Abstract—The objective of this paper is to investigate and find a solution by designing the intelligent controllers for controlling water level system, such as fuzzy logic and neural network. The controllers also can be specifically run under the circumstance of system disturbances. To achieve these objectives, a prototype of water level control system has been built and implementations of both fuzzy logic and neural network control algorithms are performed. In fuzzy logic control, Sugeno model is used to control the system. In neural network control, the approach of Model Reference Adaptive Neural Network Control based on the backpropagation algorithm is applied on training the system. Both control algorithms are developed to embed into a stand-alone DSP-based micro-controller and their performances are compared.

I. INTRODUCTION

Traditionally, accurate mathematical model-based strategies have been applied to deal with control problems. However, water level control system, for example, is very complex system, because of the nonlinearities and uncertainties of a system. Conventional control approaches are not convenient to solve the complexities. Fuzzy logic and neural networks control have emerged over the years and become one of the most active and fruitful areas of the research in the intelligent control applications. There are two major different types of the control rules in fuzzy control: the Mamdani type and the Sugeno type. The Mamdani control rules are significantly more linguistically intuitive while Sugeno rules appear to have more interpolation power even for a relative small number of control rules. In neural network control, the most commonly used ones are supervised control, direct inverse control and neural adaptive control.

There are many papers addressed the fuzzy or neural networks control in the water or liquid level control system. Niimura et al. utilized the fuzzy logic for water level control in small hydro-generating units[1]. Roubos et al. chose Sugeno fuzzy model as the model structure for a linear model based predictive control of the liquid level [2]. Ghwanmeh et al. showed a self-learning fuzzy logic control applied to a non-linear process and demonstrated the robust and repeatable performance [3]. Naman et al. presented an adaptive model-reference fuzzy controller for controlling the water level in a water tank [4]. Xiao et al. provided a backpropagation neural network algorithm used to adjust the parameters of the PID controller and control the liquid level of molten steel smelling non-crystalloid [5].

In this paper, we first elaborate the configuration of the water level control system. Then, we introduce Sugeno fuzzy control [7], [8] and model reference adaptive neural network control (MRANNC) [6], [9], [10] strategies based on backpropagation algorithm. Finally, some experimental and real-time results using DSP controller are presented.

II. SYSTEM CONFIGURATION

Figure 1 shows the system configuration. The system consists of the servo motor, valves, a pump, an infra-red sensor and a DSP controller.

III. FUZZY LOGIC CONTROL

In Fuzzy control, two inputs for the system are chosen. They are an error (\( e \)) and an error derivative (\( \dot{e} \)). The error is calculated by taking the difference between the reference
signal and the current water height. The error derivative is calculated by subtracting a previous error from the current error. The output of the system is a voltage that is sent to a servo motor to control the valve 1.

The Sugeno model is used in the system. The Sugeno model is computationally efficient, and works well with optimization and adaptive techniques, so it is popular for control problems, in particular for dynamic nonlinear systems. The Sugeno fuzzy model takes the form:

\[
\text{IF } e \text{ is } A \text{ AND } \dot{e} \text{ is } B, \text{ THEN SERVO is } f(e, \dot{e})
\]

The output function \(f(e, \dot{e})\) takes the form:

\[
A_2 e + A_1 \dot{e} + A_0
\]

The parameters of the output function \(A_2, A_1, \text{ and } A_0\) can be modified through the micro-controller interface.

The rule base used here is shown in Table I. Five membership functions for the error are shown in Figure 2(a), and three for the error derivative are shown in Figure 2(b). The rules mapping is shown in Figure 2(c). Labels in Table I and Figure 2 are as follows: HN=High Negative; N=Negative; S=Small; P=Positive; HP=High Positive;

For the Sugeno model, output functions must be defined. This will decide how the system provides the proper control through the manipulation of the valve 1. Table II shows the output functions and nominal tuning values that the controller uses on start up. Each of these parameters can be modified using the user interface of the micro-controller.

IV. NEURAL NETWORK CONTROL

In this paper, the Model Reference Adaptive Neural Network Control approach based on backpropagation algorithm is applied to implement the water level control system. Figure 3 shows the control structure. TDL in Figure 3 denotes time delays of the input or output signals.

A. Neural Network for Identifier

The neural network for identifier is designed as a three-layer neural network. It has an input layer, a hidden layer, and an output layer. \(N^1, N^2, \text{ and } N^3\) are their output values respectively. The neuron numbers in the hidden layers can be chosen depending on the practical training result.

The neural network identifier models are trained to learn the forward dynamics of the plant. Six inputs and one output are selected as the identifier model for the system. These six inputs are the control signals: \(u(t - i), i = 0...2\), and the previous output signals: \(y(t - j), j = 1...3\). These signals provide the
current and previous two control signals, and also incorporate in the historical trend from the last three time steps in the output of the system. A set of corresponding input-output training patterns is selected from the open-loop continuous signal response of the system. The control input signal is directly added to servo motor of the system and system output signal is reflected with the sensor, which is actual water level.

In training a neural network to learn a forward dynamic model of a plant, the backpropagation error signal between the output and the hidden layer is expressed as

$$\delta_k = T_k - N^i_k$$  \hspace{1cm} (1)

where $T_k$ is the target pattern and $N^i_k$ is the actual output of the identifier, and between the hidden and input layers, it is expressed as

$$\delta_j = f'(net_j) \sum_k \delta_k \cdot w_{kj}$$  \hspace{1cm} (2)

Here, $f'(net_j)$ is the derivative of the activation function $f(net_j)$ where

$$f(net_j) = \frac{1}{1 + exp(-net_j)}$$  \hspace{1cm} (3)

The weights between the input and hidden layers are updated as

$$\Delta w_{ji}(t + 1) = \eta \cdot \delta_j \cdot N^i_j + \alpha \cdot \Delta w_{ji}(t)$$  \hspace{1cm} (4)

and the weights between the hidden and output layer are updated as

$$\Delta w_{kj}(t + 1) = \eta \cdot \delta_k \cdot N^j_k + \alpha \cdot \Delta w_{kj}(t)$$  \hspace{1cm} (5)

where $N^i_j$ and $N^j_k$ are the outputs of the input and hidden layers, respectively. $\eta$ is the learning rate, and $\alpha$ is the momentum coefficients. The constants $\eta$, $\alpha$ are all chosen empirically.

B. Neural Network for Controller

The neural network controller is created directly based on the neural network identifier. Its design is fully incorporated the learning strategy into the trained identifier. The weights of the neural network identifier are constantly verified against the actual plant output. This ensures that the weights allow the neural network identifier to properly predict actual plant output. The neural network identifier is used as means to backpropagate the performance error to get the equivalent error at the output of the neural network controller.

The accuracy of the plant model is critical in ensuring that the controller is accurate as well. The error between the plant output and the identifier output is also checked for the accuracy level of the identifier. This error is used to backpropagate and adjust the weights of the identifier to provide the most accurate representation of the plant. The neural network for controller is also designed as a three-layer neural network. It has a input layer, a hidden layer, and an output layer. $N^i_k$, $N^j_k$ and $N^c_k$ are their output values respectively. The neuron numbers in the hidden layers can be chosen also depending on the practical training result.

$$\delta^i_k = \frac{\partial E^i}{\partial N^c_k} = \frac{\partial E^i}{\partial net^i_k} = \frac{\partial E^i}{\partial N^i_k}$$  \hspace{1cm} (9)

where $N^i_k$ and $N^j_k$ are the input and output of the identifier input layer neurons. Further using chain rule

$$\delta^j_k = \frac{\partial E^i}{\partial net^j_k} \frac{\partial net^j_k}{\partial N^j_k}$$  \hspace{1cm} (10)

Hence,

$$\frac{\partial E^i}{\partial net^j_k} = \delta^j_k$$

Fig. 4. The Connection Between the Controller and the Identifier

To illustrate the derivation of the error signals for the neural network controller, Figure 4 provides the connections between the controller and identifier networks.

The adaptation of the weights of the neural network controller between the hidden and output layers can be derived as follows:

$$\Delta w^c_{kj} = \eta \cdot \frac{\partial E^i}{\partial net^c_k} \cdot \frac{\partial net^c_k}{\partial w^c_{kj}}$$  \hspace{1cm} (6)

where

$$E^i = \frac{1}{2} \cdot (y_p - r)^2$$  \hspace{1cm} (7)

where $y_p$ and $r$ are the actual and desired plant output. The superscript $c$ denotes the variables belonging to the neural network controller and superscript $i$ denotes the variables belonging to the neural network identifier.

Using chain rule, equation (6) can be expanded as follows:

$$\Delta w^c_{kj} = \eta \cdot \delta^c_k \cdot N^c_j$$  \hspace{1cm} (8)

so

$$\Delta w^c_{kj} = \eta \cdot \frac{\partial E^i}{\partial net^c_k} \cdot \frac{\partial net^c_k}{\partial w^c_{kj}}$$

where

$$\delta^c_k = \frac{\partial E^i}{\partial net^c_k}$$

and

$$N^c_j = \frac{\partial net^c_k}{\partial w^c_{kj}}$$

$\delta^c_k$ is the error signal between the hidden and output layers of the neural network controller. Linear functions are used at the input and output neurons of the neural networks between the controller and identifier. Therefore, $\delta^c_k$ can be represented as follows:

$$\delta^i_k = \frac{\partial E^i}{\partial N^c_k} = \frac{\partial E^i}{\partial net^i_k} = \frac{\partial E^i}{\partial N^i_k}$$

Received February 28, 2005.
and
\[ \frac{\partial \text{net}^i_j}{\partial N^c_i} = w^i_j \]
so,
\[ \delta^c_k = \delta^i_j \cdot w^i_j \]  
(11)
The error signal between the input and hidden layer of the neural network controller can be derived as follows:
\[ \delta^c_j = \frac{\partial E^c_j}{\partial \text{net}^c_j} = \frac{\partial E^c_j}{\partial \text{net}^c_k} \cdot \frac{\partial \text{net}^c_k}{\partial \text{net}^c_j} \]  
(12)
By using chain rule
\[ \frac{\partial E^c_j}{\partial \text{net}^c_k} = \sum \frac{\partial E^c_i}{\partial \text{net}^c_k} \cdot \frac{\partial \text{net}^c_k}{\partial \text{net}^c_j} \]
and
\[ \frac{\partial \text{net}^c_j}{\partial \text{net}^c_k} = f'(\text{net}^c) \]
where
\[ f'(\text{net}^c) = N^c_j \cdot (1 - N^c_j) \]
so,
\[ \delta^c_j = N^c_j \cdot (1 - N^c_j) \cdot \left( \sum \delta^c_k \cdot w^c_k \right) \]  
(13)
The weights of the neural network identifier can be further improved online if necessary. This can be reached by back-propagation of the following error equation through the neural network identifier at every sample. \( y_p \) and \( y_i \) are the outputs of the actual plant and neural network identifier, respectively.
\[ e = \frac{1}{2} \cdot (y_p - y_i)^2 \]  
(14)
V. EXPERIMENTAL RESULTS
The neural network control algorithm were first trained and tested with C language with a normal PC. Then, the algorithm were ported to DSP-based controller and tested.

A. Training and Testing for Neural Network Control

1) Parameters Setup: In order to have quicker calculation without sacrificing performance, some parameters were setup as shown in Table III. In Table III, \( n_i, n_j \) and \( n_k \) denote the number of nodes in the input layer, hidden layer and output layer, respectively, \( \eta \) and \( \alpha \) denote learning rate and momentum term, respectively.

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 10 1</td>
<td>10 20 1</td>
</tr>
<tr>
<td>0.00001</td>
<td>0.00005</td>
</tr>
</tbody>
</table>

Table III
PARAMETERS SETUP

2) Identifier Training and Testing: The trained identifier will be used as a representation of the system plant to effectively create a controller for the system. Then, the on-line training of the controller will be feasible to reach.

1. Training Data
An open loop system was used to obtain a system response curve for the plant response to a random input signal. Figure 5 shows the input signal used for training of the NN identifier. This was done by sending a voltage signal through a potentiometer into the servo motor, and it controls the valve 1. By creating a random input control signal and measuring the sensor output while a random and a constant disturbance exists. A set of data was created and indicated the behavior of the plant response to a characteristic input signal. It was designed to fall within the operating range of the sensor. As shown in the figure, the input signal \( u \) varies between values of 2V and 4V.

2. Testing Data
Once the training portion of the Identifier is completed, it needs to be verified with the training results. A selection of similar data was created for testing the trained network. Figure 6 shows a set of testing data used for the Identifier.

3. Identifier Training
Identifier training resulted to a level of error of 7.9842 after 150,000 epochs. At the cut-off of 150,000 epochs, the identifier was still converging on an error, but extremely slowly. Then, this level of an error was used to move onto
controller training. Figure 7 shows the trend of decreasing error in the identifier training.

4. Identifier Testing  
The error seen as a result of testing is shown in Figure 8. Since the level of error seen is on the same scale of that for the training, then moving to controller training became the next step. It is believed that a more rigorous training could result in better error and better performance for controller training and final implementation.

3) Controller Training and Testing: In order to effectively create a controller for the system, the existing system was analyzed in an off-line setup. The goal in the project is to develop a controller through simulation and use more computer resources in a highly effective manner. It proved far more effective to design the training and testing in an off-line format.

1. Training Data  
Figure 9 shows the reference signal used for training of the neural network controller. It was designed to maintain a level within the operating range of the sensors. A random signal was created that would provide the controller with typical values that a user could choose during normal operation. These values were held within the ideal operating range for which the identifier was trained.

2. Testing Data  
Figure 10 shows a set of testing data used for the neural network controller. As can be seen, this pattern is somewhat similar to that seen in the training. As was the case in the identifier, once the training portion of the controller is completed, it needs to verify that training. A set of data was selected for testing the controller training. Since the training data used was designed to be a component of typical operational values, a similar set of data was created for testing the trained network.

3. Controller Training  
In Figure 11, convergence towards the goal of small error is clearly seen in the controller training algorithm. However, the error does not reach the desired cut-off. Despite this, the weights trained as a result of this training provide a very good response when the testing data is used to evaluate the controller weights.

4. Controller Testing  
As shown in Figure 12, the weights for both the identifier and controller had been trained to a level that provided an acceptable level of prediction, and showed that the controller was working as expected.
B. Experiments on micro-controller

The fuzzy and neural network control algorithms were ported to a stand-alone VRP MTSC32 micro-controller. The main board of micro-controller is based on Texas Instruments' TMS320C32 DSP with a CAN bus communication port that facilitates the communication with a PC for monitoring online parameters. The control algorithms were developed with C language, complied with TI Compiler and run from the micro-controller.

To provide an objective comparison between the fuzzy and neural network control algorithms, two tests were carried out in the VRP MTSC32 DSP micro-controller. Figure 13 shows fuzzy control result and Figure 14 shows the neural control result.

VI. CONCLUSION

This paper proposed two control schemes for the water level control system. From the results of the experimental studies, the following summary can be obtained.

For the fuzzy control, in order to ensure the best performance, a number of factors and values need to be online determined heuristically or by trial and error, for example, the membership functions. For neural network control, the learning parameters and prior well-training is essential for the success of the control. Once trained, the neural network does not require for tuning. Table IV shows a summary of comparisons between the two control schemes based on the experimental results.

<table>
<thead>
<tr>
<th>Plant Math Model</th>
<th>FLC</th>
<th>NNC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computation</td>
<td>Light</td>
<td>Heavy</td>
</tr>
<tr>
<td>Tracking Performance</td>
<td>Good</td>
<td>Better</td>
</tr>
<tr>
<td>Disturbance Rejection</td>
<td>Good</td>
<td>Better</td>
</tr>
</tbody>
</table>

TABLE IV
COMPARISON BETWEEN THE FUZZY CONTROL AND NEURAL NETWORK CONTROL BASED ON THE EXPERIMENTAL RESULTS

REFERENCES

Received February 28, 2005.