Physics-Guided Neural Networks for Inversion-Based Feedforward Control of a Hybrid Stepper Motor

Fan, Daiwei

Award date:
2022

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering

Physics-Guided Neural Networks for Inversion-Based Feedforward Control of a Hybrid Stepper Motor

by

D. Fan

MSC THESIS

Assessment committee

Chair:	Prof. dr. S. Weiland
Member 1:	Dr. M. Curti
Member 2:	Dr. M. Lazar
Advisory member 1:	Dr. S. Koekebakker
Advisory member 2:	MSc. M. Bolderman

Graduation

Program:	Electrical Engineering
Capacity group:	Control Systems Group
Supervisor 1:	Dr. M. Lazar
Supervisor 2:	MSc. M. Bolderman
Date of defense:	December 1, 2022
Student ID:	1594516
Study load (ECTS):	45

Declaration concerning the TU/e Code of Scientific Conduct for the Master's thesis

I have read the TU/e Code of Scientific Conduct[i].

I hereby declare that my Master's thesis has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct

Date

November 22 2022……..

Name

Daiwei Fan……..…….

ID-number

1594516……..……..

Signature

Daiwei Fan

……..……..

Submit the signed declaration to the student administration of your department.

[i] See: https://www.tue.nl/en/our-university/about-the-university/organization/integrity/scientific-integrity/
The Netherlands Code of Conduct for Scientific Integrity, endorsed by 6 umbrella organizations, including the VSNU, can be found here also. More information about scientific integrity is published on the websites of TU/e and VSNU

Physics-Guided Neural Networks for Inversion-Based Feedforward Control of a Hybrid Stepper Motor

Daiwei Fan
Eindhoven University of Technology
Department of Electrical Engineering
Control Systems Group

***Abstract*—Rotary motors such as hybrid stepper motors are widely used for different tasks. With the increasing requirement for accurate motion, providing a simple and feasible method to improve performance becomes a goal in industrial and commercial applications. One such tool is inversion-based feedforward control, its performance is limited by the accuracy of the inverse model. Typically, a model is derived from first-principle models based on physical knowledge. However, these models are unable to describe complex nonlinear phenomena that are omnipresent in practical applications. With the aim to also include these phenomena, a model based on the black-box neural network becomes popular due to its universal approximation capabilities. But it is difficult to train the neural network to capture the real dynamics. To address these issues, this paper investigates the design of physics-guided neural networks (PGNN) for nonlinear system identification and thereby combines physical knowledge in parallel with a neural network. The PGNN framework is validated on a real-life industrial hybrid stepper motor. Moreover, the PGNN feedforward controller does not require any modifications of the motors. In comparison with the physical model-based feedforward, the results show that the PGNN feedforward controller achieves significant improvements in tracking performance.**

***Index Terms*—Feedforward control, Machine learning, Stepper motor, System identification**

I. Introduction

With the increasing need for productivity and efficiency, there is a tendency to refine the automatic industrial process. Stepper motors generally have a simple structure and are manufactured at a low cost [1]. Ascribed to this, many automated applications such as printers, robotics, and automotive widely use stepper motors as actuators to handle different tasks [2], [3].

However, although stepper motors are easy to implement in open-loop by controlling the pulse amount and pulse frequency, they are prone to mechanical resonances due to the ripple generated in the electromagnetic torque [4], [3]. Additionally, some stepper motors have a step response with significant overshoot and a long settling time [5].

In order to improve performance, control algorithms such as micro-stepping achieve improvements by increasing the step resolution [4], [6]. There are still position-tracking errors when operating the stepper motor on a constant velocity reference [6], [7]. Moreover, there is a lack of ability to reject the load disturbance [3], [6]. These factors restrict the performance of the stepper motor with open-loop in high-precision positioning applications. Attempts have been made in the past decades to use closed-loop control for enhancing the tracking performance of stepper motors. Among these closed-loop control techniques, is field-oriented control (FOC) with proportional-integral (PI) feedback control [6]. In FOC, the stator current vector is generally controlled by the PI controller such that it is orthogonal to the rotor magnetic field [8]. Consequently, electromagnetic torque retains maximum value constantly, which reduces torque ripples and leads to a smoother operation of the motor [3]. However, the traditional linear PI controller cannot efficiently compensate for the effects of back electromotive force (EMF) and inductance [9]. Furthermore, this method requires coordinate transformation to linearize the dynamics [10]. In [11], it is mentioned that a closed-loop commutation delay may result in unusual changes in the behavior of stepper motors.

To improve the tracking performance of motion systems further, feedforward control is a technique, which is independent of feedback control. In comparison with model-free feedforward, the inversion-based feedforward signal is generated by passing the reference through a model of the inverse system dynamics [12], [13]. Tracking performance achieved by the inversion-based feedforward controller is limited by the accuracy of the model describing the inverse system dynamics [12]. This method also has robustness against different references. Therefore, the inversion-based feedforward controller is appealing but its performance is determined by the quality of the inverse model.

An inverse system model can be obtained based on physical knowledge, for example, the first principle modelling, see, e.g., [14]. However, there exist some uncertainties and unknown dynamics such as the parasitic torques caused by manufacturing tolerances, and ripple torques caused by detent torque and back electromotive force (back EMF). Therefore, it is a challenge to identify the complete dynamics only based on physical knowledge due to its limited approximation capabilities.

To solve this problem, increasing attention has been paid to the use of black-box neural networks (NN) to approximate the complete system dynamics, which is explained by their universal approximation capability [15], see also [16]. However, during the process of training, the black-box NN could learn a pseudo relation, especially when the training data set is small.

In spite of using cross-validation or Bayesian regularization, this phenomenon still exists [17]. As a consequence, the output of the black-box NN lacks compliance with respect to the known laws of physics [18]. The identified NN model can perform well on the training data, even when it has learned spurious relationships. Under these circumstances, the feedforward signal generated by the incorrect model can yield unsafe behavior. Therefore, the lack of physical interpretability and unsafety impedes applying NN-based controllers in industrial applications.

Many strategies have been proposed to overcome these undesirable effects. In [19], it is mentioned that a linear model can be inserted additively in parallel to a nonlinear model. An advanced method called physics-guided neural networks (PGNN) feedforward controller is proposed in [17]. The PGNN embeds an a priori known physical model within a black box neural network structure. A different approach called physics-informed neural networks (PINN) penalizes deviations of the NN output with respect to the physical model output in the training loss function, see [12], [18].

The above articles demonstrated that these advanced methods outperform physics-based methods on a coreless linear motor. However, applying these advanced methods to a rotary motor is not considered yet. In this paper, we investigate the design of PGNN to identify the inverse system of a two-phase hybrid stepper motor in discrete–time domain. Notice that, compared to a linear motor, besides the different parameters and dynamics, the rotor position of a hybrid stepper motor has recurring behaviour due to its rotary motion. Consequently, this paper also discusses how to solve this issue. The feedforward controller based on a PGNN is validated in the real-life device and works with typical FOC control. The experimental results show that PGNN feedforward outperforms the benchmarks, which are the physics-based and black-box NN-based feedforward. Overall, this paper enhances the work in [17] by demonstrating a new practical application: a hybrid stepper motor.

The remainder of this paper is structured as follows: Section 2 presents a mathematical model of a hybrid stepper motor, FOC control with dq transformation, inverse system identification, and feedforward design. The problem statement, motivation, and goal are described successively in Section 3. Section 4 demonstrates the advanced methodology for discrete–time inverse system identification, which is used to design the feedforward controller. In Section 5, the performance of advanced methodology is validated. Finally, the conclusions are summarized in Section 6.

II. Preliminaries

A. System description and modelling

In this work, we consider a two-phase hybrid stepper motor that is provided by Canon Production Printing, see Fig. 1. Typically, the modelling of a hybrid stepper motor can be divided into two independent parts, the mechanical part and the electromagnetic part.

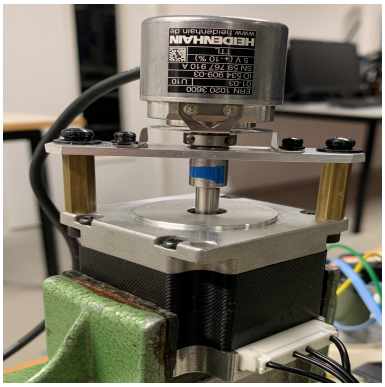

Fig. 1. Hybrid stepper motor FL57STH51-2804A manufactured by FULLING MOTOR.

The electromagnetic part is modelled in the stationary reference frame (ab-frame) as

$$\begin{bmatrix} \frac{di_a}{dt} \\ \frac{di_b}{dt} \end{bmatrix} = \frac{1}{L}\left(\begin{bmatrix} v_a \\ v_b \end{bmatrix} - R \begin{bmatrix} i_a \\ i_b \end{bmatrix} - \begin{bmatrix} F_{emf_a} \\ F_{emf_b} \end{bmatrix} \right), \quad (1)$$

where L is phase inductance and R is resistance. The currents in coils are represented by i_a,i_b and v_a,v_b are terminal voltages respectively. Additionally, F_{emf_a} and F_{emf_b} are self-induced voltage, also known as the back electromotive force (back EMF), which is modelled as

$$\begin{aligned} F_{emf_a} &= -K_m \omega \sin(Ny) \\ F_{emf_b} &= K_m \omega \cos(Ny), \end{aligned} \quad (2)$$

where K_m is the motor constant and N is the number of rotor teeth. The mechanical rotor angle and the mechanical angular velocity are y and ω, respectively. The electromagnetic part uses the current $i_{a,b}$ in the coil windings to produce an electromagnetic torque T_e. The resulting torque acts on the mechanical part and generates the motion of the rotor [2]. These ordinary differential equations show that the electromagnetic part not only contains nonlinearities but is also influenced by the rotor position from the mechanical part.

The model of mechanical part can be described as

$$\frac{dy}{dt} = \omega \quad (3a)$$

$$\frac{d\omega}{dt} = \frac{1}{J}(T_e - T_f - T_d - T_l), \quad (3b)$$

where J is the moment of inertia of the rotor. The electromagnetic torque T_e is the input from the electromagnetic part in the driving direction, which can be modelled as

$$T_e = K_m(-i_a \sin(Ny) + i_b \cos(Ny)). \quad (4)$$

The total mechanical friction torque T_f is usually modelled as

$$T_f = B\omega, \quad (5)$$

where B is the viscous friction coefficient. The detent torque T_d is generated due to the permanent magnetic rotor attracting the salient iron teeth of the stator. Note that this torque is

always present, even if the motor is de-energized [2]. The detent torque is expected to repeat with every stator phase and every rotor pole which determines the steady states of the rotor. The detent torque can be modelled by Fourier-approach that accounts for multiple spectral components [2] as

$$T_d = \sum_{k=1}^{n} a_k \cdot \sin(kNy), \tag{6}$$

where a_k is the amplitude of the k-th component and n is the total number of modelled spectral components. The load torque T_l depends on the different applications, which could be time-varying.

However, modelling based on existing knowledge can not describe the inherent dynamics without any deviations. For example, in practice, there are parasitic effects due to manufacturing tolerances such as the variation of coil dimensions, the variation of magnet properties and dimensions, and the variation of assembly. These tolerances generate parasitic Lorentz torques and parasitic reluctance torques in multiple directions [20].

Additionally, there is the ripple torque caused by the variation of the phase inductance with rotor position and the harmonics from the back EMF. From [3], it is mentioned that salient Hybrid Stepper Motor has some similar behaviors to Permanent Magnet Synchronous Motor due to the variable reluctance. Therefore, the inductance L is nonlinear with the rotor position y rather than a constant

$$L = L_0 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \Delta L \begin{bmatrix} \cos(2Ny) & \sin(2Ny) \\ \sin(2Ny) & \cos(2Ny) \end{bmatrix}, \tag{7}$$

where L_0 is the nominal inductance and ΔL is the variation of inductance. Therefore, the electromagnetic dynamics (1) is rewritten in matrix notation as

$$\begin{bmatrix} \frac{di_a}{dt} \\ \frac{di_b}{dt} \end{bmatrix} = L^{-1} \left(\begin{bmatrix} v_a \\ v_b \end{bmatrix} - R \begin{bmatrix} i_a \\ i_b \end{bmatrix} - \frac{d(Ny)}{dt} \frac{\partial L}{\partial (Ny)} \begin{bmatrix} i_a \\ i_b \end{bmatrix} - \begin{bmatrix} F_{emf_a} \\ F_{emf_b} \end{bmatrix} \right). \tag{8}$$

The performance of controllers is degraded when the controller design ignores the presence of these mentioned parasitic torques, electromagnetic disturbances, and other unknown dynamics.

B. Traditional control design

As shown in Fig. 2, in the outer loop, a linear proportional-integral-derivative (PID) position feedback controller to track the desired position y^* is designed as

$$u_{fb} = k_P (y^* - y) + k_I \int_0^t (y^* - y)\, dt + k_D \left(\frac{dy^*}{dt} - \frac{dy}{dt} \right). \tag{9}$$

The feedforward u_{ff} is designed to improve the tracking performance. Therefore, the complete control law is denoted as

$$T_e^* = u_{fb} + u_{ff}, \tag{10}$$

where T_e^* is the desired torque.

In fact, the stator current $i_{a,b}$ is the actual input in a hybrid stepper motor. The inner loop formed by the red dashed line in Fig. 2 is FOC control with DQ transformation, which aims to maximize the torque to improve performance. As shown in Fig. 3, the original sinewave reference signal is avoided by using FOC with DQ transformation. The current i_a, i_b in the stationary reference frame are converted into d-axis and q-axis components as i_d, i_q by DQ transformation [6]

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} \cos(Ny) & \sin(Ny) \\ -\sin(Ny) & \cos(Ny) \end{bmatrix} \begin{bmatrix} i_a \\ i_b \end{bmatrix}. \tag{11}$$

The inverse DQ transformation reverts voltage v_d, v_q from dq coordinate to ab coordinate as winding voltages v_a, v_b as

$$\begin{bmatrix} v_a \\ v_b \end{bmatrix} = \begin{bmatrix} \cos(Ny) & -\sin(Ny) \\ \sin(Ny) & \cos(Ny) \end{bmatrix} \begin{bmatrix} v_d \\ v_q \end{bmatrix}. \tag{12}$$

Applying (11) and (12) on the electromagnetic part (1) results in a linear model as

$$\begin{bmatrix} \frac{di_d}{dt} \\ \frac{di_q}{dt} \end{bmatrix} = \frac{1}{L} \left(\begin{bmatrix} v_d \\ v_q \end{bmatrix} - \begin{bmatrix} R & -LN\omega \\ LN\omega & R \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} - \begin{bmatrix} 0 \\ K_m \omega \end{bmatrix} \right). \tag{13}$$

Therefore, the nonlinear control problem is converted to a linear control problem. Typically the PI controllers are used to generate voltage input by minimizing the error between the desired current and the actual current. The current i_d is forced to zero, then the total stator current is equal to q-axis stator current i_q since

$$\vec{I}_{dq} = \sqrt{i_d^2 + i_q^2} = i_q \quad \text{for } i_d = 0. \tag{14}$$

To couple the inner loop with the outer loop, the desired current i_q^* is calculated based on (4) as

$$i_q^* = \frac{1}{K_m} T_e^*. \tag{15}$$

In the remainder of the paper, for ease of analysis, the torque T_e^* regarded as input of the system is denoted as $u(t)$. The mechanical rotor angle as the system output is still denoted as $y(t)$.

C. System identification and feedforward control design

To control the hybrid stepper motor in real life, the feedback controllers should be discretized. Likewise, in order to implement the feedforward design, the inverse dynamical system in the discrete-time domain can be written as

$$u(t) = \xi\big(\phi(t)\big), \tag{16}$$

where $\xi(\cdot)$ is a nonlinear function that describes the dynamics with the regressor $\phi(t) = [y(t+n_a), ..., y(t-n_b), u(t-1), ..., u(t-n_c)]^T$ that corresponds to a nonlinear ARX model. The parameters n_a, n_b, and n_c describe the order of the system.

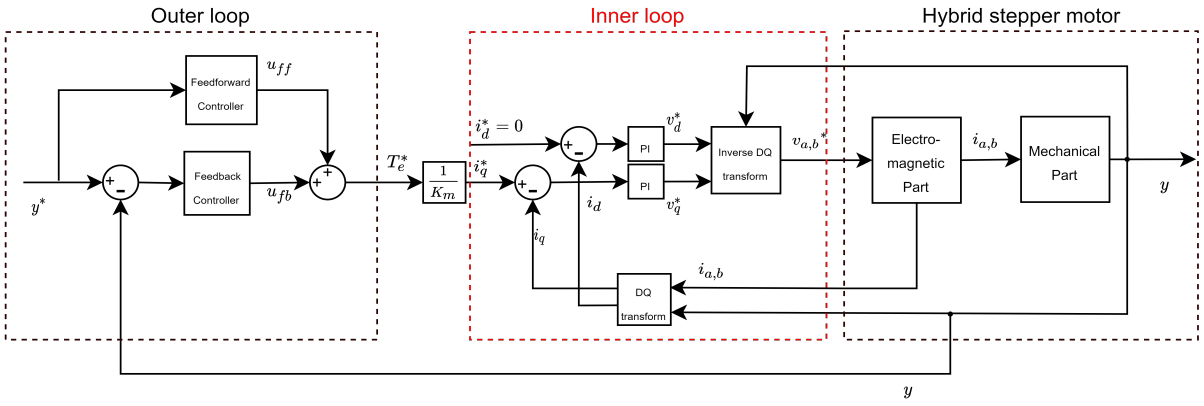

Fig. 2. The total control scheme of the hybrid stepper motor. The area in the red dashed line is the schematic control loop of the electromagnetic part.

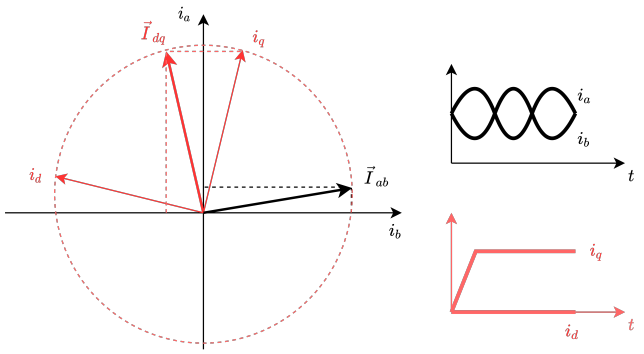

Fig. 3. The illustration of Field Oriented Control. The black line is current in the stationary reference frame and the red line is current in the rotating reference frame.

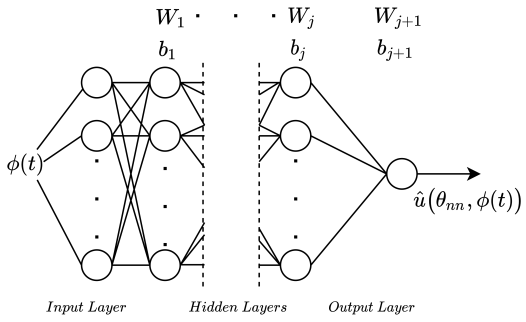

Fig. 4. Black-box neural network structure.

We aim to approximate an inverse system with the model for inversion-based feedforward controller synthesis with the following steps

1) Data generation.
2) Model structure.
3) Train Model.
4) Performance testing.

The training data set $Z^N = \{u(0), \dots, u(N-1), y(0), \dots, y(N-1)\}$ is generated on the system, which satisfies the inverse dynamics (16).

Definition II.1. A physics–based model is defined as

$$\hat{u}\left(\theta_{phy}, \phi(t)\right) = f\left(\theta_{phy}, \phi(t)\right), \tag{17}$$

where $f(\cdot)$ is a function that describes the known dynamics with the parameter θ_{phy}.

Therefore, (16) can be rewritten with (17) as

$$u(t) = f\left(\theta_{phy}, \phi(t)\right) + g\left(\phi(t)\right), \tag{18}$$

where $g\left(\phi(t)\right) := \xi\left(\phi(t)\right) - f\left(\theta_{phy}, \phi(t)\right)$ is a nonlinear function that describes all unmodelled dynamics, such as the aforementioned parasitic effects, electromagnetic disturbances, etc.

Definition II.2. To identify the inverse dynamics (16) by a nonlinear model, another method as shown in Fig. 4, the black box NN model is defined as

$$\hat{u}\left(\theta_{nn}, \phi(t)\right) = W_{j+1} a_i\left(\cdots a_1(W_1\phi(t) + b_1)\right) + b_{j+1}, \tag{19}$$

where the parameter θ_{nn} contains all the weights W and biases b. Additionally, a_i denotes the nonlinear activation functions used in the i-th hidden layer and j is the number of hidden layers.

In order to fit the model, e.g., (17) or (19), we choose the parameters theta according to an identification criterion. Typically, the identification criterion is chosen as the minimization of a cost function

$$\hat{\theta} = \arg\min_{\theta} V\left(\theta, Z^N\right), \tag{20}$$

where the cost function is often the quadratic function

$$V\left(\theta, Z^N\right) = \frac{1}{N}\sum_{t=0}^{N-1}\left(\hat{u}(\theta, \phi(t)) - u(t)\right)^2. \tag{21}$$

After the system identification procedure, the inversion–based feedforward is generated by passing the desired reference $r(t)$ through the estimated model as

$$u_{ff}(t) = \hat{u}\left(\hat{\theta}, \phi_{ff}(t)\right), \tag{22}$$

where estimated parameter $\hat{\theta}$ is obtained by (20) and $\phi_{ff}(t) := [r(t+n_a), ..., r(t-n_b), u_{ff}(t-1), ..., u_{ff}(t-n_c)]^T$.

Remark. ***2.1:***

For simplicity, we consider a physical model with $T_d = 0$ and $T_l = 0$. In addition, electrical dynamics is much faster than mechanical dynamics by using PWM drivers assembled with PI current controller [7], [21], thus it is possible to only consider the mechanical dynamics [22]. Thereby, we derive a physics-based model that is linear in the parameters as an example of (17), which is given as

$$\begin{aligned} \hat{u}\big(\theta_{phy}^{lip}, \phi(t)\big) &= \theta_{phy}^{lip} T\big(\phi(t)\big) \\ &= \begin{bmatrix} J \\ B \end{bmatrix}^T \begin{bmatrix} \delta^2 y(t) \\ \delta y(t) \end{bmatrix}, \end{aligned} \tag{23}$$

where $T(\cdot)$ is a set of functions that reshape the regressor $\phi(t) := [y(t), y(t-1), y(t-2)]^T$ based on physical knowledge, and $\delta := \frac{1-q^{-1}}{T_s}$ denotes the backward Euler discretization with backward shift operator q^{-1} and sampling time T_s. After identification of $\hat{u}\big(\theta_{phy}^{lip}, \phi(t)\big)$ that is obtained by using a least squares solver $''/''$ in Matlab, the physics-based feedforward controller is given as

$$\begin{aligned} u_{ff}(t) &= \hat{\theta}_{phy}^{lip} T\big(\phi_{ff}(t)\big) \\ &= \begin{bmatrix} \hat{J} \\ \hat{B} \end{bmatrix}^T \begin{bmatrix} \delta^2 r(t) \\ \delta r(t) \end{bmatrix}. \end{aligned} \tag{24}$$

III. Problem Statement

The performance of the inversion-based feedforward depends on the accuracy of the model. A linear physics-based model usually is considered because it is easy to design and implement. In some circumstances, we are not satisfied with the linear model. The physics-based model (23) is unable to capture the complete inverse system dynamics due to its limited approximation capabilities. It cannot describe the $g\big(\phi(t)\big)$ in (18). On the other hand, the black-box NN model (19) is theoretically able to approximate any dynamical system satisfying (16) [12]. But it is generally difficult to learn and respect underlying physical laws.

As shown in Fig. 5, results from the simulation revealed that although the position tracking error is reduced by the traditional physics-based feedforward, there are still imperfections, especially at changing acceleration. Additionally, the feedforward control signals generated by the black box NN model have some peaks and different magnitudes. However, it performs worse than the physics-based model. There are many possible reasons to explain it. For example, even though the training data set Z^N is generated on the system (16), it is still possible that the data set does not cover the domain of interest, or that training is not enough convergence.

Therefore, the goal of this paper is to improve the tracking performance of a hybrid stepper motor by designing a PGNN inversion–based feedforward controller. Motivated by the mentioned limitations of traditional methods, the following issues are considered:

- How to use the benefits of physical models and NN to design an advanced model with higher accuracy?

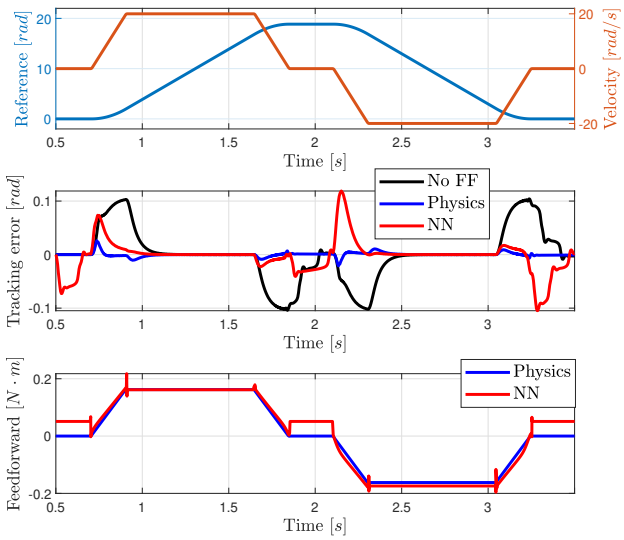

Fig. 5. The tracking error and feedforward signal by using traditional methods in the simulation: the physics-based model (23) and the NN-based model (19). The parameters $\hat{J}$ and $\hat{B}$ are identified by equations (20).

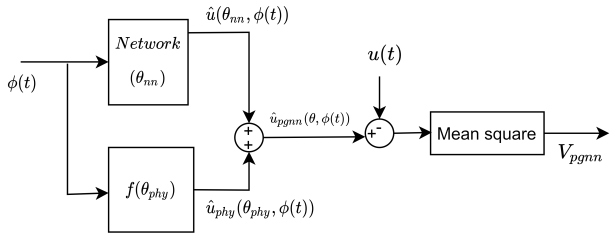

Fig. 6. Physics-Guided Neural Network (PGNN) schematic illustration.

- Without any modifications of the motor structure, how to design an implementable feedforward controller? The developed method should work with existing feedback controllers in FOC.

IV. Physics-Guided Neural Network

This section will elaborate on the procedures to obtain a PGNN feedforward controller by following the steps in Section 1. II-C.

Data generation:

In this paper, the training data set Z^N is generated by collecting the input $u(t)$ and output $y(t)$ with sampling time $T_s = 6.25 \times 10^{-4} s$. As mentioned in [17], closed-loop identification requires exciting the system persistently. A third-order exciting reference signal is designed, where position $r(t) \in \{-3 \cdot 2\pi, 3 \cdot 2\pi\}$ radians. The maximum of velocity, acceleration, and jerk are chosen as $|\frac{d}{dt} r(t)| < 15 \ \frac{rad}{s}$, $|\frac{d^2}{dt^2} r(t)| < 80 \ \frac{rad}{s^2}$, and $|\frac{d^3}{dt^3} r(t)| < 1000 \ \frac{rad}{s^3}$, respectively. Moreover, a zero–mean white noise with a variance of $2 \times 10^{-4} N \cdot m$ is added to dither the input $u(t)$ to allow the system to explore the various possible positions and velocities.

Define PGNN structure:

As shown in Fig. 6, the PGNN combines the predicted output of the NN and the output of the physical model as a single output, which is defined as

$$\hat{u}\left(\theta_{pgnn}, Z^N\right) = \hat{u}\big(\theta_{nn}, T(\phi(t))\big) + \hat{u}(\theta, \phi(t)). \tag{25}$$

In this paper, the NN model $\hat{u}(\theta_{nn}, \phi(t))$ has one hidden layer and eight neurons. The sigmoid function is selected as the activation function

$$a_1(x) = \frac{1}{1+e^{-x}}. \tag{26}$$

The simulation results show that increasing the dimensions of NN such as increasing the number of hidden layers or the number of neurons does not improve the performance of feedforward. Other activation functions such as tanh and Relu are worse than the sigmoid function when using the same NN dimension. To improve the PGNN, we design another physical model based on exploiting the known dynamics further, thus the physics-based model (23) is rewritten as

$$\begin{aligned}\hat{u}\big(\theta_{phy}^{lip}, \phi(t)\big) &= \theta_{phy}^{lip} T\big(\phi(t)\big) \\ &= \begin{bmatrix} J \\ B \\ 0 \end{bmatrix}^T \begin{bmatrix} \delta^2 y(t) \\ \delta y(t) \\ y(t) \end{bmatrix},\end{aligned} \tag{27}$$

which represents that we know some parasitic effects are related to the motor position $y(t)$. In fact, $g\big(\phi(t)\big)$ is impossible to model by position $y(t)$, thus its parameter in θ_{phy}^{lip} is defined as zero. The PGNN only allows the black box NN to explore the parasitic effects based on the training data set Z^N. Therefore the total PGNN model (25) is given by

$$\begin{aligned}\hat{u}\left(\theta_{pgnn}, Z^N\right) =& W_2\big(a_1(W_1 T\big(\phi(t)\big) + b_1)\big) + b_2 \\ &+ \theta_{phy}^{lip} T\big(\phi(t)\big).\end{aligned} \tag{28}$$

The experimental results show that the PGNN (28) improves the performance in comparison to PGNN using a standard physics-based model (23).

Train PGNN:

The PGNN model is trained based on (20) to find the parameters $\hat{\theta}$. The loss function (21) can be rewritten as

$$\begin{aligned}V_{pgnn}\left(\theta, Z^N\right) = \frac{1}{N}\sum_{t\in Z^N}^{N} \big(\hat{u}(\theta_{nn}, \phi(t)) + \\ \hat{u}(\theta_{phy}, \phi(t)) - u(t)\big)^2.\end{aligned} \tag{29}$$

There are two alternative options to identify the nonlinear system:

(a) Simultaneous: train the nonlinear θ_{nn} and the linear model θ_{phy} together.

(b) Sequential: first identify $\hat{\theta}_{phy}$ in (27) using (20), (21), then identify $\hat{\theta}_{nn}$ using (29) with $\theta_{phy} = \hat{\theta}_{phy}$ fixed.

Option (a) generally will yield a higher model accuracy at least on the training data set because more parameters are available to be trained in comparison with option (b). But it can result in overparameterization because NN also identifies the physical model. After training PGNN with θ_{nn} and θ_{phy} together, the estimated value of rotor inertia $\hat{J}$ and friction coefficient $\hat{B}$ can be identified as negative values, which is physically inconsistent. In fact, option (b) can put constraints by fixing θ_{phy}. Therefore, in the remainder of this paper, we select option (b). The training environment is Python 3.8 with the Adam optimization algorithm which is improved stochastic gradient descent. By utilizing the Nvidia Cuda to increase the training speed, the average time consumption is 90 seconds, which does not require expensive GPU cards. In order to reduce the risk of converging into a local minima during training (but not prevented), each network is trained ten times with random weight initialization. Then the best model is determined by choosing the lowest training loss.

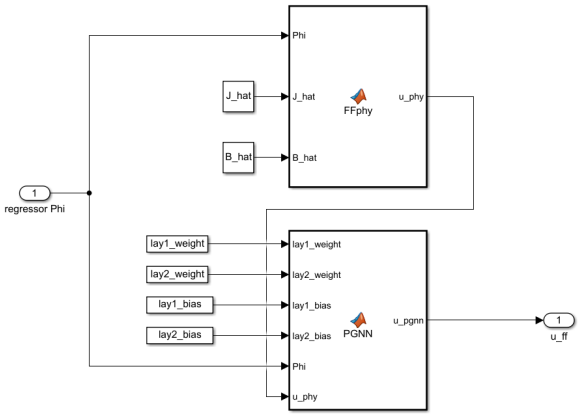

Fig. 7. The PGNN feedforward controller in Simulink.

Performance testing:

The performance of the PGNN will be evaluated on a real-life device. As shown in Fig. 7, the PGNN feedforward controller (28) is implemented by Simulink block with Matlab function, which is compiled by dSpace. The tracking error $e(t) := r(t) - y(t)$ is defined by the difference between the reference $r(t)$ and the actual position $y(t)$. The numerical assessments of the tracking performance in this paper are the mean absolute value and the maximum absolute value of $e(t)$, which are

$$MAE(e(t)) = \frac{1}{N_s}\sum_{t=0}^{N_s-1} |e(t)|, \tag{30}$$

with N the number of samples of the reference and

$$MAX(e(t)) = max(|e(t)|). \tag{31}$$

Remark. ***4.1:***

The rotor position of a hybrid stepper motor has recurring behaviour due to its rotary motion, that is $y(t) = y(t) + 2n\pi$ with $n \in \{0, 1, 2, 3\cdots\}$. Therefore, we reshape the physics-based model from (27) by limiting the motor position $y(t)$ between $\{0, 2\pi\}$ as

$$\begin{aligned}\hat{u}\big(\theta_{phy}^{lip}, \phi(t)\big) &= \theta_{phy}^{lip} T\big(\phi(t)\big) \\ &= \begin{bmatrix} J \\ B \\ 0 \end{bmatrix}^T \begin{bmatrix} \delta^2 y(t) \\ \delta y(t) \\ mod\big(y(t)\big) \end{bmatrix},\end{aligned} \tag{32}$$

where modulo operation $mod\big(y(t)\big)$ takes the remainder after division $\frac{y(t)}{2\pi}$.

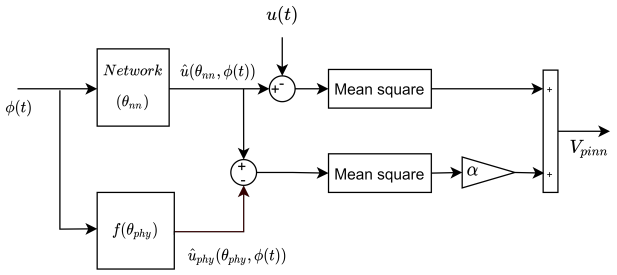

Fig. 8. Physics-Informed Neural Network (PINN) schematic illustration.

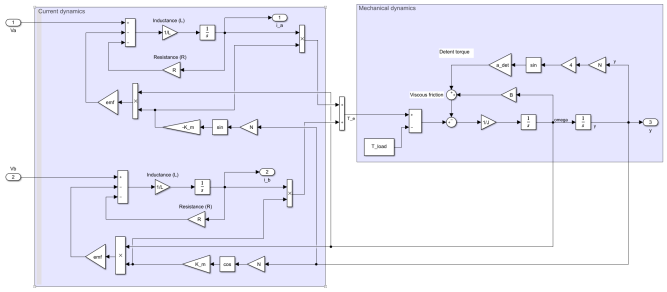

Fig. 9. The hybrid stepper motor in the Simulink.

Remark. ***4.2:***

In comparison with improving the physical model, another approach is to modify the structure of NN. Therefore, the recurrent neural network (RNN) is considered in this paper. The black box NN model (19) can be changed by simple Elman RNN as

$$h(t) = a_i(W_{hh}h(t-1) + b_{hh} + W_{ih}T\big(\phi(t)\big) + b_{ih}), \quad (33a)$$
$$\hat{u}\big(\theta_{rnn}, \phi(t)\big) = W_o h(t) + b_o, \quad (33b)$$

where $h(t)$ is the hidden state and $h(t-1)$ is the hidden state of the previous layer or the initial hidden state. In (33), the RNN has an internal memory hidden state h that enables it to remember historical input. In another word, the previous output of the hidden state is fed as input to the current step. Therefore, the RNN is able to make decisions by considering current input alongside learning from previous input. Additionally, a complicated RNN long short-term memory (LSTM) is used in to test the performance. The LSTM uses gates to regulate the flow of information as

$$i_{ls}(t) = \sigma\left(W_{ii}T\big(\phi(t)\big) + b_{ii} + W_{hi}h_{ls}(t-1) + b_{hi}\right) \quad (34a)$$
$$f_{ls}(t) = \sigma\left(W_{if}T\big(\phi(t)\big) + b_{if} + W_{hf}h_{ls}(t-1) + b_{hf}\right) \quad (34b)$$
$$g_{ls}(t) = \tanh\left(W_{ig}T\big(\phi(t)\big) + b_{ig} + W_{hg}h_{ls}(t-1) + b_{hg}\right) \quad (34c)$$
$$o_{ls}(t) = \sigma\left(W_{io}T\big(\phi(t)\big) + b_{io} + W_{ho}h_{ls}(t-1) + b_{ho}\right) \quad (34d)$$
$$c_{ls}(t) = f_t \odot c_{ls}(t-1) + i_{ls}(t) \odot g_{ls}(t) \quad (34e)$$
$$h_{ls}(t) = o_{ls}(t) \odot \tanh\left(c_{ls}(t)\right), \quad (34f)$$

where i_{ls} f_{ls} g_{ls} o_{ls} are the input, forget, cell, and output gates, respectively. σ is the sigmoid function in (26) and $\odot$ is the Hadamard product.

Remark. ***4.3:***

The PGNN introduces physical knowledge into the model set by combining the linear model in parallel with the NN model. In comparison with PGNN, another advanced model, e.g. see [12], physics-informed neural network (PINN) introduces physical knowledge in the loss function. As shown in Fig. 8, the loss function of PINN is given as

$$\begin{aligned} V_{pinn}\left(\theta, Z^N\right) &= V_D + \alpha V_{phy} \\ &= \frac{1}{N}\sum_{t\in Z^N}^{N}\left(\hat{u}(\theta_{nn}, \phi(t)) - u(t)\right)^2 + \\ &\alpha\frac{1}{N}\sum_{t\in Z^N}^{N}\left(\hat{u}(\theta_{nn}, \phi(t)) - \hat{u}(\theta_{phy}, \phi(t))\right)^2. \end{aligned} \quad (35)$$

where hyperparameter α is tuned to determine the relative weight of physical model compliance with respect to the data fit.

V. Simulation and Experiment

A. Simulation results

TABLE I
THE MOTOR PARAMETERS USED IN THE SIMULATION.

Parameters	Values
Moment of Inertial $J[\ \mathrm{kg}\cdot\mathrm{m}^2]$	2.8×10^{-5}
Friction Coefficient $B[\ \mathrm{N}\cdot\mathrm{m}\cdot\mathrm{s/rad}]$	8.0×10^{-3}
Resistance $R[\Omega]$	0.83
Inductance $L[\mathrm{H}]$	2.2×10^{-3}
Motor Constant $K_m[\ \mathrm{N}\cdot\mathrm{m/A}]$	0.36
Number of Poles N	50

Before proceeding to the real-life stepper motor, the hybrid stepper motor and controller design should be validated in the simulation. As shown in Fig. 9, the structure of a simulated motor by the Matlab Simulink is based on the equations explained in Section II-A. The mechanical part (3b) and electromagnetic part (1) are coupled by (4). The harmonic of the detent torque with $k = 4$ is typically dominant, such that (6) is simplified in the simulation as

$$T_d = a_d \sin(4Ny). \quad (36)$$

The detent torque and back-EMF are simulated by following (36) and (2), respectively. The values of motor parameters are listed in Table I.

Fig. 10 is the control scheme in the simulation by following the design of Fig. 2. The continuous-time position feedback controller is designed as

$$\begin{aligned} C_{fb}(s) &= C_{gain}C_{lead/lag}C_{integrator}C_{lpf} \\ &= \frac{6.013\times10^{-3}s^2 + 0.5907s + 7.54}{1.179\times10^{-5}s^3 + 7.626\times10^{-3}s^2 + s}, \end{aligned} \quad (37)$$

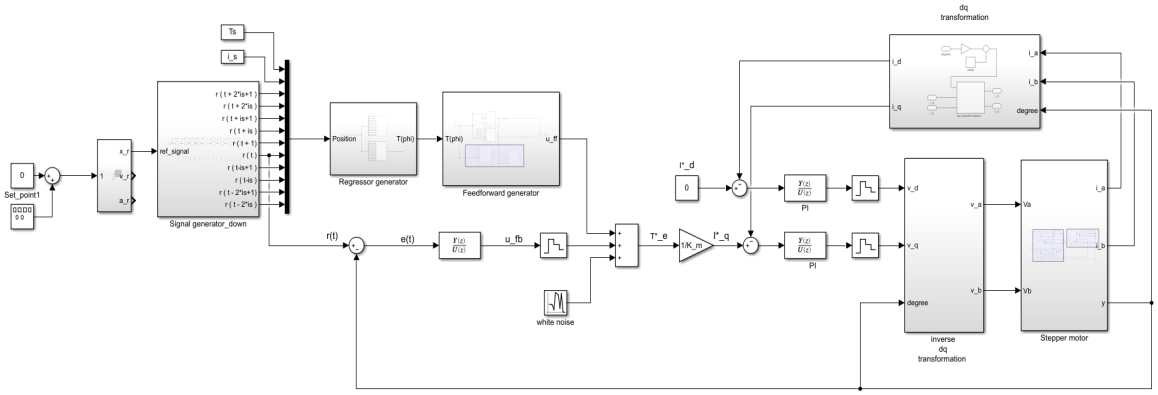

Fig. 10. Control scheme with the position feedback controller and feedforward controller in the simulation.

which is designed by using the loop–shaping technique for the linear mechanical dynamics (3a) and (3b)

$$P_{me}(s) = \frac{1}{Js^2 + Bs}. \tag{38}$$

This gives a 12.1 Hz open–loop bandwidth, a modulus margin of 2.3 dB, a phase margin of 74.4° and a gain margin of 18.5 dB. The controller $C_{fb}(s)$ is discretized by zero–order–hold with sampling time as

$$C_{fb}(z) = \frac{0.2693z^2 - 0.5224z + 0.2532}{z^3 - 2.64z^2 + 2.308z - 0.6676}. \tag{39}$$

In the traditional current loop design, only the linear dynamics of the electromagnetic part is considered. The open-loop transfer function based on (1) is

$$P_{current}(s) = \frac{1}{Ls + R}. \tag{40}$$

In order to avoid the impact on electronic devices caused by the large overshoot, the current loop is often designed as an overdamped system. In the cascade control, the bandwidth of the inner loop should be much faster than the outer loop [3]. Therefore, the PI controller is designed as

$$\begin{aligned} C_{current}(s) &= K_p + \frac{K_i}{s} \\ &= 30\big(12.1 \cdot 2\pi\big)L + \frac{30\big(12.1 \cdot 2\pi\big)R}{s}, \end{aligned} \tag{41}$$

which gives a 363.7 Hz open–loop bandwidth that is a factor of thirty larger than the bandwidth of the mechanical part P_{me}.

TABLE II
THE TRACKING RESULTS OBTAINED BY DIFFERENT FEEDFORWARD CONTROLLERS IN THE SIMULATION.

Evaluation / Cases	Loss	$MAE[rad]$	$MAX[rad]$
No FF	5.06×10^{-2}	8.99×10^{-2}	5.024×10^{-1}
Physics	2.90×10^{-3}	1.6×10^{-2}	1.02×10^{-1}
NN	5.04×10^{-4}	1.6×10^{-2}	9.08×10^{-2}
PGNN	4.54×10^{-4}	8.7×10^{-3}	7.3×10^{-2}
PINN	6.05×10^{-4}	9.8×10^{-3}	6.7×10^{-2}
PG-RNN	4.52×10^{-4}	8.5×10^{-3}	5.87×10^{-2}
PI-RNN	4.82×10^{-4}	9.3×10^{-3}	8.03×10^{-2}
PG-LSTM	3.91×10^{-4}	6.6×10^{-3}	8.18×10^{-2}
PI-LSTM	4.06×10^{-4}	8.3×10^{-3}	7.8×10^{-2}

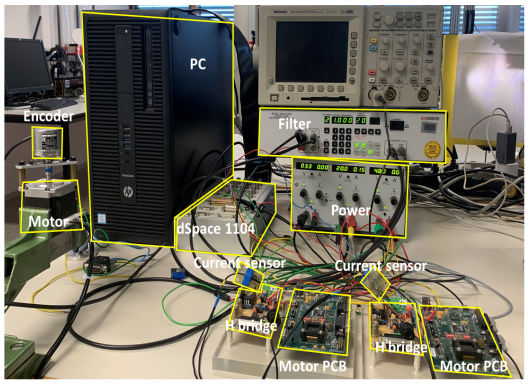

Fig. 11. The experimental device.

From Table II, PGNN and PINN all outperform the traditional feedforward methods. However, the structure of RNN and LSTM just improves the performance slightly. The stepper motor in simulation still can not present all behaviours of the motor in reality. The RNN and LSTM are expected to have an impressive performance in a complex environment. In this paper, the load T_l that can be time-varying is unavailable to implement in the real-life motor due to the time. Therefore, we still focus on the PGNN in the experimental results. Additionally, typical L_1 or L_2 regularization can also be implemented with PGNN, see appendix.

B. Experimental results

As shown in Fig. 11, the experimental device contains the hybrid stepper motor with a position encoder, the power supply, the current sensors, the H bridge, the motor PCB, the filter, and the dSpace 1104 connected to the computer. The dSpace MicrolabBox is able to run the files generated from Simulink. However, since the noise that originated from the MicrolabBox can not be reduced, the signal is amplified using an external analog filter [23].

Focusing on one example of a realistic application, the hybrid stepper motor starts from a standstill (0 RPM) to 15 $\frac{rad}{s}$ (approximately 143 RPM), then decelerates to the standstill. Note that reference $r(t)$ is a part of the reference in data

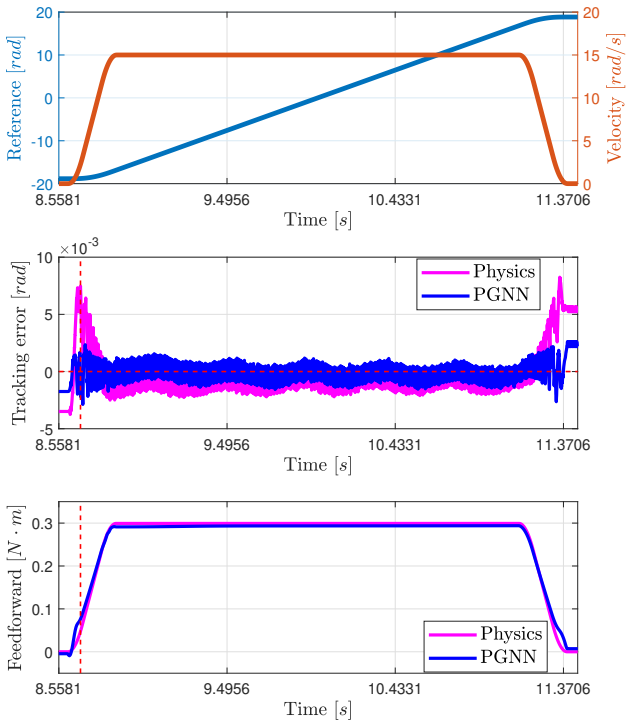

Fig. 12. The tracking results from the real-life hybrid stepper motor. The top subfigure is the reference $r(t)$; the middle subfigure is the tracking error $e(t)$; the bottom subfigure is the feedforward signal by using the regressor in (27).

generation. As shown in Fig. 12, in comparison with the physics-based controller, the transient peak error of PGNN at the beginning is reduced significantly. The mean absolute error of PGNN keeps sticking on the zero line.

TABLE III
THE TRACKING RESULTS BY USING THE REGRESSOR IN (27) IN THE REAL-LIFE DEVICE.

Evaluation / Cases	Loss	$MAE[rad]$	$MAX[rad]$
No FF	4.16×10^{-2}	5.2×10^{-2}	6.1×10^{-2}
Physics	1.70×10^{-4}	1.3×10^{-3}	8.2×10^{-3}
NN	9.07×10^{-5}	1.1×10^{-3}	8.3×10^{-3}
PGNN	8.64×10^{-5}	6.4×10^{-4}	2.9×10^{-3}
PINN($\alpha = 0.00001$)	8.89×10^{-5}	7.6×10^{-4}	3.3×10^{-3}

In some particular cases, engineers focus on the electrical angle $y_e[^\circ]$, which is $y_e = y\frac{N \cdot 180}{\pi}$. By applying this metric to the tracking evaluation Table III, PGNN feedforward can reduce the peak error from 23.5° to 8.3° and mean absolute error from 3.7° to 1.8°. Nevertheless, the regressor in (27) only considers the mechanical dynamics without the electromagnetic dynamics, where mechanical dynamics are still not incomplete. The results show that PGNN improves performance and enhances approximation capabilities.

Notice that, in contrast to identifying θ_{phy}^{lip} in (23), finding a solution of PGNN (28) based on (20) is a non–convex optimization problem. Despite the fact that PGNN could be

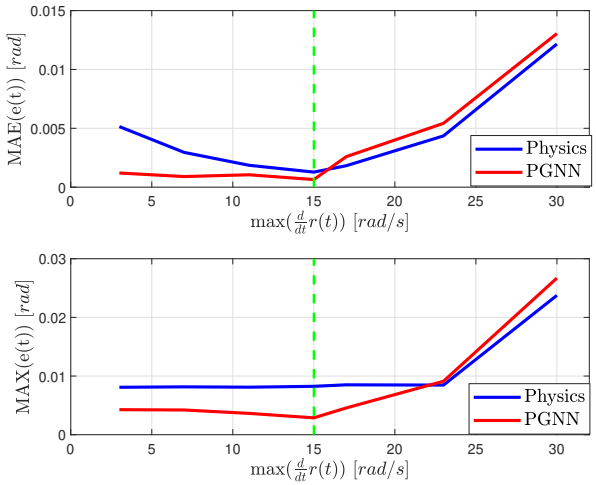

Fig. 13. MAE and MAX for the different feedforward controllers when used on a reference with varying maximum velocities. The maximum velocity in data generation is 15 $\frac{rad}{s}$.

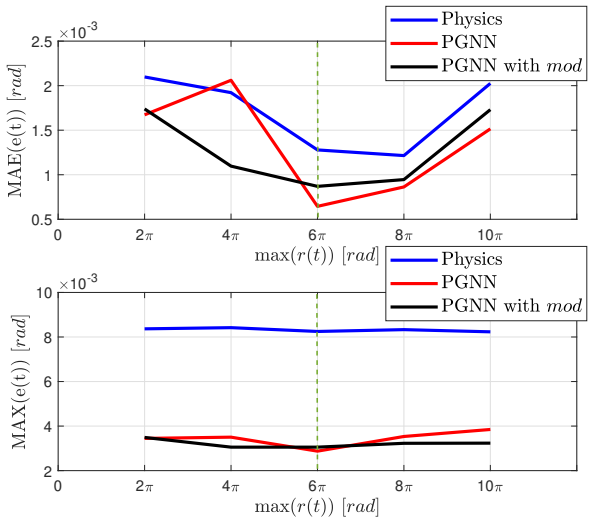

Fig. 14. MAE and MAX for the different feedforward controllers when used on a reference with varying maximum positions. The maximum position in data generation is 6π.

stuck in the local minimum, the training loss is still less than the physics-based model.

Moreover, only θ_{nn} is updated based on the data set Z^N by fixing the physics-based model θ_{phy}^{lip}, which reduces the approximation ability of the PGNN. In comparison with the black box neural network, PGNN still achieves improvements in tracking performance with a factor of approximately three.

To test the robustness of the PGNN feedforward controller, validation references are designed with different maximum velocities $|\frac{d}{dt}r(t)| \neq 15\ \frac{rad}{s}$. Fig. 13 shows the robustness of the PGNN. When the maximum velocity does not exceed the value in data generation, the PGNN improves tracking performance with respect to the physics-based feedforward both in $MAX(e(t))$ and $MAE(e(t))$. However, the PGNN is sensitive to the velocities that are not shown in the training

data set. In particular, if the dynamics contained within the NN are mostly velocity-dependent, PGNN has poor extrapolation for velocities. Similarly, in [24], the friction is mostly position-dependent in the coreless linear motor, which explains the sensitivity for positions.

As shown in Fig. 14, the PGNN with the modulo operation on position $y(t)$ imposes a form of graceful degradation, however, it is more robust to different positions. For example, in the paper-handling system of a printer, hybrid stepper motors typically transport paper sheets constantly in one direction. The PGNN with modulo operation can reduce energy consumption.

VI. Conclusion

In this paper, we investigate the design of physics-guided neural networks to identify the inverse system dynamics in the discrete-time domain on a hybrid stepper motor. Based on the estimated inverse model, the feedforward controller is implemented in the real device that also works with FOC control. From the perspective of system identification, it provides a generalized approach to improve the accuracy of an estimated model. Therefore, as shown in the results, it outperforms two traditional feedforward controllers even though using an incomplete physical model. Stepper motors have a huge market. The physics-guided neural network method is simple and easy to implement. Consequentially, in the foreseeable future, it has the potential to be employed in varying industries. As a next step, in a more complex environment, we will exploit different physics-based models and structures of neural networks to improve performance.

References

[1] S. Derammelaere, B. Vervisch, F. De Belie, B. Vanwalleghem, J. Cottyn, P. Cox, G. Van den Abeele, K. Stockman, and L. Vandevelde, "The efficiency of hybrid stepping motors: Analyzing the impact of control algorithms," *IEEE Industry Applications Magazine*, vol. 20, no. 4, pp. 50–60, 2014.

[2] B. Henke, O. Sawodny, S. Schmidt, and R. Neumann, "Modeling of hybrid stepper motors for closed loop operation," *IFAC Proceedings Volumes*, vol. 46, no. 5, pp. 177–183, 2013, 6th IFAC Symposium on Mechatronic Systems.

[3] T. Hoang, A. Das, S. Koekebakker, and S. Weiland, "Sensorless field-oriented estimation of hybrid stepper motors in high-performance paper handling," in *CCTA 2019 - 3rd IEEE Conference on Control Technology and Applications*. United States: Institute of Electrical and Electronics Engineers, Aug. 2019, pp. 252–257.

[4] T. Kenjo and A. Sugawara, "Stepping motors and their microprocessor controls, Second Edition,1994 Oxford, Oxford University Press ISBN 0 19 859385 6," *European Journal of Engineering Education*, vol. 20, no. 3, pp. 386–386, 1995.

[5] G. Feng, "Position control of a pm stepper motor using neural networks," in *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No.00CH37187)*, vol. 2, 2000, pp. 1766–1769 vol.2.

[6] C. Wang and D. Cao, "New sensorless speed control of a hybrid stepper motor based on fuzzy sliding mode observer," *Energies*, vol. 13, no. 18, 2020.

[7] W. Kim, C. Yang, and C. C. Chung, "Design and implementation of simple field-oriented control for permanent magnet stepper motors without DQ transformation," *IEEE Transactions on Magnetics*, vol. 47, no. 10, pp. 4231–4234, 2011.

[8] C. Obermeier, H. Kellermann, and G. Brandenburg, "Sensorless field oriented speed control of a hybrid and a permanent magnet disk stepper motor using an extended kalman filter," in *1997 IEEE International Electric Machines and Drives Conference Record*, 1997, pp. MC3/5.1–MC3/5.3.

[9] W. Kim, D. Shin, and C. C. Chung, "Microstepping with nonlinear torque modulation for position tracking control in permanent magnet stepper motors," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, 2011, pp. 915–921.

[10] R. H. Park, "Two-reaction theory of synchronous machines generalized method of analysis-part I," *Transactions of the American Institute of Electrical Engineers*, vol. 48, no. 3, pp. 716–727, 1929.

[11] P. Krishnamurthy and F. Khorrami, "An analysis of the effects of closed-loop commutation delay on stepper motor control and application to parameter estimation," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 1, pp. 70–77, 2008.

[12] M. Bolderman, D. Fan, M. Lazar, and H. Butler, "Generalized feedforward control using physics—informed neural networks," *IFAC-PapersOnLine*, vol. 55, no. 16, pp. 148–153, 2022, 18th IFAC Workshop on Control Applications of Optimization CAO 2022.

[13] S. Devasia, "Should model-based inverse inputs be used as feedforward under plant uncertainty?" *IEEE Transactions on Automatic Control*, vol. 47, no. 11, pp. 1865–1871, 2002.

[14] Y.H.Yuen, "Data-driven neural feedforward controller design for industrial linear motors," *Master thesis*, 2019, eindhoven University of Technology.

[15] M. B. S. K. Hornik and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, p. 359–366, 1989.

[16] O. Sørensen, "Additive feedforward control with neural networks," *IFAC Proceedings Volumes*, vol. 32, no. 2, pp. 1378–1383, 1999, 14th IFAC World Congress 1999, Beijing, China, 5-9 July.

[17] M. Bolderman, M. Lazar, and H. Butler, "Physics-guided neural networks for inversion-based feedforward control applied to linear motors," in *2021 IEEE Conference on Control Technology and Applications (CCTA)*, 2021.

[18] A. Karpatne, W. Watkins, J. S. Read, and V. Kumar, "Physics-guided neural networks (PGNN): an application in lake temperature modeling," *CoRR*, vol. abs/1710.11431, 2017. [Online]. Available: http://arxiv.org/abs/1710.11431

[19] O. Nelles, *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models*, ser. Engineering online library. Springer, 2001.

[20] T. Nguyen, "Identification and compensation of parasitic effects in coreless linear motors," Ph.D. dissertation, Electrical Engineering, Oct. 2018, proefschrift.

[21] G. Baluta, "Microstepping mode for stepper motor control," *IEEE Int. Symp. Signals, Circuits Syst.*, vol. 2, pp. 1–4, 2007.

[22] D. Chen and B. Paden, "Adaptive linearization of hybrid step motors: stability analysis," *IEEE Transactions on Automatic Control*, vol. 38, no. 6, pp. 874–887, 1993.

[23] S. van Schalm, "Advanced stepper motor control: Design of the test set-up for feed-forward control," *Bachelor thesis*, 2022.

[24] M. Bolderman, M. Lazar, and H. Butler, "Physics-guided neural networks for feedforward control: From consistent identification to feedforward controller design," 2022. [Online]. Available: https://arxiv.org/abs/2204.00431

APPENDIX A

TABLE IV
THE EXACT MOTOR PARAMETERS AND IDENTIFIED PARAMETERS ON REAL-LIFE MOTOR.

Parameters	Exact values	Identified values
Moment of Inertial $J[\ \mathrm{kg}\cdot\mathrm{m}^2]$	2.8×10^{-5}	2.7498×10^{-5}
Friction Coefficient $B[\ \mathrm{N}\cdot\mathrm{m}\cdot\mathrm{s/rad}]$	–	0.019949
Resistance $R[\Omega]$	0.83	–
Inductance L[H]	2.2×10^{-3}	–
Motor Constant $K_m[\ \mathrm{N}\cdot\mathrm{m/A}]$	–	–
Number of Poles N	50	–

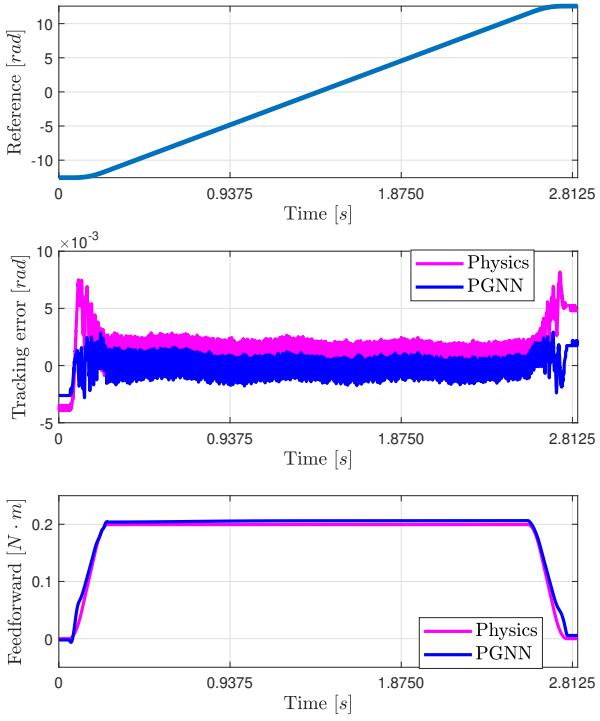

Fig. 15. The tracking results from the validation reference. The top subfigure is the validation reference $r_1(t)$; the middle subfigure is the tracking error $e(t)$; the bottom subfigure is the feedforward signal by using the regressor in (27).

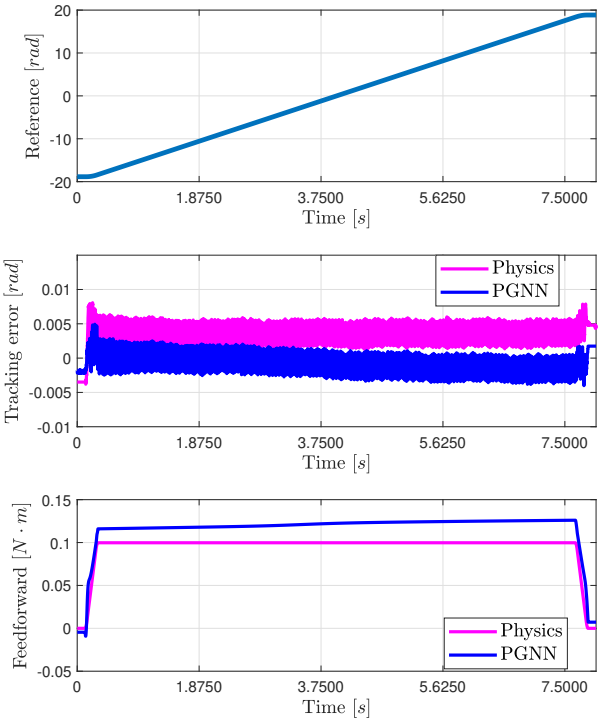

Fig. 16. The tracking results from the validation reference. The top subfigure is the validation reference $r_2(t)$; the middle subfigure is the tracking error $e(t)$; the bottom subfigure is the feedforward signal by using the regressor in (27).

TABLE V
THE TRACKING RESULTS OF FIG. 15 AND FIG. 16

	Evaluation / Cases	$MAE[rad]$	$MAX[rad]$
r_1	No FF	3.7×10^{-2}	4.3×10^{-2}
	Physics	1.7×10^{-3}	8.2×10^{-3}
	PGNN	7.4×10^{-4}	2.9×10^{-3}
r_2	No FF	2.3×10^{-2}	2.7×10^{-2}
	Physics	3.9×10^{-3}	8.1×10^{-3}
	PGNN	1.1×10^{-3}	4.9×10^{-3}

TABLE VI
THE CONVERGED LOSS VALUE AND THE MEAN ABSOLUTE ERROR OF PGNN(ϕ_1) WITH REGULARIZATION L_2

L_2 Regularization			
-	Loss_{train}	*MAX(e(t))*	*MAE(e(t))*
$\lambda=1\times10^{-1}$	0.000970	0.0190	0.1324
$\lambda=1\times10^{-3}$	0.000543	0.0177	0.1002
$\lambda=1\times10^{-5}$	0.000462	0.0130	0.0934
$\lambda=1\times10^{-7}$	0.000457	0.0094	0.0729
$\lambda=0$	0.000454	0.0087	0.0727