Implementation of Neural Network Based Control Scheme on the Benchmark Conical Tank Level System

A project report submitted for the partial fulfillment of the Bachelor of Technology Degree in Electronics & Communication Engineering under Maulana Abul Kalam Azad University of Technology

BY

Sabarna Ghosh Dastidar
(ROLL NO: 10400319068, REGISTRATION NO: 029431 of 2019-20)

Under the Guidance of: **Prof. Dr. Atanu Panda**

Department of **Electronics and Communication Engineering**

For the Academic Year 2019 – 23

Institute of Engineering & Management

Y-12, Salt Lake, Sector-V, Kolkata-700091

Affiliated To:

Maulana Abul Kalam Azad University of Technology

BF-142, Salt Lake, Sector I, Kolkata-700064

CERTIFICATE

TO WHOM IT MAY CONCERN

This is to certify that the project report entitled "**Implementation of Neural Network Based Control Scheme on the Benchmark Conical Tank Level System**", submitted by

1. Sabarna Ghosh Dastidar (*Registration No.* 029431 *of 2019-20 Roll no. 10400319068)*,

Students of **INSTITUTE OF ENGINEERING & MANAGEMENT,** in partial fulfilment of requirements for the award of the degree of **Bachelor of Technology in Electronics and Communication**, is a bona fide work carried out under the supervision and guidance of **Prof. Dr. Atanu Panda** during the final year of the academic session of 2019-2023.The content of this report has not been submitted to any other University or Institute for the award of any other degree.

It is further certified that work is entirely original and its performance has been found to be quite satisfactory.

Prof. Dr. Atanu Panda	Prof. Dr. Malay Gangopadhyay
Project Guide	H.O.D
Dept. of Electronics & Communication	Dept. of Electronics & Communication
Institute of Engineering & Management	Institute of Engineering &Management

Prof. Dr. Arun Kumar Bar
Principal
Institute of Engineering & Management
Sector-V, Salt Lake Electronics Complex, Kolkata-700091

ACKNOWLEDGEMENT

We should like to take this opportunity to extend our gratitude to the following revered persons without whose immense support, completion of this project wouldn’t have been possible.

We are sincerely grateful to our advisor and mentor **Atanu Panda** of the **Electronics and Communication**, IEM Kolkata, for his/her constant support, significant insights and for generating in us a profound interest for this subject that kept us motivated during the entire duration of this project.

We would also like to express our sincere gratitude to **Prof. Dr. Satyajit Chakrabarti** (Director**,** IEM)**, (**Principal, IEM) and **Prof. Dr. Malay Gangopadhyay**, HOD of **Electronics and Communication** and other faculties of Institute of Engineering & Management, for their assistance and encouragement.

Last but not the least, we would like to extend our warm regards to our families and peers who have kept supporting us and always had faith in our work.

Sabarna Ghosh Dastidar
Reg. No: 029431 of 2019-20
Dept. of Electronics and Communication
Institute of Engineering & Management, Kolkata

ABSTRACT

In the field of process industries, it became a very challenging problem to control the inlet flow of liquid and maintain a desired level of liquid at a constant value in a tank. In this paper, the discussion has been done on the procedure of designing a controller for a conical tank-level system. The modeling for a conical tank system is a little bit more complex than other linear tank systems (e.g. cubical, cylindrical, etc.) due to the change of its cross-sectional area from the bottom to top but it provides complete drainage as a result of it. Here the feed-forward back-propagation neural network is used to implement the architecture of the controller and then the network is trained using the Levenberg-Marquardt algorithm to control the inlet flow of the liquid in order to maintain the desired level of liquid in the tank at a constant value. From the simulation, it is observed that the neural network controller provides good results while performing the servo and regulatory response of the system.

TABLE OF CONTENTS

1. INTRODUCTION

The control of liquid level in tanks and flow between the tanks is a basic problem in process industries. The process industries require the liquids to be pumped or stored in tanks and then transfer to other tanks. Many times the liquid will be processed by chemical or mixing treatment in tanks, but always the level of the liquid in the tank must be controlled. It becomes more challenging when the process becomes nonlinear. A conical tank level system is considered here for the research work because this is a highly nonlinear process with varying cross-sectional area where its area gets steeper towards the end for the guaranteed drainage of fluids [5] in chemical industries. Conical tanks find wide applications in process industries such as food processing industries, petrochemical industries, and sewage and wastewater treatment industries. So it becomes a very challenging task to control the liquid flow in a conical tank in order to obtain the desired response.

The majority of the control theory deals with the design of linear controllers for linear systems. PID controller proved to be a perfect controller for simple and linear processes. But when it comes to the control of nonlinear and multi-variable processes, the controller parameters have to be continuously adjusted. Non-linear adaptive control strategies [2] are becoming very much useful here by adjusting the control parameters of a feedback control system in real-time based on changes in the system or its environment. The main significance of adaptive control is that it can provide improved performance and robustness in systems that are subject to changes or uncertainties. So, these methods are very effective to compute an approximated input-output model from a large number of given input and output data sets. As the process is adaptive, it updates the controller parameters in real-time to better respond to the variation of the dynamic behavior of a process due to sudden disturbance or plant parameter variation, or model uncertainties.

In this work, the neural network is used as a significant part of the proposed control scheme. This approach leverages the power of machine learning techniques to approximate the nonlinear behavior of the conical tank system [7] and adaptively control the liquid level in real-time. The actual output of the process is compared to the desired response obtained from a reference model, and based on the calculated error, the controller parameters are tuned so that the process output reaches the desired response within a finite time, even in the presence of external disturbances.

2. OBJECTIVES

The objective of this research paper on "Implementation of Neural Network Based Control Scheme on Benchmark Conical Tank Level System" is to demonstrate the effectiveness of a neural network-based control scheme for liquid level control in the process control industry. The paper aims to present the mathematical model of the Benchmark Conical Tank Level System and the implementation of a neural network-based control scheme. The paper also aims to compare the performance of the neural network-based control scheme with traditional control methods through simulation and experimental results. The overall objective is to provide a more accurate and robust control system that can handle system nonlinearities and disturbances, and to improve liquid level control in the process control industry.

3. OVERVIEW OF CONICAL TANK LEVEL SYSTEM

3.1. System Description

The schematic diagram of a conical tank level system is shown in Fig. 1. Here, the input of the system is the inlet liquid flow of the tank and the output is the liquid level of the tank which should be maintained at a constant value. This desired response can be obtained by continuously manipulating the inlet flow of liquid. Thus, the inlet flow rate is referred to as the manipulated variable, while the liquid level is known as the process variable in this system. By manipulating the inlet flow rate of liquid, the control system can maintain the liquid level at the desired set point, ensuring optimal performance of the process.

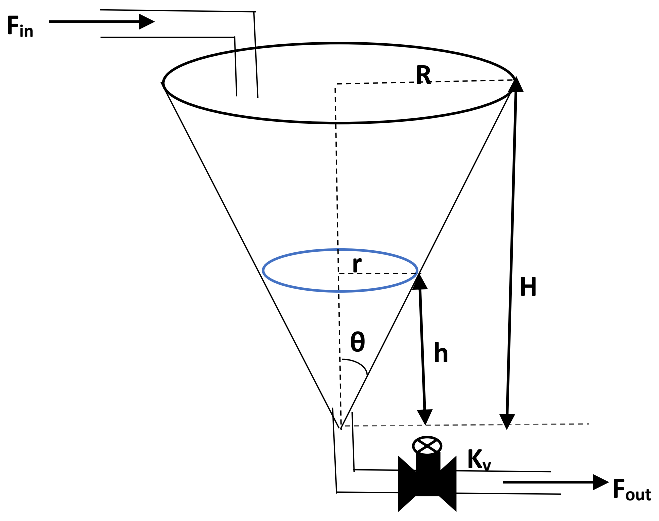

Fig. 1. Conical Tank Level System

Fin – the rate of inlet flow in the conical tank

Fout – rate of outlet flow in the conical tank

Kv – outlet valve coefficient

R – maximum radius of the conical tank

H – maximum height of the conical tank

r – the radius of the water level in the tank

h – the height of the water level in the conical tank

3.2. *Mathematical Model Implementation*

Now, Fig. 1, we can say that, $\tan\theta = \frac{r}{h} = \frac{R}{H}$ -------- (1)

Now, if we assume V as the volume of the liquid level in the tank, then we can say that the rate of accumulation of the liquid in the tank can be expressed as,

$\frac{dV}{dt} = F_{in} - F_{out}$ --------- (2)

Now, we know that the volume of the liquid in a conical tank can be expressed as,

$V = \frac{1}{3}\pi r^2 h$ ------------ (3)

Now, from equation (1), we can say, $r = \frac{R}{H}h$

So, putting the value of r in equation (3), we, get,

$$V = \frac{1}{3}\pi(\frac{R}{H}h)^2 h$$

Or, $V = \frac{\frac{1}{3}\pi R^2}{H^2} h^3$

Differentiating on both sides, we get,

$$\frac{dV}{dt} = \frac{1}{3}\pi\frac{R^2}{H^2} . 3h^2 \frac{dh}{dt}$$

Or, $\frac{dV}{dt} = \frac{\pi R^2}{H^2} . h^2 \frac{dh}{dt}$

Now, putting the value of $\frac{dV}{dt}$ in equation (2), we get,

$F_{in} - F_{out} = \frac{\pi R^2}{H^2} . h^2 \frac{dh}{dt}$ ----------- (4)

Now, the rate of outlet flow [1] can be expressed as, $F_{out} = K_v\sqrt{h}$

So, from equation (4), we get,

$F_{in} - K_v\sqrt{h} = \frac{\pi R^2}{H^2} . h^2 \frac{dh}{dt}$

Or, $\frac{dh}{dt} = \frac{1}{h^2} . \frac{H^2}{\pi R^2} . (F_{in} - K_v\sqrt{h})$ ----------- (5)

3.3. Simulink Diagram

Now, according to equation (5), we built our model of conical tank system in MATLAB Simulink, which is shown in the figure below,

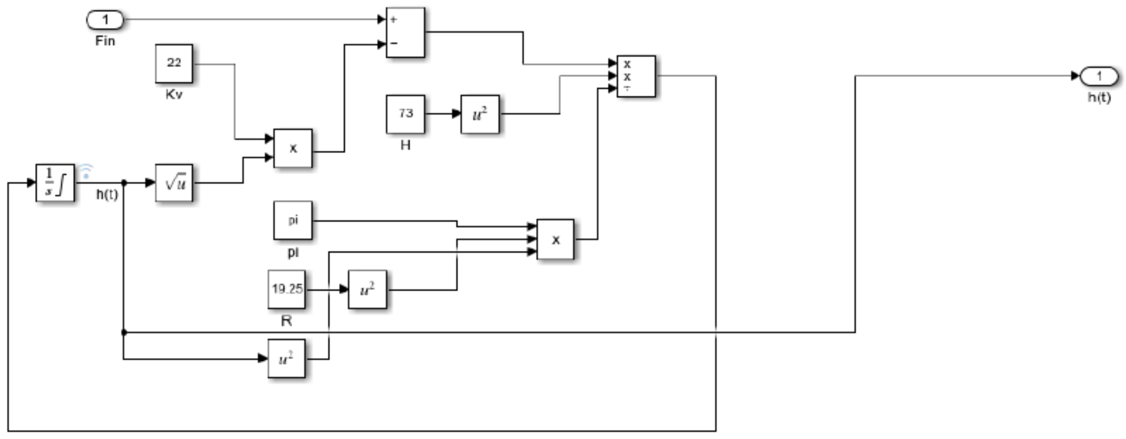

Fig. 2. Implementation of conical plant model in simulink

In the above figure, we took the specifications for our conical tank system as follows,

TABLE I. INTERNAL SPECIFICATIONS OF CONICAL TANK LEVEL SYSTEM

Name of the Variables	**Values**
Range of inlet flow (F_{in})	0-220 LPH
Outlet valve coefficient (K_v)	22
Height of the tank (H)	73 cm
Radius of the tank (R)	19.25 cm
Range of liquid level (h)	10-60 cm

4. REFERENCE MODEL IMPLEMENTATION USING A CONVENTIONAL CONTROLLER

4.1. Introduction to PI Controller

A proportional-integral (PI) controller [9] is a type of feedback control system commonly used in industrial processes to regulate a process variable such as temperature, pressure, or flow rate. In our case, we will be regulating a flow rate. The PI controller combines two control actions, proportional control, and integral control, to provide stable and accurate control of the process variable.

Proportional control is a control action that adjusts the output of the controller proportional to the error between the process variable and the desired setpoint. The error is the difference between the desired setpoint and the actual process variable. Proportional control alone can cause oscillations and instability in the control system.

Integral control is a control action that adjusts the output of the controller based on the integral of the error over time. Integral control reduces the steady-state error in the control system and improves the stability of the system. The integral control action also eliminates the offset between the setpoint and the process variable.

The PI controller combines proportional control and integral control to provide stable and accurate control of the process variable. The controller output is the sum of the proportional control action and the integral control action.

The PI controller is commonly used in industrial processes due to its simplicity, ease of implementation, and robustness. It can provide stable and accurate control of the process variable even in the presence of disturbances and uncertainties in the system. The PI controller is widely used in various applications such as temperature control, pressure control, level control, and flow control in various industries such as chemical, petrochemical, and manufacturing.

4.2. Reference Model Implementation using PI Controller

The reference model implementation using a PI controller is a technique used in the implementation of neural network-based control schemes on a benchmark conical tank level system. The reference model is a mathematical model that represents the desired behavior of the system. It is used as a reference for the control system to follow in order to achieve the desired output.

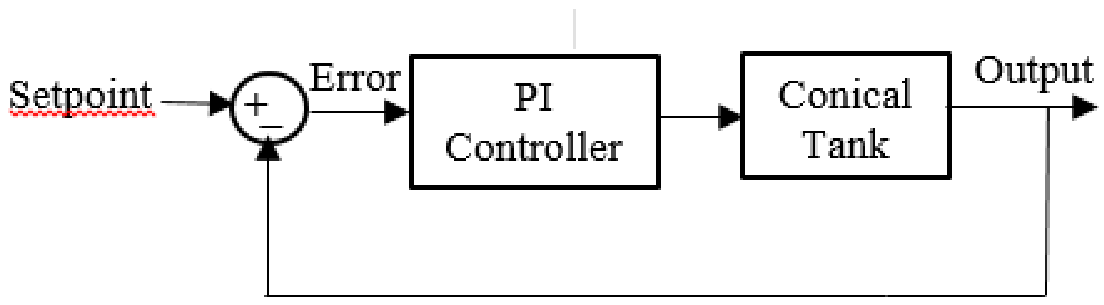

Fig. 3. Block Diagram of Conventional PI Controller

In the above figure, we used the most conventional controller i.e. the PI controller to create the dataset of desired water level of the tank corresponding to the inlet water flow in the tank. We tuned our PI controller to get the desired control flow in order to get the desired water level set by the setpoint.

The PI controller is used to regulate the process variable to follow the reference model. The controller adjusts the output based on the difference between the process variable and the reference model. The PI controller combines the proportional control action and the integral control action to provide stable and accurate control of the process variable.

The implementation of the reference model using a PI controller is essential to ensure that the control system follows the desired behavior of the system. It helps to eliminate steady-state error and improve the stability of the system. The implementation of the reference model using a PI controller can be used in various industrial processes such as chemical and petrochemical industries, where precise control of the process variable is essential.

The Simulink implementation is shown in the following figure,

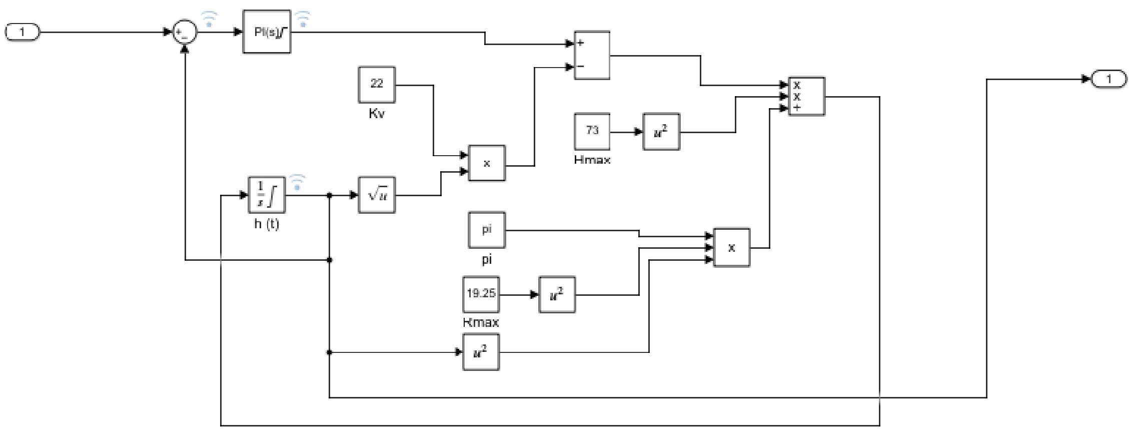

Fig. 4. Reference Model Implementation of Conical Tank using PI Controller

4.3. Response of Reference Model

The reference model is a mathematical model that represents the desired response of the system, and it is used as a reference for the control system to follow. In the context of the benchmark conical tank level system, the reference model could represent the desired behavior of the liquid level in the tank, such as maintaining a constant level, following a certain profile or trajectory, or responding to changes in the input or disturbances in the system.

The response of the reference model can be analyzed using various techniques such as mathematical modeling, simulation, or experimental testing. Once the reference model's response is determined, a PI controller can be designed and implemented to regulate the process variable to follow the reference model.

The PI controller adjusts the output based on the difference between the process variable and the reference model, and its response will depend on the accuracy of the reference model and the tuning of the PI controller parameters. The overall response of the system will also depend on other factors such as the system's dynamics, input signals, and disturbances.

We used Ziegler and Nichols method here to tune the controller parameters here and we found proportional gain, Kp = 1.3950 and Integral gain, Ki = 0.0042. So, using these values we tune our PI controller and the output response is shown in the figure below,

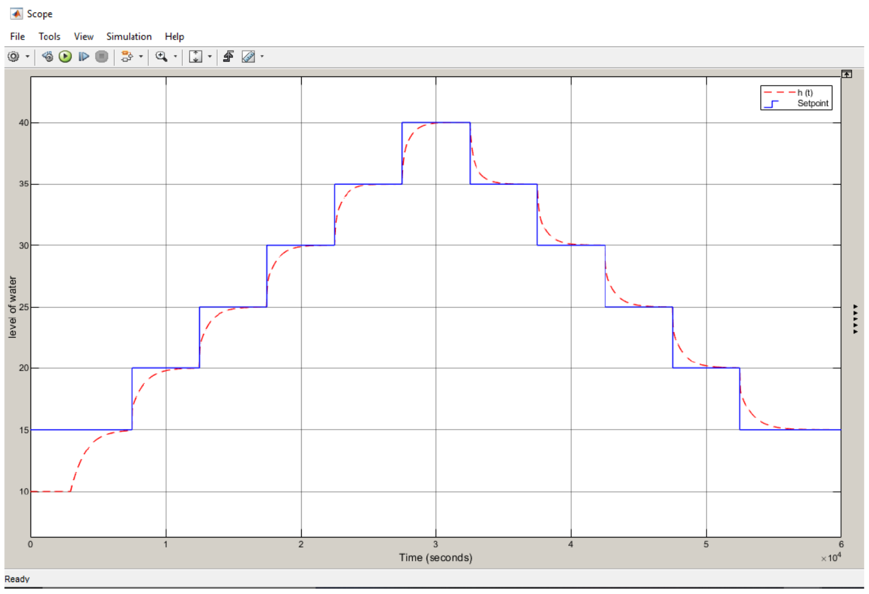

Fig. 5. Desired output response of Reference model

5. IMPLEMENTATION OF NEURAL NETWORK BASED CONTROL SCHEME

5.1. Introduction to Neural Network

Neural networks are a type of machine learning model that is inspired by the structure and functioning of the human brain. They are powerful computational models composed of interconnected nodes, called artificial neurons or nodes, organized into layers.

Neural networks have gained significant popularity and have become a fundamental tool in various fields such as computer vision, natural language processing, speech recognition, and many others. The basic building block of a neural network is the artificial neuron, also known as a perceptron. It receives one or more inputs, applies weights to those inputs, performs a mathematical operation on the weighted inputs, and produces an output. The output is usually passed through an activation function, which introduces non-linearity into the network. This non-linearity allows neural networks to model complex relationships between inputs and outputs. Neurons are organized into layers in a neural network. The input layer receives the raw input data, which is then passed through one or more hidden layers. The final layer is the output layer, which produces the network's predicted output. The layers between the input and output layers are referred to as hidden layers because they are not directly connected to the external environment. The connections between the neurons are represented by weights, which determine the strength of the connections. During the training process, these weights are adjusted iteratively to minimize the error between the predicted output and the desired output. This training process is typically done using optimization algorithms such as gradient descent.

5.2. Designing of the Controller Architecture

The neural network is an adaptive system that processes information using a connectionist approach. It consists of artificial neurons that are interconnected and its structure changes based on external or internal information that flows through the network during the learning phase. The design of a neural network is fully incorporated into the learning strategy of the trained identifier. The weights of the neural network identifier are constantly verified against the actual plant output to ensure proper prediction. The feed-forward back-propagation neural network is used as the network architecture here. It consists of one input layer, one hidden layer, and one output layer. It processes information in a unidirectional manner, from the input layer through hidden layers to the output layer, without any cycles or loops [2].

For the training of the network, the dataset obtained from the simulation of the reference model is imported into the network. From that dataset, the values of the generated errors and the previous plant outputs are imported as the input dataset and the values of the present control flow are imported as the target dataset to train the network. Hence, two neurons are taken in the input layer and one neuron is taken in the output layer of the architecture. Using six neurons in the hidden layer, a better training result is observed. In the hidden layer, the hyperbolic tangent sigmoid transfer function (tansig) is used as the activation function and the linear transfer function (purelin) is used in the output layer.

The implementation of the internal architecture of the network is shown in Fig. 6. The main concept of this architecture is to use a back-propagation algorithm, which involves adjusting the weights of the neurons in each layer based on the difference between the actual output and the desired output. The aim is to minimize the error between the actual and desired outputs, which is accomplished through multiple iterations of training.

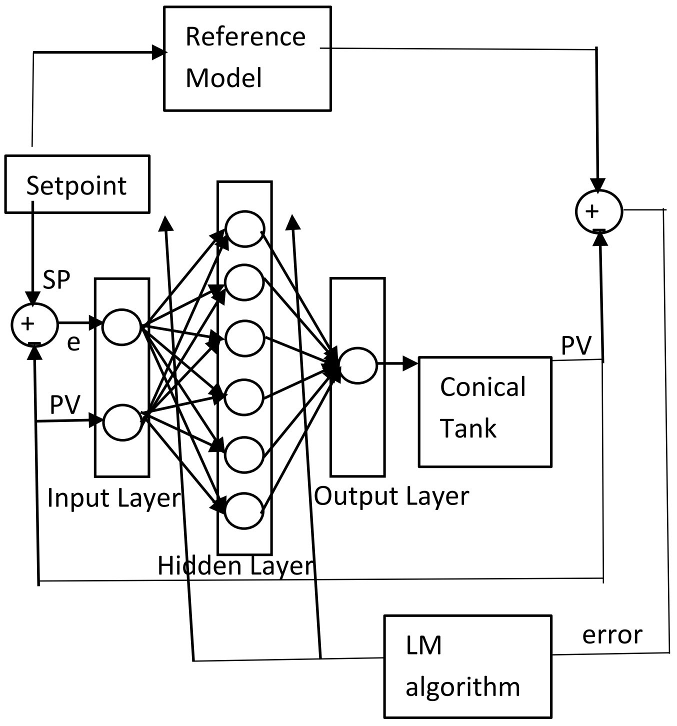

Fig. 6. Architecture of Neural Network based Model

5.3. Working Principle of Neural Network

Neural networks are composed of interconnected neurons, as depicted in Fig. 6, where each connection between neurons represents the flow of data. Each connection has a weight associated with it, which determines the signal between the two neurons. If the network output is accurate, the weight values need not be changed. However, if the output does not match the expected output, an error is calculated as the difference between the expected and actual outputs. This error is used to adapt the network and improve its performance in the next iteration.

Each neuron in the hidden and output layers can be modeled using a perceptron [6], which processes inputs to generate an output. The perceptron algorithm multiplies each input by its weight, adds the results, and applies an activation function to calculate the output. The activation function can be defined as the sign of the sum obtained, where a positive sum produces an output of 1 and a negative sum produces an output of -1. To handle cases where both inputs are zero, a bias input with a value of 1 is added.

The neural network compares its output with the expected output for each input and learns from mistakes by adjusting the weights of the perceptron. The process involves giving different inputs to the perceptron with known answers, guessing the value, calculating the error, adjusting the weight based on the error, and repeating the process. The error, defined as the difference between the expected and actual outputs, is the critical factor that determines how the perceptron weights should be varied. By continuously adapting and improving based on errors, the neural network can learn to accurately predict outputs.

The above concept is discussed below in a more detailed way:

Data Input: The neural network begins by receiving input data. This could be raw data such as images, text, or numerical values. The input data is typically pre-processed and normalized to ensure it is in a suitable format for the network.

Neuron Activation: The input data is then passed to the first layer of neurons, known as the input layer. Each neuron in the input layer receives the input data and applies weights to the inputs. The weighted inputs are then summed, and an activation function is applied to the sum to introduce non-linearity. This output is then propagated to the next layer.

Forward Propagation: The processed output from the previous layer serves as the input to the next layer. This process of passing information forward through the layers is known as forward propagation. The hidden layers in the network apply weights to their inputs, sum them up, and pass the result through an activation function.

Output Layer: The final layer of the neural network is the output layer. It produces the predicted output based on the processed information from the previous layers. The number of neurons in the output layer depends on the specific task the neural network is designed for. For example, in a binary classification task, there might be one neuron representing the probability of one class and another neuron representing the probability of the other class.

Loss Calculation: Once the predicted output is obtained, it is compared to the desired output or target value. The difference between the predicted output and the target value is quantified using a loss function. The loss function measures the error or the mismatch between the predicted and desired outputs.

Backpropagation: The next step is to update the weights in the neural network to minimize the loss. This is done through a process called backpropagation. Backpropagation involves calculating the gradients of the loss function with respect to the network's weights. The gradients indicate the direction and magnitude of the weight adjustments needed to reduce the loss.

Weight Update: The calculated gradients are then used to update the weights of the neural network. Optimization algorithms, such as gradient descent, are commonly employed to iteratively adjust the weights based on the gradients. The learning rate, which determines the step size during weight updates, is an important parameter that affects the convergence and training speed of the neural network.

Iterative Training: Steps 3 to 7 are repeated for multiple iterations or epochs. During each iteration, the network processes the training data, calculates the loss, performs backpropagation to update the weights, and refines the model's predictions. This iterative training process allows the network to learn and improve its performance over time.

Prediction: Once the neural network is trained, it can be used to make predictions on new, unseen data. The input data is passed through the network's layers using the learned weights, and the final output represents the network's prediction for the given input.

By adjusting the weights and biases of the neurons through training, a neural network can learn complex patterns and relationships in the input data, enabling it to make accurate predictions or classifications. The working principle of a neural network leverages this iterative training process to optimize its performance and generalize well to unseen data.

5.4. Levenberg-Marquardt Training Algorithm

Levenberg-Marquardt (LM) back-propagation algorithm [3] is used here to train the network. This algorithm is fast, has a very stable convergence, and provides a numerical solution for the problem of minimizing a non-linear function. This algorithm is quite a simple but extremely robust method for function approximation. It is a combination of the steepest descent algorithm and the Gauss-Newton algorithm that can converge quickly and stably for small and medium-sized problems. The LM algorithm is a numerical optimization technique that is widely used for solving non-linear least squares problems. It is a modification of the Gauss-Newton method that adds a damping parameter to improve the stability and convergence of the algorithm. This damping parameter enables the LM algorithm to handle ill-conditioned problems that the Gauss-Newton algorithm may fail to converge. This is a step-by-step overview of how the Levenberg-Marquardt training algorithm works:

Step 1: Initialize Parameters: Start by initializing the weights and biases of the neural network with small random values. These initial parameters define the initial state of the network.

Step 2: Forward Propagation: Perform forward propagation to compute the predicted output of the neural network for a given input. This involves passing the input through the network's layers and applying the activation functions.

Step 3: Compute Error: Calculate the error or the mismatch between the predicted output and the desired output using a suitable error metric, such as mean squared error.

Step 4: Compute Jacobian: Compute the Jacobian matrix, which represents the partial derivatives of the network's output with respect to its parameters (weights and biases). The Jacobian matrix provides information about how changes in the parameters affect the output.

Step 5: Compute Hessian: Calculate the Hessian matrix, which represents the second-order partial derivatives of the error function with respect to the parameters. The Hessian matrix captures information about the curvature of the error surface.

Step 6: Update Parameters: Update the parameters of the neural network using the Levenberg-Marquardt update rule. The update rule adjusts the parameters based on the error, Jacobian, and Hessian matrices. The update considers a trade-off between the Gauss-Newton method, which assumes a linear model, and the steepest descent method, which ensures more stable convergence.

Step 7: Check Convergence: Check if the training process has converged by evaluating a convergence criterion. This criterion could be based on the change in the error or the norm of the parameter updates. If the convergence criterion is met, the training process is considered complete. Otherwise, continue to the next iteration.

Step 8: Iterative Training: Repeat steps 2 to 7 until the convergence criterion is satisfied. During each iteration, the parameters of the neural network are updated using the LM update rule, and the error is evaluated to determine convergence.

The Levenberg-Marquardt algorithm efficiently combines the benefits of the Gauss-Newton method, which converges quickly in the vicinity of the minimum, and the steepest descent method, which provides more robust convergence. This makes it particularly suitable for training neural networks, which often involve non-linear optimization problems with complex error surfaces.

The mathematical expression of this algorithm can be given as:

$$(JJ^T + \lambda I)\ \delta = J^T E$$

Where J represents Jacobian matrix, λ is Levenberg damping factor, δ is the update vector of weight and E represents the error vector. The damping factor λ is adjusted for every iteration. The Jacobian matrix can be given as:

$$J = \begin{bmatrix} \frac{\partial F(x_1,w)}{\partial w_1} & \cdots & \frac{\partial F(x_1,w)}{\partial w_N} \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \frac{\partial F(x_N,w)}{\partial w_1} & \cdots & \frac{\partial F(x_N,w)}{\partial w_N} \end{bmatrix}$$

The Jacobian matrix J is a matrix of partial derivatives that describes the sensitivity of the output of the network to changes in its weights. The above equation shows the general form of the Jacobian matrix. Here, $F(x_i, w)$ represents the network function, and w is the weight vector of the network. The Levenberg damping factor λ is adjusted for every iteration to ensure that the algorithm converges to the optimal solution. If λ is too small, the algorithm may converge slowly or even diverge. If λ is too large, the algorithm may overshoot the optimal solution.

6. LYAPUNOV STABILITY ANALYSIS

Lyapunov Stability Analysis [10] is a mathematical technique used to determine the stability of a dynamical system. It is a powerful tool used in various fields such as control systems, robotics, and artificial intelligence. The Lyapunov Stability Analysis involves analyzing the behavior of a dynamical system by studying the properties of a Lyapunov function. A Lyapunov function is a scalar function that assigns a value to each point in the state space of the system, and it is used to measure the system's stability.

The Lyapunov Stability Analysis can be used to determine the stability of both linear and nonlinear systems. In the case of linear systems, the stability analysis involves finding the eigenvalues of the system's matrix. If all the eigenvalues have negative real parts, the system is stable.

In the case of nonlinear systems, the stability analysis involves finding a Lyapunov function that satisfies certain conditions. If a Lyapunov function can be found that satisfies these conditions, the system is stable. The Lyapunov Stability Analysis can also be used to determine the stability of a system's equilibrium points, which are the points where the system's derivative is zero.

The Lyapunov Stability Analysis is an important technique used in control systems to design controllers that ensure system stability. It is also used in robotics to analyze the stability of robot motion and in artificial intelligence to analyze the stability of neural networks. In the context of a conical tank level system, the Lyapunov Stability Analysis can be used to determine the stability of the system's liquid level control. The analysis involves finding a Lyapunov function that measures the stability of the system's equilibrium point, which is the point where the liquid level is constant.

The Lyapunov Stability Analysis can also be used to design controllers that ensure the stability of the system's liquid level control. The controllers can be designed to maintain a constant liquid level or follow a certain trajectory while ensuring the system's stability.

Thus the Lyapunov Stability Analysis is an important technique used in the design and implementation of control systems for conical tank level systems. It helps ensure the stability and reliability of the liquid level control and is essential in industries such as chemical and petrochemical, where precise liquid level control is critical for efficient and safe operations.

7. RESULTS AND DISCUSSION

The architecture of the neural network model is made in Matlab-Simulink software and the simulation is performed using automated training commands available in Matlab library. After the training is completed, a sequence of step inputs is given to the system as the setpoint, and the performance of the conical tank system and the neural network controller is checked for each of the responses discussed below and it is clearly observed that the controller is providing satisfactory results while tracking the setpoint. The training results are discussed below,

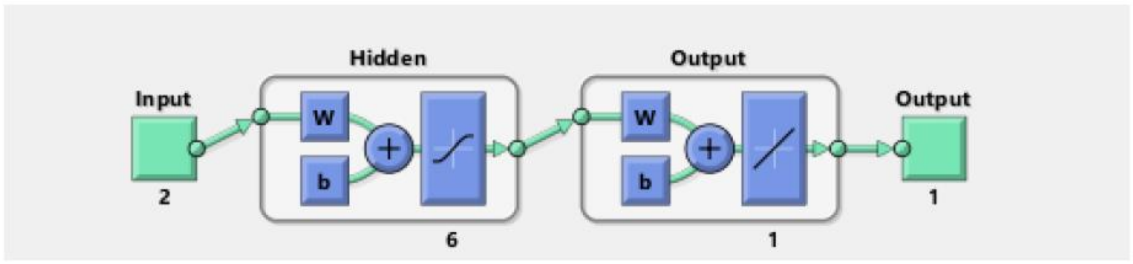

Fig. 7. Designing of architecture of Neural Network Controller in Simulink

Following are some figures related to the obtained training results,

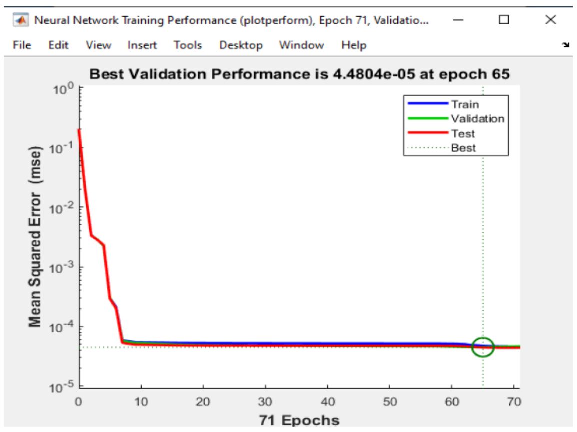

Fig. Training, Validation and Testing Performance

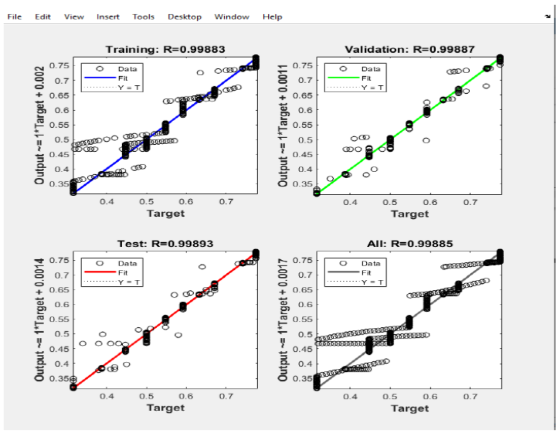

Fig. Regression

7.1. SERVO RESPONSE

Setpoint is set at 10 cm for the first 1000 seconds, then it is set at 47 cm for the next 2000 seconds and then it is set again at 23 cm for the next 2000 seconds and the output of the neural network model satisfactorily follows the setpoint. The tracking response is shown in Fig. 9 and the controller output response is shown in Fig. 10. The variation of weights in the hidden layer is shown in Fig. 11.

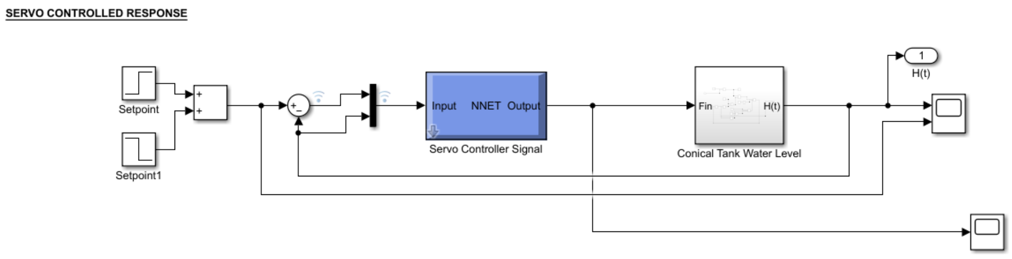

Fig. 8. Neural Network Model Implementation for Servo Control

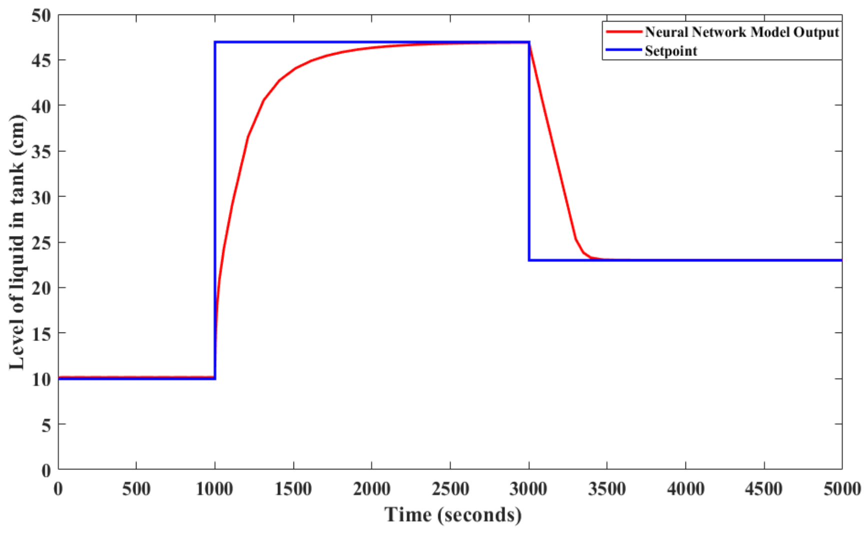

Fig. 9. Servo response of process variable

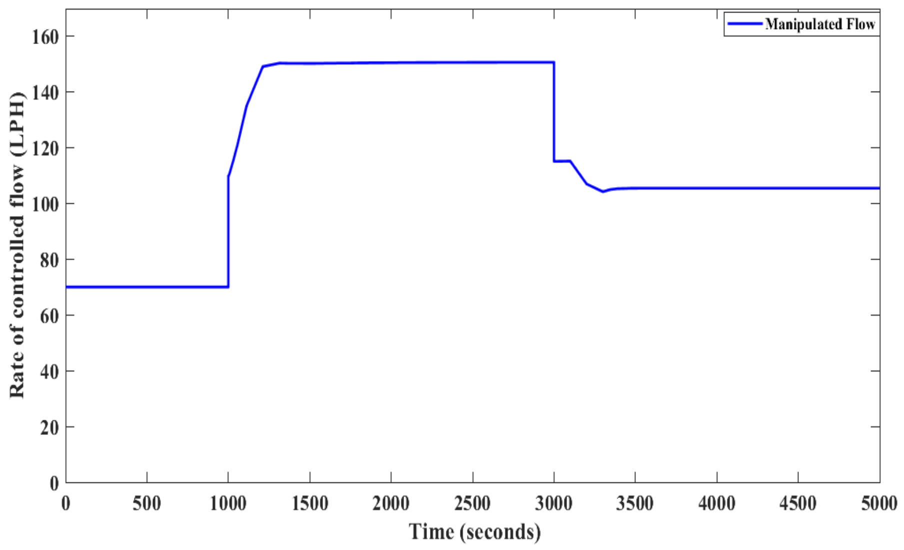

Fig. 10. Servo response of controlled variable

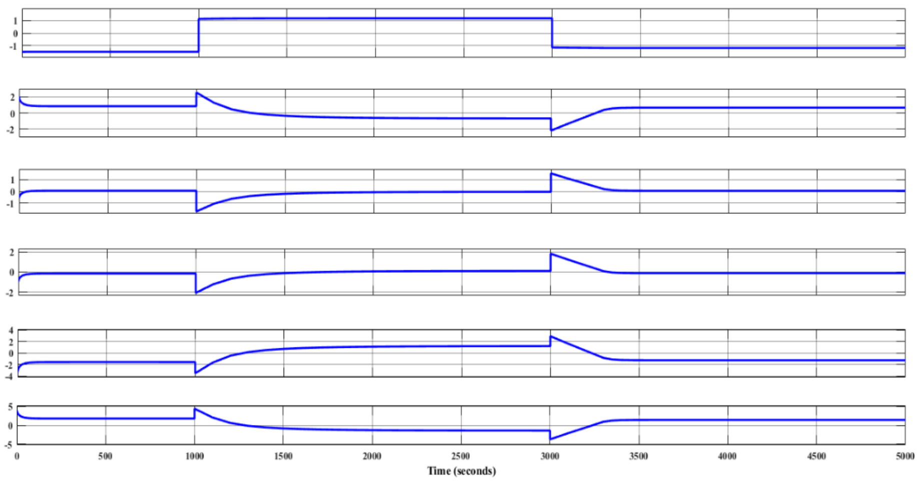

Fig. 11. Update of weights in hidden layer in servo response

7.2. *REGULATORY RESPONSE*

Setpoint is set at 10 cm for the first 100 seconds and then it is set at 27 cm for the total simulation time but during the simulation, at time, t = 2000 sec, a disturbance is added by reducing the flow by 20 LPH for a time duration of 60 seconds and at time, t = 3500 sec, another one disturbance is produced by increasing the flow by 20 LPH again for a time duration of 60 seconds. It is clearly observed that the output of the neural network model satisfactorily handles the disturbance and follows the setpoint. The tracking response is shown in Fig. 13 and the controller output response is shown in Fig. 14. The variation of weights in the hidden layer is shown in Fig. 15.

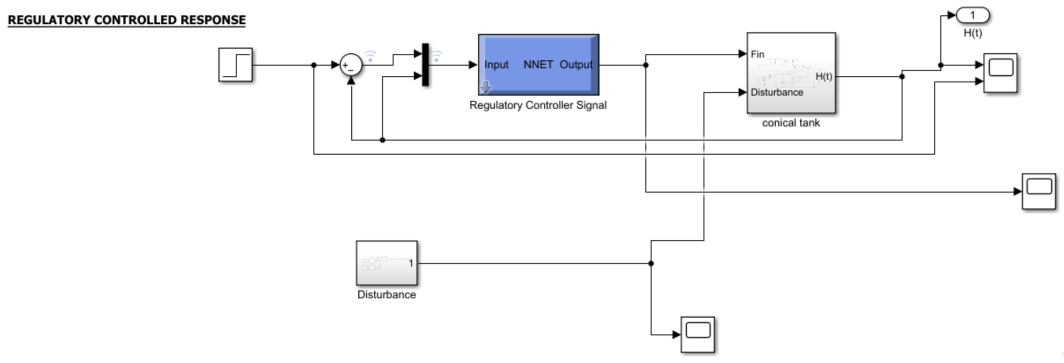

Fig. 12. NN regulatory control model

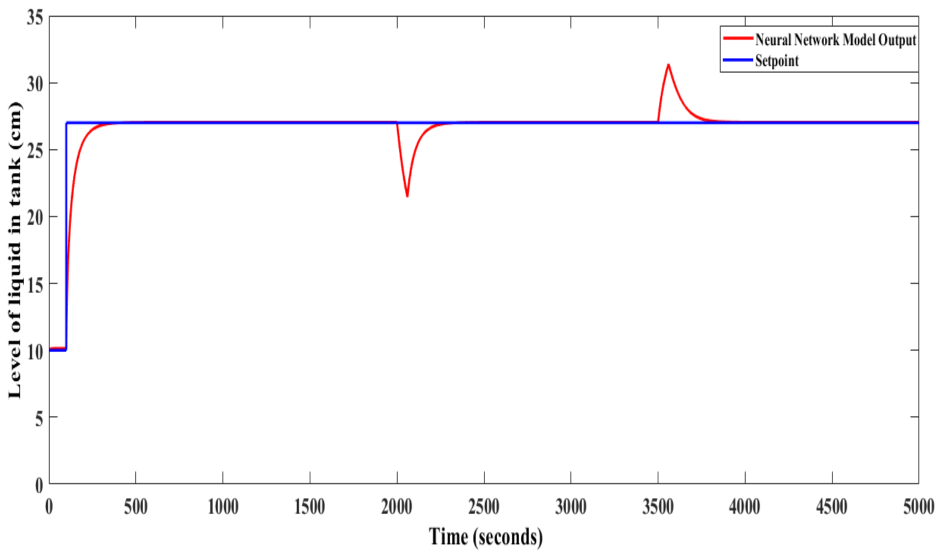

Fig. 13. Regulatory response of process variable

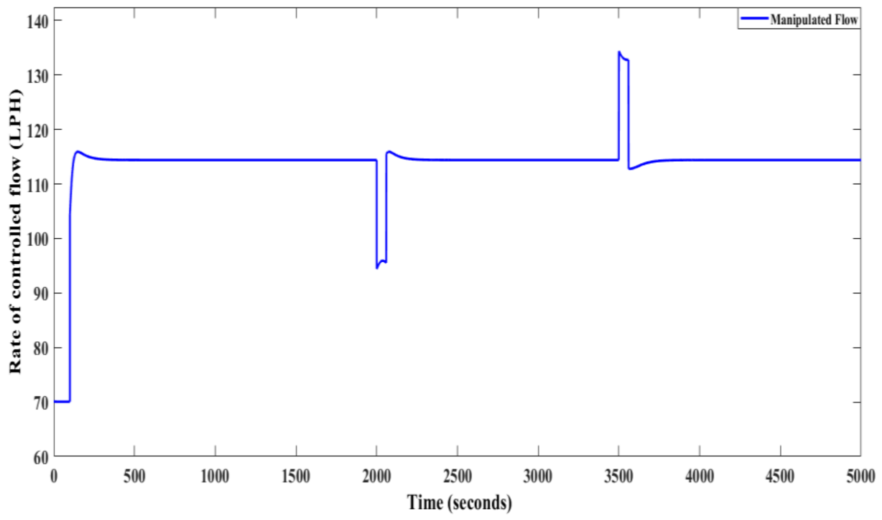

Fig. 14. Regulatory response of controlled flow

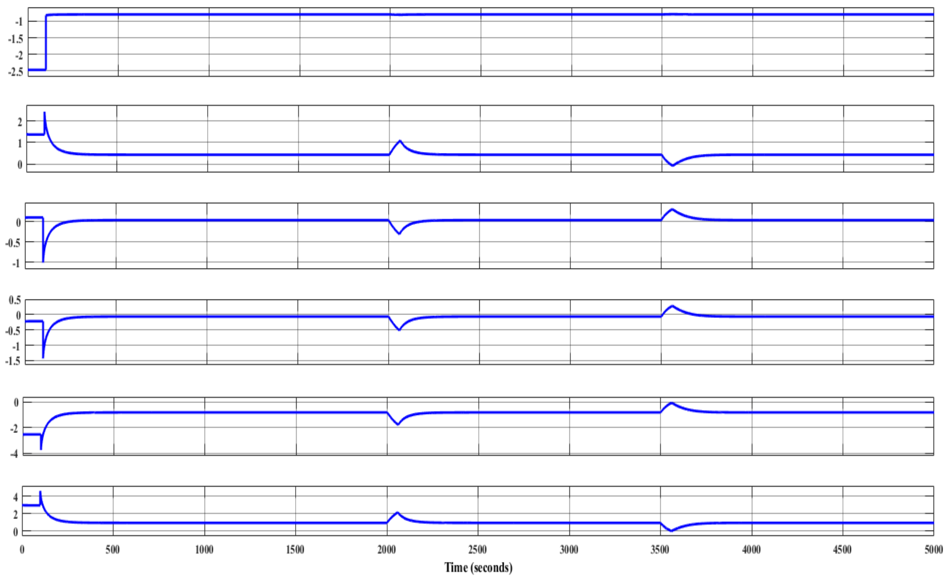

Fig. 15. Update of weights in hidden layer in regulatory response

7.3. PARAMETRIC VARIATION

Setpoint is set at 27 cm and two pulse disturbances are added at time, t =1500 sec and at time, t = 3500 sec respectively to vary the outlet valve coefficient K_v. The first one is having the amplitude = 4 and pulse width = 100 sec. The second one is having the amplitude = -4 and pulse width = 100 sec. It is clearly observed that the output of the neural network model satisfactorily handles the disturbance and follows the setpoint. The variation of K_v is shown in Fig. 16. The tracking response is shown in Fig. 17 and the controller output response is shown in Fig. 18. The variation of weights in the hidden layer is shown in Fig. 19.

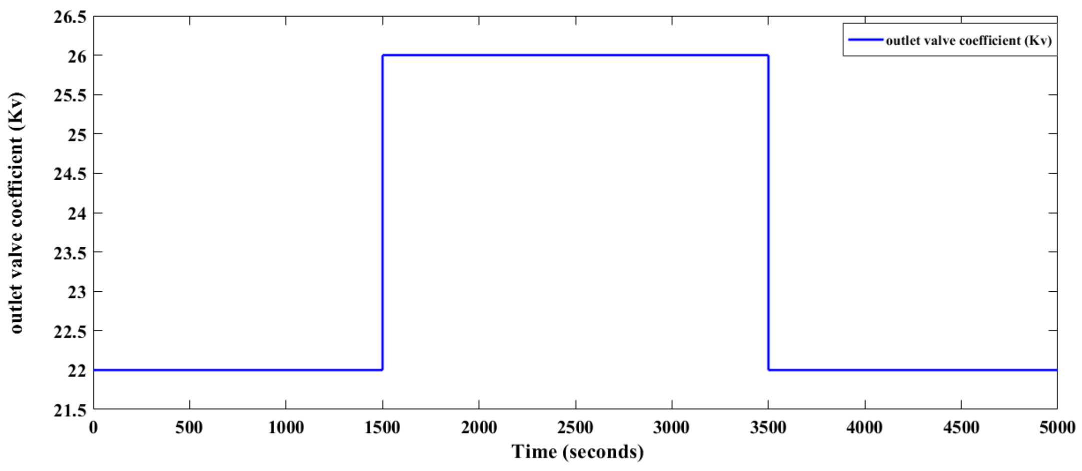

Fig. 16. Variation of outlet valve coefficient (K_v)

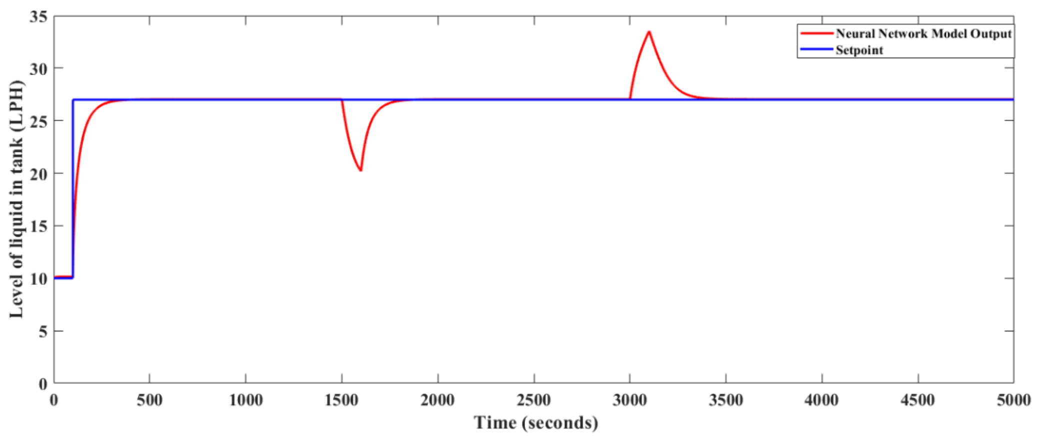

Fig. 17. Regulatory response of Process variable in parametric variation

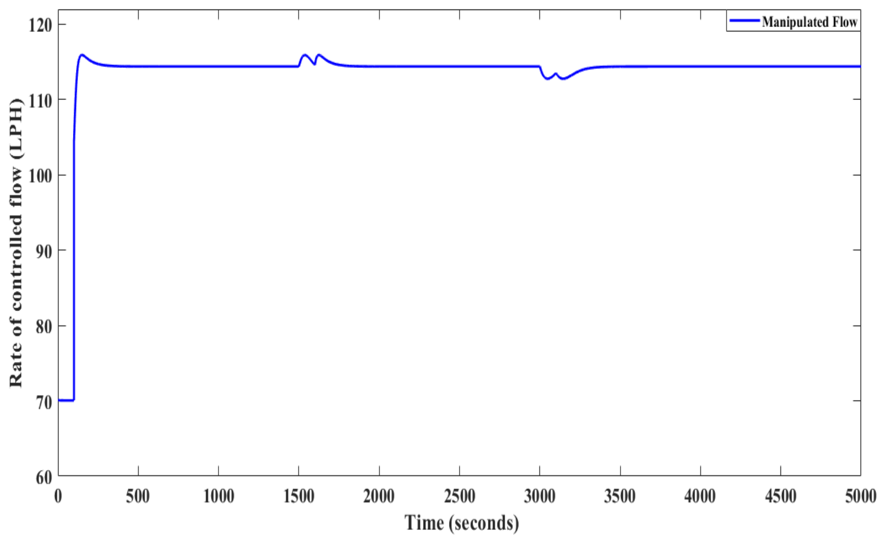

Fig. 18. Regulatory response of Controlled flow in parametric variation

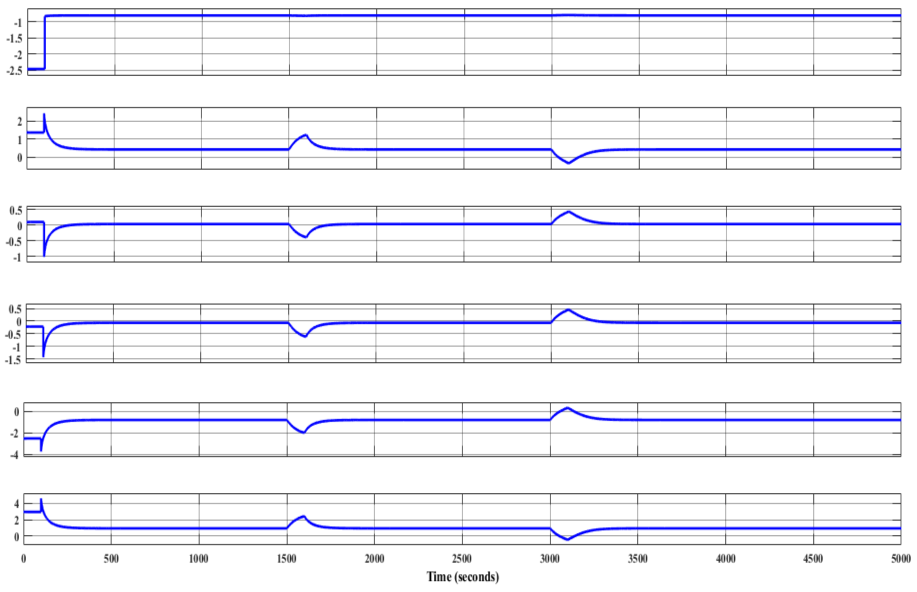

Fig. 19. Update of weights in hidden layer in regulatory response in parametric variation

8. CONCLUSION

In conclusion, the implementation of a neural network-based control scheme on the Benchmark Conical Tank Level System offers several advantages over traditional control methods. The neural network-based control scheme provides a more accurate and robust control system that can handle system nonlinearities and disturbances.

This paper presents a case study on the application of an adaptive neural network control scheme to regulate the liquid level in a nonlinear tank-level system. The simulation results demonstrate that the controller achieves the desired performance for both the servo and regulatory response of the conical tank level system. These findings are valuable for the field of process control industries, as they demonstrate the potential for effective control of liquid levels in nonlinear tank systems.

The main contribution of this research paper is the demonstration of the effectiveness of a neural network-based control scheme on the Benchmark Conical Tank Level System. This has significant implications for industries such as chemical and petrochemical where precise liquid level control is critical for efficient and safe operations.

Overall, the implementation of a neural network-based control scheme on the Benchmark Conical Tank Level System has the potential to improve liquid level control in the process control industry, and future research can explore the application of similar control schemes in other nonlinear systems.

9. REFERENCES

[1] Sanjay Bhadra, Atanu Panda, and Parijat Bhowmick, “A nonlinear model-based control scheme for benchmark industrial processes,” IEEE International Conference on Opto-Electronics and Applied Optics (Optronix), pp. 1-5, 2019.

[2] Sanjay Bhadra, Atanu Panda, Parijat Bhowmick, Shinjinee Goswami, Sayan Podder, Debayan Dutta, and Zeet Chowdhury, “Implementation of Neural Network Based Control Scheme on the Benchmark Conical Tank Level System,” IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), pp. 1-5, 2019.

[3] R J Rajesh, Preethi R, Parth Mehata, and Jaganatha Pandian B, “Artificial Neural Network Based Inverse Model Control Of A Nonlinear Process,” IEEE International Conference on Computer, Communication and Control (IC4_5270), pp. 1-6, 2015.

[4] N.S. Bhuvaneswari, G. Umab, and T.R. Rangaswamy, “Adaptive and optimal control of a non-linear process using intelligent controllers,” ELSEVIER, Applied Soft Computing 9, pp. 182-190, 2009.

[5] Sudharsana Vijayan, Dr.G Glan Devadhas, Vineed T govind, and Shinu M M, “Soft computation for Conical Tank Level Control,” IEEE International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), pp. 1776-1783, 2017.

[6] K.J. Nidhil Wilfred, S.Sreeraj, B.Vijay, and V.Bagyaveereswaran, “System Identification Using Artificial Neural Network,” IEEE International Conference on Circuits, Power and Computing Technologies (ICCPCT), pp. 1-4, 2015.

[7] C. Febina, and D. Angeline Vijula, “Model based Controller Design using Real Time Neural Network Model and PSO for Conical Tank System,” CEAI, Vol.22, No.3, pp. 13-24, 2020.

[8] Sukanya R. Warier, and Sivanandam Venkatesh, “Design of Controllers based on MPC for a Conical Tank System”, IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM), pp. 309-313, March 30, 31, 2012.

[9] T.Pushpaveni, S.Srinivasulu Raju, N.Archana, and M.Chandana, “Modeling and Controlling of Conical tank system using adaptive controllers and performance comparison with conventional PID,” International Journal of Scientific & Engineering Research, Volume 4, Issue 5, pp. 629-635, May-2013.

[10] Santanu Mallick, and Ujjwal Mondal, “Performance Analysis of a Conical Tank System using Lyapunov based MRAC Technique,” Proceedings of IEEE Region 10 Symposium (TENSYMP), pp. 168-172, 2019.