Plant-Physics-Guided Neural Network Control for Permanent Magnet Synchronous Motors

Zhenxiao Yin\textsuperscript{1}, Xu Chen\textsuperscript{1}, Yang Shen\textsuperscript{1}, Xiangdong Su\textsuperscript{1}, Dianxun Xiao\textsuperscript{1}, Dirk Abel\textsuperscript{1}, and Hang Zhao\textsuperscript{1}

\textsuperscript{1}Affiliation not available

January 10, 2024
Plant-Physics-Guided Neural Network Control for Permanent Magnet Synchronous Motors

Zhengxiao Yin, Student Member, IEEE, Xu Chen, Yang Shen, Xiangdong Su, Student Member, IEEE, Dianxun Xiao, Member, IEEE, Dirk Abel, Member, IEEE, Hang Zhao*, Member, IEEE

Abstract—In safety- and precision-critical control for permanent magnet synchronous motors (PMSMs), the spontaneous disturbance causes unexpected speed drop. The disturbance occurs without routine, so it cannot be modeled specifically. The large speed drop and slow response speed cause a reduced life of the machines driven by PMSMs. Therefore, it is crucial to implement a method that can learn the effect caused by disturbances. To this end, this paper proposes a novel approach based on the basic structure of a backpropagation neural network (BPNN) for adaptive real-time adjustment in motor control. Regarding the lack of explainability of BPNN, the electric motor physics is embedded into BPNN (BP-PHY) gradient update part to enlarge the range of stability. To overcome the shortage of a potential unstable output of neural network (NN), the learning parameter of NN is tailored based on stability theory and motor physics. Finally, the proposed methods are implemented into simulations of PMSM, while the control stability of the NN is ensured.

Index Terms—Learning rate design, neural network controller, stabilized online learning, physics-guided neural network, permanent magnet synchronous motor.

I. INTRODUCTION

Three-phase permanent magnet synchronous motor (PMSMs) represents a contemporary variant of PMSM technology, boasting notable attributes such as high energy density, robust fault tolerance, and reduced torque ripple [1], [2]. Consequently, it exhibits significant potential for deployment in future generations of automated guided vehicles, autonomous underwater vehicles, and unmanned aerial vehicles [3], [4]. In these application scenarios, numerous stochastic disturbances arise. Establishing a precise physical model for these disturbances becomes challenging [5]. These disturbances significantly degrade the operating performance of the equipment. For instance, the speed drops spontaneously and largely even with slow response speed of the actuators (PMSMs), resulting in a reduction in the service life of the machines [6]. Therefore, a controller capable of real-time adjustment is developed based on changes in external disturbances [7]. To enhance the fundamental performance of the machines, the focus should be placed on the core components of the robots, namely the motor control systems [8].

Typically, the control strategy for three-phase PMSMs relies on a position, speed, and current loop framework, employing PI-based field-oriented control [9]. However, the control performance achieved is satisfactory only under unchanged control targets and outside conditions. In industrial practice, engineers often establish tables of PI parameters tailored to different control targets, which requires a high labor cost [10]. Nonetheless, when the self-condition of the PMSMs or the external environment undergoes changes, the predefined tables may no longer provide satisfactory performance. Consequently, an auto-tuning method becomes necessary [11].

Several machine learning techniques have been adopted to address the challenge of finding table controller parameters online. The particle swarm [12] and genetic optimization [13], [14] methods have been used to fine-tune the parameters offline. These approaches are typically employed to find a single optimal parameter in all scenarios, which reduces manual optimization costs to some extent. However, it should be noted that a single optimal parameter does not necessarily guarantee optimality for all subproblems within the system [15].

In order to find the optimal solution for an individual subproblem, for example, for each speed target or load torque, the optimal parameter is different. Thus, the online neural network control method has been proposed to tackle this problem [16]. These approaches enable real-time optimization based on variables such as position, velocity error, or other cost function parameters at each step. It exhibits a certain degree of disturbance rejection capacity [17], [18]. Current popular methods for controlling neural networks based on online learning include single-neuron neural networks, radial basis neural networks, recurrent neural networks, fuzzy neural networks, and backpropagation neural networks [15], [19]. Among these, single-neuron neural networks have a simple structure and fast convergence but generally do not achieve satisfactory performances, thus limiting their widespread application. Radial basis neural networks have gained more attention due to their interpretable overshoot characteristics, but there is still room for their effectiveness improvement. Recurrent neural networks, which incorporate temporal information and iterate the optimization process based on previous steps, have been applied in control and have shown minor improvements [20].
Fuzzy neural networks typically exhibit characteristics similar to radial basis neural networks but with deeper layers, resulting in longer computation time [21]. But it delivers high-control performances. Backpropagation neural networks, known for their effective optimization guidance, have been widely applied in control due to their black-box network architecture and specific activation functions [22].

The above methods are model-free neural network self-learning approaches for motor control. And, the precise knowledge of the motor parameters are not required, which exhibits robustness in parameter variations [23]. However, from another perspective, if the basic model of the motor can be easily established and provides guiding value, it is entirely possible to incorporate the physical principles of the motor to guide the design of neural networks. Therefore, it is possible to further enhance the performance of the controller.

Through a profound analysis of the equations and the accumulation of preliminary research of neural network methods, a meticulous point in the backpropagation neural network has captured the attention. Typically, the core of the backpropagation neural network lies in the backpropagation process, which is the weight optimization process based on gradient-based optimization algorithms [24]. This optimization process usually relies on the chain rule of partial derivatives to establish the functional relationship between the objective function and the learning parameters to be adjusted. Interestingly, a partial derivative relationship exists between the motor’s desired states and the control variable’s variation within the backpropagation rule. In previous studies, this term was often approximated using the sign function [7], [15], [25], [26], implying that the optimization process of conventional neural networks employed a sliding mode control-like approach. However, this approach has the potential to lead to instability in the final control performances. Such a problem requires rigorous stability strategies, which limits the learning process of the parameters.

To further enhance the application effectiveness of neural network methods in motor control, this paper makes the following contributions.

1. An optimization method based on gradient descent with momentum is implemented for the plant-physics-guided neural network;

2. A plant-physics-guided backpropagation neural network learning method based on electric motor (plant) principles is proposed. Its feasibility and the usage of enlarging the stable range of the control system is validated;

3. The learning rates and momentum factors are designed based on stability analysis techniques with extreme value calculation, which are applied to limit the update rate of the neural network, ensuring the stability of the neural network in motor control.

4. The NN control-based scheme is given for the improvement of the dynamic performance of the electric motor, and an additional trigger scheme is given for further improvement of the NN controller in static performance.

II. THEORETICAL FOUNDATION OF PLANT PHYSICS GUIDED NEURAL NETWORK CONTROL FOR PMSM

A. Theory of BPN-Assisted Control Method

This section presents a physics-informed neural network designed for the controller of three-phase PMSMs. The update rule of a PI-formed controller is described by (1), where the control variable at time $k$ is denoted as $u_k$, the control variable at time $k-1$ is denoted as $u_{k-1}$, and the control updated increment at time $k$ is denoted as $\Delta u_k$.

$$u_k = u_{k-1} + \Delta u_k$$

(1)

The increment of the control variable $u$ is defined based on the positional PI controller form in (2). It should be noted that it provides guidance for the subsequent update of the BPNN, although it is not directly utilized in the control system.

$$\Delta u_k = K_{1k}(\dot{e}_{\omega,k} + e_{\omega,k-1}) + K_{2k}e_{\omega,k}$$

(2)

Here, $e_{\omega,k}$ and $e_{\omega,k-1}$ are the speed tracking error of the PMSMs. $K_{1k}$ and $K_{2k}$ are the control parameters. For a method based on backpropagation neural networks, the update rate of the backpropagation process is crucial. Typically, its objective function is defined as shown in (3), which represents the squared difference between the desired motor speed and the actual speed. $E_k$ is the cost function regarding the reference value $r_k$ and real output value $y_k$. Here, it differs from traditional neural networks used for system identification, where the neural network training is based on the error between the actual and desired values to approximate the mapping relationship between the data. In the proposed control method, the objective is to establish the underlying nonlinear relationship between the motor output and the controller parameter.

$$E_k = \frac{1}{2}(r_k - y_k)^2$$

(3)

The output layer and hidden layers of the neural network control algorithm still employ the incremental method for updates, as shown in (4). Here, $w$ is the weight of the neural network, and $\Delta w$ means its increment. $l$ is the order of the neurons of the output layer, $z$ is the order of the neurons of the hidden layer. $n$ is the order of the output layer. In this paper, $n_{max} = 3$, which shows a relatively shallow structure because of the computational time requirement in control tasks. Here, $L = \{l \in N_+ | l \in [1,4]\}$, $Z = \{z \in N_+ | z \in [1,8]\}$, $J = \{j \in N_+ | j \in [1,2]\}$, $n = \{n \in N_+ | n \in [1,8]\}$.

$$w_{l,z,k} = w_{l,z,k-1} + \Delta w_{l,z,k}$$

(4)

The weight increment can be computed using the gradient descent method with respect to the evaluation function. To mitigate the impact of oscillations during the optimization process, a momentum factor, denoted $\alpha_{BP}$, is introduced. The updated law is presented in (5). $\eta_{BP}^{(n)}$ is the basic learning rate of the backpropagation neural network for the $n$-th layer.

$$\Delta w_{l,z,k}^{(n)} = -\eta_{BP}^{(n)} \frac{\partial E_k}{\partial w_{l,z,k}^{(n)}} + \alpha_{BP}^{(n)} \Delta w_{l,z,k-1}^{(n)}$$

(5)

The calculation of the gradient term in (5) is carried out through a series of steps, using the chain rule to calculate the
partial derivatives of various components within the BPNN network. The specific calculation process is illustrated in (6). The variable $y_l$ represents either the current, speed, or position, depending on the desired control loop to be tuned.

$$ \frac{\partial E_k}{\partial w_{l,z,k}} = \frac{\partial y_k}{\partial \Delta u_{k-1}} \frac{\partial \Delta u_{k-1}}{\partial O_{l,k-1}} \frac{\partial O_{l,k-1}}{\partial \Delta u_{k-1}} $$

where $\frac{\partial E_k}{\partial w_{l,z,k}}$ is the partial derivative of cost function $E_k$ and weight of output layer $w_{l,z,k}$; $\frac{\partial y_k}{\partial \Delta u_{k-1}}$ is the partial derivative of $y_k$ and control increment $\Delta u_{k-1}$; $\frac{\partial \Delta u_{k-1}}{\partial O_{l,k-1}} \frac{\partial O_{l,k-1}}{\partial \Delta u_{k-1}}$ is the partial derivative of $\Delta u_{k-1}$ and the forward calculation of neural network from 1 to n, namely net$^{l}_{l,k}$. $\frac{\partial \Delta u_{k-1}}{\partial O_{l,k-1}}$ is the partial derivative of $O_{l,k-1}$ and the forward calculation of neural network from 1 to n, namely net$^{l}_{l,k-1}$. Net$^{l}_{l,k}$ and net$^{l}_{l,k-1}$ is the partial derivative which contains the influence of activation function $\sigma$.

The aforementioned equation is incorporated into the chain rule of derivative formulas for updating the weights in the third and second layers, as illustrated in (7) and (8) refer to [15]. In these equations, the activation function $\sigma_l$ is applied as a hyperbolic tangent (tanh) function to constrain the hidden layer’s output range between -1 and 1. On the other hand, the activation function $\sigma_2$ is implemented as a sigmoid function to restrict the neural network’s output within the range of 0 to 1.

$$ \Delta w_{l,z,k}^{(3)} = \eta (\omega_l) \frac{\partial \Delta u_{k-1}}{\partial \Delta u_{k-1}} \frac{\partial y_k}{\partial \Delta u_{k-1}} \frac{\partial \Delta u_{k-1}}{\partial O_{l,k-1}} \frac{\partial O_{l,k-1}}{\partial \Delta u_{k-1}} - \eta \sigma_2(\text{net}_{l,k-1}) + \alpha \Delta w_{l,z,k-1} $$

$$(1 - \sigma_2(\text{net}_{l,k-1}))O_{l,k-1}^{(2)} + \alpha \Delta w_{l,z,k-1}$$

$$ \Delta v_{z,j,k}^{(2)} = -\eta \left(1 - \sigma_2^2(\text{net}_{z,k-1}^{(2)}) \right) \frac{\partial \Delta u_{k-1}}{\partial \Delta u_{k-1}} \frac{\partial y_k}{\partial \Delta u_{k-1}} \frac{\partial \Delta u_{k-1}}{\partial O_{l,k-1}} \frac{\partial O_{l,k-1}}{\partial \Delta u_{k-1}} - \eta \sigma_2^2(\text{net}_{z,k-1}^{(2)}) + \alpha \Delta v_{z,j,k-1}^{(2)}$$

It is important to note that there is one term, denoted as $\frac{\partial y_k}{\partial \Delta u_{k-1}}$ for which a detailed explanation has not been provided in previous research [7], [15], [25], [26]. Interestingly, this Jacobian term has been identified as a crucial factor affecting the final results in this study.

**B. Design of Jacobian Term based on Plant-Physics**

Typically, the jacobian term $\frac{\partial y_k}{\partial \Delta u_{k-1}}$ is approximated by solely employing the sign function, as it cannot be accurately determined when there is no model available for the controlled object. In previous studies, this term is commonly approximated using the following expression:

$$ \frac{\partial y_k}{\partial \Delta u_{k-1}} = \text{sign} \left( \frac{y_k - y_{k-1}}{\Delta u_{k-1} - \Delta u_{k-2}} \right) $$

The method described above offers a distinct advantage as it facilitates the straightforward acquisition of the update direction and enables normal updates during transient states. However, certain drawbacks arise, particularly in the context of steady-state processes, attributable to the presence of the sign function since the character of vibration of the $\text{sign}$ function leads to oscillation and even instability.

In this paper, it is aimed to introduce modifications to address the above issues. The initial variant involves eliminating the sign function and the introduction of a bias term in the denominator. This modification mitigates the problem of infinite results and reduces the occurrence of oscillations and instability during stable states. Specifically, the updated formulation entails:

$$ \frac{\partial y_{k}}{\partial \Delta u_{k-1}} = \frac{y_{k} - y_{k-1}}{\Delta u_{k-1} - \Delta u_{k-2} + \varepsilon} $$

The second variant involves incorporating the model information into the updated law, thereby demonstrating the applicability of the motor physics-guided neural network method. The discrete-time state-space function for a three-phase permanent magnet synchronous motor (PMSMs) is introduced using Euler discretization. This choice is motivated by the simplicity of the partial derivative in the subsequent update law, which requires minimal computational memory for evaluation.

$$ \begin{bmatrix} \omega_{k} \\ I_{d,k} \\ I_{q,k} \end{bmatrix} = \begin{bmatrix} \omega_{k-1} \\ I_{d,k-1} \\ I_{q,k-1} \end{bmatrix} + \begin{bmatrix} -Bp_n/J_m & 1.5p_n\psi_f/J_m & 0 \\ 0 & -R/L_d & 0 \\ 0 & 0 & -R/L_q \end{bmatrix} \begin{bmatrix} \omega_{k-1} \\ I_{d,k-1} \\ I_{q,k-1} \end{bmatrix} + \begin{bmatrix} V_{d,k-1}/L_d \\ V_{q,k-1}/L_q \end{bmatrix} T_s $$

where

$$ R_1 = 1.5p_n I_{q,k} I_{d,k} (L_d - L_q)/J_m - T_L/J_m $$

$$ R_2 = p_n \omega_{k-1} L_q I_{q,k}/L_d $$

$$ R_3 = (-p_n \omega_{k-1} L_d I_{d,k} - p_n \omega_{k-1} \psi_f)/L_q $$

Here, $\omega_k$ is the rotational speed at time $k$. $I_{d,k}$ and $I_{q,k}$ represent two orthogonal currents of two rotational axes at time $k$, which are usually considered as control variables of the medium-frequency task. $B$ is the damping ratio, $J_m$ is the inertia of PMSM. $p_n$ is the number of pole pairs fixed after manufacturing. $R$, $L_d$, $L_q$ are the electric parameters of the PMSMs, which represent resistance, d-axis inductance, and q-axis inductance. $\psi_f$ is the magnetic parameter, namely the flux linkage. $V_d$ and $V_q$ are the voltage of the d, q axis that is generally regarded as the control variable of high-frequency task. $T_s$ represents the sampling time in the equation after Euler discretization.

The next step involves the design of a physics-informed gradient-based neural network optimization method. Firstly, rethink the control objective, where $\Delta u$ is designed as a positional PI controller. In (15), the velocity loop’s output and the current loop’s input are utilized, assuming that the reference current can perfectly track the system’s desired current. Building upon this assumption, the concept of perfect tracking is embedded in the gradient of the neural network, which theoretically accelerates the convergence of the neural...
network. The Jacobian term integrated with the motor physics is expressed as follows:

\[
\frac{\partial y_{\omega,k}}{\partial \Delta u_{k-1}} = \frac{\partial \omega_k}{\partial \Delta u_{k-1}} \approx \frac{\partial \omega_k}{\partial \Delta u_{q,k-1}} = \left(3p_n \psi_f + 3p_n (L_d - L_q) \Delta i_{d,k-1} \right) T_s
\]  

(15)

Theoretically, if the neural network control method is applied to the current loop, its optimization gradient can also be computed based on the current model. However, considering the limitations of experimental equipment and the computational demands of the high-frequency calculations in the current loop, the integration of this neural network into the current loop is temporarily excluded.

C. Simulation Results of BPNN Control with Different Jacobian Information

In 1, the different results are shown. PI means proportional-integral controller. BP-FD means neural network controller using finite difference rule of Jacobian information. SIFD means a finite difference inside the sign function. PHY means physics-guided update law. PHY to Inf

FD to Inf
SIFD,FD and PHY to Inf
SIFD,FD and PHY to Inf

The difference between the discrete Lyapunov function of time \(k\) and \(k-1\) is shown in:

\[
\Delta V_k = \frac{1}{2}(e_{k+1}^2 - e_k^2) = \frac{1}{2} (\Delta e_k^2 + 2e_k \Delta e_k)
\]  

(17)

The calculation of \(\Delta e_k\) involves the partial derivative of the formula in (18), which incorporates the parameter weight update and the information of the Jacobian system within the partial derivative term. The term \(\Delta e_k\) employs a linear approximation, neglecting the impact of higher-order terms, to simplify the design procedure for the learning rate.

\[
\Delta e_k = \sum_{l=1}^{2} \sum_{z=1}^{8} \Delta w_{l,z,k}^{(3)T} \frac{\partial e_k}{\partial w_{l,z,k}^{(3)}} + \sum_{j=1}^{4} \sum_{z=1}^{8} \Delta w_{z,j,k}^{(2)T} \frac{\partial e_k}{\partial w_{z,j,k}^{(2)}}
\]  

(18)

The maximum values for each element are selected in (19), which results in the extreme value of error increment \(\Delta e_{extr,k}\). Then, the maximum values are multiplied by the number of connections between the neurons. \(|L|, |J|, |Z|\) are the number of set \(L, J, Z\).

\[
\Delta e_{extr,k} = |L| |Z| \max_{l \in L, z \in Z} \left( \frac{\partial e_k}{\partial w_{l,z,k}^{(3)}} \right) + |J| |Z| \max_{j \in J, z \in Z} \left( \frac{\partial e_k}{\partial w_{z,j,k}^{(2)}} \right)
\]  

(19)

For simplicity, all the weights \(w\), learning rates \(\eta\), and momentum factors \(\alpha\) occur in this upper bound derivation with

Fig. 1. Comparison of basic controller, BP-FD, BP-SIFD, BP-PHY of tracking 500, 1000, 1500 rpm by using learning rate (Lr) and Momentum Factor (Mf) of 0.005, 0.05, 0.1, 0.5, 1 with speed NN control in simulation.
absolute value are all added with a *

\[
\Delta e_{extr,k} = \Delta e^*_k = - |L| |Z| \eta_{BP}^{(3*)} e_k \frac{\partial e^T_k}{\partial w_{1,z,k}^{(3*)}} \frac{\partial e_k}{\partial w_{1,z,k}^{(3*)}} \\
+ |L| |Z| \alpha_{BP}^{(3*)} \Delta w_{1,z,k}^{(3*)T} \frac{\partial e_k}{\partial w_{1,z,k}^{(3*)}} \\
- |J| |Z| \eta_{BP}^{(2*)} e_k \frac{\partial e^T_k}{\partial w_{2,j,k}^{(2*)}} \frac{\partial e_k}{\partial w_{2,j,k}^{(2*)}} \\
+ |J| |Z| \alpha_{BP}^{(2*)} \Delta w_{2,j,k}^{(2*)T} \frac{\partial e_k}{\partial w_{2,j,k}^{(2*)}}
\]  

(20)

Define:

\[
D = \left[ \frac{\partial e_k}{\partial w_{1,z,k}} \frac{\partial e_k}{\partial w_{2,j,k}} \right], \Delta W = \left[ \Delta w_{1,z,k}^{(3*)T} \Delta w_{2,j,k}^{(2*)T} \right].
\]  

(21)

From (20) and (21), the difference of the discrete Lyapunov function is calculated as:

\[
\Delta V_k = \frac{1}{2} \Delta W^T \alpha^* D \Delta W + e_k \Delta W^T \alpha^* D \\
- e_k \Delta W^T \alpha^* D \Delta V_k
\]

(22)

with

\[
\eta^* = \left[ |L| |Z| \eta_{BP}^{(3*)} 0 |L| |Z| \eta_{BP}^{(2*)} \right] \\
\alpha^* = \left[ |L| |Z| \alpha_{BP}^{(3*)} 0 |L| |Z| \alpha_{BP}^{(2*)} \right].
\]

(23)

According to the (22), when each part is smaller than 0, then \( \Delta V_k < 0 \), such that:

\[
\frac{1}{2} \Delta W^T \alpha^* D \Delta W + e_k \Delta W^T \alpha^* D \\
- e_k \Delta W^T \alpha^* D < 0
\]

(24)

and

\[
\frac{1}{2} \Delta W^T \alpha^* D \Delta W + e_k \Delta W^T \alpha^* D \\
- e_k \Delta W^T \alpha^* D < 0
\]

(25)

Then the learning rate of the second layer and the third layer are defined as:

\[
\eta_{BP}^{(3*)} < \left( \frac{1}{2} \frac{|L| |Z| e_k \frac{\partial e_k}{\partial w_{1,z,k}^{(3*)}}}{\eta_{BP}^{(3*)}} \right)^{-2} \\
\eta_{BP}^{(2*)} < \left( \frac{1}{2} \frac{|L| |Z| e_k \frac{\partial e_k}{\partial w_{2,j,k}^{(2*)}}}{\eta_{BP}^{(2*)}} \right)^{-2}
\]

(26)

Additionally, the range of the momentum factor can be mathematically expressed as follows.

\[
\alpha_{BP}^{(3*)} < \left| 2e_k \left( \frac{\partial e_k}{\partial w_{1,z,k}^{(3*)}} \right)^2 \frac{\partial e^T_k}{\partial w_{1,z,k}^{(3*)}} \right| \frac{\eta_{BP}^{(3*)}}{\Delta w_{1,z,k}^{(3*)T}}
\]

(27)

\[
\alpha_{BP}^{(2*)} < \left| 2e_k \left( \frac{\partial e_k}{\partial w_{2,j,k}^{(2*)}} \right)^2 \frac{\partial e^T_k}{\partial w_{2,j,k}^{(2*)}} \right| \frac{\eta_{BP}^{(2*)}}{\Delta w_{2,j,k}^{(2*)T}}
\]

IV. IMPLEMENTATION

In order to facilitate the reproducibility of the findings of this paper and enable the utilization of this method in other scenarios, the implementation of the proposed neural network (NN) controller is elucidated. This will be accomplished by providing a comprehensive explanation of the block diagram and subsequently presenting the implementation.
A. Block Diagram for NN Controller

Fig. 3 depicts the block diagram of the neural network (NN) controller utilized within the permanent magnet synchronous motor (PMSMs) control loop. Fig. 3 (a) illustrates the conventional configuration of the NN controller. The input layer of the NN controller consists of four neurons: the reference value, the actual value, the error, and a bias term with a fixed value of 1. The number of neurons in the hidden layer can be customized considering the desired accuracy and execution speed. Two activation functions are incorporated within the controller, enabling a tailored NN controller design for specific tasks. The cost function $E$ is used to optimize the weight parameter $W$ through gradient descent, with the sign function applied outside the finite difference of Jacobian information (SIFD).

Fig. 3 (b) presents the NN controller guided by plant physics, which incorporates stability assurance. The acronym PHY denotes the physics-guided update law for the Jacobian information of the PMSMs. A suitable learning rate can be determined by considering both the best-case scenario and the actual tracking processes. Fig. 3 (c) represents the field-oriented control (FOC) loop based on the plant physics-guided NN controller. Additionally, the figure includes a proportional-integral controller (PI), coordinate transformation matrices (T1 and T2), and the space vector pulse width modulation method (SVPWM).

B. Program Implementation for NN Controller

The following is the realization of the physics-informed NN controller proposed in this paper, which outlines the strict stability insured by the physics-guided update law of the neural network control:
1. Initialize the number of neurons and layers in the neural network.
2. Randomly initialize the weights and corresponding update parameters.
3. Enter the system loop.
4. At each step:
   a. Calculate the output error, such as the speed error for the speed loop or the current error for the current loop.
   b. Normalize the input values to accommodate the characteristics of the activation function.
   c. Obtain the control parameters as the output of the neural network controller.
   d. Apply the control value for this step to the motor.
   e. Update the incremental weight of the neural network in reverse order using the physics-guided Jacobian information.
5. Check the stable range and assign a safe update parameter to optimize the neural network controller.
6. Repeat steps 4 for subsequent execution cycles.

The above algorithm outlines initializing the neural network, calculating errors, obtaining control parameters, updating weights, and optimizing the neural network controller while ensuring strict stability.

V. Experimental Validation

A. Experimental Set-Up

As shown in Fig. 4, the experimental setup in this study consists of two small inertia permanent magnet synchronous motors. One motor serves as the test motor to verify the algorithm, while the other is the auxiliary motor for loading the test motor, enabling specific loading experiments. Table I presents the motor parameters of the test motor, revealing its small inertia, relatively low inductance, and magnetic flux. Consequently, it is easier to observe speed fluctuations, and the experimental results are more pronounced. In the basic control loop of the motor, the speed loop control frequency is set at 900 Hz, and the MOSFET switching frequency is 16,000 Hz. Motor speed and current are measured using a Tektronix oscilloscope, while a motor control workbench controls the start-stop state.

B. Dynamic and Static Performances using BPNN

Fig. 5-7 in this chapter demonstrate the performances of the NN-assisted PI controller after stability constraints. Fig. 5 shows the results of the motor accelerating from a stationary state to 900 rpm. Among the three methods, the overshoot of the BP-FD-based method is comparatively smaller, and the rise time during the acceleration phase is shorter, indicating faster acceleration. In this operating condition, the BP-PHY method exhibits a slightly larger overshoot. Overall, BP-based methods demonstrate faster acceleration compared to the PI controller. However, from a static performance perspective, the drawback of the BP method is evident, as it introduces oscillations during the steady-state process, which aligns with the issues identified in theory. Fig. 6 illustrates accelerating from 500 rpm to 900 rpm, showing that all three BP methods exhibit fast transient speeds. Nevertheless, the main problem remains...
in the oscillations during steady-state, where the oscillation frequency decreases at lower speeds, the oscillation amplitude increases. Fig. 7 presents the results of the motor decelerating from 900 rpm to 500 rpm. although BP-PHY exhibits a larger overshoot, it also exhibits a relatively quick response time. The common drawback of the three methods remains the issue of steady-state oscillations. Part IV.D proposes a trigger condition to address the aforementioned issues to optimize the neural network control method.

C. Performances of Anti-disturbance using BPNN

Measuring the performance of a controller its disturbance rejection capability is indispensable. Therefore, this section primarily analyzes the disturbance rejection performances of the tested motor under sudden load changes. Since a small inertia motor was used in the experiment, even a small sudden torque would cause a significant overshoot in the rotational speed. In this study, an additional torque of 0.2 Nm was applied. As shown in Fig. 8, all three machine learning methods, especially BP-SIFD, exhibited a larger overshoot than the baseline PI controller. However, recovery time was significantly improved, with the three methods reducing recovery time by three-quarters compared to the PI control. Fig. 9 shows the unloaded state with a torque of 0.2 Nm, and once again, the advantage lies in the recovery time. Combining both figures, regardless of the motor load, the oscillations in the motor’s steady state are significant, indicating a need to address this issue.

D. Dynamic and Static Performances using BPNN with Trigger

Based on the issue raised in the previous part, the continuous application of neural network learning during the steady-state phase results in oscillations in the steady-state range. This part proposes a small modification by adding a trigger condition. When this condition is met, the neural network control method is activated and when the condition is not met, the neural network control method is deactivated. From an intuitive perspective, the neural network control method is deactivated when the motor speed tracking error is below a certain threshold with 80 rpm. In contrast, when the speed tracking error exceeds this threshold, the neural network control method is activated, as shown in Fig. 10. This approach should effectively reduce oscillations in the steady-state range.
After this small modification, significant changes can be observed in the results from Fig. 11 to Fig. 13. The curve illustrates the motor start process in Fig. 11. The rise time maintains its characteristic fast response while reducing overshoot. The oscillations in the steady-state range have significantly decreased to an acceptable level, with the oscillations in the BP-PHY-TR (TR: Trigger) method almost disappearing. Fig. 12 demonstrates the superiority of the trigger approach during the acceleration process from 500 rpm to 900 rpm. It can be observed that the overshoot is nearly eliminated and that the significant oscillations at low speeds are effectively suppressed. The slightly larger oscillations in the BP-FD-TR method at 900 rpm are due to its division calculation method, while the oscillations in the BP-SIFD-TR and BP-PHY-TR methods are significantly reduced. Fig. 13, which shows the deceleration process from 900 to 500 rpm, further confirms the method’s superiority, as it shows a fast adjustment time and minimal steady-state oscillations.

E. Performances of Anti-disturbance using BPNN with Trigger

This part verifies the anti-disturbance capability of the entire control system using the trigger method. From Fig. 14, it can be seen that the three BPNN methods maintain the advantage of ultra-fast recovery time and exhibit a significant reduction in overshoot. Among the three methods, BP-PHY-TR has the smallest overshoot, confirming the superiority of the physics-guided neural network. Additionally, the static oscillations of the motor during loaded operation are significantly reduced. From Fig. 15, it can be observed that the neural network control methods maintain superior results across multiple indicators under the unloaded condition.

VI. Conclusion

This paper presents the design of a motor physics-guided neural network controller in field-oriented control for permanent magnet synchronous motors. The key innovation lies in identifying the critical component that influences the neural network controller, the Jacobian information term. The motor physics highly improves the explainability of NN controller, which guides the neural-network-based motor control parameter design in engineering practice. Comparisons of performances were conducted using three different methods: sign function with finite difference (SIFD), finite difference (FD),
and motor physics (PHY), resulting in significant improvements, especially the stable range. Furthermore, the stability issue, which is inherent in all neural network controllers, was addressed, making it convincing for industrial applications in the future. The update parameter was specially designed considering the discrete Lyapunov function to ensure the stability of the neural network controller. The simulation demonstrated that all designed neural network controllers were stable, validating the proposed theory. In the experimental results, this control strategy is prior to the recovery time, which reduces more than 2/3. Thus, the NN controller significantly improves dynamic performance compared to the widely used PI controller in PMSM.

In future research, several areas can be explored. First, there is a need to develop more precise discretization methods for the physics component of the neural network controller. Different optimization algorithms should be designed and compared with the existing optimization method. Optimal selection of the update parameter while ensuring stability should be investigated. The proposed scheme can further be extended to predictive neural network-based control to embed the explainable physical model into the neural network and compared with traditional model predictive control methods.

REFERENCES


Zhenxiao Yin (Graduate Student Member, IEEE) received the B.Sc. degree in mechanical engineering from the Paderborn University, Paderborn, Germany, in 2018. Then he received double M.Sc. degrees in Mechanical Engineering, Mechatronics and Robotics from the Technical University of Munich, Munich, Germany, in 2021 and 2022. He is currently working toward a Ph.D. degree in Robotics and Autonomous Systems at the Hong Kong University of Science and Technology (Guangzhou), Guangzhou and Hong Kong SAR, China. His research interests include learning-based control for electric machines and learning-based electric machine design.

Xu Chen received the B.S. degree from the Tongji University, Shanghai, China, in 2017 and M.S. degree from the University of Stuttgart, Stuttgart, Germany, in 2021, respectively. He is currently working toward the Ph.D. degree in Automatic Control at the RWTH Aachen University, Aachen, Germany. His research interests include data-driven modeling and model predictive control application.

Yang Shen received the B.S. and M.S. degrees from the North China University of Technology, Beijing, China, in 2018 and 2021, respectively. He is currently working toward the Ph.D. degree in Robotics and Autonomous Systems at the Hong Kong University of Science and Technology (Guangzhou), Guangzhou and Hong Kong SAR, China. His research interests include the research of special motor control and linear motor drive.

Xiangdong Su (Graduate Student Member, IEEE) Xiangdong Su received the B.Eng. and M.Eng. degrees in mechanical engineering from Shandong University, in 2017 and 2020 respectively. After that he joined SAIC group as an assistant engineer. He is currently the PhD student in The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China. His research interests include analytical calculation and multiphysics analysis of permanent-magnet synchronous machines.

Hang Zhao (Member, IEEE) received the B.Eng. and M.Eng. degrees from Huazhong University of Science and Technology, Wuhan, China, and the Ph.D. degree from City University of Hong Kong, Hong Kong SAR, China, all in electrical engineering in 2015, 2017, and 2021, respectively. He is currently an Assistant Professor at the Robotics and Autonomous Systems Hub, Systems Hub, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China, and also an Affiliate Assistant Professor at the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong SAR, China. He was a Research Fellow at The University of Hong Kong, Hong Kong SAR, China, in 2021. His research interests include electric machines and drives, applications of motor drives in robots and electrified transportation.