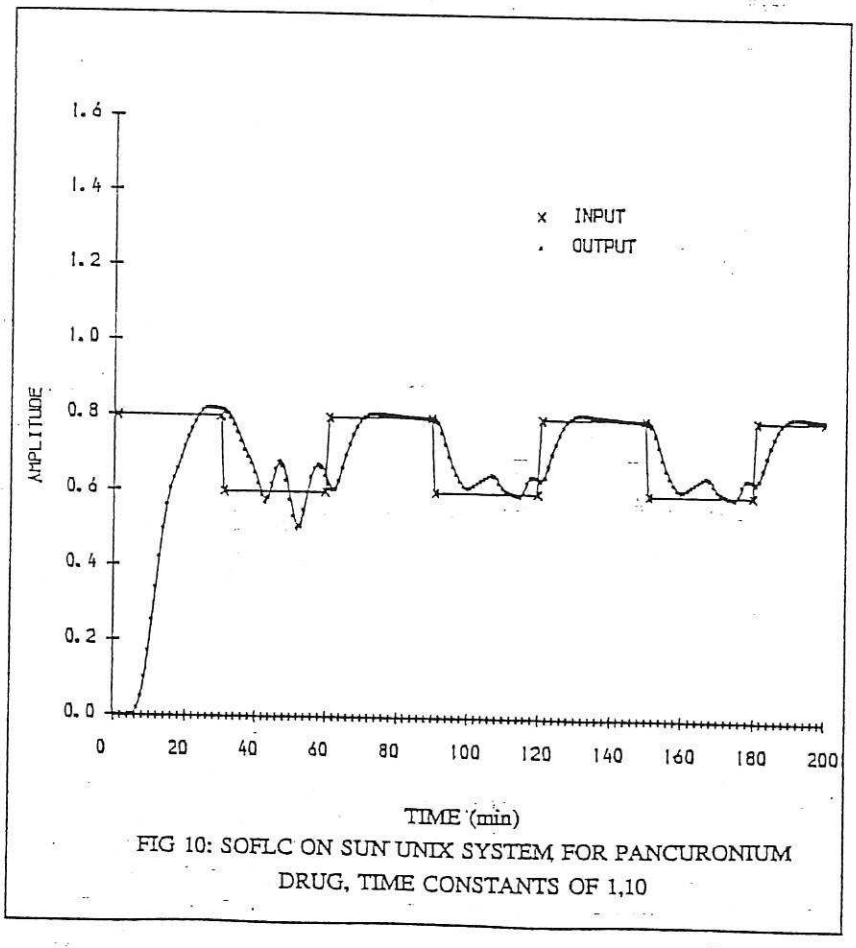

FIG 10: SOFLC ON SUN UNIX SYSTEM FOR PANCURONIUM DRUG, TIME CONSTANTS OF 1,10

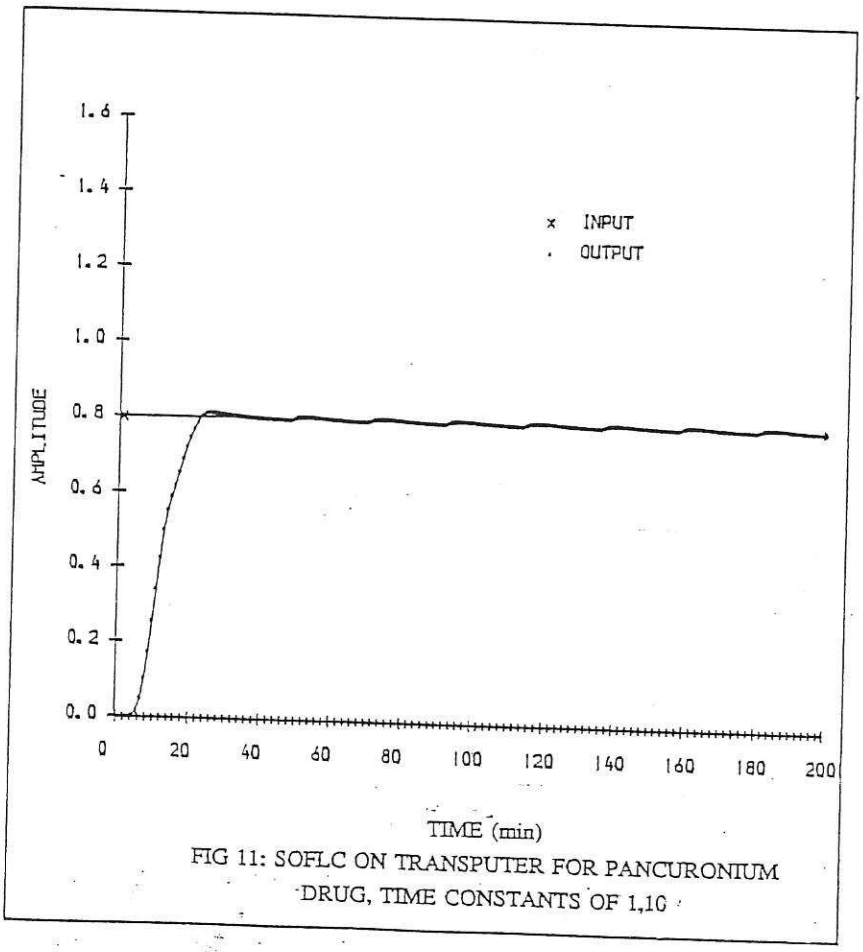

FIG 11: SOFLC ON TRANSPUTER FOR PANCURONIUM DRUG, TIME CONSTANTS OF 1,10

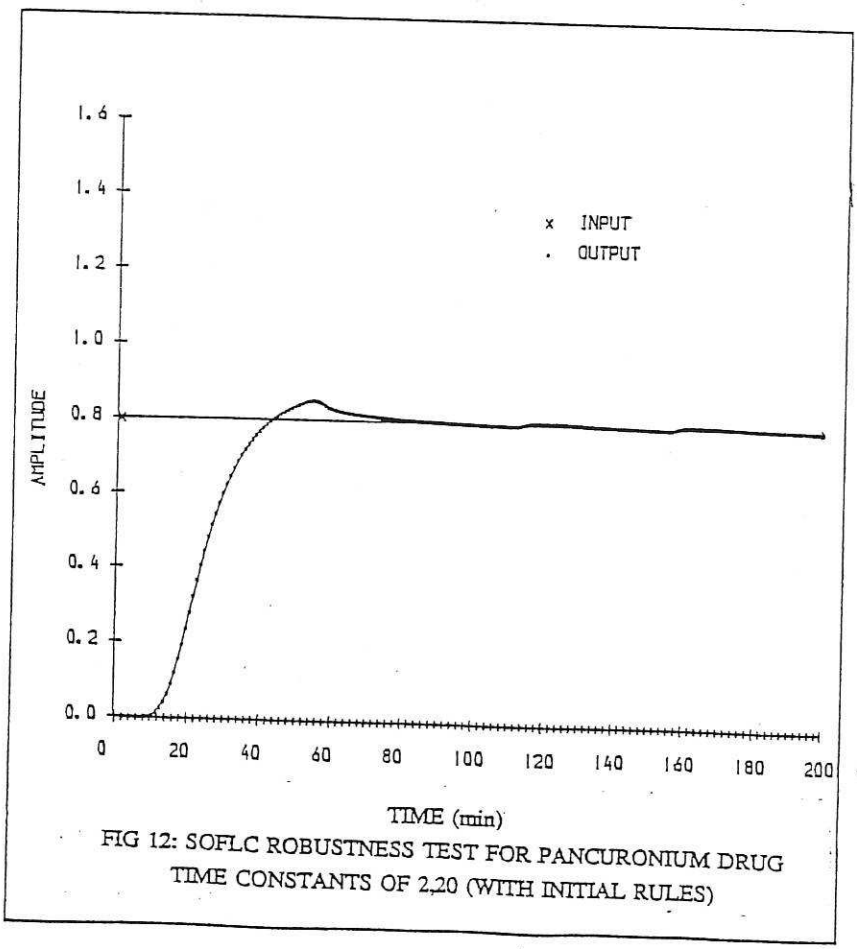

FIG 12: SOFLC ROBUSTNESS TEST FOR PANCURONIUM DRUG TIME CONSTANTS OF 2,20 (WITH INITIAL RULES)

In the simulations performed in section 5.2 SOFLC was able to control the Pancuronium process but with a change in the scaling factors as the parameter of the controlled process changed. In order to put SOFLC to more stringent tests, two experiments were performed, the details of which are as follows.

EXPERIMENT 1: Starting with an empty controller SOFLC was trained on the following parameters for equation (21) representing the Pancuronium process:

T_1= 1 min

T_2 = 1 min

T_3 = 10 min

The scaling factors used were,

GE = 10

GC = 10

GU = 0.0365

The rules generated were stored and used as initial rules with the same scaling factors for a new combination of parameters in the controlled process, i.e,

T_1= 1 min

T_2 = 2 min

T_3 = 20 min

The results of this experiment, for step change in set point is shown in Fig 12. These results show better convergence than in the simulations performed earlier without a training session.

EXPERIMENT 2: In the second experiment a different model for the anaesthetic drug Atracurium was used. First, the SOFLC was applied to the Atracurium process, starting with no rules, so that the control rules were learned. The scaling factors used to control the process were as follows:

GE = 12

GC = 12

GU = .14

The resulting response for a step and change in set points is shown in Fig 13. The next step is similar to experiment 1. The rules were learned starting from an

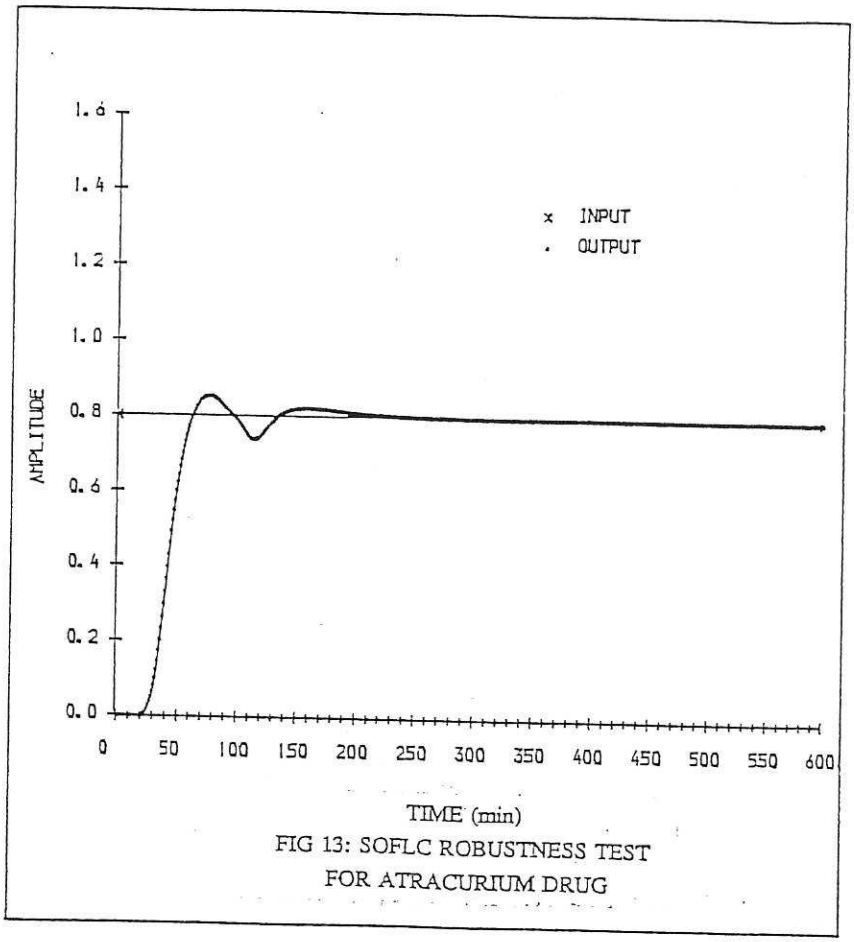

FIG 13: SOFLC ROBUSTNESS TEST FOR ATRACURIUM DRUG

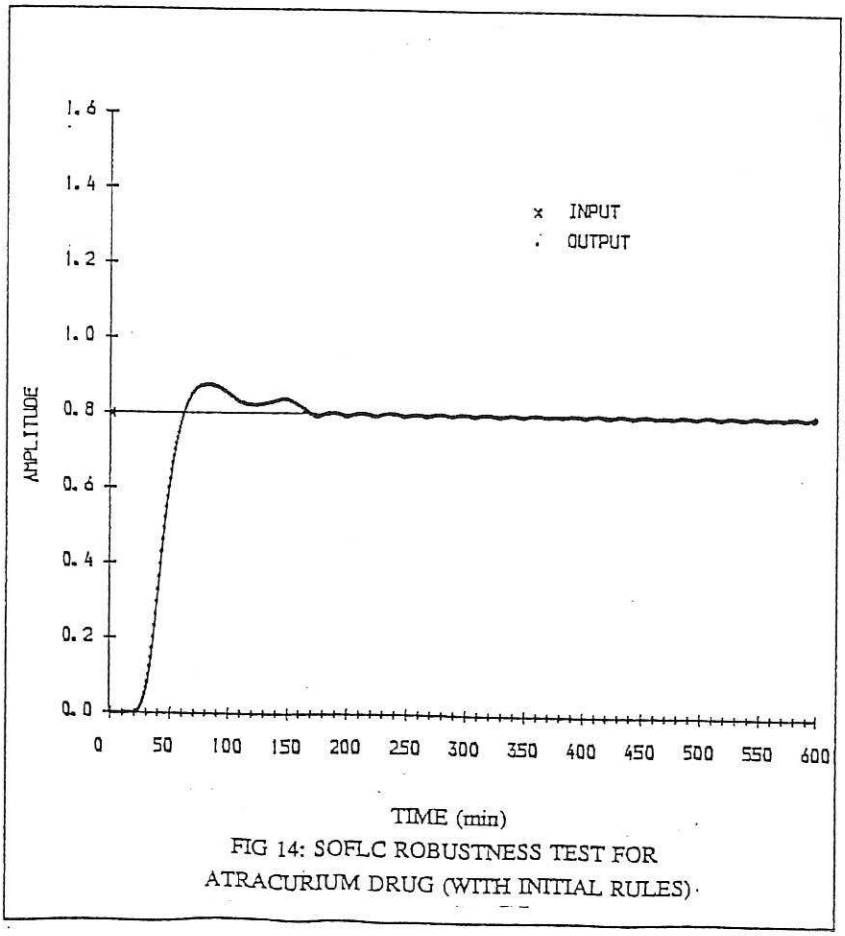

FIG 14: SOFLC ROBUSTNESS TEST FOR ATRACURIUM DRUG (WITH INITIAL RULES)

empty controller for the time constants of $T_2 = 1$ and $T_3 = 10$ of the Pancuronium process for a step input and these rules were applied as initial rules to control the Atracurium process. The result is shown in Fig 14, which is an acceptable output considering the fact that the process has been significantly changed and the rules of a different model were used as starting rules.

To study the behaviour of SOFLC under noise conditions, white noise was added to the anaesthetic drug process. Fig 15 and Fig 16 for Pancuronium and Atracurium respectively illustrate the capability of SOFLC to handle noise without significant deterioration in the output response. The final rules obtained for the Atracurium process are provided in Table 6. It should be noted that the number of rules increased, with an increase in noise variance, for example changing from 6 to 13 when a variance of 10% was used. SOFLC can give acceptable control if the variance is less than 20% of the set point value.

Table 6: FINAL RULES (FOR ADDED NOISE)

Sample No: 600

ERROR	CHANGE IN ERROR												
	-6	-5	-4	-3	-2	-1	0	+1	+2	+3	+4	+5	+6
-6													
-5													
-4													
-3													
-2						2.0							
-1							3.0	-1.0					
-0						7.0	7.0						
+0							5.0	5.0					
+1							7.0	7.0					
+2													
+3							7.0						
+4													
+5													
+6							7.0	7.0					7.0

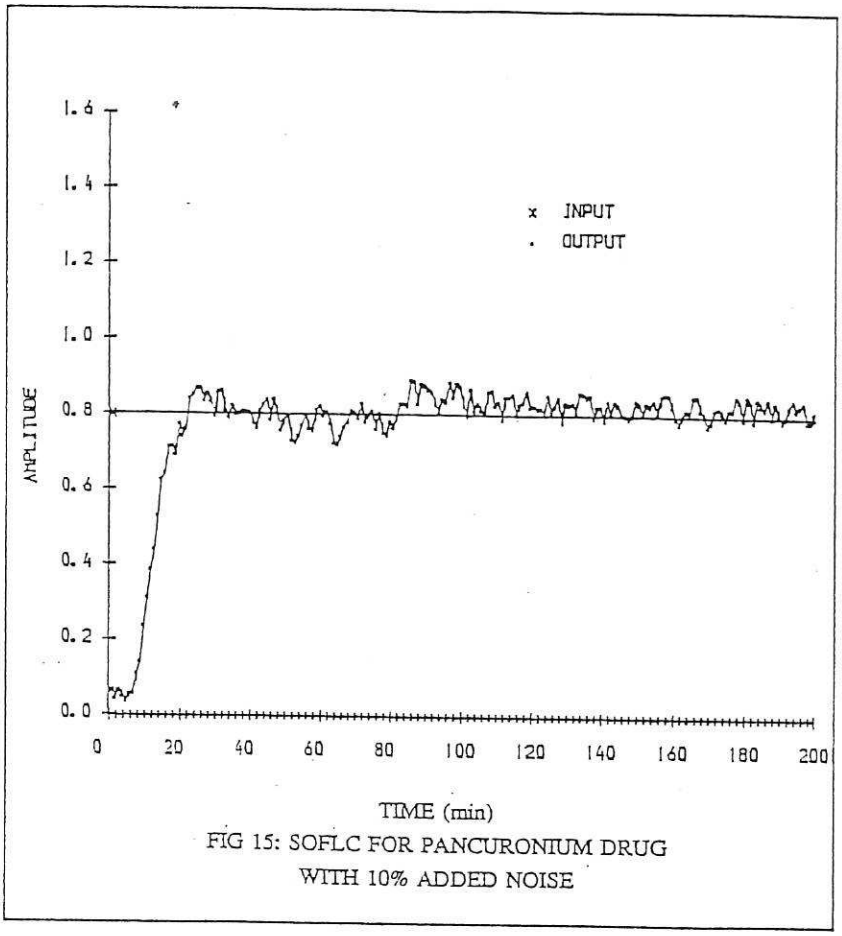

FIG 15: SOFLC FOR PANCURONIUM DRUG
WITH 10% ADDED NOISE

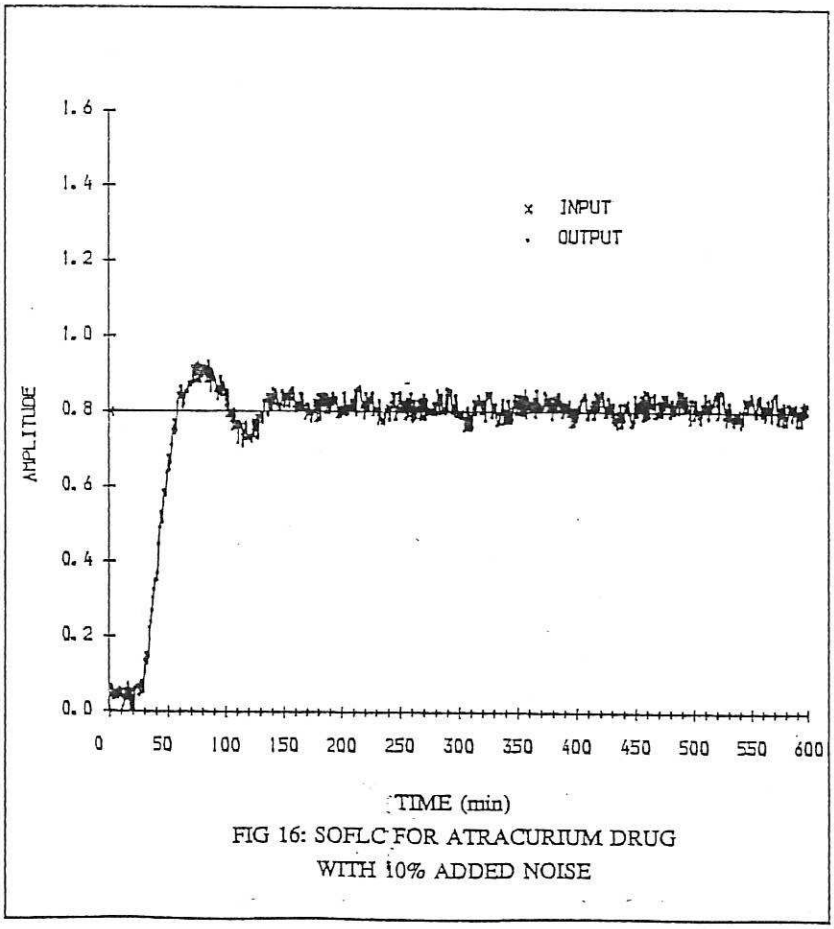

FIG 16: SOFLC FOR ATRACURIUM DRUG
WITH 10% ADDED NOISE

6. CONCLUSIONS

The application of fuzzy logic to the control of industrial processes has had considerable exposure and success, as described briefly in the introduction. In the area of biomedicine, however, it is only recently that it has been applied. This is surprising, since living systems are often hard to model mathematically and exhibit extensive non-linear behaviour for which fuzzy logic could appear to be a promising candidate for control purposes. Thus Linkens and Mahfouf [1988] described the application of a simple fuzzy logic controller to the case of automated drug infusion for muscle relaxant anaesthesia. Extensive simulation studies using a non-linear model illustrated good transient response performance, but also highlighted two problems. These were the difficulty in eliciting a suitable set of rules, and the sensitivity of the system to the fuzzy logic scale factors. Since that time, other applications in biomedicine have been reported, mainly in the area of blood pressure management in intensive care units. Thus Ying et al [1988], used a fuzzy logic 'inference shell' to produce a controller for artrial blood pressure, and Vishnoi and Gingnich [1987] report an application for gaseous anaesthetic agent delivery. The problem of knowledge elicitation motivated the work of this present paper into Self-Organising Fuzzy Logic Control (SOFLC) for the same application of muscle relaxant administration. It was found that SOFLC with a simply chosen performance index table could give good control with the nominal model for Pancuronium commencing with a zero rule based-set. This represents a particularly challenging set of conditions , and the resulting fast transient response was impressive. A further experiment was performed in which a rule-base found for Pancuronium was applied as the initial conditions for another drug, Atracurium, which has a different dynamic model structure and a different range of time constants. Good performance was also obtained under these conditions. Tests were carried out to determine the effect of noise contamination on the performance of SOFLC, and the results were good, upto 20% added noise variance. Another trial involved changing parameters of model during a run, to investigate the adaptive ability of the algorithm, with successful results again. These experiments indicate the robustness of SOFLC under a wide range of conditions.

SOFLC is a comparatively computing intensive algorithm, and this paper has described initial attempts to employ parallel computing via transputers, for its application. A simple fuzzy logic controller was coded in Occam and performed satisfactorily with the non-linear drug model. Recognising the descirability of using an A.I style language for a fuzzy rule-based system, the system was mapped into a Lisp environment. Unfortunately, the Lisp available at the time was an interpreter coded in 'C', and did not provide a fast enough system, although the principle has been verified. Thus, the full SOFLC algorithm has been coded in Occam , and provides a fast execution time even for a single transputer.

With the impending availability of languages such as Prolog for multitransputer system [Kascuk and Futo, 1989], future implementations of SOFLC could be targeted using either A.I languages or Parallel 'C' which is currently available. Further work is planned for multivariable control of anaesthesia, whereby interactive effects between different drug administrations for uncouscioosness as well as muscle relaxation will be considered. Fuzzy logic shows promise for such applications, having been validated in industrial systems such as cement kiln control where models are almost impossible to ascertain. Similar remarks apply to multivariable control of blood pressure, which has been studied via numerical algorithms of self-tuning control [Linkens and Mansoor, 1989]. This will enable a comparative study to be made between self-adaptive methods based on techniques (i,e SOFLC) and those based on quantitative approaches (such as qualitative pole-placement self-tuning). These comparisons should prove useful in advocating styles of control for processes with imprecisely-known models, such being common in both industrial situations and the life sciences.

References

DALEY, S, (1984). ''Analysis of fuzzy logic controller,'' *Ph.D Thesis, Leeds University.*

DALEY, S AND GILL, K.F, (1986). ''A design study of a self-organising fuzzy logic controller,'' *Proc Inst Mech Engrs*, vol. 200, pp. 59-69.

HARRIS, C.J AND MOORE, C.G, (1989). ''Intelligent identification and control for autonomous guided vehicles using adaptive fuzzy based algorithms,'' *Department of Aeronautics & Astronautics, Southampton University.*

KICKERT, W.J.M AND LEMKE, VAN NAUTA H.R, (1976). ''Application of fuzzy logic controller in a warm water plant,'' *Automatica*, vol. 12, pp. 301-308.

KASUK, P AND FUTO, I, (1989). ''Multi-transputer implementation of CS-Prolog,'' in *Parallel Processing and Artificial Intelligence*, ed. M. Reeve and S. Zenith, pp. 131-147, John Wiley and Sons.

LARSEN, P.M, (1979). ''Industrial applications of fuzzy logic control,'' *Int. J. Man-Machine Stud*, vol. 12, pp. 67-70.

LARKIN, L.I, (1985). ''A fuzzy controller for aircraft flight control,'' in *Industrial Applications of Fuzzy Control*, ed. Sugeno, M, Elsevier Science Publisher, North Holland.

LINKENS, D.A, ASBURY, A.J, RIMMER, S.J, AND MENAD, M, (1982 (a)). ''Identification and control of muscle-relaxant anaesthesia,'' *IEE Proceedings, D*, vol. 129, pp. 251-261.

LINKENS, D.A, MENAD, M, MORT, N, GRAY, L.S, AND BENNETT, S, (1982(b)). ''PSICON: A simulation package for the design of digitally controlled system,'' *Trans Inst M.E*, vol. 4, no. 3, pp. 153-160.

LINKENS, D.A AND MAHFOUF, M, (1988). ''Fuzzy logic knowledge based control for muscle relaxant anaesthesia,'' *IFAC Modelling and Control in Medicine, Vience, Italy*, pp. 185-190.

LINKENS, D.A AND MAHFOUF, M, (1989). ''Generalised predictive control of muscle relaxant anaesthesia,'' *IFAC workshop on "Decision Support for Patient Management", City University, London*, pp. 251-261.