# Feedforward Control for an Interventional X-Ray
## A Parallel Physical Model and Neural Network Approach

Kon, Johan; de Vos, Naomi; Bruijnen, Dennis; van de Wijdeven, Jeroen; Heertjes, Marcel; Oomen, Tom

**DOI**
10.1109/TCST.2025.3576972

**Publication date**
2025

**Document Version**
Final published version

**Published in**
IEEE Transactions on Control Systems Technology

**Citation (APA)**
Kon, J., de Vos, N., Bruijnen, D., van de Wijdeven, J., Heertjes, M., & Oomen, T. (2025). Feedforward Control for an Interventional X-Ray: A Parallel Physical Model and Neural Network Approach. *IEEE Transactions on Control Systems Technology*, *33*(6), 2194-2207. https://doi.org/10.1109/TCST.2025.3576972

# Feedforward Control for an Interventional X-Ray: A Parallel Physical Model and Neural Network Approach

Johan Kon, *Graduate Student Member, IEEE*, Naomi de Vos, Dennis Bruijnen, Jeroen van de Wijdeven, Marcel Heertjes, and Tom Oomen, *Senior Member, IEEE*

***Abstract*—Hard-to-model, often nonlinear dynamics limit the tracking performance of physical-model-based feedforward control in medical interventional X-ray (IX) systems. In this article, these unknown dynamics are compensated using a physics-guided neural network (PGNN) feedforward controller. The PGNN consists of a parallel combination of a physical model describing the equations of motions of the IX and a feedforward neural network. To ensure that the neural network compensates only for the dynamics not included in the physical model, the neural network output in the subspace spanned by the physical model is penalized through regularization. As a result, the physical model serves as a baseline for performance, and the neural network contribution can be monitored or disabled. The PGNN feedforward controller is validated on an experimental IX setup, illustrating its superior performance over a physical-model-based feedforward controller.**

***Index Terms*—Feedforward control, gray-box identification, multibody dynamics, neural networks, precision motion control.**

## I. Introduction

IMAGE-GUIDED therapy (IGT) in general and interventional X-rays (IXs) specifically [1] are key technologies in healthcare that directly improve treatment quality by enabling minimally invasive therapies through visualization of patient tissue [2]. IX is one of the main IGT systems and is able to create 3-D images of the relevant tissue by combining a sequence of 2-D X-ray snapshots [3]. It is specifically geared toward use during surgery to generate real-time images of the relevant tissue, enabling the use of small surgical tools as opposed to making large incisions, resulting in faster recovery times.

Accurate feedforward control [4], [5] is essential during the operation of an IX system to guarantee both high image quality [1] and patient and operator safety. First, accurate feedforward control enables accurate tracking of the desired setpoint for the imaging sequence, minimizing visual artifacts such as motion blur. Second, the difference between the input torques predicted by the feedforward controller and the actual applied torques, i.e., the feedback controller contribution, can be used as a basis to distinguish nominal operating conditions from anomalies not included in the feedforward controller, e.g., for detecting collisions, providing a basis for safety-promoting procedures.

To achieve accurate feedforward control, the feedforward controller should be able to compensate for the nonlinear dynamics of the IX. Specifically, the movement of IX is influenced by cable, friction, and inertial forces that nonlinearly depend not only on the current configuration but also the current velocity and acceleration [1]. If these forces are not compensated by the feedforward controller, these result in tracking errors that have to be compensated by a feedback controller, deteriorating image quality and collision detection. Simultaneously, for medical certification, the feedforward controller should guarantee a baseline level of performance through the use of an interpretable physical model.

Current feedforward approaches either are not flexible enough to compensate for the IX dynamics or are not based on an interpretable physical model.

1) First, the state-of-practice in IX feedforward control is based on a physical model [1]. However, the performance of such an approach is limited by the cable and friction forces in the IX. Specifically, these are hard to express quantitatively through mathematical expressions, such that including them in a feedforward controller is infeasible.
2) Second, to overcome the limited flexibility of physical models, neural networks have been used as feedforward controllers to express the feedforward torques as a

Received 12 November 2024; revised 4 April 2025; accepted 29 May 2025. Date of publication 19 June 2025; date of current version 23 October 2025. This work was supported in part by the Holland High Tech—TKI HSTM through the PPS Allowance Scheme for Public-Private Partnerships, in part by ASML, and in part by Philips Engineering Solutions. Recommended by Associate Editor H. Fourati. *(Corresponding author: Johan Kon.)*

Johan Kon is with the Control Systems Technology Group, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands (e-mail: j.j.kon@tue.nl).

Naomi de Vos is with the Control Systems Technology Group, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands, and also with Philips Engineering Solutions, Philips, 5656 AE Eindhoven, The Netherlands (e-mail: n.m.j.d.vos@student.tue.nl).

Dennis Bruijnen is with Philips Engineering Solutions, Philips, 5656 AE Eindhoven, The Netherlands (e-mail: dennis.bruijnen@philips.com).

Jeroen van de Wijdeven is with ASML, 5504 DR Veldhoven, The Netherlands (e-mail: j.j.m.v.d.wijdeven@gmail.com).

Marcel Heertjes is with the Control Systems Technology Group, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands, and also with ASML, 5504 DR Veldhoven, The Netherlands (e-mail: m.f.heertjes@tue.nl).

Tom Oomen is with the Control Systems Technology Group, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands, and also with the Delft Center for Systems and Control, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: t.a.e.oomen@tue.nl).

Digital Object Identifier 10.1109/TCST.2025.3576972

function of the desired setpoint [6], [7], [8]. Similarly, in system identification, black-box parameterizations based on neural networks are used to increase model expressivity to allow for more accurate models of nonlinear systems [9], [10]. Both fields exploit the guarantee that given enough degrees of freedom, neural networks can learn all predictable dynamics [11], [12], [13] and, thus, also the hard-to-model dynamics that limit performance. Other flexible function approximators have also been used for this purpose [14], e.g., polynomials [15] and fuzzy approximators [16]. However, feedforward controllers based on neural networks and other function approximators result in uninterpretable control inputs and do not provide a baseline performance guarantee.

3) Last, physical models have been augmented with a neural network in an attempt to obtain both flexibility and interpretability in both a feedforward control [17], [18] and system identification setting [19], [20], [21], [22], [23]. Such augmented models are also referred to as hybrid feedforward controllers or physics-guided neural networks (PGNNs) [18], [24], [25], [26]. For example, a parallel physical model and a neural network feedforward controller are used to control a linear motor [17], a robotic manipulator [18], a strong nonlinear turntable servo system [27], and a 3-D printer [24]. In a feedforward control setting, these augmented models achieve superior feedforward accuracy compared to their purely physical-model-based counterparts [17]. However, in all these approaches, the neural network can still learn dynamics that could have been captured by the physical model, i.e., the augmented model is overparameterized [26]. As a result, the neural network can potentially negate the physical model, resulting in a physical model that can have arbitrary values for its physical parameters and, consequently, uninterpretable feedforward signals.

Approaches to address this overparameterization appeared in [28] and bib:29. Specifically, in [28], the differences between the augmented model's physical parameters and an available physical model are regularized during optimization. However, as also pointed out in [29], a small change in parameters can result in significantly different responses, and this regularization does not guarantee complementarity in terms of responses. In [29], a regularization is introduced, which ensures that deviations from the physical model can only happen in regions where data are available. However, this approach requires a calibrated physical model, which, in an IX setting, is not available a priori but also has to be estimated from data, as each IX has different physical parameters. If estimation of a physical model is required, joint optimization of the physical model and the neural network can achieve the highest prediction performance [30].

Although physical models and neural networks each have unique properties that are necessary for accurate feedforward control of an IX, at present, these controllers cannot be directly combined according to the constraints in medical imaging. The aim of this article is to develop a PGNN for the IX that combines a neural network and a physical model in a complementary way. This is achieved by penalizing neural network contributions that could have been explained by the physical model by modifying the cost function, guaranteeing that everything that can be captured by the physical model ends up in the physical model. This results in a physical model with interpretable parameters, allows for disabling the neural network component if desired, and guarantees that the physical model provides a baseline level of performance.

The main contribution of this article is a feedforward control method that effectively combines a physical model and a neural network according to the constraints in medical imaging and its deployment on an experimental IX setup. This consists of the following subcontributions:

- C1 a PGNN feedforward controller for the IX, consisting of a physical model describing its equations of motion and a neural network with physics-guided inputs (see Section III);
- C2 a criterion to learn the parameters of the PGNN feedforward controller including an orthogonal projection-based regularizer to ensure complementarity between the physical model and the neural network (see Section IV);
- C3 the application of the developed feedforward controller to an experimental IX setup, illustrating improved feedforward accuracy with respect to the current state-of-practice (see Section V).

Preliminary results of this research are reported in [31], in particular, results for a single axis of the IX. The current article presents a major extension. Specifically, it presents a multiaxis PGNN feedforward controller design, an optimization procedure that optimizes the PGNN for both axes simultaneously thus taking into account the coupling, and a more thorough experimental validation.

## II. Control Problem

In this section, the experimental IX setup and its associated control challenges are described. These challenges then motivate the control approach.

### A. Experimental IX Setup

The considered IX, depicted in Fig. 1, is specially designed for use around medical personnel and is currently used in operating rooms. The X-ray source and receiver are mechanically coupled using the roll body, which can rotate around the $\theta_3$-axis through rolling within the propeller body through roller-based guidance. The propeller body can rotate around the $\theta_2$-axis, moving both the propeller and roll body. Last, the base can rotate around the $\theta_1$-axis and is connected to the fixed world at the point of rotation. In this way, the X-ray beam traveling between the source and receiver can be positioned in three degrees of freedom.

All three axes are driven by a permanent magnet dc motor and amplifier with a maximum input voltage of 5 V. Each axis is equipped with a drivetrain consisting of multiple belt- and gear-based transmissions to match the available motor torque to the gravity load. The rotation of the propeller and roll axis is measured using incremental encoders with an effective resolution of 0.0669° and 0.0119°, respectively. The real-time

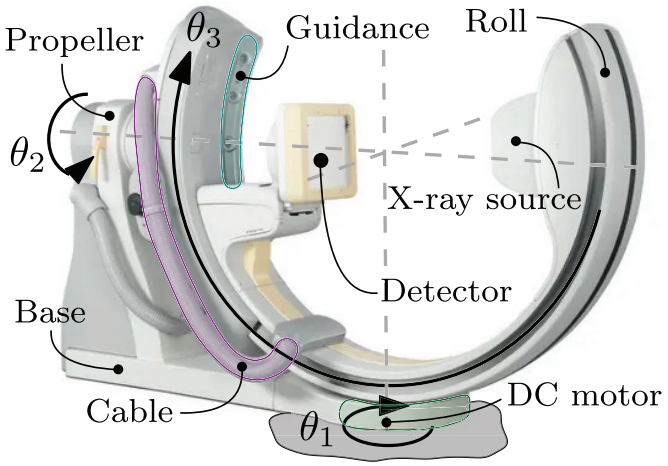


Fig. 1. IX with three degrees of freedom $\theta_1, \theta_2, \theta_3$ [rad]. The X-ray beam traveling between the source and detector can be positioned in three dimensions by rotating the base ($\theta_1$), the propeller ($\theta_2$), and the roll ($\theta_3$) body to the desired configuration. The axis is rotated using a permanent magnet dc motor ( for $\theta_1$). The roll body rolls within the propeller body using a roller-based guidance ( ) and is supplied with power through a cable ( ).

control and data acquisition are carried out using a Speedgoat Performance real-time target machine in combination with MATLAB Simulink, where both the input and output rate are $f_s = 500$ Hz.

Sensor readings and control voltages are transported to and from the real-time target machine through the base to the propeller and roll axis via the two cables at the side of the IX. In addition, these cables are used to supply the axis and the X-ray with power.

### B. Control Challenges

The mechanical design of the IX, motivated by the use of medical personnel, introduces hard-to-model nonlinear dynamics. Specifically, the following dynamics are present.

1) The centers of mass of the propeller and roll body do not coincide with the center of rotation, i.e., these bodies are not balanced. As a result, the inertia felt by the motors is configuration-dependent. The same holds for the torques required to compensate for gravitational forces.
2) The cables connecting the base, propeller, and roll body represent configuration-dependent inertia depending on the system's configuration. In addition, for specific configurations, the cable between the propeller and roll body is tensioned such that it acts as a one-sided spring.
3) The guidance for the roll body exhibits nonlinear friction characteristics. In addition, the friction can be expected to be dependent on the normal force exerted on the rollers and is, thus, also configuration-dependent.
4) Last, the system contains dynamics similar to those often found in high-tech motion systems, e.g., finite stiffness of the drivetrain transmissions resulting in flexible dynamics [32], motor characteristics resulting in torque ripple, and imperfections in drivetrain components resulting in position-dependent force disturbances [33].

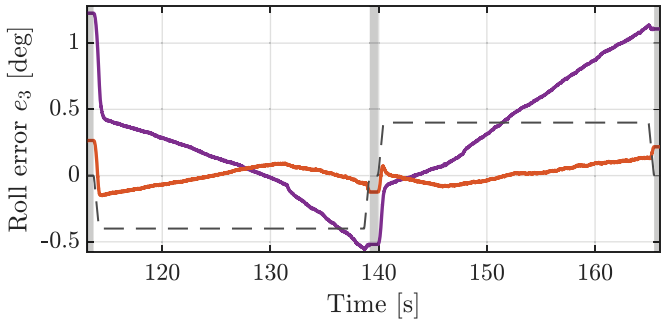


Fig. 2. Roll errors $e_3 = \theta_{3,d} - \theta_3$ without feedforward (—) and with physical-model-based feedforward (—) based on a multibody model for the IX for a specific velocity reference (– –, scaled). Although the physical model improves tracking performance, the error still clearly contains a predictable trend, especially at constant velocity intervals, originating from uncompensated dynamics not included in the model, e.g., position-dependent friction.

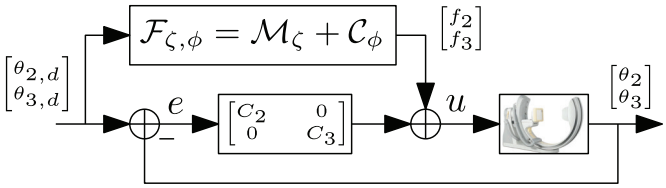


Fig. 3. Two degree-of-freedom control configuration with diagonal feedback controller $C$ and MIMO feedforward controller parameterization $\mathcal{F}_{\zeta,\phi}$, consisting of a physical model $\mathcal{M}_\zeta$ and a neural network $\mathcal{C}_\phi$.

The above dynamics inherently limit the performance of physical-model-based feedforward control: 1) whereas effect can be effectively described by physical models; 2) nonlinear effects; and 3) generally only qualitatively understood. In other words, it is known that these dynamics are present, but they are hard to model such that no physics-based expressions are available that describe their torque contributions. Consequently, these hard-to-model dynamics cannot be accurately compensated using physical-model-based feedforward control and deteriorate tracking performance.

To illustrate this deterioration, Fig. 2 shows the error for the roll ($\theta_3$) axis without feedforward control and with a state-of-practice physical-model-based feedforward controller (see Section III for details). It is clear that the physical-model-based controller improves the tracking performance with respect to feedback control only. At the same time, the error still contains a predictable trend that is known to be a direct consequence of uncompensated dynamics, e.g., errors resulting from configuration-dependent friction or stiffness.

Since the control challenges are mainly present in the propeller ($\theta_2$) and roll ($\theta_3$) axis, the rotation of the base ($\theta_1$) is disregarded in the remainder of this article.

### C. Control Architecture

To address the hard-to-model dynamics, a two-degree-of-freedom control architecture consisting of a feedback controller $C$ and feedforward controller $\mathcal{F}_{\zeta,\phi}$ is employed (see Fig. 3). In this architecture, the feedback controller $C$ acts on the difference between the outputs $\theta_2$ and $\theta_3$ and the desired angles $\theta_{2,d}$ and $\theta_{3,d}$ and is given by a diagonal controller that

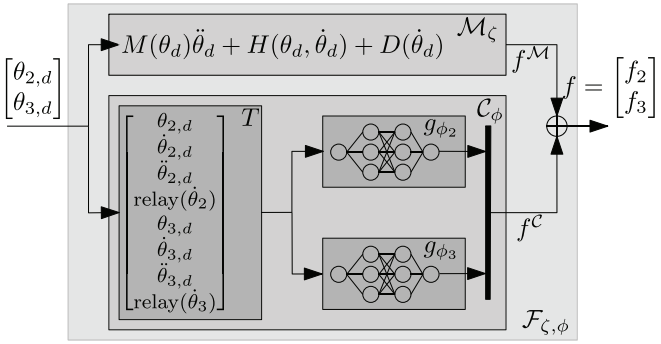


Fig. 4. PGNN feedforward controller $\mathcal{F}_{\zeta,\phi}$ consists of a physical model $\mathcal{M}_\zeta$ and a neural network $\mathcal{C}_\phi$ (see also Definition 1). Here, $\mathcal{M}_\zeta$ is a nonlinear multibody model, and $\mathcal{C}_\phi$ is a feedforward neural network with physically relevant input $T(\theta_d)$, as detailed in Sections III-A and III-B, respectively.

consists of a simple PD and low-pass filter in each axis. This controller is fixed by external considerations, results in a stable feedback loop, and is not further discussed.

The goal of this article is to obtain a feedforward controller $\mathcal{F}_{\zeta,\phi}$ that generates a discrete-time feedforward signal $f$ with $f(k) = \begin{bmatrix} f_2(k) & f_3(k) \end{bmatrix}^T \in \mathbb{R}^2$, $k \in \mathbb{Z}$ such that the output $\theta$ with $\theta(k) = \begin{bmatrix} \theta_2(k) & \theta_3(k) \end{bmatrix}^T \in \mathbb{R}^2$ tracks reference $\theta_d$, i.e., that compensates the hard-to-model dynamics, specifically the configuration-dependent friction and cable forces. This is achieved through first parameterizing $\mathcal{F}_{\zeta,\phi}$ as a PGNN (see Section III) and then learning its parameters from data (see Section IV).

## III. PGNN Feedforward Parameterization

In this section, the PGNN feedforward controller parameterization is detailed, constituting Contribution C1. The feedforward controller $\mathcal{F}_{\zeta,\phi}$ consists of the parallel combination of a physical-model-based feedforward controller $\mathcal{M}_\zeta$ and a neural network feedforward controller $\mathcal{C}_\phi$, as defined in the following.

*Definition 1:* The PGNN feedforward parameterization $\mathcal{F}_{\zeta,\phi} : \theta_d \to f$ with parameters $\zeta$ and $\phi$ is given by

$$f = \mathcal{F}_{\zeta,\phi}(\theta_d) = \mathcal{M}_\zeta(\theta_d) + \mathcal{C}_\phi(\theta_d) \tag{1}$$

with reference signal $\theta_d$, $\mathcal{M}_\zeta : \theta_d \to f^{\mathcal{M}}$, $f^{\mathcal{M}}(k) \in \mathbb{R}^2$, and $\mathcal{C}_\phi : \theta_d \to f^{\mathcal{C}}$, $f^{\mathcal{C}}(k) \in \mathbb{R}^2$.

The components $\mathcal{M}_\zeta$ (with parameters $\zeta$) and $\mathcal{C}_\phi$ (with parameters $\phi$) are further detailed in Sections III-A and III-B, respectively. A complete overview of the PGNN is given in Fig. 4. Intuitively, $\mathcal{M}_\zeta$ expresses all dynamics that are known about the system, e.g., inertia and mass unbalance, through equations of motions derived by first-principle modeling. Since the torque contributions of the hard-to-model dynamics such as cable forces are hard to express quantitatively through physics-based expressions, the neural network $\mathcal{C}_\phi$ is used as a function that is flexible enough to express these complex torque contributions as a function of $\theta$ given enough degrees of freedom [12] and can, thus, be used to learn the hard-to-model dynamics from data.

*Remark 1:* The neural network component $\mathcal{C}_\phi$ can be substituted by a different function approximator, e.g., wavelets [14], polynomials [15], or fuzzy approximators [16], as long as they can be optimized using gradient-based solvers. Here, $\mathcal{C}_\phi$ is chosen specifically as a neural network due to their ability to effectively address high-dimensional inputs [34] (see Fig. 4 and Section III-B).

TABLE I
Parameters of the Physical Model for the IX

| Parameter(s) $\zeta$ | | Definition |
|---|---|---|
| $m_2, m_3$ | $\mathbb{R}_{>0}$ | Mass of propeller and roll axis |
| $J_{2,yy}$ | $\mathbb{R}_{>0}$ | Moment of inertia of propeller axis |
| $J_{3,xx}, J_{3,yy}, J_{3,zz}$ | $\mathbb{R}_{>0}$ | Moments of inertia of roll axis |
| $J_{3,xy}, J_{3,xz}, J_{3,yz}$ | $\mathbb{R}$ | Products of inertia of roll axis |
| $x_2, z_2$ | $\mathbb{R}$ | Coordinates of center of mass of propeller with respect to origin of rotational frame |
| $x_3, y_3, z_3$ | $\mathbb{R}$ | Coordinates of center of mass of roll with respect to origin of rotational frame |
| $d_1, d_2$ | $\mathbb{R}$ | Viscous friction coefficient |

### A. Physical Model Feedforward Controller

The physical-model-based feedforward controller $\mathcal{M}_\zeta$ consists of a first-principle physical model for the IX derived using the Euler-Lagrange formalism. The inverse dynamics of this model then constitute the physical model feedforward controller. To simplify the derivation of the physical model, the following modeling assumptions are made.

- A1 The drivetrain and bodies of the IX are infinitely stiff.
- A2 The inputs to the IX are confined within the actuator saturation limits.

The above modeling assumptions are justified in practice. Specifically, in practice, the spectral content of the reference $\theta_d$ is confined to low frequencies, for which the flexible dynamics induced by the drivetrain stiffness are well-approximated by a constant gain, justifying Assumption A1. In addition, the reference is designed such the input required to track this reference is smaller than the actuator limits, justifying Assumption A2. The above assumptions allow for rigid body modeling of the system, resulting in a physical model with as many degrees of freedom as inputs, such that it can directly be inverted to obtain a physical-model-based feedforward controller.

Based on the above assumptions, the equations of motion for the generalized coordinates $\theta = \begin{bmatrix} \theta_2 & \theta_3 \end{bmatrix}^T$ with input torques $\tau = \begin{bmatrix} \tau_2 & \tau_3 \end{bmatrix}^T$ are derived using the Euler-Lagrange modeling formalism [35] and are given by

$$\tau = M(\theta)\ddot{\theta} + H(\theta, \dot{\theta}) + D(\dot{\theta}) \tag{2}$$

in which $M(\theta)\ddot{\theta}$ represents the inertial torque contributions, $D(\dot{\theta})$ represents the viscous friction, and $H(\theta, \dot{\theta})$ represent the gravity, Coriolis, and centrifugal torque contributions, with physical parameters as in Table I. $D(\dot{\theta})$ is given by

$$D(\dot{\theta}) = \begin{bmatrix} d_2\dot{\theta}_2 & d_3\dot{\theta}_3 \end{bmatrix}^T \tag{3}$$

with damping coefficients $d_1, d_2 \in \mathbb{R}$. $M(\theta)$ reads as

$$M(\theta) = \begin{bmatrix} m_{22}(\theta) & m_{23}(\theta) \\ m_{23}(\theta) & m_{33}(\theta) \end{bmatrix} = M(\theta)^T \tag{4}$$

in which the entries are given by

$$\begin{aligned} m_{22}(\theta) &= J_{2,yy} + J_{3,zz} + m_3\left(x_3^2 + y_3^2\right) + m_2\left(x_2^2 + z_2^2\right) \\ &\quad + \left(J_{3,yy} - J_{3,zz}\right)\cos(\theta_3)^2 + \left(J_{3,yz} + m_3 y_3 z_3\right) \\ &\quad \times \sin(2\theta_3) + m_3\left(-y_3^2 + z_3^2\right)\cos(\theta_3)^2 \\ m_{23}(\theta) &= \left(J_{3,xz} + m_3 x_3 z_3\right)\sin(\theta_3) \\ &\quad - \left(J_{3,xy} + m_3 x_3 y_3\right)\cos(\theta_3) \\ m_{33}(\theta) &= m_3\left(y_3^2 + z_3^2\right) + J_{3,xx}. \end{aligned} \tag{5}$$

Last, $H(\theta, \dot{\theta})$ is given by

$$H(\theta, \dot{\theta}) = \begin{bmatrix} h_2(\theta, \dot{\theta}) & h_3(\theta, \dot{\theta}) \end{bmatrix}^T \tag{6}$$

in which

$$\begin{aligned} h_2\left(\theta, \dot{\theta}\right) &= g\left(m_2 z_2 + m_3 z_3\cos(\theta_3) + m_3 y_3\sin(\theta_3)\right)\sin(\theta_2) \\ &\quad - \left(m_3 x_3 + m_2 x_2\right) g\cos(\theta_2) - 2\left(m_3 y_3 z_3 + J_{3,yz}\right) \\ &\quad \times \dot{\theta}_2\dot{\theta}_3 + \left(m_3\left(y_3^2 - z_3^2\right) - J_{3,yy} + J_{3,zz}\right)\dot{\theta}_2\dot{\theta}_3 \\ &\quad \times \sin(2\theta_3) + 4\left(m_3 y_3 z_3 + J_{3,yz}\right)\dot{\theta}_2\dot{\theta}_3\cos(\theta_3)^2 \\ &\quad + \left(J_{3,xy} + m_3 x_3 y_3\right)\dot{\theta}_3^2\sin(\theta_3) \\ &\quad + \left(J_{3,xz} + m_3 x_3 z_3\right)\dot{\theta}_3^2\cos(\theta_3) \end{aligned} \tag{7}$$

$$\begin{aligned} h_3\left(\theta, \dot{\theta}\right) &= m_3 g\left(y_3\cos(\theta_3) - z_3\sin(\theta_3)\right)\cos(\theta_2) \\ &\quad + \frac{1}{2}\dot{\theta}_2^2\left(m_3\left(\left(z_3^2 - y_3^2\right)\sin(2\theta_3) - 2\ y_3 z_3\cos(2\theta_3)\right)\right. \\ &\quad + \frac{1}{2}\dot{\theta}_2^2\left(\left(J_{3,yy} - J_{3,zz}\right)\sin(2\theta_3) - 2J_{3,yz}\cos(2\theta_3)\right). \end{aligned} \tag{8}$$

The $n_\zeta = 16$ parameters of the physical model and their interpretation are summarized in Table I. The physical model does not depend on $y_2$, representing the translational invariance of the equations along the propeller axis. Note that in this section, the parameter $\zeta$ is considered to be unknown. Instead, these are learned from data in Section IV.

The equations of motion (2) can be used to obtain the physical-model-based feedforward controller. First, the continuous time map from a desired reference $\theta_d, \dot{\theta}_d, \ddot{\theta}_d$ to the required torques, i.e., the inverse physical model, is trivially obtained evaluating the right-hand side of (2) for a desired $\theta_d$, obtaining the required $\tau$ as output. This is possible because of the rigid-body nature of the model. Second, the feedforward controller is digital such that $\theta_d, \dot{\theta}_d, \ddot{\theta}_d$ are sampled, resulting in a sampled physical-model-based feedforward controller, as formalized in the following.

*Definition 2:* Given reference $\theta_d$ with $\theta_d(k) \in \mathbb{R}^2$, the physical-model-based feedforward controller $\mathcal{M}_\zeta : \theta_d \to f^{\mathcal{M}}$ with parameters $\zeta \in \mathbb{R}^{n_\zeta}$ is given by

$$f^{\mathcal{M}}(k) = M(\theta_d(k))\ddot{\theta}_d(k) + H(\theta_d(k), \dot{\theta}_d(k)) + D(\dot{\theta}_d(k)) \tag{9}$$

in which $\dot{\theta}_d(k)$ is the discrete-time derivative of $\theta_d(k)$.

Here, it is chosen to calculate $\dot{\theta}_d(k) = f_s(\theta_d(k+1) - \theta_d(k))$ with sampling rate $f_s$. Of course, other discrete approximations of the time derivative are possible.

In conclusion, a physical-model-based feedforward controller $\mathcal{M}_\zeta$ is obtained that contains physics-based expressions for the torqued required to compensate for inertial, viscous friction, Coriolis, and gravitational effects.

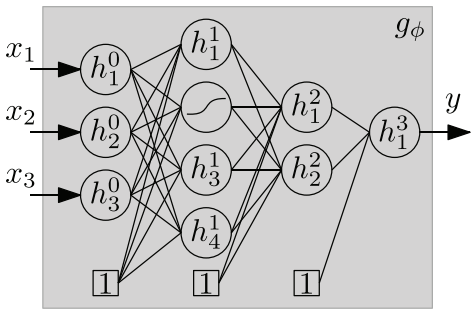


Fig. 5. Example of a feedforward neural network with $L = 3$ hidden layers of size $n_1 = 4, n_2 = 2, n_3 = 1$, $n_0 = n_x = 3$ inputs, and $n_4 = n_y = 1$ output. Each connection between nodes represents an entry of the weight matrices/bias vector.

### B. Neural Network Feedforward Controller

To capture torque contributions that cannot be captured by the physical-model-based feedforward $\mathcal{M}_\zeta$, $\mathcal{M}_\zeta$ is complemented by a neural network feedforward controller $\mathcal{C}_\phi$. $\mathcal{C}_\phi$ consists of two neural networks $g_{\phi_2}$ and $g_{\phi_3}$, one for the propeller axis ($g_{\phi_2}$) and one for the roll axis ($g_{\phi_3}$), both acting on the same physically relevant input vector $T(\theta_d(k))$. The structure of $g_{\phi_2}, g_{\phi_3}$ and the transformation $T$ are outlined in the following.

*1) Neural Network Structure:* In this article, $g_{\phi_2}$ and $g_{\phi_3}$ are parameterized as feedforward neural networks (a neural network without internal feedback loops), as defined in the following.

*Definition 3:* A feedforward neural network $y = g_\phi(x)$ with parameter $\phi$, output $y \in \mathbb{R}$, and input $x \in \mathbb{R}^{n_x}$ is given by

$$\begin{aligned} h^0(x) &= x \\ h^l(x) &= \sigma\left(W^{l-1}h^{l-1}(x) + c^l\right), \quad \text{for } l = 1, \ldots, L \\ g_\phi(x) &= W^L h^L(x) \end{aligned} \tag{10}$$

with hidden layers $h^l \in \mathbb{R}^{n_l}$, $l = 1, \ldots, L$, weight matrices $W^l \in \mathbb{R}^{n_{l+1} \times n_l}$, $l = 0, 1, \ldots, L$, $n_0 = n_x$, $n_{L+1} = 1$, bias vectors $c^l$, $l = 1, \ldots, L$, and elementwise nonlinear activation function $\sigma(\cdot)$. The parameters of the neural network are given by $\phi = [\mathrm{vec}(W^0), c^0, \mathrm{vec}(W^1), c^1, \ldots, \mathrm{vec}(W^L)]$.

The first layer $h^0$ is called the input layer, the last layer $g_\phi(x)$, the output layer, and $h^l$, $l = 1, \ldots, L$ and the $L$ hidden layers of the neural network. At each hidden layer, the data are transformed by an affine mapping governed by $W^{l-1}$ and $c^l$, followed by an elementwise nonlinear activation function $\sigma(\cdot)$ such as a rectified linear unit (ReLU), sigmoid, or hyperbolic tangent tanh. An example of a neural network is schematically represented in Fig. 5. The number of hidden layers, the size of each hidden layer, and the type of activation functions are user-defined hyperparameters and are detailed for this application in Section V.

*2) Physically Relevant Input:* To learn the required input corresponding to a desired output $\theta_d$, the input to the neural network should contain all relevant physical quantities. For the IX, the physically relevant input $T(\theta_d(k))$ is defined as

$$\begin{aligned} T(\theta_d(k)) &= \begin{bmatrix} T_2(\theta_d(k))^T & T_3(\theta_d(k))^T \end{bmatrix}^T \\ T_i(\theta_d(k)) &= \begin{bmatrix} \theta_{i,d}(k) & \dot{\theta}_{i,d}(k) & \ddot{\theta}_{i,d}(k) & \mathrm{relay}(\dot{\theta}_{i,d}(k)) \end{bmatrix}^T \end{aligned} \tag{11}$$

in which the relay function is defined as

$$\text{relay}(x(k)) = \begin{cases} 1, & \text{if } x(k) > 0 \\ -1, & \text{if } x(k) < 0 \\ \text{relay}(x(k-1)), & \text{otherwise} \end{cases} \tag{12}$$

with $\text{relay}(x(k)) = 0$ for $k \leq 0$. This input vector encapsulates the qualitative prior knowledge on the unmodeled dynamics: the position-varying inertia and stiffness of the cable are a function of both $\theta_2$ and $\theta_3$, requiring that both are included as the input to the neural network. In addition, the unmodeled dynamics are not only dependent on position but also on velocity (nonlinear friction) and acceleration (position-varying inertia), necessitating their inclusion in $T(\theta_d)$. Last, the relay function is included as a rudimentary heuristic to approximate hysteresis characteristics by keeping track of the history of the sign of the velocity.

*3) Neural Network Feedforward Controller:* Using the general feedforward neural network in Definition 3 and physically relevant input (11), the neural network feedforward controller is defined as follows.

*Definition 4:* Given the reference signal $\theta_d$, the neural network feedforward controller $\mathcal{C}_\phi \colon \theta_d \to f_\mathcal{C}$ with parameters $\phi = [\phi_2^T, \phi_3^T]^T \in \mathbb{R}^{n_\phi}$ is given by

$$f_\mathcal{C}(k) = \begin{bmatrix} f_2^\mathcal{C}(k) \\ f_3^\mathcal{C}(k) \end{bmatrix} = \mathcal{C}_\phi(\theta_d(k)) = \begin{bmatrix} g_{\phi_2}(T(\theta_d(k))) \\ g_{\phi_3}(T(\theta_d(k))) \end{bmatrix} \tag{13}$$

with $T(\cdot)$ as in (11) and $g_{\phi_2}(\cdot), g_{\phi_3}(\cdot)$ neural networks as in Definition 3.

The neural network feedforward controller $\mathcal{C}_\phi$ represents two independently parameterized neural networks: one for the propeller axis ($g_{\phi_2}$) and one for the roll axis ($g_{\phi_3}$). In general, the two networks have a different structure, i.e., differing in the number of hidden layers and their size, denoted by the two separate parameter sets $\phi_2$ and $\phi_3$. Both have the same physically relevant input vector $T(\theta_d)$.

With this neural network feedforward controller, the PGNN feedforward parameterization $\mathcal{F}_{\zeta,\phi}$ is fully defined.

*Remark 2:* Note that the stability of the equilibrium configuration of the IX (see Fig. 1) is unaltered by the feedforward controller $\mathcal{F}_{\zeta,\phi}$ as $f$ is not a function of the internal states of the loop. Stability guarantees in terms of forward invariance of a safe set are guaranteed by a safety layer already implemented on the IX.

## IV. Learning Feedforward Parameters From Data

Given the PGNN feedforward controller parameterization $\mathcal{F}_{\zeta,\phi}$, in this section, it is detailed how its parameters $\zeta, \phi$ are optimized from data, enabling the learning of the hard-to-model dynamics, constituting Contribution C2.

### A. Cost Criterion for Optimization

Intuitively, $\mathcal{F}_{\zeta,\phi}$ is optimized such that given a measured output $\theta$, it accurately predicts the corresponding input $u$. To achieve the highest prediction performance, $\mathcal{M}_\zeta$ and $\mathcal{C}_\phi$ should be optimized simultaneously [30]. Thus, given a dataset $\mathcal{D}$ with $N$ input voltages and output angles, i.e., $\mathcal{D} = \{u(k), \theta(k)\}_{k=1}^N$ with $u(k) = \begin{bmatrix} u_2(k) & u_3(k) \end{bmatrix}^T$ and $\theta(k) = \begin{bmatrix} \theta_2(k) & \theta_3(k) \end{bmatrix}^T$, $\mathcal{F}_{\zeta,\phi}$ is optimized as follows.

*Definition 5:* Given $\mathcal{D}$, $\mathcal{F}_{\zeta,\phi}$ is optimized according to $\arg\min_{\zeta,\phi} J(\zeta, \phi)$ with

$$J(\zeta, \phi) = J_{\text{LS}}(\zeta, \phi) + \mu_1 R_1(\phi) + \mu_2 R_2(\zeta) \tag{14}$$

with

$$J_{\text{LS}}(\zeta, \phi) = \sum_{k=1}^{N} \left\| \begin{bmatrix} u_2(k) \\ u_3(k) \end{bmatrix} - \begin{bmatrix} f_2^\mathcal{M}(k) + f_2^\mathcal{C}(k) \\ f_3^\mathcal{M}(k) + f_3^\mathcal{C}(k) \end{bmatrix} \right\|_2^2 \tag{15}$$

in which $f^\mathcal{M}(k) = \mathcal{M}_\zeta(\theta(k))$ and $f^\mathcal{C}(k) = \mathcal{C}_\phi(\theta(k))$ and regularization terms $R_1 \colon \mathbb{R}^{n_\phi} \to \mathbb{R}_{\geq 0}$ and $R_2 \colon \mathbb{R}^{n_\zeta} \to \mathbb{R}_{\geq 0}$ with weighting $\mu_1 \in \mathbb{R}_{\geq 0}$ and $\mu_2 \in \mathbb{R}_{\geq 0}$.

The cost criterion $J(\zeta, \phi)$ in (14) consists of three contributions.

1) A least-squares contribution $J_{\text{LS}}(\zeta, \phi)$ that simultaneously optimizes $\zeta, \phi$ such that $\mathcal{F}_{\phi,\zeta}(\theta)$ accurately predicts $u$. This can be interpreted as a system identification criterion in terms of the inverse dynamics, i.e., identifying the system with input $\theta$ and output $u$ [36].
2) An orthogonal projection-based regularization $R_1(\phi)$ that penalizes neural network outputs $f^\mathcal{C}(k)$ that could have been generated by the physical model, promoting that $\mathcal{M}_\zeta$ and $\mathcal{C}_\phi$ are complementary.
3) A physics-based regularization $R_2(\zeta)$ incorporating prior knowledge on physical quantities, e.g., mass $m$ should be positive.

Regularization contributions $R_1(\phi)$ and $R_2(\zeta)$ are further detailed in the following. The section concludes with an overview of the optimization procedure. The details about $\mathcal{D}$ for the IX application can be found in Section V.

### B. Orthogonal Projection-Based Regularization

The orthogonal projection-based regularization $R_1(\phi)$ is required to enforce complementarity between $\mathcal{M}_\zeta$ and $\mathcal{C}_\phi$. Specifically, since the neural network $\mathcal{C}_\phi$ is rich enough to also learn dynamics included in $\mathcal{M}_\zeta$, there exist multiple parameters $\zeta, \phi$ that result in the same input-output map $\mathcal{F}_{\zeta,\phi}$, i.e., $\mathcal{F}_{\zeta,\phi}$ is unidentifiable [26]. As a result, $f_2^\mathcal{M}$ can be negated by $f_2^\mathcal{C}$ while still correctly predicting $u$ in (15). This unidentifiability can result in physically inconsistent parameters and a physical model that cannot be used as a stand-alone baseline. Regularization $R_1(\phi)$ prevents this negation by penalizing the contribution of $f^\mathcal{C}$ in the subspace in which the physical model $\mathcal{M}_\zeta$ can generate outputs, thus promoting complementarity. The remainder of this section further details these statements.

*1) Example of Unidentifiability:* As an illustrative example, consider for a moment that the physics-based torque contribution $f_2^\mathcal{M}(k)$ only consists of $J_{2,yy}\ddot{\theta}_2(k)$, i.e., an inertia term. Since $\mathcal{C}_\phi$ is a neural network, there exists parameters $\phi_2$ such that $g_{\phi_2}(T(\theta(k))) = f_2^\mathcal{C}(k) \approx s\ddot{\theta}_2(k)$ for any scaling $s \in \mathbb{R}$. Thus, there exists infinite $J_{2,yy}$ and $\phi_2$ for which $f_2^\mathcal{M}(k) + f_2^\mathcal{C}(k) = (J_{2,yy} + s)\ddot{\theta}_2(k)$ remains unchanged, i.e., $\mathcal{M}_\zeta$ can have any value for $J_{2,yy}$ and $\mathcal{F}_{\zeta,\phi}$ is unidentifiable. As a consequence, the optimized value for $J_{2,yy}$ can be completely different from the true physical quantity.

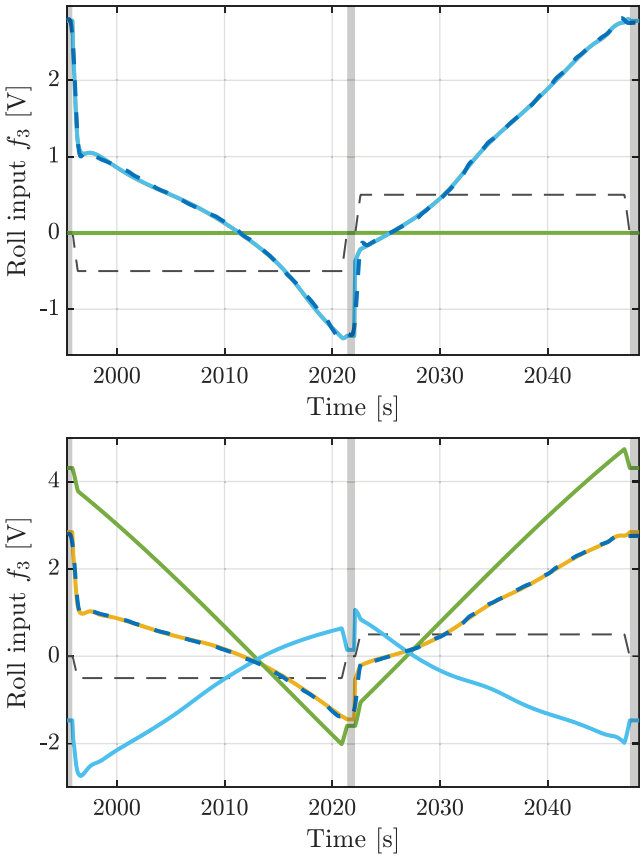


Fig. 6. Two realizations of physical model (—) and neural network (—) feedforward signals $f_{3,\mathcal{M}}, f_{3,\mathcal{C}}$ that together (—) generate the required input $\hat{u}_3$ (- -) but are not complementary, i.e., the neural network can capture all dynamics resulting in a trivial physical model contribution (upper), or the signals can even be opposite (lower). Both realizations have the same input-output relation, illustrating that the parallel combination $\mathcal{F}_{\zeta,\phi}$ is unidentifiable, such that the minimizer to $J_{\text{LS}}$ is nonunique, resulting in a physical model that cannot be used as a baseline. (- -) represents the scaled velocity reference.

Although the above example might seem toy, this mechanism of unidentifiability is also observed in practice. As an example, Fig. 6 shows two realizations of the learned input signals $f_3^{\mathcal{M}}, f_3^{\mathcal{C}}$ after optimizing $\mathcal{F}_{\zeta,\phi}$ according to $\arg\min_{\zeta,\phi} J_{\text{LS}}(\zeta,\phi)$, i.e., without $R_1(\phi)$ and $R_2(\zeta)$. It can be observed that $f_3^{\mathcal{C}}$ can fully negate $f_3^{\mathcal{M}}$ (upper) or that $f_3^{\mathcal{C}}$ can have similar but opposing contributions compared to $f_3^{\mathcal{M}}$ (lower), while, in both cases, $\mathcal{F}_{\zeta,\phi}$ accurately predicts $u_3$. In other words, the contributions of $\mathcal{M}_\zeta$ and $\mathcal{C}_\phi$ are not complementary. As a consequence, $\mathcal{M}_\zeta$ cannot be used as an effective stand-alone feedforward controller, $\mathcal{C}_\phi$ cannot be disabled, and no physically meaningful coefficients $\zeta$ are obtained from the optimization.

Thus, there exist multiple parameters $\zeta, \phi$ with the same input-output behavior, i.e., $\mathcal{F}_{\zeta,\phi}$ is unidentifiable. This unidentifiability is investigated in the following.

*2) Unidentifiability:* To investigate the unidentifiability of $\mathcal{F}_{\zeta,\phi}$, $J_{\text{LS}}(\zeta,\phi)$ in (15) is decomposed in two complementary subspaces, one describing the subspace in which $\mathcal{M}_\zeta$ can generate outputs given the dataset $\mathcal{D}$ and one describing its orthogonal complement. From this decomposition, it is apparent that the neural network has a contribution in the subspace in which the physical model can generate outputs.

To obtain this decomposition of $J_{\text{LS}}(\zeta,\phi)$, first define $\zeta_l = [m_2, m_3, J_{2,yy}, J_{3,xx}, J_{3,yy}, J_{3,zz}, J_{3,xy}, J_{3,xz}, J_{3,yz}, b_2, b_3]^T \in \mathbb{R}^{n_{\zeta_l}}$ and $\zeta_n = [x_2, z_2, x_3, y_3, z_3] \in \mathbb{R}^{n_{\zeta_n}}$ with $n_{\zeta_l} = 11$, $n_{\zeta_n} = 5$. Then, the physical-model-based feedforward signal $f^{\mathcal{M}}$ can be written as follows.

*Lemma 1:* Given $\zeta_l, \zeta_n$, $f^{\mathcal{M}}(k)$ can be written as

$$\begin{bmatrix} f_2^{\mathcal{M}}(k) \\ f_3^{\mathcal{M}}(k) \end{bmatrix} = \begin{bmatrix} X_{2,\zeta_n}(\theta(k)) \\ X_{3,\zeta_n}(\theta(k)) \end{bmatrix} \zeta_l = X_{\zeta_n}(\theta(k))\zeta_l \tag{16}$$

in which $X_{2,\zeta_n}(\theta(k)), X_{3,\zeta_n}(\theta(k)) \in \mathbb{R}^{1\times n_{\zeta_l}}$ are given by

$$\begin{bmatrix} X_{2,\zeta_n}(\theta(k)) \\ X_{3,\zeta_n}(\theta(k)) \end{bmatrix} = \nabla_{\zeta_l} f^{\mathcal{M}}(k) \tag{17}$$

with $f^{\mathcal{M}}(k)$ as in (9).

*Proof:* By inspection of (2)–(9), it follows that $f^{\mathcal{M}}(k)$ is linear in parameters $\zeta_l$. By this linearity, $f^{\mathcal{M}}(k)$ can be written as (16) with $X_{2,\zeta_n}(\theta(k)), X_{3,\zeta_n}(\theta(k))$ as in (17).□

The above lemma expresses the response of $\mathcal{M}_\zeta$ as a linear combination of $\zeta_n$- and $\theta$-dependent basis functions. As an example, by (2)–(8), the first basis function of $X_{2,\zeta_n}(\theta_d)$ corresponding to coefficient $m_2$ is given by

$$\left(x_2^2 + z_2^2\right)\ddot{\theta}_2(k) - g\left(x_2\cos(\theta_2(k)) + z_2\sin(\theta_2(k))\right). \tag{18}$$

Expressing $f^{\mathcal{M}}(k)$ as a sum of basis functions allows for investigating the subspace in which $\mathcal{M}_\zeta$ can generate outputs given the dataset $\mathcal{D}$ of length $N$. Specifically, define the finite-time vectorized representation of $\theta$ with $\theta(k) \in \mathbb{R}^2$ as

$$\underline{\theta} = \begin{bmatrix} \theta^T(1) & \theta^T(2) & \cdots & \theta^T(N) \end{bmatrix}^T \in \mathbb{R}^{2N} \tag{19}$$

and, similarly, for $\underline{u}$, $\underline{f}^{\mathcal{M}}$, and $\underline{f}^{\mathcal{C}}$. Furthermore, define

$$X_{2,\zeta_n}(\underline{\theta}) = \begin{bmatrix} X_{2,\zeta_n}^T(\theta(1)) & \cdots & X_{2,\zeta_n}^T(\theta(N)) \end{bmatrix}^T \in \mathbb{R}^{N\times n_{\zeta_l}} \tag{20}$$

as the vectorized response of $X_{2,\zeta_n}(\theta(k))$ for $k = 1, \ldots, N$, and define $X_{3,\zeta_n}(\underline{\theta})$, $g_{\phi_2}(T(\underline{\theta}))$, and $g_{\phi_3}(T(\underline{\theta}))$ similarly. Given this vectorized representation of $X_{i,\zeta_n}(\underline{\theta})$, define its singular value decomposition (SVD) as

$$X_{i,\zeta_n}(\underline{\theta}) = \begin{bmatrix} U_{i,\zeta_n} & W_{i,\zeta_n} \end{bmatrix} \begin{bmatrix} \Sigma_{i,\zeta_n} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_{i,\zeta_n} & V_{i,\zeta_n}^{\perp} \end{bmatrix}^T \tag{21}$$

with orthonormal matrices $U_{i,\zeta_n} \in \mathbb{R}^{N\times r_i}$, $V_{i,\zeta_n} \in \mathbb{R}^{n_{\zeta_l}\times r_i}$, $W_{i,\zeta_n} \in \mathbb{R}^{N\times N-r_i}$, $V_{i,\zeta_n}^{\perp} \in \mathbb{R}^{n_{\zeta_l}\times n_{\zeta_l}-r_i}$, and diagonal matrix $\Sigma_{i,\zeta_n} \in \mathbb{R}^{r_i\times r_i}$ with $r_i = \text{rank}(X_{i,\zeta_n}) \leq n_{\zeta_l}$ for $i = 2, 3$.

Using these vectorized representations and their SVD, the cost criterion $J_{\text{LS}}(\zeta,\phi)$ can be decomposed as follows.

*Theorem 1:* Given $f^{\mathcal{M}}(k)$ as in Lemma 1 and SVD (21), $J_{\text{LS}}(\zeta,\phi)$ in (15) can be written as

$$\begin{aligned} J_{\text{LS}}(\zeta,\phi) = &\left\| U_{2,\zeta_n}^T\underline{u}_2 - \left(\Sigma_{2,\zeta_n}V_{2,\zeta_n}^T\zeta_l + U_{2,\zeta_n}^T g_{\phi_2}(T(\underline{\theta}))\right)\right\|_2^2 \\ &+ \left\| W_{2,\zeta_n}^T\underline{u}_2 - W_{2,\zeta_n}^T g_{\phi_2}(T(\underline{\theta}))\right\|_2^2 \\ &+ \left\| U_{3,\zeta_n}^T\underline{u}_3 - \left(\Sigma_{3,\zeta_n}V_{3,\zeta_n}^T\zeta_l + U_{3,\zeta_n}^T g_{\phi_3}(T(\underline{\theta}))\right)\right\|_2^2 \\ &+ \left\| W_{3,\zeta_n}^T\underline{u}_3 - W_{3,\zeta_n}^T g_{\phi_3}(T(\underline{\theta}))\right\|_2^2. \end{aligned} \tag{22}$$

*Proof:* First, given $f^{\mathcal{M}}(k)$ as in Lemma 1, $J_{\text{LS}}(\zeta,\phi)$ in (15) can be written as

$$J_{\text{LS}}(\zeta,\phi)=\sum_{k=1}^{N}\left\|\begin{bmatrix}u_2(k)\\u_3(k)\end{bmatrix}-\begin{bmatrix}X_{2,\zeta_n}(\theta(k))\\X_{3,\zeta_n}(\theta(k))\end{bmatrix}\zeta_l+\begin{bmatrix}f_2^{\mathcal{C}}(k)\\f_3^{\mathcal{C}}(k)\end{bmatrix}\right\|_2^2 \tag{23}$$

which, by using the finite-time vectorized representation of $u(k), X_{2,\zeta_n}(\theta(k)), X_{3,\zeta_n}(\theta(k)), f_2^{\mathcal{C}}(k) = g_{\phi_2}(\theta(k)), f_3^{\mathcal{C}}(k) = g_{\phi_3}(\theta(k))$, can be equivalently written as

$$J_{\text{LS}}(\zeta,\phi)=\left\|\begin{bmatrix}\underline{u}_2\\\underline{u}_3\end{bmatrix}-\begin{bmatrix}X_{2,\zeta_n}(\underline{\theta})\zeta_l+g_{\phi_2}(T(\underline{\theta}))\\X_{3,\zeta_n}(\underline{\theta})\zeta_l+g_{\phi_3}(T(\underline{\theta}))\end{bmatrix}\right\|_2^2. \tag{24}$$

Decomposing $X_{2,\zeta_n}(\underline{\theta})$ and $X_{3,\zeta_n}(\underline{\theta})$ according to (21) gives that (24) can be written as

$$J_{\text{LS}}(\zeta,\phi)=\sum_{i=2}^{3}\left\|\underline{u}_i-U_{i,\zeta_n}\Sigma_{i,\zeta_n}V_{i,\zeta_n}^T\zeta_l-g_{\phi_i}(T(\underline{\theta}))\right\|_2^2. \tag{25}$$

Now note that by the properties of this SVD, it holds that

$$U_{i,\zeta_n}U_{i,\zeta_n}^T+W_{i,\zeta_n}W_{i,\zeta_n}^T=I_N \qquad U_{i,\zeta_n}^TU_{i,\zeta_n}=I_{r_i} \tag{26}$$

for $i=2,3$. Consequently, $(U_{2,\zeta_n}U_{2,\zeta_n}^T+W_{2,\zeta_n}W_{2,\zeta_n}^T)\underline{u}_2=\underline{u}_2$ and, similarly, for $\underline{u}_3$, $g_{\phi_2}(T(\underline{\theta}))$, and $g_{\phi_3}(T(\underline{\theta}))$ such that (24) can be written as

$$\begin{aligned}&J_{\text{LS}}(\zeta,\phi)\\&=\sum_{i=2}^{3}\Big\|W_{i,\zeta_n}\left(W_{i,\zeta_n}^T\underline{u}_i-W_{i,\zeta_n}^Tg_{\phi_i}(T(\underline{\theta}))\right)\\&\qquad+U_{i,\zeta_n}\left(U_{i,\zeta_n}^T\underline{u}_i-\Sigma_{i,\zeta_n}V_{i,\zeta_n}^T\zeta_l-U_{i,\zeta_n}^Tg_{\phi_i}(T(\underline{\theta}))\right)\Big\|_2^2.\end{aligned} \tag{27}$$

Since $U_{i,\zeta_n}$ and $W_{i,\zeta_n}$ are orthonormal, i.e., it holds that $U_{i,\zeta_n}^TW_{i,\zeta_n}=0$, the norm of the sum in (27) is equal to the sum of the norms such that expanding (27) gives

$$\begin{aligned}&J_{\text{LS}}(\zeta,\phi)\\&=\sum_{i=2}^{3}\Big\|W_{i,\zeta_n}\left(W_{i,\zeta_n}^T\underline{u}_i-W_{i,\zeta_n}^Tg_{\phi_i}(T(\underline{\theta}))\right)\Big\|_2^2\\&\quad+\Big\|U_{i,\zeta_n}\left(U_{i,\zeta_n}^T\underline{u}_i-\Sigma_{i,\zeta_n}V_{i,\zeta_n}^T\zeta_l-U_{i,\zeta_n}^Tg_{\phi_i}(T(\underline{\theta}))\right)\Big\|_2^2.\end{aligned} \tag{28}$$

Last, $U_{i,\zeta_n}, W_{i,\zeta_n}$ can be removed from the norm as they are orthonormal, i.e., $U_{i,\zeta_n}^TU_{i,\zeta_n}=I_{r_i}, W_{i,\zeta_n}^TW_{i,\zeta_n}=I_{N-r_i}$ such that (28) is equivalent to (22), completing the proof.□

SVD (21) immediately shows the subspace in which $\mathcal{M}_\zeta$ can generate outputs for this dataset $\mathcal{D}$, namely, in the images of $U_{2,\zeta_n}, U_{3,\zeta_n}$. Similarly, $W_{2,\zeta_n}, W_{3,\zeta_n}$ represent the subspaces in which $\mathcal{M}_\zeta$ cannot generate outputs. Note that these subspaces are dependent on $\zeta_n$. The terms $U_{i,\zeta_n}^T\underline{u}_i$ and $W_{i,\zeta_n}^T\underline{u}_i$ represent the contribution of the vectorized input $\underline{u}_i$ in these subspaces. Consequently, the decomposition of $J_{\text{LS}}(\zeta,\phi)$ according to these subspaces shows that if $W_{i,\zeta_n}^T\underline{u}_i\neq 0$, then the physical model cannot explain this contribution, as evidenced by the absence of $\zeta_l$ in the second and fourth terms of (22). In addition, this decomposition immediately shows that $\mathcal{C}_\phi$ can negate $\mathcal{M}_\theta$: since $\mathcal{C}_\phi$ is a neural network, it can also generate outputs in $U_{2,\zeta_n}, U_{3,\zeta_n}$, i.e., $U_{i,\zeta_n}g_{\phi_i}(T(\underline{\theta}))\neq 0$ for $i=2,3$. Thus, contributions can be freely exchanged between $U_{2,\zeta_n}g_{\phi_2}(T(\underline{\theta}))$ and $\Sigma_{2,\zeta_n}V_{2,\zeta_n}^T\zeta_l$ as exemplified in Fig. 6. Furthermore, as long as $W_{2,\zeta_n}g_{\phi_2}(T(\underline{\theta}))$ remains the same, the combined output $f^{\mathcal{M}}+f^{\mathcal{C}}$ and, thus, $J_{\text{LS}}(\zeta,\phi)$ remain unchanged. This option to exchange of contributions between $\mathcal{M}_\zeta$ and $\mathcal{C}_\phi$ illustrates the unidentifiability of $\mathcal{F}_{\zeta,\phi}$ and the mechanism behind $\mathcal{C}_\phi$ negating $\mathcal{M}_\zeta$.

*3) Orthogonal Projection-Based Regularization:* To counteract the negation of $\mathcal{M}_\zeta$ by $\mathcal{C}_\phi$ and obtain physically relevant parameters when optimizing the unidentifiable parameterization $\mathcal{F}_{\zeta,\phi}$, orthogonal projection-based regularization is employed [26], which is defined as follows.

*Definition 6:* Given parameters $\phi$ and data $\mathcal{D}$, orthogonal projection-based regularization $R_1(\phi)$ is defined as

$$R_1(\phi)=\left\|U_{2,\zeta_n^0}^Tg_{\phi_2}(T(\underline{\theta}))\right\|_2^2+\left\|U_{3,\zeta_n^0}^Tg_{\phi_3}(T(\underline{\theta}))\right\|_2^2 \tag{29}$$

with $\zeta_n^0$ being a priori fixed approximation of $\zeta_n$.

Regularization $R_1(\phi)$ penalizes the contribution of $\mathcal{C}_\phi$ in the subspace in which the physical model can generate outputs for a given $\zeta_n^0$, i.e., it penalizes $U_{2,\zeta_n^0}^Tg_{\phi_2}(T(\underline{\theta}))$ and $U_{3,\zeta_n^0}^Tg_{\phi_3}(T(\underline{\theta}))$. If $\zeta_n^0$ is appropriately chosen, subspaces $U_{2,\zeta_n^0}, U_{3,\zeta_n^0}$ correspond to $U_{2,\zeta_n}, U_{3,\zeta_n}$, i.e., the subspace in which contributions can be exchanged between $\mathcal{C}_\phi$ and $\mathcal{M}_\zeta$, as illustrated by decomposition (22) (see Theorem 1). Consequently, $R_1(\phi)$ penalizes the exchange of contributions between $\mathcal{M}_\zeta$ and $\mathcal{C}_\phi$, promoting complementarity and preventing the negation of $f^{\mathcal{M}}$ by $f^{\mathcal{C}}$.

The choice to fix $\zeta_n$ to a fixed approximation $\zeta_n^0$ in $R_1(\phi)$ is motivated by a computational argument. Specifically, to guarantee complementarity between $\mathcal{M}_\zeta$ and $\mathcal{C}_\phi$, the true $U_{2,\zeta_n}, U_{3,\zeta_n}$ should be used in $R_1(\phi)$. However, this would require updating $U_{2,\zeta_n}, U_{3,\zeta_n}$ during optimization, as changing $\zeta_n$ also changes the output space of $\mathcal{M}_\zeta$. This update introduces a significant computational burden, especially for large datasets. As an alternative, $\zeta_n$ can be set to a fixed approximation $\zeta_n^0$. Even though $\zeta_n^0$ might not exactly correspond to the true value of $\zeta_n$, for the IX application, $\zeta_n$ is known to be within reasonable bounds already before identification from mechanical design specifications. Consequently, in practice, fixing $\zeta_n$ to $\zeta_n^0$ results in $U_{2,\zeta_n^0}, U_{3,\zeta_n^0}$ that offer a good enough approximation of the true $U_{2,\zeta_n}, U_{3,\zeta_n}$. This is further supported by the fact that updating $U_{i,\zeta_n}$ during optimization does not improve results.

This orthogonal projection-based regularization $R_1(\phi)$ is added to $J_{\text{LS}}(\zeta,\phi)$ as in (14) with weighting $\mu_1$, which represents the degree of desired complementarity: it creates a spectrum between the unregularized case $J_{\text{LS}}$ with $\mu_1=0$ and full orthogonality for $\mu_1\to\infty$.

### C. Physics-Based Regularization

In addition to the orthogonal projection-based regularization, other physical prior knowledge on physical quantities is imposed during optimization through regularization $R_2(\zeta)$. Specifically, $R_2(\zeta)$ is defined as follows.

*Definition 7:* Given parameters $\zeta$, physics-based regularization $R_2(\zeta)$ is defined as

$$R_2(\zeta)=\sum_{i=1}^{9}\max(0,\exp(-h_i(\zeta))-1) \tag{30}$$

**Algorithm 1** Optimization Procedure for $\mathcal{F}_{\zeta,\phi}$

1: **inputs**: $\mathcal{M}_\zeta$ in (9), $\mathcal{C}_\phi$ in (13), a dataset $\mathcal{D} = \{u_2(k), u_3(k), \theta_2(k), \theta_3(k)\}_{k=1}^N$, regularization weighting $\mu_1, \mu_2 \in \mathbb{R}_{\geq 0}$, $\zeta_0$ as the best physical model fit and $\phi_0$.
2: **calculate** the basis function matrix $X_{i,\zeta_n^0}(\underline{\theta})$, $i = 2, 3$ of $\mathcal{M}_\zeta$ for $\zeta^0$, see Lemma 1.
3: **calculate** the orthogonal basis $U_{i,\zeta_n^0}$, $i = 2, 3$ of the output space of $X_{i,\zeta_n^0}(\underline{\theta})$ through an SVD, see (21).
4: **set** $j = 0$.
5: **repeat**
6: **calculate** least-squares costs $J_{LS}(\zeta_j, \phi_j)$ in (15).
7: **calculate** orthogonal projection-based regularization $R_1(\phi_j)$ in (29).
8: **calculate** physics-based regularization $R_2(\zeta_j)$ in (30).
9: **calculate** total cost function $J(\zeta_j, \phi_j)$ as $J(\zeta_j, \phi_j) = J_{LS}(\zeta_j, \phi_j) + \mu_1 R_1(\phi_j) + \mu_2 R_2(\zeta_j)$.
10: **update** parameters $\zeta, \phi$ according to an adaptive gradient descent update, e.g.

$$\begin{bmatrix} \zeta_{j+1} \\ \phi_{j+1} \end{bmatrix} = \begin{bmatrix} \zeta_j \\ \phi_j \end{bmatrix} - \begin{bmatrix} \nabla_\zeta J(\zeta, \phi)\big|_{\zeta=\zeta_j, \phi=\phi_j} \\ \nabla_\phi J(\zeta, \phi)\big|_{\zeta=\zeta_j, \phi=\phi_j} \end{bmatrix}.$$

11: **until** convergence, e.g., $|J(\zeta_j, \phi_j) - J(\zeta_{j+1}, \phi_{j+1})| < \epsilon$.

with physical constraints $h_i(\cdot)$ defined such that $h_i(\cdot) > 0$ when the constraint is satisfied.

$R_2(\zeta)$ as in Definition 7 embeds physical prior knowledge in (14) through regularization with the penalty function $\max(0, \exp(-h_i(\zeta)) - 1)$ [37], i.e., through increasing the costs $J(\zeta, \phi)$ if physical constraints $h_i(\zeta)$ are violated.

*Remark 3:* As opposed to this approach based on soft constraints, $h_i$ can also be implemented as hard constraints [37]. However, since this penalty function method ensures that all constraints are met for the IX application, and most automatic differentiation frameworks for neural network optimization do not allow for incorporating hard constraints out of the box, the penalty function is deemed satisfactory.

The physical constraints for the IX application are given as follows.

1) Mass and moments of inertia are positive, i.e.,

$$h_1 = m_2, \quad h_2 = m_3, \; h_3 = J_{2,yy}$$
$$h_4 = J_{3,xx}, \quad h_5 = J_{3,yy}, \; h_6 = J_{3,zz}. \tag{31}$$

2) The inertia matrix $M(\theta)$ of (2) is positive definite, i.e.,

$$h_7 = \min_i \lambda_i(M(\theta)). \tag{32}$$

3) The viscous friction coefficients are positive, i.e.,

$$h_8 = b_2, \quad h_9 = b_3. \tag{33}$$

$R_2(\zeta)$ is added to $J_{\text{LS}}(\zeta, \phi)$ as in (14) with weighting $\mu_2$ to enforce physics-based constraints.

### D. Training Procedure

With all components of $J_{\text{LS}}(\zeta, \phi)$ as in (14) defined, Algorithm 1 summarizes all aspects of the optimization procedure for $\mathcal{F}_{\zeta,\phi}$ in the form of pseudocode.

Since $J(\zeta, \phi)$ is nonconvex in both physical parameters $\zeta$ and neural network parameters $\phi$, it is necessary to resort to iterative methods to minimize $J(\zeta, \phi)$. Algorithm 1 uses a gradient descent method to minimize $J(\zeta, \phi)$. Consequently, only convergence to a local minimum (or a saddle point) can be guaranteed. The gradients of $J(\zeta, \phi)$ in step 10 can be efficiently computed by any modern automatic differentiation library, such as PyTorch [38]. Algorithm 1 can readily be modified to employ other iterative optimization schemes such as ADAM [39].

To summarize, the developed feedforward approach consists of the following steps[1]:

1) a parameterization of a physical-model-based feedforward controller (see Section III-A);
2) a parameterization of the neural network feedforward controller (see Section III-B), deciding on the number of hidden layers, the number of neurons per layer and the activation function;
3) the derivation of the basis function matrix of the physical model (see Section IV-B);
4) a choice for the regularization weightings $\mu_1$ and $\mu_2$ associated with the orthogonal projection-based regularization and physics-based regularization, defining costs $J(\zeta, \phi)$;
5) the collection of a dataset and optimization of $\mathcal{F}_{\zeta,\phi}$ according to Algorithm 1.

## V. Experimental Validation on IX Setup

In this section, the parallel feedforward controller $\mathcal{F}_{\zeta,\phi}$ in Definition 1 optimized according to cost function $J(\zeta, \phi)$ in Definition 5 is evaluated on the IX, constituting Contribution C3. The benefits of the developed framework are illustrated through comparison to a purely physical-model-based feedforward controller and to a parallel feedforward controller optimized according to just $J_{\text{LS}}(\zeta, \phi)$ in (15).

### A. Implementation Details

*1) Dataset:* A dataset for learning of $\mathcal{F}_{\zeta,\phi}$ is obtained as follows.

1) A set of References is designed such that all relevant positions and velocities within the operating range of the IX are covered. Specifically, similar to state-of-practice calibration routines, the operating range of the propeller axis is swept using a third-order setpoint [40] for various positions of the roll axis, and vice versa; the roll axis is swept for various positions of the propeller axis. This results in a dataset of length $N = 1 \cdot 10^6$. Although this type of calibration results in a large amount of data, this choice is made to remain close to current industrial practices. In case measurement time is limited, there exist methods to more efficiently cover the full operating range [41], [42].
2) Since the output $\theta$ is quantized and cannot be differentiated, the reference $\theta_d$ is used as a surrogate for $\theta$, which

[1]A numerical implementation of these steps for an academic example can be found at https://gitlab.tue.nl/kon/orthogonal-projection-based-regularization

has well-defined derivatives at the sample instances. Consequently, the dataset for optimization is given by $\theta_d$ and corresponding closed-loop control input $u$, i.e., $\mathcal{D} = \{u_2(k), u_3(k), \theta_{2,d}(k), \theta_{3,d}(k)\}_{k=1}^{N}$. By choosing this dataset and optimizing $\mathcal{F}_{\zeta,\phi}$ according to criterion $J(\zeta,\phi)$ in (14), it is implicitly assumed that the discrepancy between $\theta_d$ and $\theta$ is negligible, i.e., it is assumed that $u$ results in $\theta_d$ when applied to the IX. Even though this is never exactly true in practice, $\theta_d(k) - \theta(k) \approx 0$ for references of which the frequency content is below the feedback controller bandwidth. Consequently, for sufficiently slow references, the discrepancy between $\theta_d$ and $\theta$ is negligible.

3) Last, the feedback inputs are filtered with a second-order Butterworth low-pass filter with a 1-[Hz] cutoff frequency in both forward and reverse directions (zero phase).

*2) Hyperparameters:* Based on a hyperparameter optimization [35], the structure of both neural networks $g_{\phi_2}$ and $g_{\phi_3}$ (see Definition 3) is chosen as two hidden layers of 30 neurons each with tanh activation functions. The hyperparameter study shows that these settings provide just enough degrees of freedom to capture all dynamics while preventing overfitting. The tanh slightly outperforms the ReLU, whereas the sigmoid is observed to perform worst. Furthermore, $\mu_1$ and $\mu_2$ are set as $\mu_1 = 10^{-5}$ and $\mu_2 = 1$.

*3) Training Procedure:* The dataset is split randomly into smaller datasets: 80% for training, 10% for validation, and 10% for testing. At each iteration $k$, the training dataset is split randomly into minibatches of size $10^4$, and each training iteration $k$ loops over all these minibatches. In addition, using the validation set, training is terminated if no improvement of the validation costs is noticed during the last five training iterations [43]. Minimization is carried out using ADAM with a learning rate of $10^{-3}$ [39]. The parameters $\zeta$ of the physical model component $\mathcal{M}_\zeta$ are initialized based on minimizing $J_{\text{LS}}$ using only $f_{\mathcal{M}}$, i.e., as the best physical-model approximation.

Using the above data, hyperparameters, and training settings, $\mathcal{F}_{\zeta,\phi}$ is optimized on an HP Z-book G5 using an NVIDIA Quadro P2000 GPU in 45 iterations, taking a total time of 1112 s, corresponding to approximately 18.5 min. Fig. 7 visualizes the convergence of $J_{\text{LS}}(\zeta,\phi)$ evaluated on both the training and validation set during optimization.

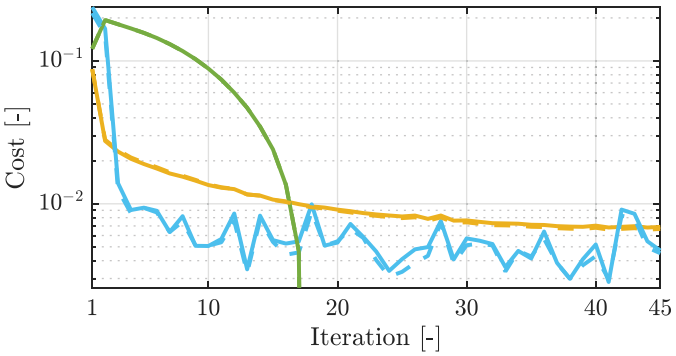


Fig. 7. Least-squares costs $(1/N)J_{\text{LS}}$ (—), orthogonal-projection-based regularization $R_1(\phi)$ (—), and physics-based regularization $R_2(\zeta)$ (—) during optimization of $\mathcal{F}_{\zeta,\phi}$ for both the training data (—) and the validation data (- -). $J_{\text{LS}}$ decreases by a factor of 10 during training, indicating that $\mathcal{F}_{\zeta,\phi}$ can capture the measured inputs $u$. Furthermore, $J_{\text{LS}}$ is approximately the same for the validation and train set, indicating that $\mathcal{F}_{\zeta,\phi}$ generalizes to data that was not trained on. Last, $R_2(\zeta) = 0$ after iteration 17, indicating that the parameters of the physical model satisfy the physical constraints $h_i(\zeta) > 0$.

## B. Performance Metrics

The performance of $\mathcal{F}_{\zeta,\phi}$ is evaluated based on two signals:

1) the difference between the measured input voltages and the voltages predicted by $\mathcal{F}_{\zeta,\phi}$ for the test dataset, i.e., based on $\epsilon_i(k) = u_i(k) - f_i(k)$, $i = 2, 3$ (see Section V-C);
2) the resulting tracking error $e_i(k) = \theta_{d,i}(k) - \theta_i(k)$ when applying $\mathcal{F}_{\zeta,\phi}$ to the IX for a new but similar reference (see Section V-D).

The performance is evaluated graphically and through relevant norms of $\epsilon_i(k)$ and $e_i(k)$. Specifically, the root mean square (rms), mean absolute (MA), and infinity (Inf) norm of $\epsilon_i(k)$ and $e_i(k)$ are considered, which are defined as

$$\text{MA}(\epsilon_i) = \frac{1}{N}\sum_{k=1}^{N} |\epsilon_i(k)| \tag{34}$$

$$\text{rms}(\epsilon_i) = \sqrt{\frac{1}{N}\sum_{k=1}^{N} \epsilon_i^2(k)} \tag{35}$$

$$\text{Inf}(\epsilon_i) = \max_{k\in\{1,\ldots,N\}} |\epsilon_i(k)| \tag{36}$$

and, similarly, for $e_i(k)$.

## C. Performance on the Test Dataset

Parameters $\zeta, \phi$ of $\mathcal{F}_{\zeta,\phi}$ are optimized according to $J(\zeta,\phi)$ in (14) with settings as in Section V-A. Fig. 8 shows the resulting signals $[f_2(k), f_3(k)]^T = \mathcal{F}_{\zeta,\phi}(\theta_d(k))$ after optimization, as well as the neural network and physical model contributions $f_i^{\mathcal{M}}(k) = \mathcal{M}_\zeta(\theta_d(k))$, $f_i^{\mathcal{C}} = \mathcal{C}_\phi(\theta_d(k))$ for $i = 2, 3$. It is observed that $f_2$ and $f_3$ closely match measured input $u$ in both axes. In addition, $f_{i,\mathcal{M}}$ and $f_{i,\mathcal{C}}$ are complimentary in both axes. In addition, the following is observed.

1) For both axes, the neural network enables predicting parts of the input signal $f_2$ that cannot be captured by $\mathcal{M}_\zeta$, e.g., the position-dependent friction and cable forces. Moreover, in the propeller axis, $\mathcal{C}_\phi$ even manages to learn the locally increased resistance from the cable getting stuck between the base and propeller axis (see Fig. 8, inset of the upper figure).
2) For both axes, $\mathcal{F}_{\zeta,\phi}$ has trouble capturing $u$ just before and right after periods of zero velocity. In these periods, IX suffers from static friction such that the difference between $\theta_d$ and $\theta$ is no longer negligible, violating the assumption made on the dataset $\mathcal{D}$. Specifically, whereas, in these periods, $\theta_d$ has nonzero velocity, the true system is not moving due to friction such that $\theta$ is constant in these periods. As a result, it is impossible to predict $u$ based on $\theta_d$ with the nonrecurrent structure of $\mathcal{C}_\phi$.

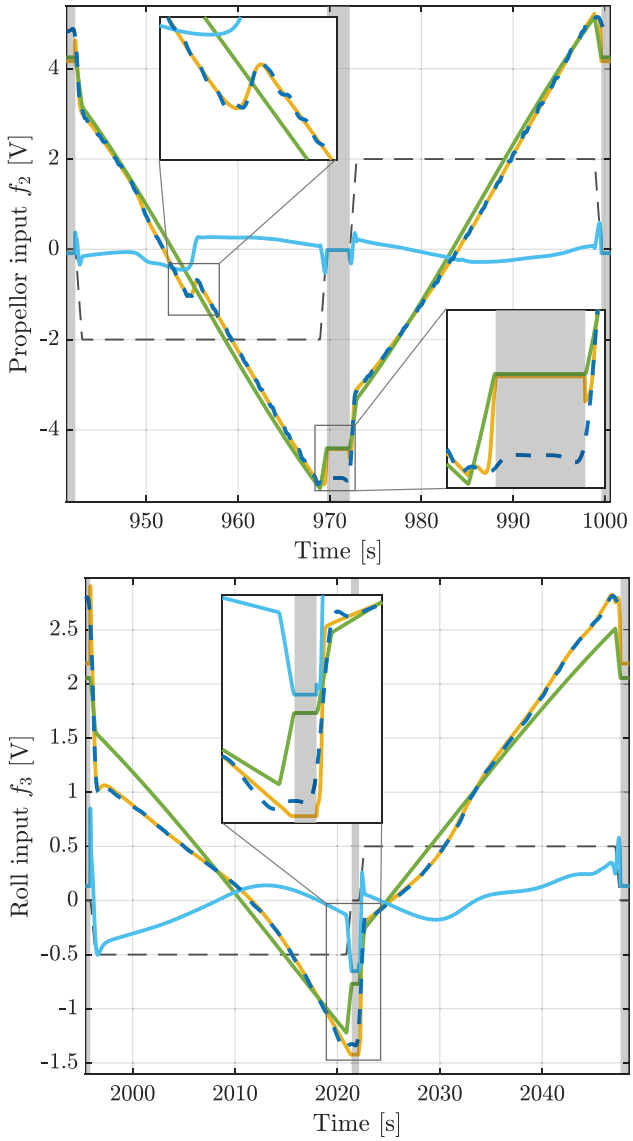


Fig. 8. Input $f_i$, $i = 2, 3$, generated by the PGNN $\mathcal{F}_{\zeta,\phi}$ (—) trained with $J(\zeta, \phi)$ is able to generate the measured input $u_i$ (- -) and can capture unmodeled dynamics such as position-dependent friction. In addition, the orthogonal projection-based regularization ensures complementarity between the physical model (—) and neural network (—) contributions $f_{i,\mathcal{M}}, f_{i,\mathcal{C}}$, in both propeller ($\theta_2$, top) and roll axis ($\theta_3$, bottom). (- -) represents the scaled velocity reference.

TABLE II
RESIDUAL $\epsilon_2 = u_2 - f_2$ NORMS [V] FOR $\theta_2$-AXIS

| | MA($\epsilon_2$) | RMS($\epsilon_2$) | Inf($\epsilon_2$) | RMS($f_{2,\mathcal{C}}$) |
|---|---|---|---|---|
| Physical model | 0.162 | 0.219 | 1.877 | 0.000 |
| PGNN with $J(\zeta, \phi)$ | 0.031 | 0.071 | 1.439 | 0.235 |
| PGNN with $J_{LS}(\zeta, \phi)$ | 0.030 | 0.064 | 1.423 | 2.953 |

Tables II and III show the norms of the prediction residuals $\epsilon_i(k) = u_i(k) - f_i(k)$ for three feedforward controllers: $\mathcal{F}_{\zeta,\phi}$ optimized according to $J(\zeta, \phi)$ in (14), $\mathcal{F}_{\zeta,\phi}$ optimized according to $J_{\text{LS}}(\zeta, \phi)$ in (15), and $\mathcal{M}_\zeta$ optimized according to $J_{\text{LS}}(\zeta)$, i.e., a purely physical-model-based feedforward controller with $\mathcal{C}_\phi = 0$. In addition, these tables show the energy in the neural network signal $f_2^{\mathcal{C}}$. It is observed that both $\mathcal{F}_{\zeta,\phi}$ are

TABLE III
RESIDUAL $\epsilon_3 = \hat{u}_3 - f_3$ NORMS [V] FOR $\theta_3$-AXIS

| | MA($\varepsilon_3$) | RMS($\epsilon_3$) | Inf($\epsilon_3$) | RMS($f_{3,\mathcal{C}}$) |
|---|---|---|---|---|
| Physical model | 0.262 | 0.321 | 1.131 | 0.000 |
| PGNN with $J(\zeta, \phi)$ | 0.016 | 0.038 | 0.842 | 0.325 |
| PGNN with $J_{LS}(\zeta, \phi)$ | 0.020 | 0.046 | 0.711 | 1.410 |

able to capture $u$ significantly better than $\mathcal{M}_{\zeta,\phi}$ due to the added flexibility of the neural network $\mathcal{C}_\phi$. The decrease in the worst case prediction error Inf($\epsilon_i$) is limited by the static friction effects. In addition, optimizing according to $J(\zeta, \phi)$ as opposed to $J_{\text{LS}}(\zeta, \phi)$, i.e., including orthogonal projection-based regularization $R_1(\phi)$ and physics-based regularization $R_2(\zeta)$ in the optimization, ensures order of magnitude smaller neural network contribution while maintaining a similar prediction performance.

These observations show that the developed PGNN can capture more dynamics of the IX than a purely physical-model-based approach through the inclusion of a neural network, and the neural network contribution remains small due to the inclusion of the orthogonal projection-based regularization. However, they also illustrate limitations, i.e., that the neural network is not able to capture all dynamics, such as stick-slip effects around zero velocity. Even though the neural network can approximate any function, the unmodeled dynamics might manifest as dynamic filters with memory, necessitating the use of recurrent neural networks.

*Remark 4:* Note that even though it is observed that more dynamics are captured by the neural network, no claims can be made on how accurately the true dynamics are approximated as these are unknown. Moreover, since neural networks do not extrapolate [9], the PGNN can only approximate the true dynamics in regions where data are provided, which is why it is only used as an augmentation of a physical model that can be used as a baseline.

### D. Real-Time Performance

In Section V-C, it is shown that given $\theta_d$, $\mathcal{F}_{\zeta,\phi}$ is able to generate $f$ that accurately matches the $u$ required for this $\theta_d$. Thus, $\mathcal{F}_{\zeta,\phi}$ is evaluated for a different but similar reference, and the generated $f$ is applied as a feedforward signal to the experimental IX setup. On the considered real-time target machine, $\mathcal{F}_{\zeta,\phi}$ takes at maximum $2.3 \cdot 10^{-6}$ s to evaluate, which is approximately 1% of the sampling time.

Fig. 9 shows the resulting tracking error for the propeller and roll axis and compares it to the tracking errors when only applying the optimized physical-model-based feedforward controller $\mathcal{M}_\zeta$ and to the tracking error when only using feedback control. Although $\mathcal{M}_\zeta$ reduces tracking errors compared to a situation with only feedback, predictable errors due to unmodeled dynamics are still present. In contrast, $\mathcal{F}_{\zeta,\phi}$ significantly improves the tracking performance by compensating for most of these unmodeled dynamics, especially for the roll axis. The following is observed.

1) The tracking errors are centered around 0, indicating successful compensation of position-dependent friction and cable-induced stiffness contributions, which result

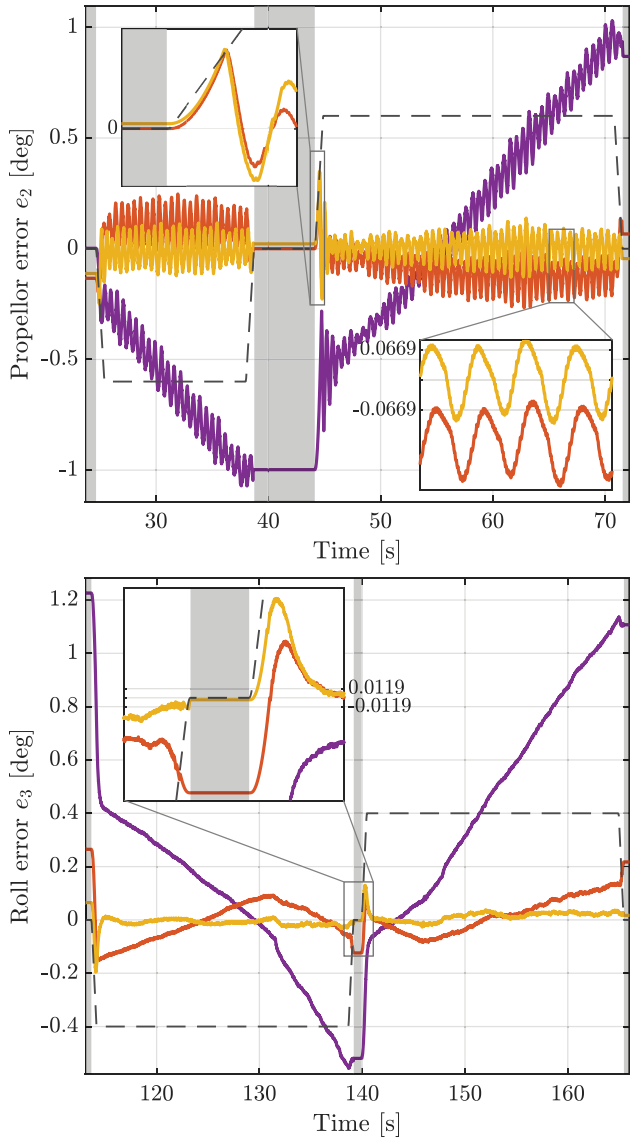


Fig. 9. PGNN feedforward (—) improves the tracking error both with respect to no feedforward (—) and physical model-based feedforward (—) in the propeller axis (top) but especially in roll axis (bottom), where unmodeled dynamics are more pronounced. (- -) represents the scaled velocity reference.

in an offset and slope in the tracking errors when using $\mathcal{M}_\zeta$ or only a feedback controller.

2) For the roll axis, the error is generally flat and within a few encoder counts. However, right after stationary periods, static friction prevents the roll axis from moving, which, combined with the constant acceleration phase of the setpoint, generates a quadratic error profile, resulting in transients. As discussed in Section V-C, these errors cannot be compensated by the neural network.
3) For the propeller axis, a clear single harmonic component is observed in the error during the constant velocity interval. Further investigation shows that both the frequency and amplitude of this harmonic vary with the velocity $\dot{\theta}_2$. It is, thus, hypothesized that a cause for this harmonic is gear imperfection in the drivetrain. Learning and compensating for these imperfections and the resulting parasitic forces requires either an extra sensor or a feedforward controller that estimates these gear positions since these gear positions are not statically but dynamically related to $\theta_2$ through the drivetrain dynamics. Consequently, $\mathcal{F}_{\zeta,\phi}$ is not rich enough to capture these disturbances.

TABLE IV
ERROR NORMS FOR $\theta_2$-AXIS

| | MA($e_2$) | RMS($e_2$) | Inf($e_2$) |
|---|---|---|---|
| Feedback only | 0.428 | 0.507 | 1.073 |
| Physical model $\mathcal{M}_\zeta$ | 0.072 | 0.089 | 0.351 |
| PGNN $\mathcal{F}_{\zeta,\phi}$ | 0.031 | 0.046 | 0.423 |

TABLE V
ERROR NORMS [DEG] FOR $\theta_3$-AXIS

| | MA($e_3$) | RMS($e_3$) | Inf($e_3$) |
|---|---|---|---|
| Feedback only | 0.402 | 0.522 | 1.401 |
| Physical model $\mathcal{M}_\zeta$ | 0.095 | 0.117 | 0.279 |
| PGNN $\mathcal{F}_{\zeta,\phi}$ | 0.020 | 0.029 | 0.269 |

Tables IV and V quantify the performance of the three methods in terms of norms of the tracking error. These tables illustrate that both the MA and rms error are improved by factors of 2 and 5 for the propeller respectively roll axis with respect to just $\mathcal{M}_\zeta$. The worst case error is not improved due to the static friction and transients right after stationary periods.

These results illustrate that the developed PGNN results in an improved tracking error compared to a purely physical-model-based feedforward controller. However, similar to Section V-C, predictable errors remain, which are not compensated by the neural network, as it is limited by its nonrecurrent structure and by the data collection process (see 2) in Section V-C).

## VI. CONCLUSION

PGNNs for feedforward control are used to compensate for the hard-to-model dynamics of an IX system. This PGNN consists of a physical multibody model of the IX in parallel with a neural network. To ensure that the neural network only compensates for unmodeled effects, orthogonal projection-based regularization is applied such that the physical model and the neural network are complementarities, resulting in a physical model that serves as a baseline and a small neural network contribution that can be monitored. The developed PGNN is able to successfully compensate the position-varying friction and position-varying cable forces present in the IX setup, as evidenced by a reduction of the tracking error by factors of 2 and 5 in respective axes compared to a physical-model-based approach.

Future work focuses on improving the feedforward accuracy of the PGNN method by developing a PGNN with a recurrent neural network. In addition, it focuses on obtaining an improved dataset that describes the required input to compensate for the stiction after stationary periods. Last, a benchmark of the developed framework compared to a broader range of feedforward control methods is considered.

## REFERENCES


[1] R. van der Maas, "Advanced geometric calibration and control for medical X-ray systems," Ph.D. dissertation, Dept. Mech. Eng., Eindhoven Univ. Technol., Eindhoven, The Netherlands, 2016.

[2] F. A. Jolesz, *Intraoperative Imaging and Image-Guided Therapy*. New York, NY, USA: Springer, 2014.

[3] N. J. Pelc, "Recent and future directions in CT imaging," *Ann. Biomed. Eng.*, vol. 42, no. 2, pp. 260–268, Feb. 2014.

[4] G. M. Clayton, S. Tien, K. K. Leang, Q. Zou, and S. Devasia, "A review of feedforward control approaches in nanopositioning for high-speed SPM," *J. Dyn. Syst., Meas., Control*, vol. 131, no. 6, Nov. 2009, Art. no. 061101.

[5] J. A. Butterworth, L. Y. Pao, and D. Y. Abramovitch, "A comparison of control architectures for atomic force microscopes," *Asian J. Control*, vol. 11, no. 2, pp. 175–181, Mar. 2009.

[6] K. Hunt, D. Sbárbaro, R. Żbikowski, and P. J. Gawthrop, "Neural networks for control systems—A survey," *Automatica*, vol. 28, no. 6, pp. 1083–1112, Nov. 1992.

[7] O. H. Sørensen, "Additive feedforward control with neural networks," in *Proc. 14th IFAC World Congr.*, vol. 32, Jul. 1999, pp. 1378–1383.

[8] G. Otten, T. J. A. de Vries, J. van Amerongen, A. M. Rankers, and E. W. Gaal, "Linear motor motion control using a learning feedforward controller," *IEEE/ASME Trans. Mechatronics*, vol. 2, no. 3, pp. 179–187, Mar. 1997.

[9] J. Schoukens and L. Ljung, "Nonlinear system identification: A user-oriented road map," *IEEE Control Syst. Mag.*, vol. 39, no. 6, pp. 28–99, Dec. 2019.

[10] L. Ljung, C. R. Andersson, K. Tiels, and T. B. Schön, "Deep learning and system identification," in *Proc. 21st IFAC World Congr.*, vol. 53, Jan. 2020, pp. 1175–1181.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[12] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[13] A. M. Schäfer and H. G. Zimmermann, "Recurrent neural networks are universal approximators," in *Proc. Int. Conf. Artif. Neural Netw.*, Sep. 2006, pp. 632–640.

[14] J. Sjöberg et al., "Nonlinear black-box modeling in system identification: A unified overview," *Automatica*, vol. 31, no. 12, pp. 1691–1724, Dec. 1995.

[15] J. Paduart, L. Lauwers, J. Swevers, K. Smolders, J. Schoukens, and R. Pintelon, "Identification of nonlinear systems using polynomial nonlinear state space models," *Automatica*, vol. 46, no. 4, pp. 647–656, Apr. 2010.

[16] B. Kosko, "Fuzzy systems as universal approximators," *IEEE Trans. Comput.*, vol. 43, no. 11, pp. 1329–1333, Nov. 1994.

[17] M. Bolderman, M. Lazar, and H. Butler, "Physics-guided neural networks for inversion-based feedforward control applied to linear motors," in *Proc. IEEE Conf. Control Technol. Appl.*, Aug. 2021, pp. 1115–1120.

[18] S. Xu and Z. Wu, "Adaptive learning control of robot manipulators via incremental hybrid neural network," *Neurocomputing*, vol. 568, Feb. 2024, Art. no. 127045.

[19] D. C. Psichogios and L. H. Ungar, "A hybrid neural network-first principles approach to process modeling," *AIChE J.*, vol. 38, no. 10, pp. 1499–1511, Oct. 1992.

[20] M. L. Thompson and M. A. Kramer, "Modeling chemical processes using prior knowledge and neural networks," *AIChE J.*, vol. 40, no. 8, pp. 1328–1340, Aug. 1994.

[21] M. Forgione and D. Piga, "Continuous-time system identification with neural networks: Model structures and fitting criteria," *Eur. J. Control*, vol. 59, pp. 69–81, May 2021.

[22] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar, "Integrating scientific knowledge with machine learning for engineering and environmental systems," *ACM Comput. Surv.*, vol. 55, no. 4, pp. 1–37, Nov. 2022.

[23] W. De Groote, E. Kikken, E. Hostens, S. Van Hoecke, and G. Crevecoeur, "Neural network augmented physics models for systems with partially unknown dynamics: Application to slider–crank mechanism," *IEEE/ASME Trans. Mechatronics*, vol. 27, no. 1, pp. 103–114, Feb. 2022.

[24] C.-H. Chou, M. Duan, and C. E. Okwudire, "A physics-guided data-driven feedforward tracking controller for systems with unmodeled dynamics—Applied to 3D printing," *IEEE Access*, vol. 11, pp. 14563–14574, 2023.

[25] A. Karpatne et al., "Theory-guided data science: A new paradigm for scientific discovery from data," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 10, pp. 2318–2331, Oct. 2017.

[26] J. Kon, D. Bruijnen, J. van de Wijdeven, M. Heertjes, and T. Oomen, "Physics-guided neural networks for feedforward control: An orthogonal projection-based approach," in *Proc. Amer. Control Conf. (ACC)*, Jun. 2022, pp. 4377–4382.

[27] Y. He, S. Chen, Y. Liu, C. Xie, and J. Geng, "A feedforward control method for turntable servo system based on physics-guided neural network," *Trans. Inst. Meas. Control*, vol. 1, no. 1, pp. 1–13, Nov. 2024. [Online]. Available: https://journals.sagepub.com/doi/abs/10.1177/01423312241291086

[28] M. Bolderman, M. Lazar, and H. Butler, "On feedforward control using physics–guided neural networks: Training cost regularization and optimized initialization," in *Proc. Eur. Control Conf. (ECC)*, Jul. 2022, pp. 1403–1408.

[29] Y. Liu, R. Tóth, and M. Schoukens, "Physics-guided state-space model augmentation using weighted regularized neural networks," in *Proc. IFAC Symp. Syst. Identificat.*, vol. 58, Jan. 2024, pp. 295–300.

[30] L. Di Natale, B. Svetozarevic, P. Heer, and C. N. Jones, "Physically consistent neural networks for building thermal modeling: Theory and analysis," *Appl. Energy*, vol. 325, Nov. 2022, Art. no. 119806.

[31] J. Kon et al., "Learning for precision motion of an interventional X-ray system: Add-on physics-guided neural network feedforward control," in *Proc. IFAC World Congr.*, vol. 56, Jan. 2023, pp. 7523–7528.

[32] A. M. Rankers, "Machine dynamics in mechatronic systems, an engineering approach," Ph.D. dissertation, Dept. Mech. Eng., Univ. Twente, Enschede, The Netherlands, 1997.

[33] X. Huo, M. Wang, K.-Z. Liu, and X. Tong, "Attenuation of position-dependent periodic disturbance for rotary machines by improved spatial repetitive control with frequency alignment," *IEEE/ASME Trans. Mechatronics*, vol. 25, no. 1, pp. 339–348, Feb. 2020.

[34] A. R. Barron, "Approximation and estimation bounds for artificial neural networks," *Mach. Learn.*, vol. 14, no. 1, pp. 115–133, 1994.

[35] N. de Vos, "Feedforward control in medical X-ray systems: Combining physical models with neural networks," Dept. Mech. Eng., Eindhoven Univ. Technol., Eindhoven, The Netherlands, Tech. Rep., 2022.

[36] L. Blanken and T. Oomen, "Kernel-based identification of non-causal systems with application to inverse model control," *Automatica*, vol. 114, Apr. 2020, Art. no. 108830.

[37] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York, NY, USA: Springer, 2006.

[38] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Conf. Neural Inf. Process. Syst.*, 2019.

[39] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, 2015, pp. 1–15.

[40] P. Lambrechts, M. Boerlage, and M. Steinbuch, "Trajectory planning and feedforward design for electromechanical motion systems," *Control Eng. Pract.*, vol. 13, no. 2, pp. 145–157, Feb. 2005.

[41] V. Smits and O. Nelles, "Space-filling optimized excitation signals for nonlinear system identification of dynamic processes of a diesel engine," *Control Eng. Pract.*, vol. 144, Mar. 2024, Art. no. 105821.

[42] M. Kiss, R. Tóth, and M. Schoukens, "Space-filling input design for nonlinear state-space identification," in *Proc. IFAC Symp. Syst. Identificat.*, May 2024, pp. 562–567.

[43] Y. Yao, L. Rosasco, and A. Caponnetto, "On early stopping in gradient descent learning," *Constructive Approximation*, vol. 26, no. 2, pp. 289–315, Aug. 2007.


**Johan Kon** (Graduate Student Member, IEEE) received the B.Sc. degree (cum laude) in mechanical engineering and the M.Sc. degree (cum laude) in systems and control from Eindhoven University of Technology, Eindhoven, The Netherlands, in 2018 and 2021, respectively, where he is currently pursuing the Ph.D. degree with the Control Systems Technology Group, Department of Mechanical Engineering.

His research interests include machine learning for system identification and control applied to precision mechatronics.

**Naomi de Vos** received the B.Sc. degree in mechanical engineering and the M.Sc. degree in systems and control from Eindhoven University of Technology, Eindhoven, The Netherlands, in 2020 and 2022, respectively

She is currently a Systems Engineer at Philips, Eindhoven, where she focuses on product development.

**Dennis Bruijnen** received the M.Sc. and Ph.D. degrees in mechanical engineering from Eindhoven University of Technology, Eindhoven, The Netherlands, in 2003 and 2007, respectively.

In 2008, he joined Philips, Eindhoven, as a Control Technologist within Philips Engineering Solutions. His main areas of research interest are advanced control, robotics, high-precision systems, algorithms, and optimization.

**Jeroen van de Wijdeven** received the M.Sc. and Ph.D. degrees in mechanical engineering from Eindhoven University of Technology, Eindhoven, The Netherlands, in 2004 and 2008, respectively.

Since 2008, he has been a Mechatronics Design Engineer at ASML, Veldhoven, The Netherlands. His research interests include motion control and dynamics for high-precision mechatronic systems.

**Marcel Heertjes** received the M.Sc. and Ph.D. degrees from Eindhoven University of Technology, Eindhoven, The Netherlands, in 1995 and 1999, respectively.

After being with Philips Center for Industrial Technology, Eindhoven, from 2000 to 2005, he joined ASML, Veldhoven, The Netherlands, in 2006. In 2019, he was appointed (part-time) as a Full Professor of Industrial Nonlinear Control for High-Precision Systems.

Dr. Heertjes was a recipient of IEEE Control Systems Technology Award in 2015 for Variable Gain Control and Its Applications to Wafer Scanners. He has been serving as an Associate Editor for *IFAC Mechatronics* since 2016.

**Tom Oomen** (Senior Member, IEEE) received the M.Sc. degree (cum laude) and the Ph.D. degree from Eindhoven University of Technology, Eindhoven, The Netherlands.

He is a Full Professor with the Department of Mechanical Engineering, Eindhoven University of Technology. He is also a part-time Full Professor with Delft University of Technology, Delft, The Netherlands. He was a Visiting Scholar at KTH Royal Institute of Technology, Stockholm, Sweden, and the University of Newcastle, Callaghan, NSW, Australia. His research interests are in the field of data-driven modeling, learning, and control, with applications in precision mechatronics.

Dr. Oomen was a member of Eindhoven Young Academy of Engineering. He was a recipient of the 7th Grand Nagamori Award, the Corus Young Talent Graduation Award, the IFAC 2019 TC 4.2 Mechatronics Young Research Award, the 2015 IEEE Transactions on Control Systems Technology Outstanding Paper Award, the 2017 IFAC Mechatronics Best Paper Award, the 2019 IEEJ Journal of Industry Applications Best Paper Award, and a Veni and Vidi Personal Grant. He is currently a Senior Editor of IEEE CONTROL SYSTEMS LETTERS (L-CSS) and the Co-Editor-in-Chief of *IFAC Mechatronics*. He has served on the Editorial Board of IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY. He has also been the Vice-Chair for IFAC TC 4.2.