BP neural network PID controller based on Sigmoid function and Simulink simulation

1st Tengyue Wu
Electrical and computer enginerring
Vanderbilt University
Nashville, Unite States
tengyue.wu@Vanderbilt.Edu

Abstract*—The Backpropagation (BP) neural network is a widely used algorithm in artificial neural networks due to its strong ability to learn and adapt to nonlinear systems. This paper presents the implementation of a BP neural network PID (Proportional-Integral-Derivative) controller using MATLAB, with the Sigmoid function employed as the activation function to enhance the controller's adaptability and precision.Based on the MATLAB implementation, a Simulink simulation model is developed to evaluate the performance of the BP neural network PID controller. The model integrates the neural network's adaptive tuning capabilities with the classic PID control strategy, providing a flexible and efficient control solution for dynamic systems. To validate the proposed approach, the simulation model is applied to nonlinear systems, highlighting its effectiveness in handling complex and time-varying behaviors.The results demonstrate that the BP neural network PID controller achieves superior performance compared to traditional PID controllers. By dynamically adjusting PID parameters in real-time, the controller effectively improves response speed, reduces overshoot, and enhances stability and robustness. These characteristics make it particularly suitable for controlling nonlinear systems where conventional methods often fail to deliver optimal results.This paper contributes to the field of intelligent control by offering a practical and efficient method for combining BP neural networks with PID controllers. The use of MATLAB and Simulink ensures that the approach is both accessible and versatile for further development and application in various engineering and industrial scenarios. The findings provide a strong basis for extending neural network-based control strategies to more complex and dynamic systems.

***Index Terms*—BP Neural Network, PID Controller, Sigmoid Function, Simulink Simulation, Nonlinear Systems**

I. Introduction

Proportional-Integral-Derivative (PID) controllers are among the most widely used control strategies in industrial applications due to their simplicity [1], effectiveness, and reliability in managing linear systems. However, conventional PID controllers face significant challenges when applied to complex, nonlinear, or time-varying systems, where precise tuning of control parameters is required to maintain optimal performance. These limitations have driven the exploration of adaptive and intelligent control methods, including those based on artificial neural networks (ANNs), which have shown significant promise in overcoming these challenges.

Backpropagation (BP) neural networks, a prominent type of ANN, are particularly suited for control applications due to their ability to learn complex nonlinear mappings and self-adaptively adjust system parameters [2]. By integrating BP neural networks with PID controllers, a hybrid control strategy is achieved that offers enhanced adaptability and robustness for systems with dynamic and uncertain behaviors. Among the activation functions available for neural networks, the Sigmoid function is especially critical in this context due to its smoothness, bounded range, and effectiveness in handling nonlinearity. Incorporating the Sigmoid function into a BP neural network-based PID controller enables real-time, dynamic adjustment of PID parameters. This results in a controller capable of responding adaptively to variations in the controlled system, maintaining performance under varying conditions.

This study focuses on the design and implementation of a BP neural network PID controller, leveraging the Sigmoid function as the activation mechanism. The controller's performance is validated through simulations in Simulink, a powerful modeling and simulation tool widely used in control system analysis. The proposed controller is designed to dynamically adjust PID parameters in real-time, allowing it to handle nonlinearities and uncertainties in the controlled system with high precision and stability.

To illustrate the effectiveness of the proposed approach, this study uses a permanent magnet synchronous motor (PMSM) as an example system. The PMSM, a common actuator in industrial applications, presents challenges due to its nonlinear dynamic characteristics, especially during rapid speed adjustments. In the simulation, an overshoot module is employed to drive the motor's speed from zero to a target value of one unit. The control system ensures that the BP neural network PID controller tracks the target speed accurately, with the simulation functions modified accordingly to achieve seamless control transitions. The simulation results demonstrate that the proposed controller not only achieves rapid speed tracking but also effectively mitigates overshoot and undershoot, ensuring smooth and stable operation.

The main contributions of this work can be summarized as follows: 1. A novel BP neural network PID controller is designed, incorporating the Sigmoid function to enhance the system's ability to handle nonlinearity and adapt to dynamic

Identify applicable funding agency here. If none, delete this.

changes in the controlled environment. 2. A comprehensive Simulink model of the PMSM control system is developed, incorporating an overshoot module to simulate real-world scenarios where the motor's speed transitions dynamically [**?**]. 3. The performance of the proposed controller is evaluated and compared against conventional PID controllers, demonstrating superior precision, stability, and robustness. The results show that the proposed controller effectively tracks the target function and adjusts its parameters dynamically to maintain optimal control under varying conditions.

This study establishes a framework for using intelligent control strategies in complex dynamic systems, providing insights into the potential of neural network-based PID controllers for practical applications. The findings indicate that the integration of BP neural networks with Sigmoid activation functions offers a powerful and adaptive solution for managing nonlinear and time-varying systems, such as PMSMs.

II. Related Work

The integration of BP (Backpropagation) neural networks with PID (Proportional-Integral-Derivative) controllers has gained significant attention in recent years due to their ability to dynamically and adaptively optimize control systems. Traditional PID controllers are widely used in industrial automation because of their simplicity and effectiveness. However, their reliance on fixed parameters limits their performance in systems with nonlinearity, time-varying characteristics, or external disturbances. To overcome these limitations, researchers have explored intelligent mechanisms for real-time tuning of PID parameters [3].

Neural networks, particularly BP neural networks, have demonstrated great potential in control applications. Their ability to approximate nonlinear functions and adaptively learn from data makes them well-suited for dynamic systems. The BP neural network, with its iterative error correction mechanism based on the gradient descent algorithm, is particularly effective in identifying and controlling complex nonlinear systems. By combining BP neural networks with PID controllers, it becomes possible to adaptively adjust the PID parameters—proportional gain (K_p), integral gain (K_i), and derivative gain (K_d)—in real time. This adaptive tuning improves the controller's performance in handling uncertainties, reducing overshoot, shortening settling time, and minimizing steady-state error.

The sigmoid function plays a crucial role in BP neural network-based PID controllers [4]. As a widely used activation function, the sigmoid function offers smooth, differentiable, and bounded properties that enable the neural network to model complex relationships effectively. In the context of a PID controller, the sigmoid function facilitates precise adjustment of parameters, ensuring system stability and robust performance under varying conditions. This makes the sigmoid-based BP neural network a promising approach for improving control accuracy.

Simulink has emerged as a powerful tool for designing, simulating, and analyzing control systems, providing researchers with a robust platform to test BP neural network-based PID controllers. Through Simulink simulations, researchers can model and validate the performance of these controllers in various applications, such as robotics, process control, and power systems. These simulations serve as a critical step in refining theoretical models and optimizing controller designs before implementation in real-world systems.

Recent advancements in this field include the integration of optimization algorithms, such as genetic algorithms (GA) and particle swarm optimization (PSO), to enhance learning efficiency and performance. However, challenges remain, such as managing computational complexity, addressing the need for large training datasets, and ensuring stability in real-time applications. Despite these challenges, the combination of BP neural networks with PID controllers represents a significant advancement in intelligent control systems, with the sigmoid function and Simulink simulations playing pivotal roles in their development and validation. This paper builds on existing research by exploring the advantages of sigmoid-based BP neural network PID controllers and their performance through simulation.

III. PID Implementation

The PID controller is a mainstream control strategy widely used in industrial applications. To better understand its components, consider the example of adjusting the temperature of bathwater:

Proportional Term (Kp): The proportional term adjusts the control output in direct proportion to the error. For instance, the greater the difference between the desired temperature and the current temperature, the larger the adjustment in turning the heating knob. As the difference decreases, the adjustment becomes smaller.

Integral Term (Ki): The integral term addresses any residual error that remains after proportional control has been applied over a period. If the actual temperature does not reach the desired temperature despite proportional control, the integral term continuously adjusts the heating knob. This cumulative process helps eliminate steady-state errors.

Derivative Term (Kd): The derivative term reacts to the rate of change of the error, essentially predicting future error behavior. In the context of bathwater temperature, it adjusts the control output based on the speed of temperature change, ensuring a smoother and faster approach to the target temperature without overshooting. Then add the above three items together and you get the basic PID control:

$$output = K_p \cdot \mathrm{err} + K_I \cdot \int \mathrm{err}\, dt + K_D \cdot \frac{d\,\mathrm{err}}{dt} eq \qquad (1)$$

From the implementation point of view, there are actually two different implementation methods, one is called position PID, and the other is incremental PID. The known PID formula is as follows:

$$u = K_p \cdot err + K_I \cdot \int err\, dt + K_D \cdot \frac{derr}{dt} eq \qquad (2)$$

Formula (2) can be written as follows according to the PID position control algorithm:

$$u(k) = K_p \cdot err(k) + \sum_{i=0}^{k} K_I \cdot err(i) + K_D \cdot [err(k) - err(k-1)] \tag{3}$$

Position PID is the deviation between the actual position of the current system and the expected position you want to achieve, and PID control is performed. Because the error integral err(i) is accumulated, that is, the current output u(k) is related to all the past states, and the accumulated value of the error is used; (error err will have error accumulation), the output u(k) corresponds to the actual position of the actuator. Once the control output is wrong (the current state value of the control object has a problem), a large change in u(k) will cause a large change in the system and when the integral term of the position PID reaches saturation, the error will continue to accumulate under the integral action. Once the error begins to change in the opposite direction, the system needs a certain amount of time to exit the saturation zone. Therefore, when u(k) reaches the maximum and minimum, the integral action must be stopped, and there must be integral limiting and output limiting. Therefore, when using position PID, we generally use PD control directly, and position PID is suitable for objects whose actuators do not have integral components, such as the control of the upright and temperature control systems of servos and balancing cars. Formula (2) is written as follows according to the PID incremental control algorithm:

$$u(k) = u(k-1) + K_p \cdot [err(k) - err(k-1)] + K_I \cdot err(k) + K_D \cdot [err(k) - 2 \cdot err(k-1) + err(k-2)] \tag{4}$$

The incremental PID controller can be clearly understood from its formula [5]. Once the proportional (P), integral (I), and derivative (D) parameters are determined, the control increment $\Delta u(k)$ can be calculated directly using the errors from the current and the two most recent measurements. Importantly, the calculated control increment $\Delta u(k)$ is based on the recent changes in position error, rather than the cumulative error associated with the actual position deviation. This eliminates the need for error accumulation in incremental PID control.

In other words, the incremental PID controller operates without cumulative summation of errors. The determination of the control increment $\Delta u(k)$ relies only on the most recent three sampling values, making it easier to achieve optimal control performance through proper weighting of these values. Furthermore, incremental PID control has a distinct advantage in that, if a system issue occurs, the controller will not drastically affect the system's operation. This is because the control output is determined incrementally, rather than based on the absolute error.

To summarize, the incremental PID controller can be seen as the derivative form of the positional PID controller. In this approach, the controller output is the difference between the calculated control outputs at two consecutive sampling instants. The result is an incremental value, representing the adjustment (increase or decrease) required to the previous control output. A positive increment indicates an increase in control output, while a negative value represents a reduction [6]. This incremental nature allows for smoother control adjustments and minimizes the impact of system disturbances, making it more robust in practical applications.

IV. BP NEURAL NETWORK

The neural network model is based on neurons in the brain. The general structure of neurons is as shown in Figure 2 below: Neurons are composed of cell bodies and protrusions, and the

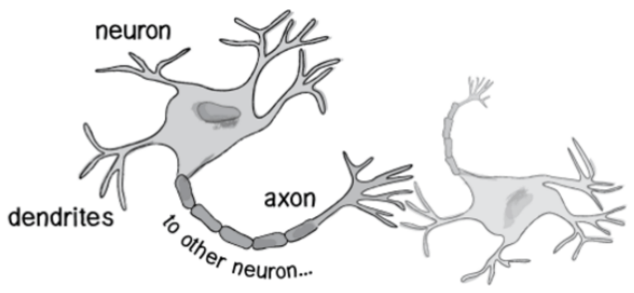

Fig. 1. neurons.

protrusions are divided into dendrites and axons. Dendrites are the network-like protrusions on the left side of the above picture. They are receivers that connect to the axons of other neuronal cells and receive signals from other neuronal cells. Axons are the long strip-shaped protrusions on the right side of the above picture. They are transmitters responsible for sending signals to other neuronal cells. Countless neurons form a huge neural network called a nerve center, and people can produce various behaviors based on the instructions sent by the nerve center. The neural network is a computer structure that scientists simulated using computers after understanding the structure of biological neurons. Why do scientists simulate the brain? The reason is that they hope to give the machine the ability to "think" of the brain. The simplest neuron model is called a "perceptron".As shown in Figure 3:

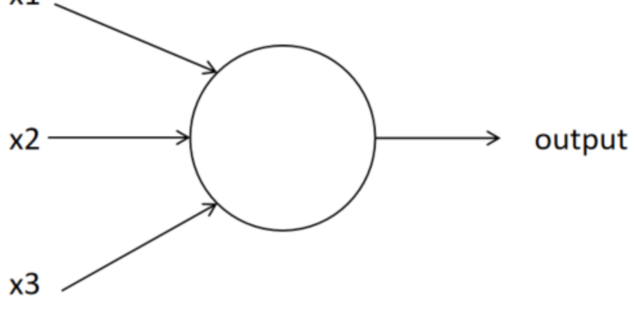

Fig. 2. Neurons.

The x1, x2, and x3 above simulate the dendrites of biological neurons, and the output represents the axon of the neuron. Countless neurons are connected front and back to form the ”brain” in the chip. However, the dendrites of the commonly used neuron model will also have ”weights” added to them. The input is multiplied by the weight and then added. If the final value is greater than the threshold of the neuron, the neuron will generate an output signal. This simulates the process in the biological brain that only when a neuron is stimulated enough can an electrical signal be generated. The output is:

$$y_i = \sum_{i=1}^{N} w_i x_i - \theta_i \tag{5}$$

The threshold term θi of the neuron means that the neuron will have an output only when the linear sum of the input values is greater than this value. In order to modify the value of the weight, the learning rule is introduced. The learning rule can enable the simulated neurons to acquire the ability to ”learn” in some way. Its main function is to modify the connection strength between neurons, that is, to modify the ”weight”. Different learning rules can form different neural networks, and the specific properties of these neural networks are also different. The more commonly used learning rules are Hebb, Perceptron, Delta, BP, etc [7]. Learning is divided into unsupervised learning and supervised learning. The simple distinction is whether to introduce expected values. The general δ learning rules are as follows:

$$\Delta w_i(t) = \eta r x_i(t) \tag{6}$$

The equation indicates that the weight change at time t, denoted as $\Delta w_i(t)$, is proportional to the product of the input at time t, $x_i(t)$, and the learning signal r. The proportional coefficient, η, is called the learning rate constant, which determines the speed of learning. So we can get the weighting for the next moment:

$$w_i(t+1) = \Delta w_i(t) + \eta r x_i(t) \tag{7}$$

V. BP NEURAL NETWORK ADAPTIVE PID CONTROL

A. *Framework of the Proposed BP Neural Network PID Controller*

The framework of the proposed BP neural network PID controller is illustrated in Figure 1. This architecture combines the adaptability of BP neural networks with the precision of PID control to enhance performance in handling nonlinear and time-varying systems.

Error Signal (“e”): The difference between the reference input (“r”) and the system output (“y”) is computed to form the error signal, which drives the control process.

Learning Algorithm: A BP neural network learning algorithm is employed to adjust the network parameters based on the error signal. This dynamic learning mechanism ensures that the controller adapts to changing system dynamics in real time.

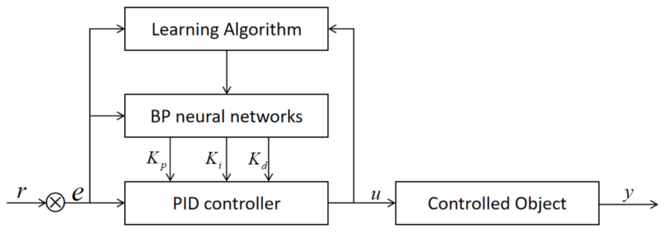

Fig. 3. Model Diagram.

BP Neural Network: The BP neural network computes the PID parameters (“Kp”, “Ki”, “Kd”) based on the current error signal and its historical data. The Sigmoid activation function is used to handle nonlinearities and ensure smooth parameter adjustments.

PID Controller: The PID controller receives the parameters from the BP neural network and generates the control signal (“u”) to drive the controlled object.

Controlled Object: The controlled object represents the system or process being regulated by the controller. This integrated framework ensures real-time adaptability and robustness, making it suitable for applications in complex and nonlinear control systems. The proposed design is further validated and analyzed through detailed Simulink simulations, as described in subsequent sections.

B. *Sigmoid function writing format*

The basic format of S function is as follows: function [sys, x0, str, ts] = function name (t, x, u, flag) where t is the simulation time, corresponding to the continuous system t is continuous, and for the discrete system t is a series of sampling points; x is the state variable of the system, including continuous and discrete states. If there is no state variable, NumStates=0; u is the input vector, sys is the system output, and the meaning of sys is different under different calling functions; x0 is the initial value of the system state variable; str is a reserved parameter and is always an empty set; ts is the sampling time; flag is the simulation process control flag variable, which is used to control the calling sequence and process of the program. Table 1 shows the calling function and function description corresponding to flag. When writing an S function, you must know clearly what information the system needs at different times. For example, how many state variables, input variables, and output variables do the system have. Which of them are continuous variables,

TABLE I
FLAG CORRESPONDING CALLING FUNCTION AND FUNCTIONAL DESCRIPTION

Flag	Calling Function	Functional Description
0	mdlInitializeSize	System model initialization function
1	mdlDerivatives	Continuous system state variable derivatives
2	mdlUpdate	Discrete system state variable update
3	mdlOutputs	System model output
4	mdlGetTimeOfNextVarHit	Compute the time of the next sampling point
9	mdlTerminate	Simulation termination function

which are discrete variables, and the initial conditions of these variables. This information can be obtained by setting flag=0 in the S function. If the system is a continuous variable, the derivative of the state variable must be obtained according to the condition, which can be obtained by setting flag=1; if it is a discrete variable, the sampling time and the next discrete state must be determined, which can be obtained by setting flag=2; finally, setting flag=3 can obtain the output of the system. It can be seen from this that you can use the Simulink module library or write an S function to draw any complex system block diagram and then simulate and analyze the system. The simulation results can be displayed directly on the oscilloscope, or the values can be returned to MATLAB to the workspace for further processing.

C. BP neural network PID controller core part S function

The BP neural network PID controller is implemented as an S-function in MATLAB/Simulink. The core code of the S-function dynamically adjusts the PID parameters (“Kp”, “Ki”, “Kd”) based on the current system state and error signals. The implementation involves the following key steps: 1.Initialization Function: function [sys,x0,str,ts]=mdlInitializeSizes sizes = simsizes; sizes.NumContStates = 0; sizes.NumDiscStates = 3; sizes.NumOutputs = 4; sizes.NumInputs = 7; sizes.DirFeedthrough = 1; sizes.NumSampleTimes = 1; sys = simsizes(sizes); x0 = zeros(3,1); str = []; ts = [0 0]; end This function initializes the module parameters, including the number of states, inputs, and outputs. It also sets the sampling period and initial state variables. 2.Update Function: The update function computes the discrete state variables, including error, cumulative error, and error change rate, bas2ed on the system inputs. function sys=mdlUpdates(x,u) T = 0.001; x = [u(5); x(2) + u(5)*T; (u(5) - u(4))/T]; sys = [x(1); x(2); x(3)]; end 3.Output Function: function sys=mdlOutputs(t,x,u) xite = 0.2; alfa = 0.05; IN = 3; H = 5; OUT = 3; wi = rand(5,3); wo = rand(3,5); Oh = zeros(5,1); xi = [u(1), u(3), u(5)]; epid = [x(1); x(2); x(3)]; I = xi * wi’; for j = 1:5 Oh(j) = tanh(I(j)); end K1 = wo * Oh; for i = 1:3 K(i) = tanh(K1(i)); end end The output function calculates the PID parameters and control law using the BP neural network with a Sigmoid activation function. It dynamically updates the weights and adjusts the control output accordingly. The overall simulink model is shown in Figure 4:

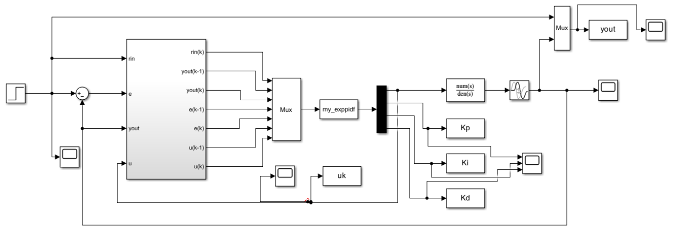

Fig. 4. Simulink model.

VI. Experiments and Result

After writing the S function, the Simulink model of the controlled system is established. The steps are as follows: 1) Enter the function variable name and parameter variable name. After dragging the S function module to the model window, edit the S function module and enter the function variable name exp_pidf (m file name, which cannot be the same as the model file name). 2) Create a subsystem. 3) Mask the subsystem [6]. The established system model is shown in Figure 4. The transfer function of the controlled object is set as:

$$G(s) = \frac{1.2}{208s + 1} e^{-80s} \tag{8}$$

The simulation results are shown in Figure 5 and Figure 6 respectively. Figure 5 shows the values of P, I, and D from top

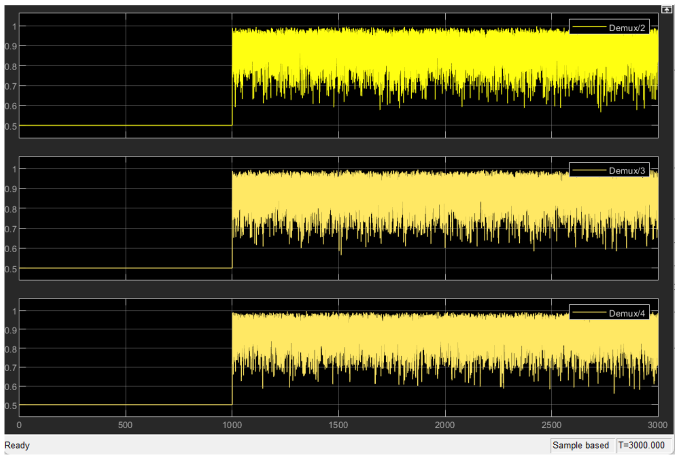

Fig. 5. P,I,D reult

to bottom. It can be seen that when the step function jumps from 0 to 1, the PID changes accordingly. Figure 6 shows the actual operation effect. It can be seen that the blue is the simulation function and the yellow is the target function. The blue is close to the target function after about 1800s. However, I did not expect the PID value to be adjusted too radically, but you can see that its actual value adjustment is smooth, so this article believes that there is no problem with the adjustment of the PID value. It is a successful convergence.

VII. Conclusion

According to the simulation results, the Sigmoid function can be used to realize the system simulation model of complex control laws. Since the BP neural network PID controller can adjust the three parameters of the PID controller in real time, it can achieve good tracking control of nonlinear objects. The simulation results show that this method has better tracking effect than ordinary PID control. Therefore, by writing Sigmoid functions to establish the Simulink simulation model of the BP neural network PID controller, or writing other more complex control law models, the respective advantages of MATLAB programming flexibility and Simulink simplicity

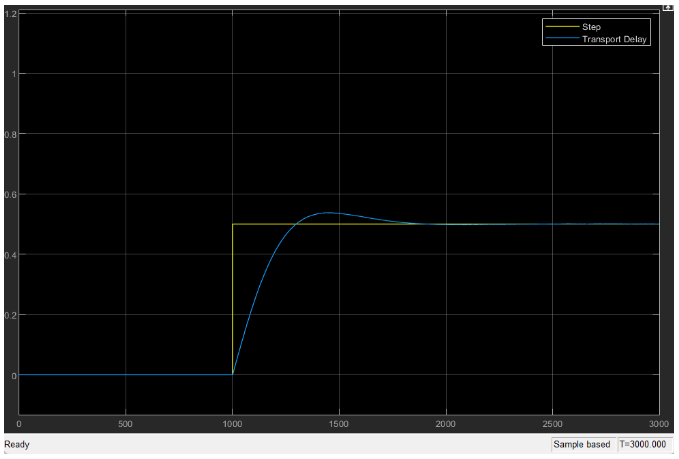

Fig. 6. simulink value and target value.

and intuition can be fully utilized, broadening the application scope of Simulink.

References

[1] Åström K J, Hägglund T. PID control[J]. IEEE Control Systems Magazine, 2006, 1066.

[2] Ding S, Su C, Yu J. An optimizing BP neural network algorithm based on genetic algorithm[J]. Artificial intelligence review, 2011, 36: 153-162.

[3] Du J, Wang B. Pitch Control of wind turbines based on BP neural network PI[C]//Journal of Physics: Conference Series. IOP Publishing, 2020, 1678(1): 012060.

[4] Song R, Li H, Li Y. An Impedance Matching Method Based on Optimized BP Neural Network for Vehicular Power Line System[J]. IEEE Access, 2024.

[5] Zhou Y, Zhang Y, Yang T. Research on load simulator control strategy based on BP neural network and PID method[C]//MATEC Web of Conferences. EDP Sciences, 2020, 306: 03002.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] Klee H, Allen R. Simulation of dynamic systems with MATLAB® and Simulink®[M]. Crc Press, 2018.