A NEURAL-NET BASED SELF-TUNING FUZZY LOOPER CONTROL FOR ROLLING MILLS [*]

F. Janabi-Sharifi
Robotics and Manufacturing Automation Laboratory
Department of Mechanical, Aerospace, and Industrial Engineering, Ryerson University
350 Victoria Street, Toronto, Ontario, Canada M5B 2K3

Abstract

Looper control is one of the challenging problems in rolling mills. The purpose of the paper is to propose an intelligent control method using fuzzy logic and neural network for improved performance of looper control over conventional loop control methods. The focus of the paper will be on the rule-tuning aspect of the proposed fuzzy looper control. Simulation reults will also be presented to verify the performance of the control system.

1 Introduction

Control of the loopers in rolling mills plays an important role in the quality of the rolled products. The loop control methods rely on an initial formation of a bar loop by utilizing looper arms located between each pair of rolling stands (Fig. 1). In loop control, the height of the formed loop could serve as a tension indicator. Maintaining a constant desired loop height will be done by adjusting the motor speed ratios and will indicate no tension/compression status. The height of the loop (H) is measured and compared to reference height (H_0, indicating zero tension condition) to obtain a correction command for the motor speed-control unit. Conventional control techniques, e.g., PD, and PID control laws are common control methods. The difficulty of looper tension

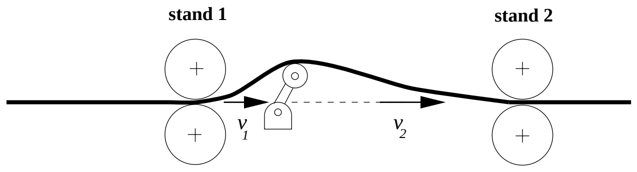

Figure 1: The rolling mill with looper control.

control lies in the varaiation and variety of plant parameters. Usually many different passes and schedules will be required to acheive a given final gauge for different grades and widths of rolled strip. For instance, it is reported that ”about 3000 schedules are available on the Senszimir mill, rolling various steel strips in widths ranging from 0.6-1.4 m and gauges of 4 down to 0.3 mm.” [8]. Many disturbances are also introduced during each run [3]. Conventional controllers cannot deal effectively with unmodeled dynamics and large system variations. The difficulty of tension control is generally manifested by the significant amount of scraps runs and damages to machinery. Therefore, many steel makers are not satisfied with the conventional controllers any more. Despite recent efforts, such as multivariable H_∞-based method [2], development of a robust looper control remains an important rolling control problem.

The potential advantages of fuzzy looper control and its design issues were discussed in our previous paper [4]. Advantages include: robustness to interfering disturbances, ease of development, and knowledge extraction. It was shown that a properly designed fuzzy controller could outperform conventional controllers. However, tuning of fuzzy control remains a bottleneck in its application. Also, the performance may still be unsatisfactory when the system parameters vary too much. The basic reason is the lack of the system learning (self-tuning) ability for dealing with different variations. Multiple tunings involved (membership functions, rules, operators, and gains) make the optimum tuning more difficult. One problem with fixed sets of membership functions and rule-set is that once they are determined, they will not change and adapt (by learning) to very different operating conditions. Therefore, many researchers have recently attempted to improve the performance of self-organizing mechanisms and to establish a more systematic method of designing and tuning fuzzy controller, e.g., [5], [1]. However, these approaches lack sufficient generalization and expressing capability for the acquired knowledge. Furthermore, our experiments indicated long training effort for these approaches. In [4], we focused on membership functions tuning and provided a self-tuning scheme for both on-line and off-line learning. In this paper, the focus will be on learning rule-base, though the methodology outlined

[*] 0-7803-7087-2/01/$10.00 © 2001 IEEE

could be applied for generating fuzzy rules and membership functions. This is particularly a very important study for different reasons. First, it provides an automatic way of tuning rule-base for fuzzy looper control and contributes towards a rapidly tuneable FLC framework for rolling mills. To the best knowledge of the author, this is the first attempt for on-line tuning of the rule-base for FLC framework in rolling mills. Second, it can suggest guidelines for designing fuzzy rule-base. Third, the effect of rule-tuning could be compared with membership function tuning for the ease of tuning focus. Our approach is based on combining neural networks with fuzzy logic.

Neural networks have a good potential due to their generalization capabilities [7]. However, training of neural nets and extraction of knowledge from a trained net might be problematic. In particular, there is a rich set of mill operator linguistic knowledge and incorporating this knowledge into controllers in a systematic way is very important. The translation of good linguistic rules, however, depends on the knowledge of the control expert. In many cases, redundant or insufficient rules might be specified. Therefore, a rational solution would be to learn rules (and/or membership functions) by neural networks and then to apply a fuzzy control based on learned rules (and/or membership functions). This approach has the advantage of extracting knowledge from the neural net by inspecting the weights associated with the fuzzy rules.

The structure of the paper is as follows. Section 2 presents the system model and Section 3 describes neuro-fuzzy looper control system. Tuning algorithm is explained in Section 4. The simulation results and discussions are given in Section 5. Finally, concluding remarks are presented in Section 6.

2 System Model

The model used for simulation purposes follows that in [4]. The plant is a rolling mill with a looper for tension control (Fig. 1). The speed difference results in the storage of the strip length $L(t)$, which can be obtained from the integral of speed difference and the looper height is related to the storage length by:

$$H(t) = \frac{1}{\alpha + \beta/\sqrt{L(t)}}, L(t) = \int_t [v_1(t) - v_2(t)]dt. \quad (1)$$

Here α and β depend on the looper parameters and are determined experimentally. For the simulation purposes, we adopted experimental results of $\alpha = 0.0001955567$ and $\beta = 0.028845145$ for $L(t)$ in mm. Change of the looper height is related to the storage length by the Jacobian of the plant, which is given by

$$J_{\Delta v_1} = \frac{\partial H(t)}{\partial(\Delta v_1)} = \frac{b(v_2(t) - v_1(t))}{2\sqrt{L(t)}(\alpha\sqrt{L(t)} + \beta)^2 \dot{v_1}(t)} \quad (2)$$

3 Control Design

The control structure of the looper system follows the one in Fig. 2. The process control can be described as follows:

$$x = [x_1(k), x_2(k)]^T = [\frac{e(k)}{K_{in1}}, \frac{\Delta e(k)}{K_{in2}}]^T, \quad (3)$$

$$e(k) = y(k) - y_r(k), \quad \Delta e(k) = e(k) - e(k-1), \quad (4)$$

$$y(k) = H(k), \quad y_r(k) = H_r(k), \quad (5)$$

$$u(k) = \Delta v_1(k) = U[x_1(k), x_2(k)]. \quad (6)$$

Here state vector x includes error $e(k)$ and error change $\Delta e(k)$ as the inputs to the controller, and the controller output $u(k)$ is generated using $U[x_1(k), x_2(k)]$, nonlinear mapping implemented using neuro-fuzzy controller. The input scaling factors are K_{in1} and K_{in2} respectively. Also, $y(k)$ and $y_r(k)$ are system output and reference command respectively. In the following, we describe how $U[x_1(k), x_2(k)]$ is implemented. The realization of

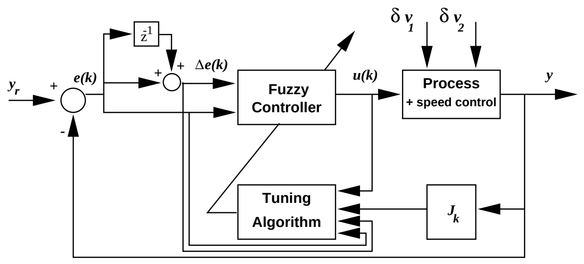

Figure 2: The structure of the looper control system.

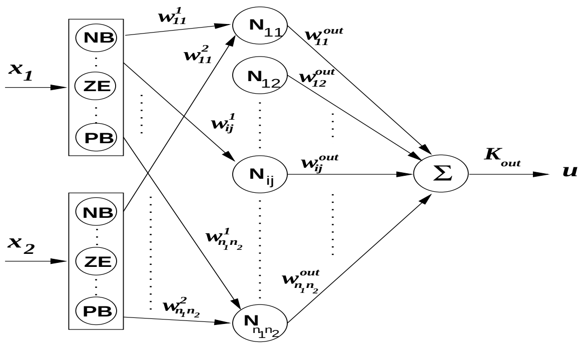

Figure 3: The structure of the neuro-fuzzy control.

the function $U[x_1(k), x_2(k)]$ follows the neural-net based fuzzy system design method of [6] and [4]. It basically consists of three stages: fuzzification, decision making fuzzy logic, and defuzzification. The process of fuzzification transforms the inputs $x_1(k)$ and $x_2(k)$ into the setting of linguistic variables which maybe viewed as labels of a fuzzy set. In this work, different linguistic variables were used for e, and Δe, represented by L_{x_1} and L_{x_2}, each with n_1 and n_2 number of membership functions respectively. Here the meaning of each variable should be clear from its mnemonic: **NB** (Negative Big), **NM** (Negative Medium), **NS** (Negative Small), **N** (Negative), **Z** (Zero), **P** (Positive), **PS** (Positive Small), **PM** (Positive Medium), and **PB** (Positive Big). The membership functions of the inputs to the controller (Fig. 4) were all assumed to be triangular. The fuzzy partitioning of the Fig. 4 has been based on normalization and choosing scaling factors by inspecting the operation range. The scaling factors are based on the expert knowledge and can be rapidly adjusted by means of a few trials. For each input x_j ($j = 1, 2$), we assign numbers $\mu_{A_{ij}}(x_j)$ using membership functions A_{ij} associated with L_{x_j}: $\mu_{A_{ij}}(x_j) = A_{ij}(x_j)$. Each membership function could

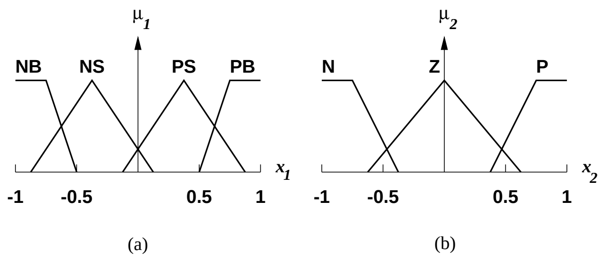

Figure 4: An example of membership functions for the state variables: (a) $x_1 = e$, (b) $x_2 = \Delta e$. The domains for both variables have been normalized to [-1, 1]. membership function.

be represented by a (coordinate of the center of triangle) and b (base length) [4]. A 3-layered neural network is used to represent the fuzzy controller (Fig. 3). The network directly maps weights of layers into fuzzy rules and membership functions. The fuzzification process is done in the first (input) layer, i.e. the outputs of the first layer will be obtained from: $o_i^j = \mu_{A_{ij}}(x_j)$, ($j = 1, 2$). The rules are represented by the hidden layer. Each rule R will be represented by a neuron N_{ij} associated with the i^{th} membership function of x_1 and j^{th} membership function associated with x_j. Also, w_{ij}^1 and w_{ij}^2 are the weights between the input and hidden layers for N_{ij}. Here, w_{ij}^{out} is the weight between the hidden and output layers from N_{ij}. The inputs represent the antecedent parts of the rules and the latter weights represent the consequent part. These weights will indicate the contribution of each rule. Sample of the rules will be given in the next section. The rules are singleton type and will have the form of:

Rule k: **If** x_1 is l_i **and** x_2 is l_j, **then** $u = w_{ij}^{out}$.

Here $l_i \in L_{x_1}$, and $l_j \in L_{x_2}$ are sets of linguistic terms attached to x_1, and x_2 respectively. Also, w_{ij}^{out} is a real number associated with the singleton controller which is a special case of Takagi-Sugeno controller [10] with all coefficients of higher order equal to zero. The middle layer neurons represent the rule-base. Product based neurons have been used in the middle layer. That is:

$$net_{ij}^{hid} = (w_{ij}^1 o_i^1).(w_{ij}^2 o_j^2). \tag{7}$$

Here, linear activation function with the slope of unity is used. Thus, $o_{ij}^{hid} = net_{ij}^{hid}$. Finally, the result of defuzzification will be calculated from:

$$u = K_{out} f(net_{ij}^{hid}) = K_{out} \sum w_{ij}^{out} o_{ij}^{out}. \tag{8}$$

4 Tuning Algorithm

Tuning FLC is not a simple task as it has more parameters to be tuned than its non-fuzzy counterparts. It is possible to tune rules, operators, and/or membership functions (MFs). Our initial work [4] focused on MF tuning. In this work, our focus will be on rule tuning for better performance. The network represented (Fig. 3) is a modified version of backpropagation net [9] and follows that of Khan and Venkatapuram [6]. The network is capable of learning both rules and membership functions. In order to learn membership functions, the input layer can be represented by a layer of neurons. We will only contempt ourselves to learning and/or evaluating rules in this paper. The net is initialized with some weight values. The input-output set are then presented to the system for weight modification and consequently rule learning. It is important to select the training set carefully to cover a vast range of input space. The weight modification algorithm is as follows.

The equivalent error at the output layer is:

$$d^{out} = (y - y_d)\frac{\partial y}{\partial u} f'(net) \tag{9}$$

because $f'(net)$ is unity for the hidden and output layer neurons. net is the input of the output layer neurons, and $\frac{\partial y}{\partial u}$ is the plant Jacobian. The weights of the output layer are modified according to:

$$w_{ij}^{out}(k+1) = w_{ij}^{out}(k) + \gamma d^{out} o_{ij}^{out} \tag{10}$$

where γ is the learning rate. The equivalent error at the hidden layer can be calculated from:

$$d_{ij}^{hid} = f'(net_{ij}^{hid}) d^{out} w_{ij}^{out} \tag{11}$$

For the input layer, the equivalent error expression becomes

$$\begin{aligned} d_{ij}^1 &= f'(net_{ij}) d_{ij}^{hid} w_{ij}^{out} w_{ij}^2 o_j^2, \\ d_{ij}^2 &= f'(net_{ij}) d_{ij}^{hid} w_{ij}^{out} w_{ij}^1 o_j^1. \end{aligned} \tag{12}$$

The inner weights are thus corrected according to:

$$\begin{aligned} w_{ij}^1(k+1) &= w_{ij}^1(k) + \gamma d_{ij}^1 o_i^1, \\ w_{ij}^2(k+1) &= w_{ij}^2(k) + \gamma d_{ij}^2 o_j^2. \end{aligned} \tag{13}$$

5 Simulation Results

The performance of fuzzy control without tuning (FLC) and with neural-net based tuning (TNFLC) on different rule sets were compared. Both off-line and on-line learning were considered but we will present off-line learning results unless otherwise specified. In off-line learning, FLC was tuned using the steady state error. Due to its advantages [4], only linearized Jacobian sign was utilized for selecting the tuning direction. The simulation parameters were chosen to be: $v_1(0)$ = 7000 mm/s, v_2 = 7000 mm/s, H_r = 200 mm, $\Delta H(0)$ = 50 mm, and ΔT = 0.05 s. The learning coefficients were determined after experiments: $\gamma = 0.01$ for off-line learning, and $\gamma = 0.005$ for on-line learning, $k_a = k_b = 0.1$, $k_w = 0.01$ for MF tuning using [4]. The scaling factors were chosen to be: $K_{in1} = 60$, $K_{in2} = 10$, and $K_{out} = 50$. The results are shown in Figs. 5 to 6 and Tables 1 to 4. The solid and dotted lines represent the responses without and with neuro-fuzzy tuning respectively.

Effect of n_1, n_2 (number of MFs) and rule-base size. Several cases with different numbers of membership functions and consequently rule-bases were considered. Only 2 cases will be shown here for the sake of brevity.
Case (1)- (7 × 7) rule-base as in Table 1.
Case (2)- (5 × 3) rule-base as in Table 2.
The weights between input and hidden layer remained 1 in order to tune the rules and not the membership functions. The weights between the hidden layer and output layer were initialized using the initial singleton values inside the brackets (Tables in 1 and 2). For better initialization, the numeric values of the centers of MFs for consequent parts of Mamdani rules have been considered. In Case (1), the [a, b] parameters of MFs for e were: NB:[-1.0, 0.7]; NM:[-0.65, 0.7]; NS:[-0.3, 0.7]; ZE:[0.0, 0.6]; PS:[0.3, 0.7]; PM:[0.65, 0.7]; and PB:[1.0, 0.7]. Those for Δe were exactly the same. In Case (2), those for e were: NB:[-1.0, 1.0]; NS:[-0.5, 1.2]; ZE:[0.0, 1.2]; PS:[0.5, 1.2]; and PB:[1.0, 1.0]. The MF parameters for Δe were: N:[-1.0, 2.0]; Z:[0.0, 2.0]; and P:[1.0, 2.0]. The net was trained and the results were compared to initial values. The results are shown in Fig. 5 and Tables 3 and 4. From the obtained results, the following observations were made. **(1)** Introduction of the neural system for rule tuning could improve the performance of the system significantly. **(2)** Higher number of MFs does not necessarily improve the performance of the system, as the performance of the controller for Case (1) is not superior to those of other cases. For FLC, the largest steady-state error belongs to Case (1) with 7 × 7 rules. Besides, higher number of MFs adds to the computational complexity of the system. The rule-base with 5 × 3 rules, for instance, seems to be sufficient for practical purposes. **(3)** One can use the outputs of the neurons in the hidden layer (Tables 3 and 4) to cancel the rules represented by very small numbers, given that a wide operating range has been used during training. Also, one can additionally inspect the significance of the rules by examining the weights between the hidden and output layers. Tables 3 and 4 can be compared with Tables 1 and 2 for this purpose. This again indicates that there would be rule redundancy in defining 7 × 7 rule-base.

Table 1: Initialization of rule-base for Case (1). The singleton values associated with fuzzy terms are: (NB:-1.00, NM:-0.70, NS:-0.25, ZE:0.10, PS:0.35, PM:0.70, PB:1.00).

$\Delta e \mid e$	**NB**	**NM**	**NS**	**ZE**	**PS**	**PM**	**PB**
NB	PB	PB	PM	PM	PS	PS	ZE
NM	PM	PM	PS	PS	ZE	ZE	NS
NS	PM	PS	PS	ZE	ZE	NS	NS
ZE	PS	PS	ZE	ZE	ZE	NS	NS
PS	PS	PS	ZE	ZE	NS	NS	NM
PM	PS	ZE	ZE	NS	NS	NM	NM
PB	ZE	NS	NS	NM	NM	NB	NB

Comparison with PID control. The PID gains were tuned to give optimum response: $K_P = K_D = 5$, $K_I = 1$. When the plant parameters α and β were changed by 20% and 40% respectively, PID control became unstable. Similar results to [4] were obtained. The resulst are not shown here for the sake of brevity. Again, Fuzzy control demonstrated a steady performance. Self-tuning was achieved using a Singleton-based control with on-line tuning. Finally, it was shown that self-tuning Fuzzy control, in comparison with PID and Fuzzy controls, provided lower steady-state error and stable behavior.

Comparison with MF tuning. It was practically important to compare the effects of MF tuning [4] and rule tuning (TNFLC). Similar initial conditions were considered: on-line, singleton, Jacobian sign, learning rate of 0.005, 4 × 3 rule-base (Table 5). The MF parameters for e were: NB:[-1.0, 1.0]; NS:[-0.5, 1.2]; PS:[0.5, 1.2]; and PB:[1.0, 1.0]. Those for Δe were: N:[-1.0, 2.0]; Z:[0.0, 2.0]; and

Table 2: Initialization of rule-base for Case (2). The singleton values associated with fuzzy terms are: (NB:-1.00, NM:-0.70, NS:-0.25, ZE:0.10, PS:0.35, PM:0.70, PB:1.00).

$\Delta e \mid e$	**NB**	**NS**	**ZE**	**PS**	**PB**
N	PB	PM	PS	ZE	NS
ZE	PM	PS	ZE	NS	NM
P	PS	ZE	NS	NM	NB

Table 3: The weights between the hidden and output layers (or the outputs of neurons in the hidden layer) for Case (1).

$\Delta e \mid e$	**NB**	**NM**	**NS**	**ZE**	**PS**	**PM**	**PB**
NB	1.00	1.00	0.70	0.63	0.32	0.35	0.10
	(0)	(0)	(0)	(0)	(0)	(0)	(0)
NM	0.70	0.70	0.35	0.28	0.07	0.10	-0.25
	(0)	(0)	(0)	(0)	(0)	(0)	(0)
NS	0.70	0.35	0.35	0.03	0.02	-0.26	-0.25
	(0)	(0)	(0.09)	(0.18)	(0.06)	(0)	(0)
ZE	0.70	0.35	0.35	0.03	-0.29	-0.34	-0.25
	(0)	(0)	(0.54)	(0.98)	(0.42)	(0)	(0)
PS	0.35	0.35	0.10	0.03	-0.33	-0.26	-0.70
	(0)	(0)	(0.08)	(0.30)	(0.10)	(0)	(0)
PM	0.35	0.10	0.10	-0.32	-0.28	-0.70	-0.70
	(0)	(0)	(0)	(0)	(0)	(0)	(0)
PB	0.10	-0.25	-0.25	-0.77	-0.70	-1.00	-1.00
	(0)	(0)	(0)	(0)	(0)	(0)	(0)

Table 4: The weights between the hidden and output layers (or the outputs of neurons in the hidden layer) for Case (2).

$\Delta e \mid e$	**NB**	**NS**	**ZE**	**PS**	**PB**
N	1.00	0.70	0.34	0.09	-0.25
	(0)	(0.00)	(0.01)	(0.00)	(0)
ZE	0.70	0.35	-0.01	-0.29	-0.70
	(0)	(0.17)	(1.00)	(0.17)	(0)
P	0.35	0.10	-0.26	-0.70	-1.0
	(0)	(0.00)	(0.00)	(0.00)	(0)

P:[1.0, 2.0]. The step responses of both training methods were compared with FLC with no tuning (Fig. 6). As it can be seen, the steady-state errors of both training methods are very close and both of them improve steady-state error of FLC. However, neural based rule-tuning indicates lower undershoot than MF tuning. The effect of rule tuning can be also observed by comparing the control surfaces (Fig. 7).

Table 5: Rule-base for comparison with MF tuning.

$\Delta e \mid e$	**NB**	**NS**	**PS**	**PB**
N	1.00	0.30	0.55	-0.35
Z	0.70	-0.10	0.30	-0.70
P	0.55	-0.35	-0.10	-1.00

Effect of rule tuning combined with MF tuning. The same 4 $\times$ 3 rule-base (Table 5) and MF parameters were used and two additional cases were considered:
Case (3)- The membership functions of neuro-fuzzy controller were initialized by normal values.
Case (4)- The membership functions of the antecedent

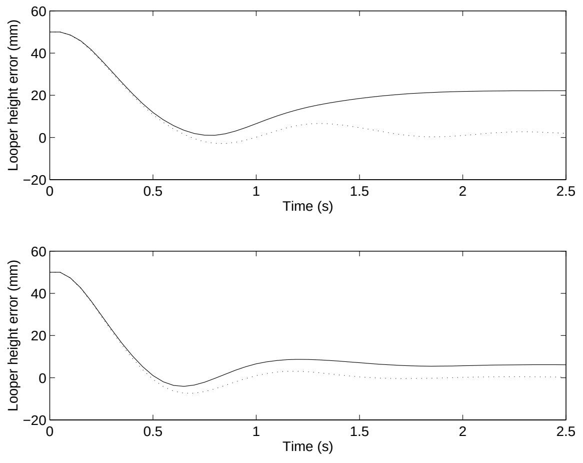

Figure 5: Comparison of step responses for FLC (solid) and TNFLC (dotted) under Case(1) (top), and under Case(2) (bottom).

parts were first tuned by MF tuning [4]. The resultant MFs were then used to initialize neuro-fuzzy controller. The results are shown in Fig. 6 which indicates that there is not any significant difference between two responses. Therefore, as far as rule-tuning is concerned, the initial tuning of the antecedent MFs might not affect the performance of the controller significantly.

6 Conclusions

A neuro-fuzzy controller has been proposed for looper control in rolling mills. The proposed system is capable of generating, tuning, and/or evaluating the fuzzy rules (and membership functions) by neural-net learning. Since the fuzzy inference and defuzzification are combined, and because less number of calculations are involved, the system cycle time is reduced. Therefore, it provides a way of reducing the number of rules and hence improving the computational cost and processor response time. Moreover, the shown technique facilitates the evaluation of the rules and membership functions often generated heuristically. The effectiveness of the method was verfied by the simulation results. Several cases were compared and practical conclusions were made regarding: the size of the rule-base, pre-tuning using MF tuning methods [4], and effectiveness of the proposed method in comparison with conventional PID control for looper control in rolling mills.

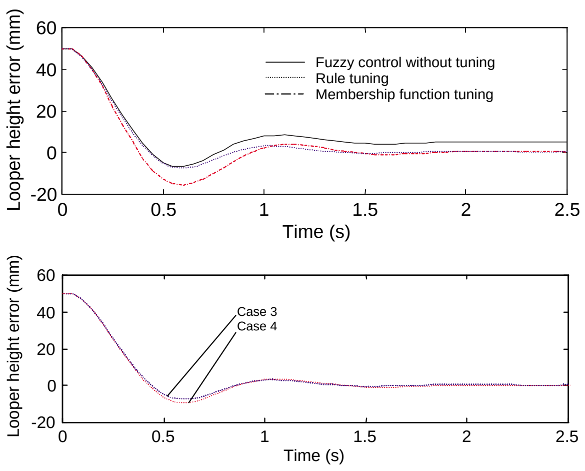

Figure 6: Comparison of the effects of rule-tuning (TN-FLC) and MF tuning: individual (top) and combined (bottom).

Acknowledgments

This research was financially supported by Quad Engineering Inc., Natural Sciences and Engineering Research Council of Canada (NSERC) under CRDPJ234028-99. The author wishes to thank the useful discussions and assistance of Jian Fan, Leon Winitsky, Jalal Biglou, and Mark Pinkus from Quad Engineering, Inc.

References

[1] I. Hayashi, H. Nomura and N. Wakami, “Artificial neural network driven Fuzzy control and its application to the learning of onverted pendulum system,” *Proc. 3rd IFSA Congress*, pp. 610-613, 1989.

[2] G. Hearns, M.R. Katebi, and M.J. Grimble, “Robust control of a hot strip mill looper,” *Proc. 1996 IFAC World Cong.*, pp. 445-450.

[3] H. Imanari, Y. Morimatsu, K. Sekiguchi, H. Ezure, R. Matuoka, A. Tokuda, and H. Otobe, “Looper H-infinity control of hot-strip mills,” *IEEE Trans. Ind. Appl.*, vol. 33, no. 3, pp. 790-796, May 1997.

[4] F. Janabi-Sharifi and J. Fan, “Self-tuning fuzzy looper control for rolling mills,” *Proc. IEEE Conf. Decision and Control*, Sydney, Australia, Dec. 2000, pp. 376-381.

[5] R. Jang, “Self-learning fuzzy controllers based on temporal back propagation,” *IEEE Trans. Neural Networks*, vol. 3, pp. 714-723, Sep. 1992.

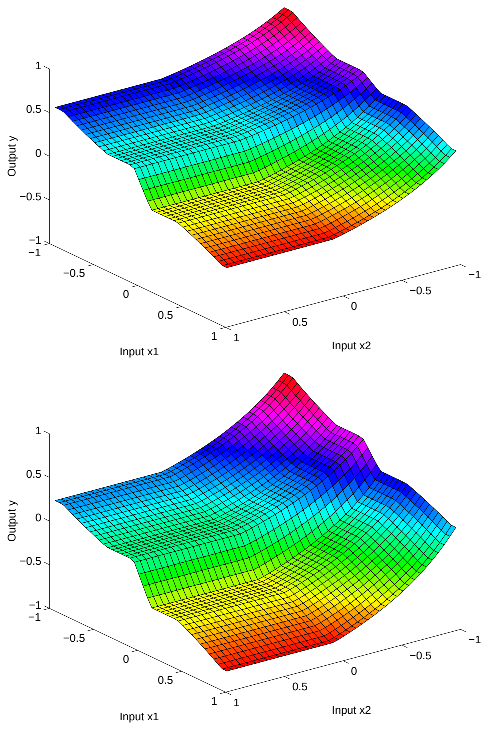

Figure 7: Control surface of initial FLC (top) and TNFLC (bottom).

[6] E. Khan and P. Venkatapuram, “Neufuz: neural network based fuzzy logic design algorithms,” *Proc. IEEE Int. Conf. Fuzzy Systems*, San Francisco, CA, pp. 647-654, 1993.

[7] K.S. Narendra and K. Parthasarathy, “Identification and control of dynamical systems using neural networks,” *IEEE Trans. Neural Networks*, vol. 1, no. 1, pp. 4-27, 1990.

[8] J. V. Ringwood, “Shape control systems for Sendzimir steel mills,” *IEEE Trans. Control Tech.*, vol. 8, no. 1, Jan. 2000, pp. 70-86.

[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagation errors,” in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Foundations*, D. E. Rumelhart and J. L. McClelland, Eds., vol. 1, MIT Press, Cambridge, MA, 1986.

[10] T. Takagi and M. Sugeno, “Fuzzy identification of systems and its applications to modeling and control,” *IEEE Trans. Systems, Man, and Cybernetics*, vol. 15, no. 1, 1985, pp. 116-132.