A Learning Fuzzy System for Looper Control in Rolling Mills

F. Janabi-Sharifi
Department of Mechanical Engineering
Ryerson Polytechnic University
Toronto, Ontario, Canada M5B 2K3

J. Fan
Quad Engineering Inc.
North York, Ontario, Canada M3B 2R2

Abstract

In this paper, several issues of looper control in rolling mills are discussed and a fuzzy logic control is proposed to address the enumerated issues. The proposed system incorporated fast tuning algorithms for both off-line and on-line learning of membership functions and singleton values. Also, it is relatively simple to design and implement. The effectiveness of the proposed system is verified by the simulations results. It is shown that the proposed system is more robust to parameter uncertainties and noises, when compared with conventional PID control.

1 Introduction

The loop control method is one of the common control techniques for producing flexible cross-sections at intermediate and finishing sub-mills. The loop control techniques rely on the initial formation of a bar loop by using mechanical deflectors and proper motor speed adjustments (Fig. 1). In fact, each stand roll speed has to be synchronized to the speed of the bar exiting the previous stand. Otherwise, *push* or *pull* conditions would occur and loop might be formed. 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 motor speed control unit. An example is the implementation of the looper control in a 17 stand bar mill in the Chapparrel Steel mill in Milwaukee by ASEA [3]. A characteristic control problem is the introduction of the disturbances of different nature. These disturbances are caused by the increasing use of low temperature heating of slabs, high-carbon steels, and high-speed rolling. Due to increasing demand for high precision in thickness, width, and crown of the strips, it is very important to develop a high-performance and robust looper control.

In this paper, the potential of fuzzy controllers will be examined and the issues on the application of fuzzy controllers for looper control will be discussed. We were motivated to explore the application of fuzzy looper control for the following reasons: (1) fuzzy controllers prove to be robust to unmodeled dynamics and noises; (2) a typical fuzzy control design does not require a formal model of the plant or an initial training set; and (3) sufficient knowledge would be available in the form of linguistic if-then rules by interviewing a mill operator.

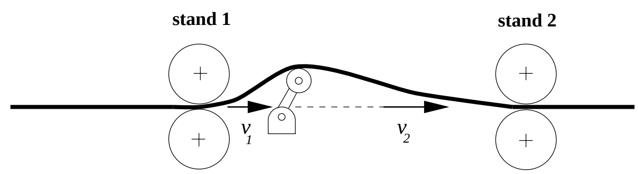

Figure 1: The rolling mill with looper control.

Despite the enumerated advantages, a few problems might arise when implementing a fuzzy logic controller (FLC). The translation of good linguistic rules depends on the knowledge of the control expert. In many cases, redundant or insufficient rules might be specified. Also, translation into fuzzy set theory is not formalized yet and arbitrary choices concerning the shape or membership functions or T-operators might be made. These uncertainties in the design of a FLC usually results in a heuristic tuning to overcome the initial design errors [2]. Therefore, it is very important to have learning procedures which can tune the system automatically. The automated FLC tuning was first proposed by Procyk and Mamdani [5]. Most of the work in automated FLC tuning has focused on neural nets [1]. However, these approaches lack sufficient generalization and expressing capability of the acquired knowledge. Furthermore, our experiments indicated long training effort.

In this paper, we will implement a self-tuning method based on descent technique for tuning membership functions of FLC. Both off-line and on-line tuning will be considered. This will contribute towards a rapidly tunable FLC framework for rolling mills. To the best knowledge of the authors, this is the first on-line tunable FLC framework for the looper control in rolling mills. Different design issues will be discussed and practical conclusions will be derived.

2 Looper System Model

The plant is a rolling mill with a looper for tension control (Fig. 1). The tension is caused by the difference between the exit speed v_1 in stand one and the entry speed v_2 in stand two. The speed difference results in the storage of the strip length, which can be obtained from the integral of speed difference:

$$L(t) = \int_t [v_1(t) - v_2(t)]dt \tag{1}$$

and the looper height is related to the storage length by

$$H(t) = \frac{1}{\alpha + \beta\sqrt{L(t)}}. \tag{2}$$

Here α and β depend on the looper parameters and are determined experimentally. For the simulation purposes, we adopted $\alpha = 0.0001955567$ and $\beta = 0.028845145$ for $L(t)$ in mm. The Jacobian of the plant is then given by

$$J_{\Delta v_1} = \frac{\partial H(t)}{\partial(\Delta v_1)} = \frac{b(v_2(t) - v_1(t))}{2\sqrt{L(t)}(\alpha\sqrt{L(t)} + \beta)^2 v_1(t)} \tag{3}$$

3 Design of Fuzzy Controller

The control structure of the looper system is shown in Fig. 2. In this section, we will not consider any tuning aspect and the focus will be on FLC. The process control can be described as follows:

$$x(k) = [x_1(k), x_2(k)]^T = [e(k)/K_{in1}, \Delta e(k)/K_{in2}]^T, \tag{4}$$

$$e(k) = y(k) - y_r(k), \quad \Delta e(k) = e(k) - e(k-1), \tag{5}$$

$$y(k) = H(k), \quad y_r(k) = H_r(k), \tag{6}$$

$$u(k) = \Delta v_1(k) = U[x_1(k), x_2(k)]. \tag{7}$$

Here state vector x includes error $e(k)$ and error change $\Delta e(k)$ as the 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 fuzzy logic. 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 the function $U[x_1(k), x_2(k)]$ is based on a fuzzy logic method and consists of three stages: fuzzification, decision making fuzzy logic, and defuzzification.

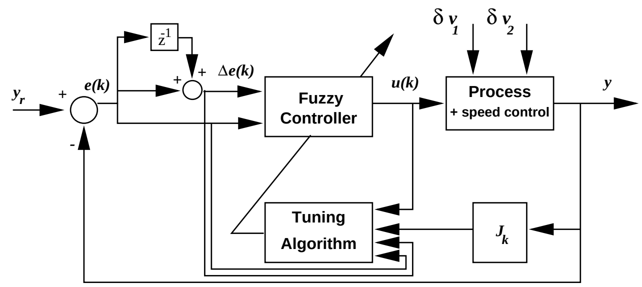

Figure 2: The structure of the lopper control system.

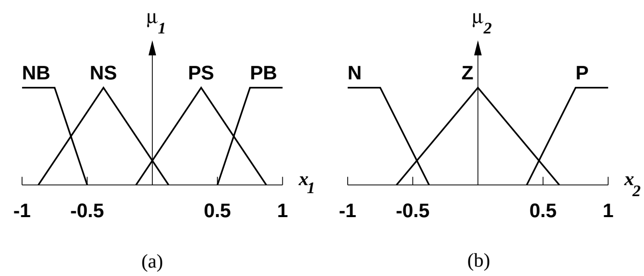

Figure 3: Membership functions for the state variables: (a) $x_1 = e$, (b) $x_2 = \Delta e$. The domains for both variables have been normalized to [-1, 1].

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, the following linguistic variables were used for e, and Δe respectively:

$$L_{x_1} = \{\textbf{NB}, \textbf{NS}, \textbf{PS}, \textbf{PB}\}, \quad L_{x_2} = \{\textbf{N}, \textbf{Z}, \textbf{P}\}. \tag{8}$$

Here the meaning of each variable should be clear from its mnemonic: **NB** (Negative Big), **NS** (Negative Small), **N** (Negative), **Z** (Zero), **P** (Positive), **PS** (Positive Small), and **PB** (Positive Big). The membership functions of the inputs to the controller (Fig. 3) were all assumed to be triangular. The fuzzy partitioning of the Fig. 3 has been based on normalization and choosing scaling factors by inspecting the operation range. The scaling factor is based on the expert knowledge and can be rapidly adjusted by means of a few trials. For each input x_1 and x_2, we assign numbers $\mu_{A_{i1}}(x_1)$ and $\mu_{A_{i2}}(x_2)$ using membership functions A_{i1} and A_{i2} associated with L_{x_1} and L_{x_2} respectively. These numbers will be used for fuzzy decision making which is described next.

Associated with the fuzzy logic decision process is a set of fuzzy rules $R = \{R_1, R_2, R_3, \cdots, R_{12}\}$. The rules are given in Fig. 4, where the values in brackets indicate Mamdani rules. A singleton rule will have a form of:

Rule i: **If** x_1 is l_{i1} **and** x_2 is l_{i2}, **then** $u = w_i$.

Δe \ e	NB	NS	PS	PB
N	w_1 (PB)	w_2 (PM)	w_3 (PS)	w_4 (PZ)
Z	w_5 (PS)	w_6 (PZ)	w_7 (NZ)	w_8 (NS)
P	w_9 (NZ)	w_{10} (NS)	w_{11} (NM)	w_{12} (NB)

Figure 4: Fuzzy rule-base.

Here $l_{i1} \in L_{x_1}$, and $l_{i2} \in L_{x_2}$ are set of linguistic terms attached to x_1, and x_2 respectively. Also, w_i is a real number associated with the singleton controller which is a special case of Takagi-Sugeno controller [7] with all coefficients of higher order equal to zero. Fuzzy rules were developed heuristically to facilitate the shaping of the looper's response and to simplify tuning of the controller. If a different response is desired for a particular range of input variables, then only a few fuzzy rules would need to be altered. The ability to modify the controlled response locally, while not significantly altering the global response, is an attractive feature of fuzzy control, in particular from learning point of view. The isosceles triangles, used for the membership functions, could be characterized by their center and width. For instance, membership function A_{ij} could be expressed by its center a_{ij} and the width b_{ij}. Therefore, the grade of membership for A_{ij}, denoted by $\mu_{A_{ij}}$, could be obtained from

$$\mu_{A_{ij}}(x_j) = 1 - \frac{2 \mid x_j - a_{ij} \mid}{b_{ij}}, \qquad j = 1, 2. \tag{9}$$

Given a pair of $e(k)$ and $\Delta e(k)$, each relevant rule first assigns $\mu_{A_{i1}}(x_1(k))$ and $\mu_{A_{i2}}(x_2(k))$. Then a value is assigned using the production *t-norm* defined as follow:

$$\mu_i(x_1(k), x_2(k)) = \mu_{A_{i1}}(x_1(k)) \cdot \mu_{A_{i2}}(x_2(k)) \tag{10}$$

for the ith rule. This process is repeated for all the relevant rules.

Basically, defuzzification is a mapping from a space of fuzzy control action defined over a universe of discourse into a space of non-fuzzy control actions. Several methods of defuzzification can be considered. The defuzzification process considered here takes a form of Takagi-Sugeno output value computation method, given by:

$$u(k) = U[x_1(k), x_2(k)] = K_{out} \frac{\sum_{i=1}^{n} \mu_i(x_1(k), x_2(k)) w_i}{\sum_{i=1}^{n} \mu_i(x_1(k), x_2(k))}, \tag{11}$$

where K_{out} is a scaling factor, and w_i is a real number associated with the consequent part of the rule (or the center of relevant membership function of the output in Mamdani rule-base).

4 Tuning Algorithm

Tuning FLC is a very difficult 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). In this work, we will focus on MF tuning for better performance. We initially focused on conventional multilayer perceptron using backpropagation algorithm [6] for tuning MFs. However, the results were not encouraging with respect to convergence and speed of tuning. Therefore, we focused on descent methods [4]. The tuning algorithm presented here is relatively simple and fast for online tuning. The objective function to be minimized is defined as:

$$E(k) = \frac{1}{2} \sum_k [y(k) - y_r]^2 \tag{12}$$

with $y(k)$ as the system output in the kth instance. It can be easily shown that the learning rules can be expressed as:

$$a_{ij}(k+1) = a_{ij}(k) - k_a \frac{\partial E(k)}{\partial a_{ij}(k)} \tag{13}$$

$$b_{ij}(k+1) = b_{ij}(k) - k_b \frac{\partial E(k)}{\partial b_{ij}(k)} \tag{14}$$

$$w_i(k+1) = w_i(k) - k_w \frac{\partial E(k)}{\partial w_i(k)} \tag{15}$$

where k_a, k_b, and k_w are learning coefficients. The gradients in equations (13), (14), and (15) can be derived using equations (9) to (11).

$$\frac{\partial E(k)}{\partial a_{ij}(k)} = \frac{\mu_i(x_1(k), x_2(k))}{\sum_{i=1}^{n} \mu_i(x_1(k), x_2(k))} \cdot \Psi_1, \tag{16}$$

$$\frac{\partial E(k)}{\partial b_{ij}(k)} = \frac{\mu_i(x_1(k), x_2(k))}{\sum_{i=1}^{n} \mu_i(x_1(k), x_2(k))} \cdot \Psi_2, \tag{17}$$

$$\frac{\partial E(k)}{\partial w_i(k)} = \frac{\mu_i(x_1(k), x_2(k))}{\sum_{i=1}^{n} \mu_i(x_1(k), x_2(k))} \sum_k J_k e(k), \tag{18}$$

$$\begin{aligned} \Psi_1 \quad = \quad & \sum_k J_k e(k)(w_i(k) - y(k)) \mathrm{sgn}(x_j(k) - a_{ij}(k)) \\ & \cdot \frac{2}{b_{ij}(k) \mu_{A_{ij}}(x_j(k))}, \end{aligned} \tag{19}$$

$$\Psi_2 = \sum_k J_k e(k)(w_i(k) - y(k)) \frac{1 - \mu_{A_{ij}}(x_j(k))}{b_{ij}(k) \mu_{A_{ij}}(x_j(k))}. \tag{20}$$

Here $J_k = (\frac{\partial H}{\partial u})_k$ is the Jacobian. The summation $\sum_k$ is for off-line learning which uses the summation of measurements at the discrete instances after the end of control cycle. No summation $\sum_k$ is required in the above equations for on-line learning. In our simulations, we will use discrete linearized Jacobian instead of (3), defined as

$$J_k = \frac{y(k) - y(k-1)}{u(k) - u(k-1)} \quad (21)$$

or its sgn defined by:

$$\text{sgn}(J_k) = \begin{cases} 1 & \text{if} \quad J_k > 0 \\ 0 & \text{otherwise.} \end{cases} \quad (22)$$

The training procedure works as follows:

step 1. Initiate the inference rules and membership functions.

step 2. Calculate y, x_1, and x_2 using (2) and (4)-(6).

step 3. Calculate Jacobian using (3), or (21), or (22).

step 4. Update the membership functions and singleton values using (12) to (18).

step 5. Calculate the control output using (7). Go to Step 2.

5 Simulation Results

Simulations were run to verify the effectiveness of the proposed method. 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: $k_a = k_b = 0.1$, $k_w = 0.01$ for off-line tuning, and $k_a = k_b = 0.002$, $k_w = 0.005$ for on-line tuning. The scaling factors were chosen to be: $K_{in1} = 60$, $K_{in2} = 10$, and $K_{out} = 50$. It was observed that input-output scaling has a drastic effect on the control performance. We will leave the presentation of these results to another publication for the sake of brevity. The initial values of [a, b] for the membership functions of e were: NB: [-1.00, 1.20]; NS: [-0.40, 1.00]; PS: [0.40, 1.00]; PB: [1.00, 1.20]; and those for Δe were: N: [-0.95, 2.00]; Z: [0.00, 2.00]; P: [0.95, 2.00]. Both Mamdani and singleton type rules were examined and indicated similar performances. For brevity singleton rules are demonstrated in these simulations. Also the initial values of output singletons (w_1 - w_{12}) were: [1.00, 0.70, 0.55, 0.3.0, 0.55, 0.30, -0.10, -0.35, -0.10, -0.35, -0.70, -1.00].

The results are shown in Figs. 5 to 10. The dotted and solid line represent the responses of fuzzy logic control without tuning (FLC) and with tuning (TFLC) respectively. Several cases were studied under different conditions as follows.
(C1) Off-line learning, Jacobian model.
(C2) On-line learning, Jacobian model.
(C3) On-line learning, linearized Jacobian model.
(C4) On-line learning, Jacobian sign model.

Off-line vs on-line learning. For the off-line learning, the parameters of the membership functions are tuned after each control cycle, while in on-line learning, the parameters are tuned after each time step. The results are shown in Figs. 5-6 which indicate lower errors when the control is tuned. Also, it is shown that off-line learning provides lower undershoots than on-line learning. As it might be expected, most of the changes affect the neighborhood of the Zero membership functions. For instance, in (C1) the tuned values of [a, b] were: NS: [-0.44, 0.98], PS: [0.05, 0.94] for e; and and N: [-0.93, 2.00], Z: [-0.93, 2.00] for Δe. For the same case, noticeable changes in output singletons was only for w_6 which changed to 0.33. Similarly, in (C2) the tuned values of [a, b] were: NS: [-0.40, 0.99], PS: [0.39, 1.00] for e. However, the changes in singleton values were more dominant than (C1). That is: w_2 - w_4 changed to 0.69, 0.51, and 0.27, and w_7, w_8 were tuned to: -0.19, and -0.36 respectively.

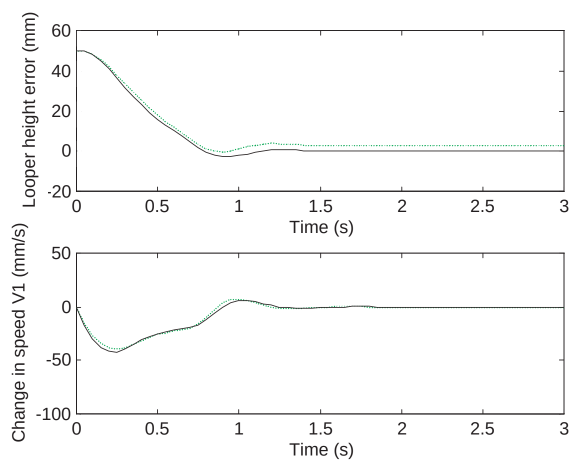

Figure 5: Comparison of step responses with FLC (dotted) and TFLC (solid) under (C1).

Effect of plant Jacobian. The plant Jacobian affects the tuning algorithm. Due to the difficulty of providing true Jacobian, approximations of Jacobian are used. A linearized model of the plant Jacobian (21) could provide an easy-to-calculate approximation. Also, in order to improve the stability of the system, we examined the sign of linearized Jacobian (22) for parameter tuning. The results are shown in Figs. 6 to 8. It was concluded that the sign of linearized

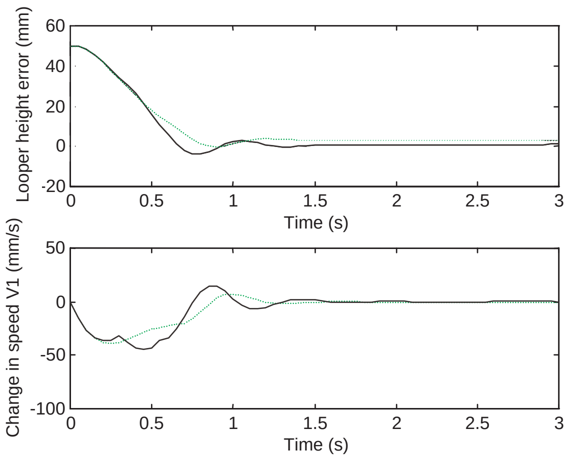

Figure 6: Comparison of step responses with FLC (dotted) and TFLC (solid) under (C2).

Jacobian could also provide good performance.

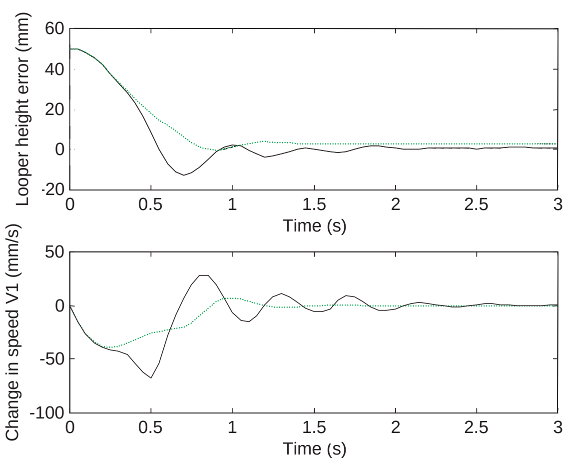

Figure 7: Comparison of step responses with FLC (dotted) and TFLC (solid) under (C3).

Disturbance rejection. To study the effectiveness of the proposed system in disturbance rejection, a step disturbance in setting of exit speed of strip in stand one (v_1) was introduced at $t = 3$ sec. As it is shown in Fig. 9, tuned controller was capable of canceling the drop caused by the disturbance. A comparison with commonly used PID control was also made. The PID gains have been 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 45% respectively, PID control became unstable (Fig. 10). However, Fuzzy control demonstrated a steady performance. Self-tuning was achieved using a Singleton-based control

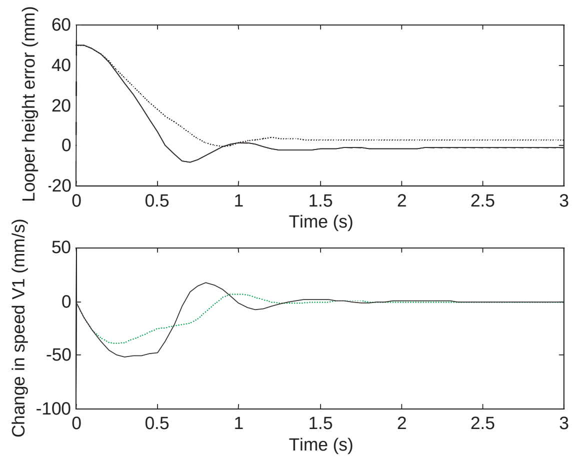

Figure 8: Comparison of step responses with FLC (dotted) and TFLC (solid) under (C4).

with on-line tuning. Finally, inspection of Fig. 10 shows that self-tuning Fuzzy control, in comparison with PID and Fuzzy controls, provides lower steady-state error and stable behavior.

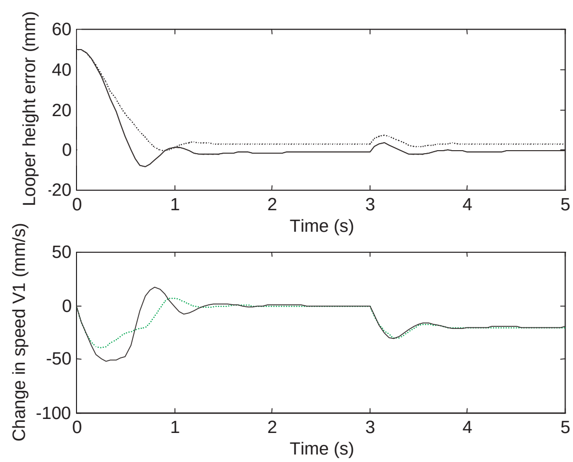

Figure 9: The effect of step disturbance on the responses of FLC (dotted) and TFLC (solid) under (C3).

6 Conclusions

In this paper, a fuzzy logic controller was proposed for looper control of rolling mills. The proposed system incorporated a learning algorithm for fine-tuning membership functions and singleton outputs. The effectiveness of the proposed system was verified by the simulation results.

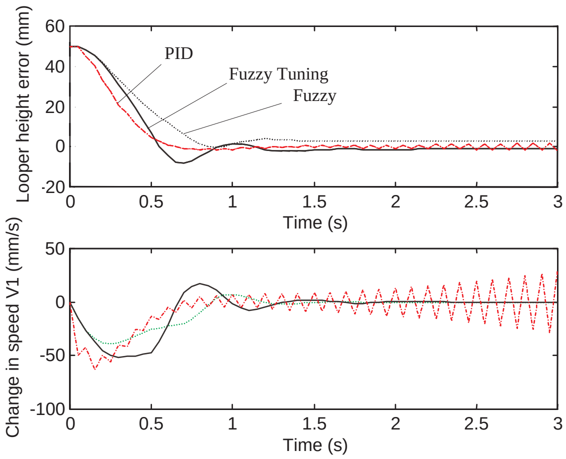

Figure 10: Comparison of the performances of PID, FLC, and TFLC under (C3) with disturbed plant parameters.

It was shown that the proposed system is more robust to parameter uncertainties and noises, when compared with classical PID control law. The incorporated tuning algorithm had the advantage of high-speed learning capability, yet it was effective for improving the control performance. Several practical issues were discussed and conclusions were made regarding different design options. Among them were: off-line vs. on-line tuning, and different Jacobian approximations.

Acknowledgments

This research was financially supported by Quad Engineering Inc. Also, we would like to thank the useful discussions and assistance of Leon Winitsky, Richard Li, Jalal Biglou, and Mark Pinkus from Quad Engineering. The second author was also supported by an industrial research fellowship from Natural Sciences and Engineering Research Council of Canada.

References

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

[2] R. Kruse, J. Gebhardt, and F. Klaonn, *Foundations of Fuzzy Systems.* Wiley, Chichester, 1994.

[3] A. Neilson, and E. Koebe, "Microprocessor control system for bar mill revamp at Chapparrel Steel," *Iron and Steel Engineer*, pp. 56-59, Apr. 1983.

[4] H. Nomura, I. Hayashi, N. Wakami, "A learning method of fuzzy inference rules by descent method," *Proc. IEEE Int. Conf. on Fuzzy Systems*, 1992, pp. 203-210, San Diego, CA.

[5] T. J. Procyk and E. H. Mamdani, "A linguistic self-organizing process controller," *Automatica*, vol. 15, pp. 15-30, 1979.

[6] 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.

[7] 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.