Physics–guided neural networks for inversion–based feedforward control applied to hybrid stepper motors*

D. Fan[1], M. Bolderman[†,1], S. Koekebakker[2], H. Butler[1,3], and M. Lazar[1]

Abstract— Rotary motors, such as hybrid stepper motors (HSMs), are widely used in industries varying from printing applications to robotics. The increasing need for productivity and efficiency without increasing the manufacturing costs calls for innovative control design. Feedforward control is typically used in tracking control problems, where the desired reference is known in advance. In most applications, this is the case for HSMs, which need to track a periodic angular velocity and angular position reference. Performance achieved by feedforward control is limited by the accuracy of the available model describing the inverse system dynamics. In this work, we develop a physics–guided neural network (PGNN) feedforward controller for HSMs, which can learn the effect of parasitic forces from data and compensate for it, resulting in improved accuracy. Indeed, experimental results on an HSM used in printing industry show that the PGNN outperforms conventional benchmarks in terms of the mean–absolute tracking error.

I. INTRODUCTION

Hybrid stepper motors (HSM) are widely used in industrial automation, such as pick–and–place robots [1], [2], additive manufacturing [3], professional printing applications [4], and more, see, e.g., [5] for an overview. HSMs can be operated in an open–loop configuration using microstepping [6]. However, the open–loop stepping often induces unwanted vibrations and is highly inefficient as it applies high currents to be robust for worst case loads. Consequently, for high–precision applications, closed–loop control schemes are often applied in the form of field–oriented control (FOC) [7], [8], see also [9] for control strategies of the inner current control loop. Since FOC requires measurements of both the currents and the angular position of the HSM, significant research has been done on sensorless FOC which does not require the additional angular position sensor, see, e.g., [4], [10].

For motion control systems, reference tracking performance is typically achieved via feedforward control, while feedback control stabilizes the system and rejects disturbances and feedforward imperfections [11]. For rotary motors however, the feedforward control design is largely neglected or restricted to be linear, see, e.g., [7] which employs a linear velocity–acceleration feedforward, or [12] which employs linear feedforward with a disturbance compensator. The complete dynamical behaviour of HSMs constitutes more complex phenomena, such as parasitic torques arising from manufacturing tolerances, as well as torque ripples caused by detent torque and back electromotive forces. Since performance achieved by feedforward control is limited by the accuracy of the model of the inverse system [13], designing a feedforward controller from a linear model intrinsically limits performance. Iterative learning control [14] provides the potential to improve tracking performance further, but requires multiple repetitions of the same reference.

Physics–guided neural networks (PGNNs) have potential to improve performance achieved by linear, physics–based, feedforward controllers by accurately identifying the inverse system dynamics from data [15]. PGNNs effectively merge physics–based and NN–based models and thereby result in nonlinear feedforward controllers with improved performance, and the same reliability as physics–based feedforward controllers [16]. This is in contrast to black box NNs, which can fail to learn from presented data. The application of a PGNN feedforward controller to a rotary machine however remains unexplored.

Hence, this motivates us to develop PGNN feedforward controllers for improving performance of HSMs. To this end, we define a PGNN architecture that embeds a simple, physics–based inverse model of the HSM within a black–box NN. Also, we impose the rotational reproducible behaviour, i.e., the same dynamics is expected for each rotation. With the PGNN architecture defined, the PGNN training identifies or learns the inverse system dynamics of the HSM from an available input–output data set, i.e., requiring measurements of the angular position. Since the PGNN feedforward controller does not require online measurements, it can also be implemented in a sensorless FOC scheme. The developed PGNN feedforward improves performance by a factor 2 in terms of the mean–absolute tracking error (MAE) in real–time on an HSM used in printing industry, without requiring additional computational hardware or measurements.

II. PRELIMINARIES

A. First–principle modeling of an HSM

Fig. 1 shows a schematic overview of the FOC structure with dq–transformation of the hybrid stepper motor, see, e.g., [17]. Note that, both the position and the current controllers are implemented in discrete–time, which are indicated by the sampler and ZOH blocks. The HSM is subdivided in a mechanical and an electromagnetic part. The mechanical dynamics, indicated with G_{me} is modeled using

[1]Control Systems Group, Eindhoven University of Technology, Groene Loper 19, Eindhoven, 5612 AP, The Netherlands

[2]Canon Production Printing, St. Urbanusweg 43, Venlo, 5900 AE, The Netherlands.

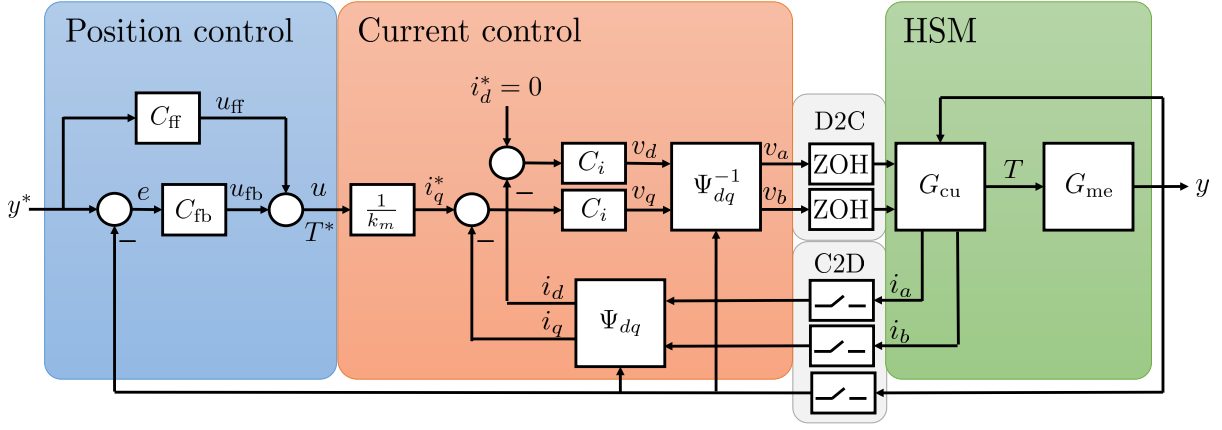[3]ASML, De Run 6501, Veldhoven, 5504 DR, The Netherlands.

Fig. 1. FOC architecture including the HSM, the current control with dq–transform and the position feedback–feedforward control setup.

Newton–Euler relations, such that

$$J \frac{d^2}{dt^2} y(t) = F(t) - f_v \frac{d}{dt} y(t), \tag{1}$$

where $y(t)$ is the position output at time $t \in \mathbb{R}_{>0}$, $J \in \mathbb{R}_{>0}$ the mass moment of inertia, $f_v \in \mathbb{R}_{>0}$ the viscous friction coefficient, and $F(t)$ the driving torque. The driving torque is modeled as [18]

$$F(t) = k_m \left(-i_a(t) \sin\left(Ny(t)\right) + i_b(t) \cos\left(Ny(t)\right) \right), \tag{2}$$

where $k_m \in \mathbb{R}_{>0}$ is the motor constant, $N \in \mathbb{Z}_{>0}$ the number of rotor teeth, and $i_a(t)$ and $i_b(t)$ the current through coils a and b, respectively.

The electromagnetic dynamics is modeled as

$$L \frac{d}{dt} i_a(t) = v_a(t) - Ri_a(t) + k_m \left(\frac{d}{dt} y(t)\right) \sin\left(Ny(t)\right),$$
$$L \frac{d}{dt} i_b(t) = v_b(t) - Ri_b(t) - k_m \left(\frac{d}{dt} y(t)\right) \cos\left(Ny(t)\right), \tag{3}$$

where $L \in \mathbb{R}_{>0}$ is the inductance, $R \in \mathbb{R}_{>0}$ the resistance, and $v_a(t)$ and $v_b(t)$ the terminal voltages of coil a and b, respectively. The latter terms in (3) are the self–induced voltage, also known as the back electromotive force.

Since the HSM has two inputs, i.e., the voltages v_a and v_b, and only a single output y, often the dq–transformation [17] Ψ_{dq} is employed

$$\begin{bmatrix} i_d(t) \\ i_q(t) \end{bmatrix} = \Psi_{dq}\left(y(t)\right) \begin{bmatrix} i_a(t) \\ i_b(t) \end{bmatrix}$$
$$:= \begin{bmatrix} \cos\left(Ny(t)\right) & \sin\left(Ny(t)\right) \\ -\sin\left(Ny(t)\right) & \cos\left(Ny(t)\right) \end{bmatrix} \begin{bmatrix} i_a(t) \\ i_b(t) \end{bmatrix}. \tag{4}$$

As a result, we observe that the driving torque in (2) simplifies to $T(t) = k_m i_q(t)$, and the mechanical dynamics into

$$J \frac{d^2}{dt^2} y(t) = k_m i_q(t) - f_v \frac{d}{dt} y(t). \tag{5}$$

Note, from (5) we observe that the position control only requires $i_q(t)$. Finally, energy consumption can be approx-

imated by the squared sum of currents, which, using dq–transformation (4), yields

$$i_a^2 + i_b^2 = i_d^2 + i_q^2. \tag{6}$$

Since i_d does not contribute to the driving torque, we aim to have it equal to zero and thereby minimize the energy consumption.

Remark 2.1: It is possible to derive a more complex description of the HSM dynamics, e.g., by including detent torque, reluctance, and other effects. However, the goal of this work is to demonstrate effectiveness of the PGNN framework for feedforward control, which should compensate for unmodeled effects by learning these effects from data.

B. Field–oriented control architecture of an HSM

The inner current control loop aims to have the driving torque $T(t)$ become equal to the input $u(t)$. The currents $i_a(t)$ and $i_b(t)$ are controlled using the voltages $v_a(t)$ and $v_b(t)$, such that, in dq–coordinates these follow the references $i_d^*(t) = 0$ and $i_q^*(t) = \frac{1}{k_m} u(t)$. In order to achieve this, the inverse dq–transformation is applied to the voltages, such that

$$\begin{bmatrix} v_a(t) \\ v_b(t) \end{bmatrix} = \Psi_{dq}^{-1}\left(y(t)\right) \begin{bmatrix} v_d(t) \\ v_q(t) \end{bmatrix}$$
$$= \begin{bmatrix} \cos\left(Ny(t)\right) & -\sin\left(Ny(t)\right) \\ \sin\left(Ny(t)\right) & \cos\left(Ny(t)\right) \end{bmatrix} \begin{bmatrix} v_d(t) \\ v_q(t) \end{bmatrix}. \tag{7}$$

Substituting the dq–transformation (4) and the inverse dq–transformation (7) in the electromagnetic model (3), gives the electromagnetic model in dq–coordinates as

$$L \frac{d}{dt} i_d(t) = v_d(t) - Ri_d(t) + LNi_q(t) \frac{d}{dt} y(t),$$
$$L \frac{d}{dt} i_q(t) = v_q(t) - Ri_q(t) - k_m \frac{d}{dt} y(t) - LNi_d \frac{d}{dt} y(t). \tag{8}$$

The voltages in dq–coordinates are computed using the discrete–time feedback controller $C_i(z)$ as

$$v_d(k) = -C_i(z) i_d(k),$$
$$v_q(k) = C_i(z) \left(i_q^*(k) - i_q(k)\right), \tag{9}$$

where $k \in \mathbb{Z}_{\geq 0}$ indicates the discrete–time instant. The inverse dq–transformation Ψ_{dq}^{-1} in (7) and dq–transformation in (4) are evaluated at discrete time indices, i.e., for $t = kT_s$, with $T_s \in \mathbb{R}_{>0}$ the sampling time. The feedback controller $C_i(z)$ is a discretized version of the PI–controller

$$C_i(s) = k_p + \frac{k_i}{s}, \tag{10}$$

with $k_p \in \mathbb{R}$ and $k_i \in \mathbb{R}$ the proportional and integral gain, respectively.

Remark 2.2: The use of the dq–transformation can be omitted by directly transforming the control input $u(k)$ into current references $i_a^*(k)$ and $i_b^*(k)$, see [7]. When following a constant velocity reference and assuming a constant load (viscous friction), both $i_a^*(k)$ and $i_b^*(k)$ follow a sinusoidal reference, whereas $i_q^*(k)$ remains constant. Correspondingly, the dq current control is expected to work better for reference tracking control.

The outer angular position control loop consists of a feedback and a feedforward controller, such that

$$u(k) = u_{\text{fb}}(k) + u_{\text{ff}}(k), \tag{11}$$

where $u_{\text{fb}}(k)$ is the feedback and $u_{\text{ff}}(k)$ the feedforward input. The feedback input is computed as

$$u_{\text{fb}}(k) = C_{\text{fb}}(z)\big(y^*(k) - y(k)\big), \tag{12}$$

where $C_{\text{fb}}(z)$ is the transfer function of the discrete–time feedback controller, and $y^*(k)$ the reference.

We develop a data–driven feedforward controller following the same steps as in [15], where linear motors were considered. First, we have an input–output data set generated on the system, i.e.,

$$Z^N := \{u_0, y_0, ..., u_{N-1}, y_{N-1}\}, \tag{13}$$

where $N \in \mathbb{Z}_{>0}$ are the number of samples, and u_i, y_i are $u(i)$, $y(i)$ for the data generating experiment. Second, we parametrize the inverse system dynamics according to

$$\begin{aligned} \hat{u}\big(\theta, \phi(k)\big) &:= f\big(\theta, \phi(k)\big), \\ \phi(k) &:= [y(k + n_k + 1), ..., y(k + n_k - n_a + 1), \\ &\qquad u(k-1), ..., u(k - n_b + 1)]^T. \end{aligned} \tag{14}$$

In (14), $\hat{u}$ is the prediction of the input u, $f : \mathbb{R}^{n_\theta \times (n_a + n_b)} \to \mathbb{R}$ is a model of the inverse dynamics, $\theta \in \mathbb{R}^{n_\theta}$ are the parameters, and $\phi(k)$ is the regressor with $n_a, n_b \in \mathbb{Z}_{\geq 0}$ describing the order of the dynamics and $n_k \in \mathbb{Z}_{\geq 0}$ the number of pure input delays. The values for n_a, n_b, and n_k can be obtained, e.g., by discretizing a first–principle model of the continuous–time dynamics, or by analyzing a frequency response function. In order to have the model (14) fit the inverse system dynamics, the parameters are chosen according to an identification criterion

$$\hat{\theta} = \arg\min_\theta \frac{1}{N} \sum_{i=0}^{N-1} \big(u_i - \hat{u}(\theta, \phi_i)\big). \tag{15}$$

Finally, the feedforward controller is obtained by computing the input that is required to follow the reference, such that

$$\begin{aligned} u_{\text{ff}}(k) &= \hat{u}\big(\hat{\theta}, \phi_{\text{ff}}(k)\big), \\ \phi_{\text{ff}}(k) &:= [y^*(k + n_k + 1), ..., y^*(k + n_k - n_a + 1), \\ &\qquad u_{\text{ff}}(k-1), ..., u_{\text{ff}}(k - n_b + 1)]^T. \end{aligned} \tag{16}$$

In order to implement the feedforward controller (16), we assume that reference values up until time $k + n_k + 1$ are known at time k.

III. PROBLEM STATEMENT

The choice of the model class f in (14) determines the effects to be identified, and, consequently, compensated for by the feedforward controller (16). For mechatronic systems, it is typically assumed that the current loop operates significantly faster compared to the position loop, such that the feedforward controller can be designed solely for the mechanical part of the dynamics, i.e., (1) with $T(k) = u(k)$. Consequently, using the physical knowledge, a suitable candidate for the model class is given as

$$\begin{aligned} \hat{u}\big(\theta, \phi(k)\big) &= f_{\text{phy}}\big(\theta_{\text{phy}}, \phi(k)\big) \\ &= \theta_{\text{phy}}^T \begin{bmatrix} \delta^2 y(k) \\ \delta y(k) \end{bmatrix}, \end{aligned} \tag{17}$$

where $\delta = \frac{q - q^{-1}}{2T_s}$ with q the forward shift operator, such that $\phi(k) = [y(k+2), ..., y(k-2)]^T$. Additionally, θ_{phy} are the physical parameters which represent the inertia J and viscous friction coefficient f_v.

Remark 3.1: It is possible to use more accurate discretization schemes to find n_a, n_b, and n_k in (17). For example, ZOH discretization is exact for linear dynamics if the input is kept constant between two consecutive samples. However, the experimental results in Sec. V show that the parasitic effects are dominant over the discretization error made in (17). Additionally, this discretization scheme has the advantage that $n_b = 0$, such that the feedforward controller (16) is stable. For $n_b > 0$, [16] presents tools to both validate (after training) and impose (during training) stability of the PGNN feedforward controllers.

The physics–based feedforward controller (17) can only identify and compensate for the inertia and viscous friction, while real–life applications comprise of more complex behaviour. Consequently, it was first proposed in [19] to employ a black–box NN as a model class (14), such that

$$\begin{aligned} \hat{u}\big(\theta, \phi(k)\big) &= f_{\text{NN}}\big(\theta_{\text{NN}}, \phi(k)\big) \\ &= W_{L+1}\alpha_L\Big(...\alpha_1\big(W_1\phi(k) + B_1\big)\Big) + B_{L+1}, \end{aligned} \tag{18}$$

where $\alpha_l : \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$ denotes the aggregation of activation functions with $n_l \in \mathbb{Z}_{>0}$ the number of neurons in layer $l \in \{0, ..., L\}$, and $L \in \mathbb{Z}_{>0}$ the number of hidden layers. The parameters $\theta_{\text{NN}} := [\text{col}(W_1)^T, B_q^T, ..., \text{col}(W_{L+1})^T, B_{L+1}^T]^T$ are the concatenation of all weights $W_l \in \mathbb{R}^{n_l \times n_{l-1}}$ and biases $B_l \in \mathbb{R}^{n_l}$, where $\text{col}(W_l)$ stacks the columns of W_l. Although the NN (18) has the potential to approximate the
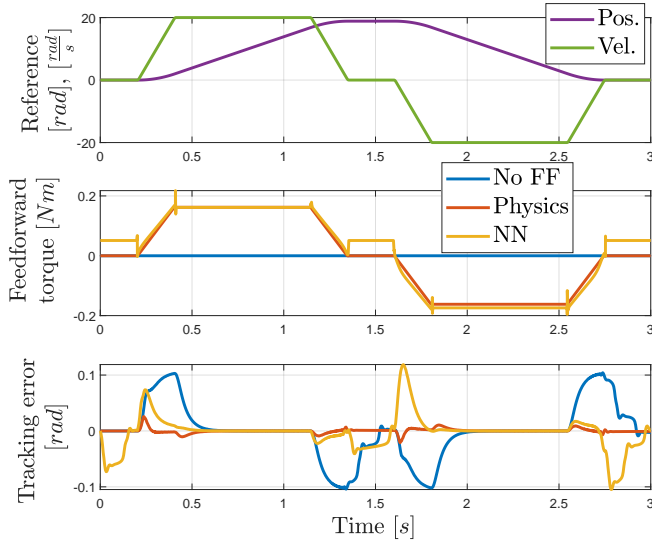
Fig. 2. Reference (top window), feedforward signal (middle window), and the resulting tracking error (bottom window) for the feedforward controllers using the physical model (17) and the NN (18) on a simulation example.

Fig. 3. Schematic overview of the physics–guided neural network.

model and the NN model are combined in a single PGNN feedforward controller.

IV. FEEDFORWARD CONTROL OF HSMS USING PHYSICS–GUIDED NEURAL NETWORKS

With the aim to obtain a feedforward controller with the same reliability as the physics–based model (17) and the high accuracy of the NN model (18), the PGNN model was first proposed in [15], see Fig. 3. The PGNN predicts the input according to

$$\hat{u}\big(\theta, \phi(k)\big) = f_{\text{phy}}\big(\theta_{\text{phy}}, \phi(k)\big) + f_{\text{NN}}\big(\theta_{\text{NN}}, T(\phi(k))\big), \quad (19)$$

where $\theta := [\theta_{\text{NN}}^T, \theta_{\text{phy}}]^T$ are the PGNN parameters, and $T : \mathbb{R}^{n_a+n_b} \to \mathbb{R}^{n_0}$ is an input transformation, with $n_0 \in \mathbb{Z}_{>0}$ the number of NN inputs.

To train the PGNN, we employ the following two–step sequential procedure. First, the physical parameters $\hat{\theta}_{\text{phy}}$ are identified according to identification criterion (15) with physics–based model (17). Afterwards, the NN parameters $\hat{\theta}_{\text{NN}}$ are identified on the residual of the identified physics–based model, such that

$$\hat{\theta}_{\text{NN}} = \arg\min_{\theta_{\text{NN}}} \frac{1}{N} \sum_{i=0}^{N-1} \big(u_i - \hat{u}([\theta_{\text{NN}}^T, \hat{\theta}_{\text{phy}}^T]^T, \phi_i)\big) \\ + \|\Lambda_{\text{NN}}\theta_{\text{NN}}\|_2^2, \quad (20)$$

where Λ_{NN} is a regularization matrix. Note that, first identifying $\hat{\theta}_{\text{phy}}$ and then identifying $\hat{\theta}_{\text{NN}}$ with $\hat{\theta}_{\text{phy}}$ fixed can yield a suboptimal solution. This is prevented by identifying $\hat{\theta}_{\text{phy}}$ and $\hat{\theta}_{\text{NN}}$ simultaneously as in, e.g., [16].

It is expected that, for each rotation, the HSM exhibits the same dynamical behaviour. It is crucial that the PGNN (19) incorporates this rotational reproducibility, since it is otherwise difficult to generate a training data set which describes all relevant rotations $y^*(k)$, e.g., when the HSM rotates in one direction. Therefore, we aim to identify a PGNN model (19) which satisfies

$$\hat{u}\big(\theta, \phi(k)\big) = \hat{u}\left(\theta, \phi(k) + \begin{bmatrix} 1^{(n_a+1)\times 1} \\ 0^{(n_b-1)\times 1} \end{bmatrix} n2\pi\right), \ n \in \mathbb{Z}. \quad (21)$$

In order to impose the rotational reproducible behaviour, i.e., to make the PGNN (19) comply with (21), we consider a specific design of the physics–guided input transform $T(\cdot)$. To do so, we restate that the system order was approximated as $n_k = 1$, $n_a = 4$, and $n_b = 0$ from the physical model (17),

inverse dynamics up to any accuracy, it lacks the robustness of the physics–based model (17). For example, the NN easily fails to learn and generalize from the presented data [20].

To illustrate this, we make use of a closed–loop simulation model of an HSM with some parasitic friction forces, see [21] for details on the parameters, feedback controllers and friction model. The simulation closely resembles the real–life setup discussed in Sec. V, and follows the same data generation experiment. We employ feedforward controllers (16) based on the physical model (17) and the NN model (18) with a single hidden layer $L = 1$ with $n_1 = 16$ neurons. Fig. 2 shows the feedforward signal and resulting tracking error resulting from both feedforward controllers on the HSM simulation. Even though the physical model (17) significantly improves performance with respect to the situation where no feedforward is applied, there remain some errors that are caused by the inability of the physical model to capture the complete dynamics. The NN (18) on the other hand, has the capability to learn more complex dynamics. However, the NN fails to learn and generalize from the presented data which is observed by, e.g., the offset during standstill, and the spikes at the start of the acceleration. This results in poor tracking performance when the NN–based feedforward controller is applied. This issue might be reduced by using a different training data set, or by adjusting the NN dimensions and regularization parameters. However, the example showcases the sensitivity of the NN.

Consequently, the goal of this work is to effectively embed the known physics–based feedforward controller within a NN–based feedforward controller, termed PGNN, to improve the tracking performance of HSMs. To this end, we will use a two–step sequential procedure: first, we identify the parameters of a physics–based feedforward controller as in (17). Second, we train a NN model (18) on the residuals of the identified physics–based model. Then, the physics–based
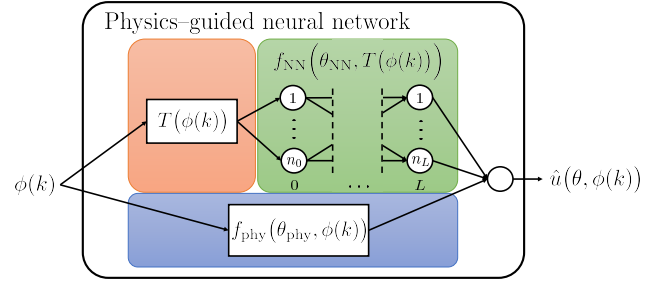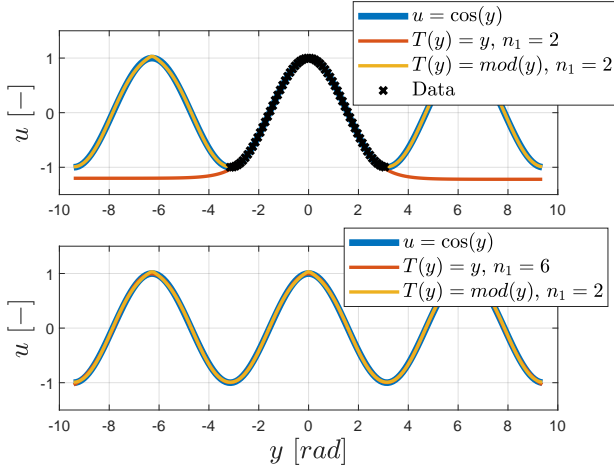
Fig. 4. Example for imposing physical knowledge via $T(\cdot)$, i.e., improved extrapolation capabilities when training a NN with $T(y) = \text{mod}(y)$ compared to $T(y) = y$ on a limited data set (top window), and the reduction of the required amount of neurons n_1 to achieve an approximation of similar quality (bottom window).

such that we consider the following physics–guided input transform

$$T\left(\begin{bmatrix} y(k+2) \\ \vdots \\ y(k-2) \end{bmatrix}\right) = \begin{bmatrix} \delta^2 y(k) \\ \delta y(k) \\ \text{mod}(y(k), 2\pi) \end{bmatrix}, \quad (22)$$

where $\text{mod}(y(k), 2\pi)$ is the remainder after division of $y(k)$ with 2π. Note that, (22) is adopted rather than wrapping all y into the domain $[0, 2\pi)$, since $\delta\text{mod}(y(k)) \neq \delta y(k)$ for all k. Therefore, $T(\cdot)$ includes discrete–time approximations of derivatives of the output $y(k)$, which can also improve the training convergence, e.g., when high sampling rates with respect to the velocity are taken, such that $y(k) \approx y(k-1)$.

Remark 4.1: The physical model (17) only inputs discrete–time angular velocity and acceleration, such that it is reproducible for any offset $y(k)+\Delta$. Then, combined with the NN using transform (22), the PGNN (19) satisfies (21).

As an example of the physics–guided input transform $T(\cdot)$, consider the situation in which a NN is used to learn

$$u(k) = \cos\big(y(k)\big), \quad (23)$$

with data generated from one period. The top window in Fig. 4 shows that $n_1 = 2$ hidden layer neurons (with $\tanh$ activation) give a reasonably accurate identification. The lack of data however, causes the NN trained with $T(y(k)) = y(k)$ to extrapolate poorly, in contrast to the NN trained with $T(y(k)) = \text{mod}(y(k))$. On the other hand, when the full range of interest is covered with data, the NN with $T(y(k)) = \text{mod}(y(k))$ requires significantly less neurons compared to the NN with $T(y(k)) = y(k)$ to yield an approximation of similar accuracy, see the bottom window of Fig. 4.

V. EXPERIMENTAL VALIDATION

The PGNN–based feedforward controller (19) is validated on a real–life HSM used in printing industry shown in Fig. 5.
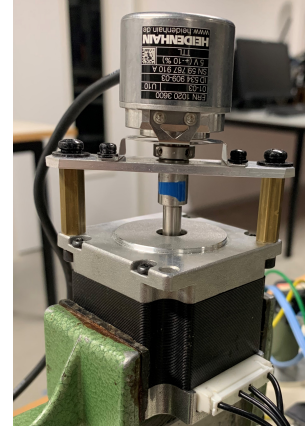

Fig. 5. HSM FL57STH51–2804A by FULLING MOTOR with encoder.

For simplicity, the current and position controllers are only proportional gains tuned as

$$C_i(s) = 6.6, \quad C_{\text{fb}}(s) = 5. \quad (24)$$

Training data is generated by sampling the input $u(k)$ and output $y(k)$ with a sampling time of $Ts = 10^{-4}$ s while operating the HSM in closed–loop with a third order reference moving back–and–forth between -3 to $+3$ rotations with a velocity of $15 \frac{\text{rad}}{\text{s}}$, acceleration of $80 \frac{\text{rad}}{\text{s}^2}$, and jerk $1000 \frac{\text{rad}}{\text{s}^3}$ for a duration of 80 s. The PGNN (19) uses the physical model (17) and a single hidden layer with $16 \tanh$ neurons with physics–guided input transform (22) trained according to (20) with $\Lambda_{\text{NN}} = 0$. It was observed that adding more neurons or hidden layers did not further improve performance.

Fig. 6 shows the reference, generated feedforward signals, and the tracking error resulting from the physics–based feedforward and the PGNN. The presented forward motion was preceded by a back–and–forward motion of the same reference to remove the transients caused by differences in initial conditions, and thereby facilitate a fair comparison. Although the physics–based and the PGNN–based feedforward inputs are largely similar, the small deviations especially during the acceleration part of the reference yield significantly smaller overshoot for the PGNN.

Fig. 7 shows the mean–absolute error (MAE)

$$\frac{1}{N_r} \sum_{t=0}^{N_r-1} |y^*(k) - y(k)|, \quad (25)$$

for a reference of $N_r \in \mathbb{Z}_{>0}$ samples as in Fig. 6 with different maximum velocities. The PGNN outperforms the physics–based feedforward controllers for all velocities smaller than $15 \frac{\text{rad}}{\text{s}}$. For velocities larger than $15 \frac{\text{rad}}{\text{s}}$, the physics–based feedforward controller achieves slightly better performance, which is explained by the fact that the training data did not contain information for velocities exceeding $15 \frac{\text{rad}}{\text{s}}$. It is possible to enhance robustness to non–training data via the regularization approach discussed in [22], which penalizes the deviation of the PGNN output with respect to the output of the physical model for non–training data.
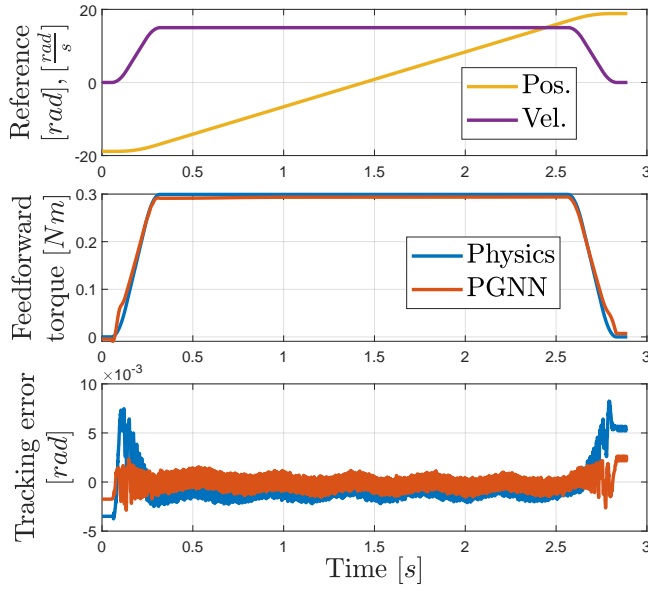
Fig. 6. Reference (top window), feedforward signal (middle window), and the resulting tracking error (bottom window) for the feedforward controllers using the physical model (17) and the PGNN (19) for the real–life HSM.
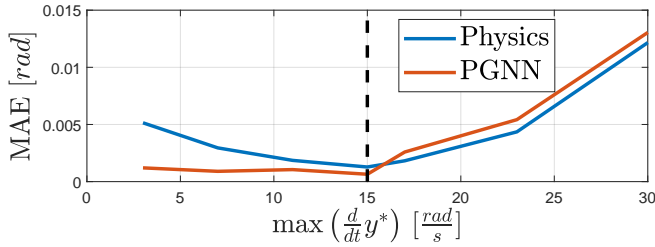

Fig. 7. MAE of the tracking error using a physics–based feedforward (17) and the PGNN (19) for references with varying velocities. The black line indicates the maximum velocities attained during training.

VI. CONCLUSIONS

A PGNN–based feedforward controller for HSMs was developed and tested in real–time experiments. The PGNN was designed to physically embed the rotational reproducible behaviour of the HSM, which improved performance with respect to a physics–based approach on a real–life HSM without requiring an increase in costs. Further research will focus on the feedforward controller design for HSMs as part of a complex industrial printer, as well as reducing the effect of predictable disturbances on the closed–loop system.

VII. ACKNOWLEDGEMENTS

REFERENCES

[1] R. V. Sharan and G. C. Onwubolu, "Simulating the arm movements of a stepper motor controlled pick–and–place robot using the stepper motor model," *International Journal of Advanced Science and Technology*, vol. 60, pp. 59–66, 2013.

[2] M. S. H. Talpur and M. H. Shaikh, "Automation of mobile pick and place robotic system for small food industry," *arXiv:1203.4475*, 2013.

[3] P. S. Kamble, S. A. Khoje, and J. A. Lele, "Recent developments in 3d printing technologies: Review," *International Conference on Intelligent Computing and Control Systems*, pp. 468–473, 2018.

[4] T. D. Hoang, A. Das, S. Koekebakker, and S. Weiland, "Sensor-less field–oriented estimation of hybrid stepper motors in high–performance paper handling," *Conference on Control Technology and Applications*, pp. 252–257, 2019.

[5] P. Acarnley, *Stepping motors: A guide to theory and practice.* IET, 2002.

[6] 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.

[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] K. M. Le, H. van Hoang, and J. W. Jeon, "An advanced closed–loop control to improve the performance of hybrid stepper motors," *IEEE Transactions on Power Electronics*, vol. 32, no. 9, pp. 7244–7255, 2017.

[9] F. Bernardi, E. Carfagna, G. Migliazza, G. Buticchi, F. Immovilli, and E. Lorenzani, "Performance analysis of current control strategies for hybrid stepper motors," *IEEE IOpen Journal of the Industrial Electronics Society*, vol. 3, pp. 460–472, 2022.

[10] 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," *IEEE International Electric Machines and Drives Conference Record*, 1997.

[11] M. Steinbuch and R. M. J. G. van de Molengraft, "Iterative learning control of industrial motion systems," *IFAC Proceedings Volumes*, vol. 33, no. 26, pp. 899–904, 2000.

[12] J. Wu, Y. Han, Z. Xiong, and H. Ding, "Servo performance improvement through iterative tuning feedforward controller with disturbance compensator," *International Journal of Machine Tools and Manufacture*, vol. 117, pp. 1–10, 2017.

[13] S. Devasia, "Should model–based inverse inputs be used as feed-forward under plant uncertainty?" *IEEE Transactions on Automatic Control*, vol. 47, pp. 1865–1871, 2002.

[14] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control," *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 96–114, 2006.

[15] M. Bolderman, M. Lazar, and H. Butler, "Physics–guided neural networks for inversion–based feedforward control applied to linear motors," *IEEE Conference on Control Technology and Applications*, pp. 1115–1120, 2021.

[16] M. Bolderman, H. Butler, S. Koekebakker, E. van Horssen, R. Kamidi, T. Spaan-Burke, N. Strijbosch, and M. Lazar, "Physics–guided neural networks for feedforward control with input–to–state stability guarantees," *arXiv:2301.08568*, 2023.

[17] C. J. O'Rourke, M. M. Qasim, M. R. Overlin, and J. L. Kirtley, "A geometric interpretation of reference frames and transformations: dq0, clarke, and park," *IEEE Transactions on Energy Conversion*, vol. 34, no. 4, pp. 2070–2083, 2019.

[18] B. Henke, O. Sawodny, S. Schmidt, and R. Neumann, "Modeling of hybrid stepper motors for closed loop operation," *IFAC Symposium on Mechatronic Systems*, pp. 177–183, 2013.

[19] O. Sørensen, "Additive feedforward control with neural networks," *IFAC Proceedings Volumes*, vol. 32, no. 2, pp. 1378–1383, 1999.

[20] P. J. Haley and D. Soloway, "Extrapolation limitations of multilayer feedforward neural networks," *in Proceedings of International Joint Conference on Neural Networks*, vol. 4, pp. 25–30, 1992.

[21] D. Fan, "Physics–guided neural networks for inversion–based feedforward control of a hybrid stepper motor," Master's thesis, Eindhoven University of Technology, The Netherlands, 2022.

[22] M. Bolderman, D. Fan, M. Lazar, and H. Butler, "Generalized feedforward control using physics–informed neural networks," *IFAC–PapersOnline*, vol. 55, pp. 148–153, 2022.