REAL-TIME FUZZY PID-CONTROLLER FOR MOTOR SPEED REGULATION

S. SUJITJORN [1], D. PUANGDOWNREONG [2], Y. PREMPRANEERAT [3]

[1] School of Electrical Engineering, Suranaree University of Technology, Nakhon Ratchasima, Thailand.
[2] Department of Electrical Engineering, South-East Asia University, Bangkok, Thailand.
[3] Department of Control Engineering, King Mongkut Institute of Technology-Ladkrabang, Thailand.

ABSTRACT

A supervisory control loop can assist a PID controller to better regulate the speed of a dc servo motor. The supervisory mode works on the concept of input adjustment. The adjustment mechanism is automated using fuzzy rule-base approach. Design of fuzzy rules is heuristic and rather an art than a science. The overall system can be regarded as a fuzzy augmented PID-control of a dc servo motor. This article gives details of real-time implementation of the fuzzy control using a low-cost 8-bit processor. The processor accomplishes its task every cycle within less than 1 milli-seconds. Even though lack of rigorous analysis, our work contributes to the promising future of fuzzy control.

KEY WORDS

motor control, fuzzy PID, supervisory control, microprocessor applications

INTRODUCTION

Today industries have been widely using dc servo motors as actuators. To drive the motors successfully requires suitable control mechanisms. PID-controller is one of the useful tools to provide automatic control of the motor. Conventional design of a PID-controller is well known and can be found in many control textbooks and handbooks.

Intelligent mechanisms may be incorporated into the PID control loop to make it capable of adapting to environmental changes and to cope with disturbances. Such a system is commonly referred to as an intelligent control system. Several intelligent techniques are available for example neural network, fuzzy logic, rule-based control, neuro-fuzzy, etc. Choosing an appropriate approach is usually application dependent. Fuzzy logic [1] is one of the widely applied control technology due to its easy-to-use, although lack of proper design in rigorous sense. In Japan, as a good example, fuzzy control has become more and more favorite to industrial use [2]. Extensive tutorials on fuzzy control can be found in [3]. Recent uses of fuzzy control are such as control of industrial robots [4], and mold level control [5].

Real-time applications to control of available AI techniques are of our prime interest. In the context of real-time control, the time interval available for a controller (usually computer) to execute its tasks may be as long as several minutes to as short as hundreds of micro-seconds. Controller implementation becomes more stringent when this time interval is short. Control of a servo motor digitally falls into this category that often requires a sampling interval in the order of milli-seconds. Our work on real-time fuzzy control is aimed for a dc servo motor system described in the next section. A conventional PID-control loop augmented with a fuzzy supervisory loop makes the system become more adaptive to environmental changes and disturbances. Our practical results confirm this claim. This article also gives discussion of the design and implementation of the fuzzy control using an economic 8-bit processor.

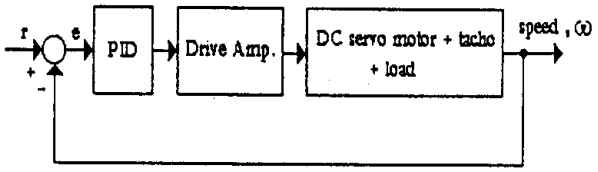

Figure 1 Block diagram represents basic control system.

SERVO MOTOR CONTROL LOOP

The block diagram shown in figure 1 represents our closed-loop speed control of a dc servo motor. This closed-loop system is used as basic control structure for our work. The motor and drive amplifier are U178T and San Drive PDT-203-30, respectively, available from Sanyo Denki™. This motor has a tachometer built-in. Considering a step-response test, the transfer

function G(s) = 1/(1+0.089s) is a good approximation for the motor dynamics. The time-constant of this motor is 89 milli-seconds.

Another separate block is a PID-controller. We utilize a digital PID that is described by

$$u(k) = u_i(k-1) + u_p(k) + u_i(k) + u_d(k) \qquad (1)$$

where

$u_p(k) = K_p e(k)$,

$u_i(k) = [K_i h/2][e(k) + e(k-1)]$,

$u_d(k) = [K_d/h][e(k) - e(k-1)]$, and

$e(k) = r(k) - c(k)$.

h is the sampling interval in seconds. At this stage, we design the controller conventionally and obtain K_p=5, K_i=125, and K_d=0.004. Our design objective is to achieve a smooth and rapid response without overshoot. The desired steady-state speed is 200 rpm. Figure 2 depicts the measured speed response to step input of the closed-loop system. The response exhibits no overshoot, time constant of 14.58 milli-seconds, and reaches the steady-state speed of 200 rpm as required (tacho gives 0.6 volts). The sampling rate of 1 kHz is adequate for the control of the U178T motor with drive that has its time-constant of 89 milli-seconds. The PID-controller may practically encounter saturation in its output u(k) because of integral windup and sudden change in the input to the D-element. This situation is handled by a simple rule "if $\{u(k) \geq u_{sat}\}$ the $\{u(k) = u_{sat}\}$" inserted in our control software, where u_{sat} is the saturation limit for the control signal u(k).

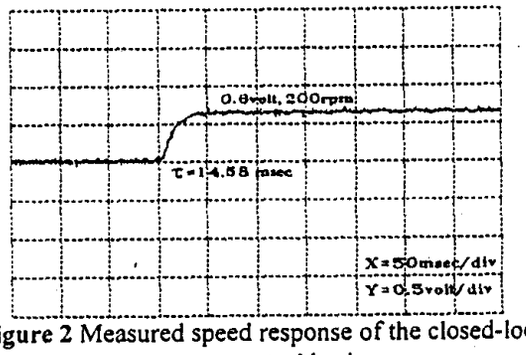

Figure 2 Measured speed response of the closed-loop motor control basic.

As far as all the elements in the control loop function well and the system is not disturbed, we can still expect the response depicted in figure 2. In actual working environment, deviation in response may occur within a reasonable range due to component aging, environmental change, model uncertainty, and other sources of disturbance. These matters are regarded in general as perturbation to the system. Under such circumstance, the PID-controller alone may not be able to recover the system performance. Thus, a supervisory control loop may be added onto the system to assist performance recovery.

SUPERVISORY CONTROL LOOP

Using a supervisory control loop is helpful when the basic control loop is hardwired or hardly accessible. Several techniques are available for design both conventional and unconventional. We have chosen fuzzy logic approach because of its simplicity in design, realization, and effectiveness. It is also challenging to implement a real-time fuzzy controller using an economic 8-bit processor to accomplish its control tasks within 1 milli-seconds sampling interval.

The supervisory control loop works under the concept of performance alteration via input adjustment. Block diagram of figure 3 illustrates the concept.

Referring to figure 3, F_p, F_i, and F_d are supervisory gain elements that perform an adaptive like functions. Their values can be varied to adjust the level of the input r(k) from zero to maximum value. Hence, each control signal element is modified and expressed by

$$u_p(k) = K_p[F_p(k)r(k)-c(k)] , \qquad (2.a)$$

$$u_i(k) = (K_i h/2)[F_i(k)r(k)+F_i(k-1)r(k-1)-c(k)-c(k-1)] , \qquad (2.b)$$

and

$$u_d(k) = (K_d/h)[F_d(k)r(k)-F_d(k-1)r(k-1)-c(k)+c(k-1)] \qquad (2.c) .$$

Experiments were conducted to investigate the effectiveness of the supervisory gain elements. Some software modifications were made to the basic control

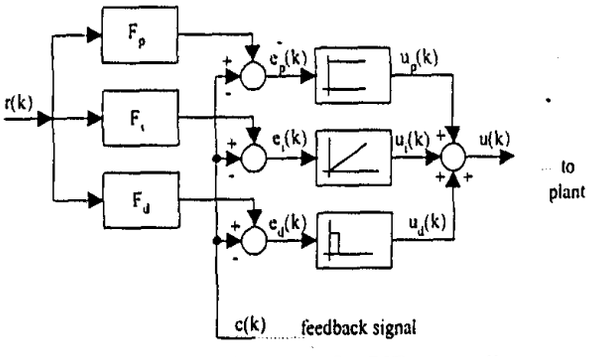

Figure 3 Input adjustment for PID-controller.

loop such that these gain elements could be set manually. Step response tests were performed on the basis of each gain element being varied from 0 to 200% independently. When one of the gain elements was varied, the rest was kept constant at 1 or 100%. It was found that (i) F_p was the most effective element to perform steady-state response recovery, (ii) F_i was effective but its function was a little slower than F_p's , and (iii) F_d was not so effective due to the nature of the step input and a rapid decay of the effect of derivative element. The results of our experimental studies are useful to the design but limited to step excitation.

Our control design based on fuzzy logic technique makes the input adjustment mechanism functioned automatically. Fuzzy rules are derived heuristically

using the knowledge gained from the experiments. Details for the controller design and implementation are given in the next section.

FUZZY CONTROLLER FOR SPEED REGULATION

The experimental studies previously described lead to construction of fuzzy rules and membership functions. The design technique used in this work is a direct fuzzy controller that contains a few rules. Each rule is fired one at a time. The membership functions used have triangular form that gives unambiguous fuzzification and defuzzification. Figure 4 shows the general structure of a fuzzy controller. The measured speed compared to the reference speed produces speed error which is crisp. This error is passed through a fuzzification process giving a fuzzified value of the speed error. The fuzzy controller consisting of fuzzy rules receives the fuzzy error as its input and infers an output as fuzzy control. This fuzzy control is in turn defuzzified into crisp form of control signal fed to actuator.

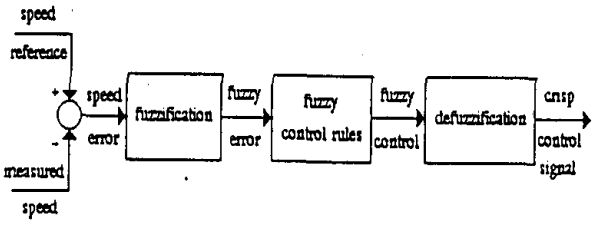

Figure 4 Structure of a fuzzy controller.

To design a fuzzy controller, the relations between the input and the output variables must be known. The relationship is expressed in the form of rule "if (condition) then (consequence)". The rule is commonly called fuzzy rule. The input variable presents in the "condition", and the output variable in the "consequence" of the fuzzy rule. The variables are linguistic or fuzzy and characterized by membership functions [1,3]. Each variable has its own fuzzy regions in which membership functions overlap each other. The more dense in the area of membership functions, the better in control resolution. The resolution or control accuracy must be considered against the number of rules and the speed of execution. Generally, more rules tend to require longer execution time. Our fuzzy controller employs the α-cut principle [1] where α = 0.5, single rule firing, and the maximum height technique for fuzzification and defuzzification.

The design of fuzzy controller starts with the first step of rule generation. The rules are heuristically generated on the basis of the results of our experimental study. The study reveals the effects of the input gain elements. The linguistic variable present in "condition" is speed error. The other present in "consequence" is input increment or decrement. By defining the speed error as the difference between the required speed and the measured speed, the speed error can be characterized by "large negative (LN)" , "medium negative (MN)" , "small negative (SN)" , "about zero (ZO)" , "small positive (SP)" , "medium positive (MP)" , and "large positive (LP)". The amount of input increment or decrement is characterized by ZO, SP, MP, and LP. Since F_p is the most effective, it is used to perform input adjustment. This implies $F_i=F_d=1$. Our fuzzy controller contains seven rules as follows :

if (speed error is LN) then (decrease input by LP)
if (speed error is MN) then (decrease input by MP)
if (speed error is SN) then (decrease input by SP)
if (speed error is ZO) then (no input adjustment)
if (speed error is SP) then (increase input by SP)
if (speed error is MP) then (increase input by MP)
if (speed error is LP) then (increase input by LP).

In this regard, it is intended to show that the controller is effective with only a small number of rules. More rules may be added with the gain element F_i brought into action for better results and refinement. In case the input is not step excitation, F_d can be effective and brought into action also.

Figure 5 illustrates the triangular membership functions that characterize speed error. The symmetrical triangular membership functions are simple and easy to use. They give unambiguous fuzzification and defuzzification results. They are suitable for an initial design phase and prototype test. For any refinement needed, a fuzzy controller may be redesigned to have optimum fuzzy rules and membership functions using some available methods such as [6,7]. Those methods generally utilize extensive simulations and experiments made to the plant to be controlled. The obtained data is used for analysis and automatic synthesis of rules an membership functions.

Referring to figure 5, the "0" point on the axis of universe of discourse or the U-axis (i.e. speed error) stands for the required speed of 200 rpm. As mentioned previously, the tacho gives 0.6 volts to represent 200 rpm. The membership functions as being symmetrical give "0.5" cross-over points. Corresponding to these

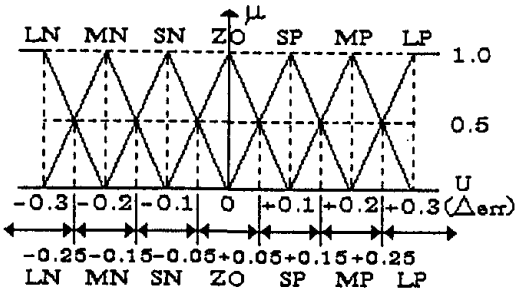

Figure 5 Linguistic characterization for speed error.

points the axis is subdivided into 7 regions. The numerical values such as -0.3, -0.25, etc. indicate speed error in terms of volts. Starting from the leftmost of the U-axis, each region is classified as follows : $\Delta err < -0.25$

for LN, -0.25≤ Δerr <-0.15 for MN, -0.15≤ Δerr <-0.05 for SN, -0.05≤ Δerr <0.05 for ZO, 0.05≤ Δerr <0.15 for SP, 0.15≤ Δerr <0.25 for MP, and Δerr ≥0.25 for LP. As an example, let the crisp value of speed error be -0.02, i.e. about zero (ZO), the rule "if (speed error is ZO) then (no input adjustment)" would be fired.

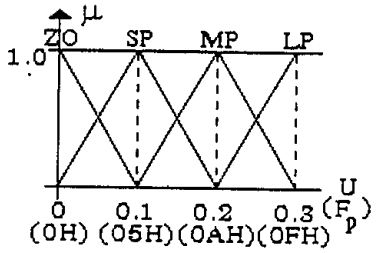

Figure 6 Linguistic characterization for input adjustment.

Figure 6 illustrates the membership functions that characterize input adjustment. The numerical values 0, 0.1, 0.2, and 0.3 on the U-axis stand for 0, 10, 20, and 30 % of the input to be decreased or increased, respectivelly. The 0, 05, 0A, and 0F are their hex equivalent. As an example, let the crisp value of speed error be 0.17, i.e. medium positive (MP), the rule "if (speed error is MP) then (increase input by MP)" would be fired. In effect, the input r(k) would be increased by 20%.

CONTROLLER IMPLEMENTATION

Hardware : The fuzzy controller uses an 8-bit Z180 single board with 10 MHz clock as its processing unit. The board has memory capacity of 128 Kbytes. One Z80CTC is available on board. The CTC is programmed to generate an interrupt signal at 1 milli-seconds' period. Additional signal conversion circuits are built using two ADC0800s and three MC1400s as major components. The resolution for signal conversion is 8-bit while arithmetic operations inside the CPU have 16-bit resolution.

Software : Two major components constitute the control software. The first one performs input adjustment. The second one does PID-control mechanism. The software is implemented in assembly with 16-bit resolution for arithmetic operations.

The input adjustment component performs logical operation according to the fuzzy rules. Figure 7 depicts the flow diagram for this software part. Referring to figure 7, Δω stands for speed error, and the values 05, 0A, and 0F hex represent absolute values of the amount of input adjustment. These hex values are added to or subtracted from the input r(k) in accordance with increment or decrement the input by 10, 20, and 30%, respectively. Execution of the fuzzy rule is sequential starting with the "LN speed error" clause. The "compare" and "jump conditioned" instruction sets play important roles in this software part. After finishing input modification or adjustment, the control software waits for the new sampling interval to begin.

The implementation of the PID-control component is rather conventional. Relevant descriptions can be found elsewhere. The difference is that our controller uses the modified input r(k) instead of original input. The control signal u(k) thus composes of the modified elements described by equations (2.a), (2.b), and (2.c).

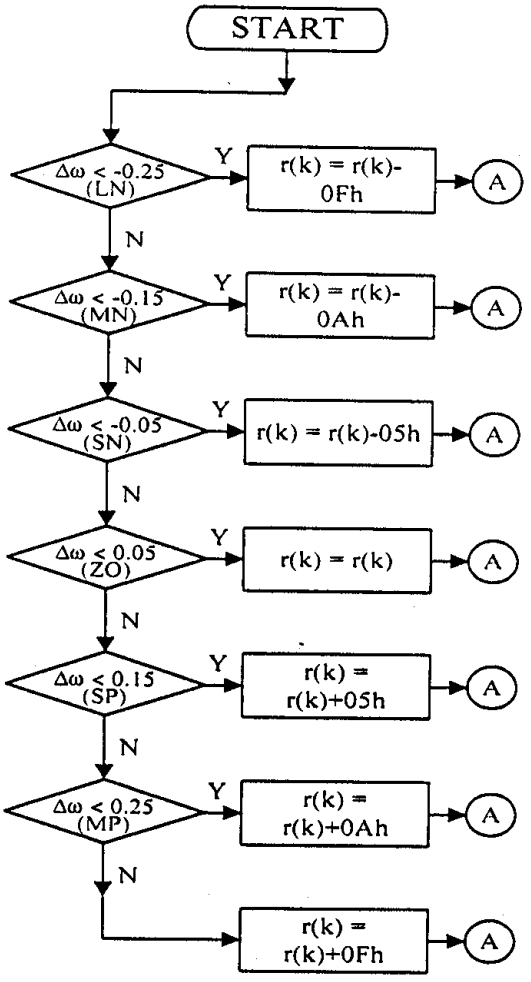

Figure 7 Flow diagram represents the execution of fuzzy rules.

The flow diagram in figure 8 gives the overall picture for the execution of our control software. The amount to modify or adjust the input r(k) is computed as the last task of each sampling interval. The control signal u(k) is updated at the beginning of each interval. After issuing the control signal u(k), the speed error is monitored and the fuzzy rules set is invoked again. This control software is implemented as an interrupt service routine for each sampling interval. It requires 0.24 milli-seconds for execution at 10 MHz.

PRACTICAL RESULTS

Practical results presented in this section illustrate the effectiveness of our control strategy. To assess its effectiveness, three experimental steps are performed as follows : (i) the motor with a conventional PID-

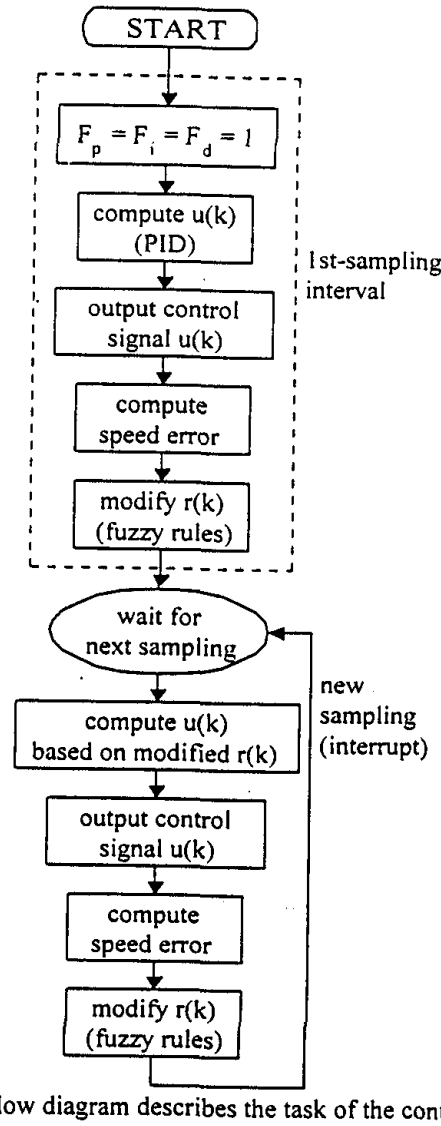

Figure 8 Flow diagram describes the task of the control software.

controller is tuned to produce a satisfactory response, (ii) the controller is detuned to imitate parameter variations in the control loop and the system response is recorded, and (iii) the fuzzy rule set is activated to regulate the system and the response is recorded. These three steps are performed automatically by the software under test. The results are presented in figure 9.

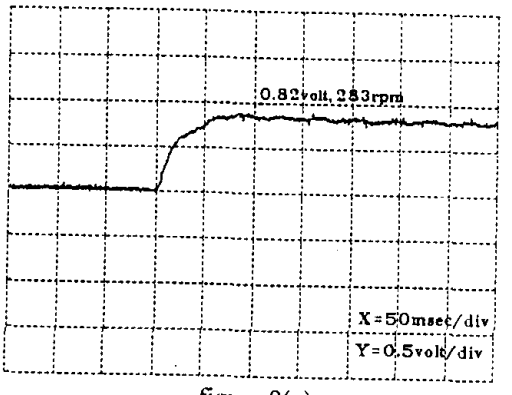

figure 9(a)

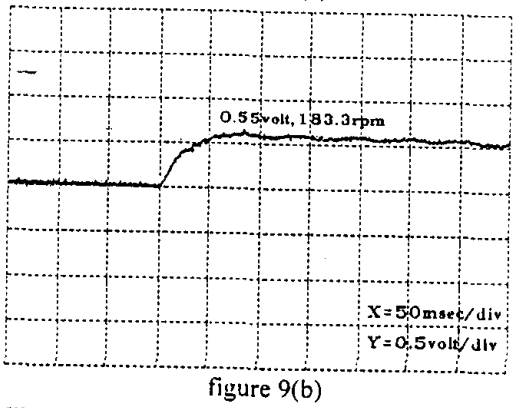

figure 9(b)

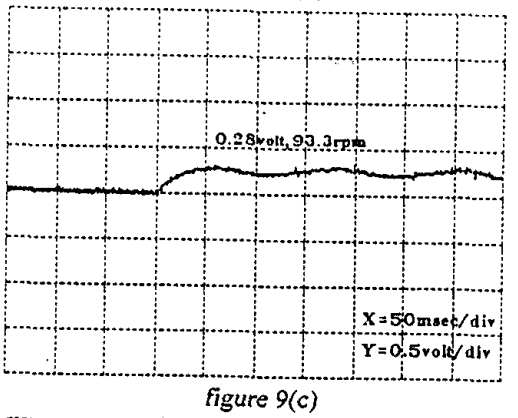

figure 9(c)

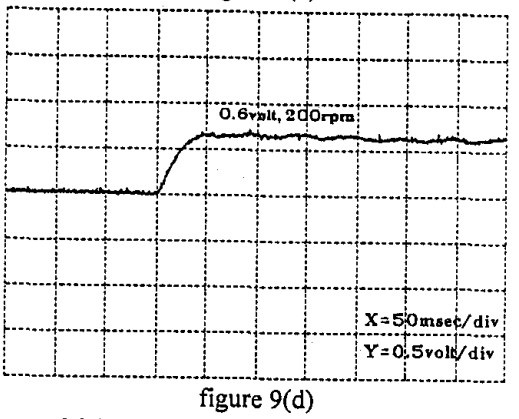

figure 9(d)

***Figure 9** Measured step response of the system with and without fuzzy supervisory control.*

Referring to figure 9, the measured speed responses are obtained from running the motor with an inertia load having its inertia about the same as that of the motor's. Figure 9(a) shows that the motor runs at higher speed (283 rpm), and time-constant is 16.7 milli-seconds. The fuzzy rule set can bring the steady-state speed back to 183.3 rpm as shown by the waveform in figure 9(b). The time-constant in this case is about the same. The difference between 183.3 and the required 200 rpm seems to be much. This can be reduced by fine tuning the corresponding fuzzy regions. Figure 9(c) depicts the speed waveform for the case of low speed of 93.3 rpm with time-constant of 20.8 milli-seconds. At low speed, low frequency oscillation becomes noticeable in the speed waveform. This is caused by imperfect coupling between the motor and load. After the activation of the fuzzy rule set, the speed is brought back to 200 rpm as shown in figure 9(d). The time-constant in this case is 18.7 milli-seconds. The results confirm the effectiveness of our fuzzy supervisory control for motor speed regulation. The controller has little influence on transient characteristic although very effective with only seven fuzzy rules.

CONCLUSION

The work presented herein confirms the usefulness of fuzzy logic to real-time control applications. The design is simple and only small number of rules can be effective. To implement a real-time fuzzy controller is not a trivial task when the sampling interval is as short as 1 milli-seconds. To keep the hardware cost low, lots of efforts are needed for software development that requires assembly programming skills. Our control software runs within 0.24 milli-seconds at 10 MHz clock by the Z180CPU. The sampling rate is 1 kHz. This approach may suit some communities where software development cost is low. In the contrary, higher performance hardware may be used if available at low cost. Experimental results show that our proposed control strategy is capable of regulating the motor speed. Even though there is no rigorous performance analysis presented, our research contributes to the promising future of fuzzy logic control to real-world applications.

REFERENCES

[1] L. A. Zadeh, Outline of a new approach to analysis of complex systems and decision process, *IEEE Trans. SME*, 3(1), 1973, 28-44.

[2] H. Takatsu and T. Itoh, Future needs for control theory in industry - report of the control technology survey in Japanese industry, *IEEE Trans. Contr. Syst. Technol.*, 7(3), 1999, 298-305.

[3] C. C. Lee, Fuzzy logic in control systems : fuzzy logic controller - part i and ii, *IEEE Trans. SME*, 20(2), 1990, 404-418 and 419-435.

[4] T. Corbet, N. Sepehri, and P. D. Lawrence, Fuzzy control of a class of hydraulically actuated industrial robots, *IEEE Trans. Contr. Syst. Technol.*, 4(4), 1996, 419-426.

[5] M. Dussud, S. Galichet, and L. P. Foulloy, Application of fuzzy logic control for continuous casting mold level control, *IEEE Trans. Contr. Syst. Technol.*, 6(2), 1998, 246-256.

[6] S. Abe, and M.-S. Lan, A method for fuzzy rules extraction directly from numerical data and its application to pattern classification, *IEEE Trans. Fuzzy Systems*, 3(1), 1995, 18-28.

[7] A. Homaifar, and E. McCormick, Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms, *IEEE Trans. Fuzzy Systems*, 3(2), 1995, 129-139.