Neural Network Augmented Physics Models for Systems with Partially Unknown Dynamics: Application to Slider-Crank Mechanism

Wannes De Groote, Edward Kikken, Erik Hostens, Sofie Van Hoecke, Guillaume Crevecoeur

Abstract—Dynamic models of mechatronic systems are abundantly used in the context of motion control and design of complex servo applications. In practice, these systems are often plagued by unknown interactions, which make the physics-based relations of the system dynamics only partially known. This paper presents a neural network augmented physics (NNAP) model as a combination of physics-inspired and neural layers. The neural layers are inserted in the model to compensate for the unmodeled interactions, without requiring direct measurements of these unknown phenomena. In contrast to traditional approaches, both the neural network and physical parameters are simultaneously optimized, solely by using state and control input measurements. The methodology is applied on experimental data of a slider-crank setup for which the state dependent load interactions are unknown. The NNAP model proves to be a stable and accurate modeling formalism for dynamic systems that ab initio can only be partially described by physical laws. Moreover, the results show that a recurrent implementation of the NNAP model enables improved robustness and accuracy of the system state predictions, compared to its feedforward counterpart. Besides capturing the system dynamics, the NNAP model provides a means to gain new insights by extracting the neural network from the converged NNAP model. In this way, we discovered accurate representations of the unknown spring force interaction and friction phenomena acting on the slider mechanism.

Index Terms—Nonlinear dynamic system modeling, neural networks, explainable artificial intelligence, load identification

I. INTRODUCTION

SERVO-controlled systems face increasingly demanding performance and efficiency requirements in industrial and manufacturing applications. In, for example, presses, pumps and compressors, linear motion must be realized in direct drive or indirectly. The latter servo systems need to drive a load system via a rotary motor system and mechanical transmissions

such as gears, cam-follower, and bar linkages. Position and speed control are critical, and their performance is limited due to unknown dynamics resulting from the external loads and uncertainties in the mechanical system such as inertia, backlash, damping, and friction [1]–[3]. To better understand these intricate and often highly nonlinear dynamics, extensive research has been conducted on developing predictive models that capture the overall behavior of servo systems. These models can then be used by a user or manufacturer to further improve the mechatronic system design [4], [5]. Furthermore, predictive models can be inserted in model-based control strategies to improve the mechatronic system's operations, (e.g. [6]).

Capturing the servo drive system dynamics is, however, often cumbersome and challenging because we face against limits of various existing modeling and system identification formalisms. The modeling of a mechatronic system can be based on expert knowledge where partial and/or ordinary differential equations are distilled from the system. These physics-based models include (lumped) physical parameters that relate directly to the actual mechatronic system's behavior. Parameter identification techniques can then be used to identify the values of these parameters based on experimental sensor data [7]. Uncertainties can, however, still remain because servo drive systems face nonlinearities such as friction forces that give rise to disturbances that are difficult to model. Alternatively, black-box models can be built in a data-driven manner. In the context of nonlinear system identification, several methodologies exist such as Gaussian processes [8], Hammerstein-Wiener structures [9], and nonlinear auto-regressive models (e.g. NARMAX) [10]. Supervised learning is typically used to build the model by relating input to output data. Supervised machine learning techniques such as deep learning have become pervasive because they have the ability to recognize patterns in data [11]. Neural networks have, for instance, been used to approximate various nonlinear relations, such as inverse kinematics within complex mechatronic systems [12]. Recurrent neural networks (RNN) have the ability to capture the nonlinear dynamics in time series by means of an internal state and are of particular interest for learning the dynamic behavior of mechatronic applications [13]. Ensembles of interconnected RNNs have demonstrated their usefulness in the modeling of complex dynamical systems [14]. However, as observed in [15], there is no physical interpretation given to this internal state, which makes the convergence rely substantially on hyperparameter

tuning and initialization choices.

Having physical interpretations to the modeling can, however, be useful, particularly to assure robust and physics consistent predictions. Physically interpretable neural-fuzzy networks have shown in this respect enhanced explainability by assigning piecewise linear models to the network nodes [16]. The dynamic behavior is typically defined by the so-called derivative function in a partial and/or ordinary differential equations-based model. Based on physical insights, it is possible to deduce an appropriate structure of a black-box model. Feedforward neural networks have been proposed in physics-informed deep learning [17] to approximate derivative functions of dynamical systems that, in contrast to traditional black-box approximations, return more interpretable models. Other recent approaches based on symbolic regression [18] and sparse regression [19] were able to return more interpretable models by discovering the underlying governing equations. Furthermore, recent developments indicate the possibilities to approximate reformulations of classical mechanics, such as Lagrangian [20] and Hamiltonian [21] mechanics by neural networks.

Next to using physical insights to deduce the structure of black-box models, one can directly use physics-based governing equations within the data-driven model. Physical laws can be included in the loss function of black-box neural networks to guide the training process towards physical consistent model predictions [22]. Conversely, neural network mappings can assist in compensating for prediction discrepancies of a complete physics-based model [15], [23]. These neural networks serve as nonlinear mappings that transform the state estimations behavior of the physics-based model. They enable predictions to be improved; however, they do not represent an interpretable physical phenomenon. An incomplete physics model was considered in [24] that was complemented with neural networks. The latter was engaged to accommodate for partially unknown dynamics within the physical system. Neural networks were trained separately and in a supervised manner on the basis of measurement data that directly relate to the unknown phenomena.

This paper proposes a neural network augmented physics (NNAP) model, encompassing trainable physical layers closely connected to a neural network to compensate for partially unknown dynamics. It is assumed that no direct measurement data on the unknown phenomena are available because they are either difficult or not possible at all to measure. Examples are unknown friction phenomena in servo mechanisms that affect the overall system behavior. The proposed NNAP models consist of an interconnected network of physics layers complemented with black-box layers. Measurement data are propagated through these layers to simultaneously identify the physics and data-driven parameters, as depicted in Fig. 1. The effectiveness of the methodology is extensively tested on experimental data collected on a servo slider-crank setup with unknown nonlinear load and unidentified key physical parameters. The objective of the presented methodology is twofold. First, to obtain a model with improved predictive capabilities by performing a joint optimization of physical and neural parameters while only using state and control input

data. Secondly, to explain the partially unknown dynamics within the system by analyzing the information captured by the black box layers after training the NNAP model. By using such an approach, new physical insights can be obtained to explain and interpret the unknown phenomena that are inaccessible by direct measurements or are difficult to model *ab initio*.
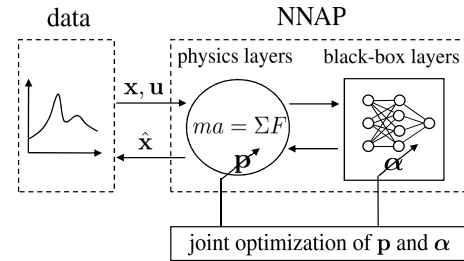

Fig. 1: NNAP model as an interconnected network of physics and black-box layers, which are defined by trainable physics parameters $\mathbf{p}$ and neural parameters $\boldsymbol{\alpha}$, respectively. All layers are simultaneously optimized solely by using control inputs $\mathbf{u}$ and state measurement $\mathbf{x}$.

II. SLIDER-CRANK SYSTEM

Various industrial applications require reciprocating linear motions. Motion control of these systems can be provided by means of linear electric motors [25]. For high-power applications, a combination of rotary motors in combination with mechanical transmissions [26] such as gears, cam-follower and bar linkage systems are the preferred driveline because they achieve energy efficient and robust operation. These drivelines exhibit highly nonlinear behavior that is often challenging to capture using first-principle physics modeling. They are often plagued by unidentified load disturbances and unknown interactions of the system with the environment that are beyond expert knowledge. These nonlinear phenomena are, henceforth, formalized by relation $\mathcal{P}$ that relates certain inputs of the system (e.g. velocity) to a response of the system (e.g. friction force). Next to these nonlinear unknown phenomena, physical parameters such as the inertia in the servo drive system can be uncertain and are formalized as $\mathbf{p}$.

This paper applies and aims at validating a methodology to accommodate for unknown $\mathcal{P}$ and $\mathbf{p}$ that can arise in servo-controlled systems. We consider a slider-crank mechanism translating a rotary motion into a linear displacement that is subject to an unidentified load. This application is of direct relevance in various industrial applications such as compressors [27], hydraulic pumps [28], weaving looms [29], and presses [30]. A system model comprehending the dynamics of the slider-crank system is built, and experimental data are collected from a practical setup.
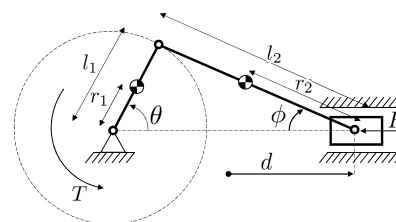

Fig. 2: Schematic of slider-crank system.

A. System Model

The slider-crank system considered in this paper is schematically depicted in Fig. 2. The setup consists of a rotary servo motor delivering torque T to a first mechanical link that rotates with angle θ, which, in turn, is connected to a second link. The latter link is connected to a slider (with relative angle ϕ between this link and slider). The dynamics of the considered slider-crank setup are governed by force interactions between the rigid mechanical links. Appendix A details the expressions that formalize the dynamics of the multibody system. A time invariant state-space model can be distilled that captures the dynamics of the state $\mathbf{x} = \begin{bmatrix} \theta & \omega \end{bmatrix}^T$. The angular speed is denoted as $\omega \equiv \dot{\theta}$. Note that the slider position d is geometrically coupled to motor angle θ and angle ϕ that, in turn, is related to θ via $\phi = \arcsin\left(\frac{l_1}{l_2}\sin(\theta)\right)$. The nonlinear state space relation can be comprehended by a derivative function that relates the state $\mathbf{x}$ and control input $u = T$ to the next state. We, furthermore, parameterize the state space model with physical parameters $\mathbf{p}$, being masses, lengths, inertia, and rotational damping coefficient, which appear in (11). Finally, the load force F depicted in Fig. 1 is the unknown nonlinear relation $\mathcal{P}$ that is an additional disturbance input of the model. Given the above, we formalize the dynamics in the slider-crank, thus, as $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, u, F; \mathbf{p})$ with a geometrical relation between the linear displacement and velocity

$$\boldsymbol{\gamma}(\mathbf{x}; \mathbf{p}) = \begin{bmatrix} d \\ v \end{bmatrix} = \begin{bmatrix} l_1 \cos(\theta) + l_2 \cos(\phi) - (l_2 - l_1) \\ -l_1 \sin(\theta)\omega - l_2 \sin(\phi)\dot{\phi} \end{bmatrix} \quad (1)$$

that is again dependent on the physical parameters $\mathbf{p}$.

B. Practical Setup

The practical slider-crank setup is depicted in Fig. 3. A 3 kW brushless servo motor with integrated drive is connected to a rigid crank having length $l_1 = 0.05$ m. The slider mechanism is connected to the crank via a rigid connecting rod with approximate length $l_2 = 0.29$ m. The rotary motion of the motor is measured by an incremental encoder (8192 CPR). A compression spring is added to invoke additional nonlinear disturbances in the external force F acting on the slider. The spring force characteristic as a function of displacement d is given in Fig. 4. This characteristic was empirically determined for validation purposes later on.
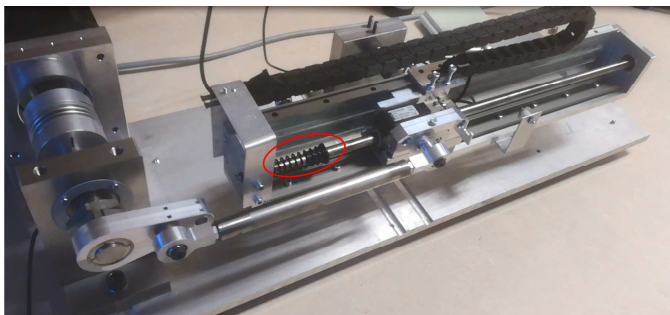

Fig. 3: Slider-crank setup with spring load indicated by red circle.

Experimental data are collected from the slider-crank setup for different torque signal inputs. These torque profiles, see Fig. 5, are chosen *ad-hoc* to have diverse excitations of the system. These torque inputs are each time applied starting from two distinct motor positions, leading to a total of $S = 20$ different trajectories of rotational movements θ and ω. The measurements are sampled at 2000 Hz, resulting in $L = 800$ samples per trajectory.
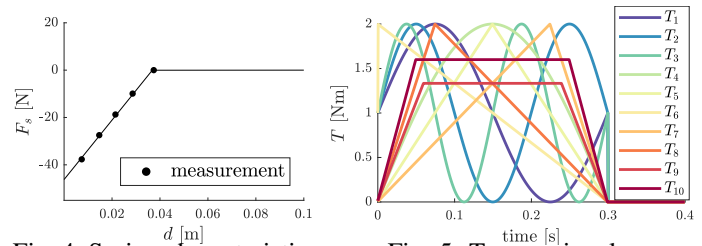
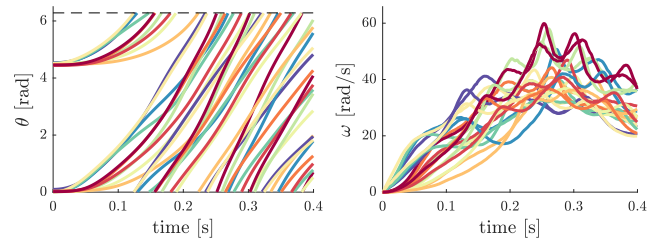

Fig. 4: Spring characteristic.

Fig. 5: Torque signals.

Fig. 6: Collected data from the slider-crank setup.

III. NEURAL NETWORK AUGMENTED PHYSICS MODEL

Having accurate predictive models of systems, such as the slider-crank setup presented in Section II, can advance the design and motion control of various manufacturing and industrial systems. To accommodate for partially unknown dynamics (unknown nonlinear disturbance $\mathcal{P}$ and uncertain physical parameters $\mathbf{p}$) arising in servo systems, the physics-based ordinary differential equations (ODE) are complemented with additional data-driven models. The physics-based derivative function $\mathbf{f}$ defines the behavior of the system state $\mathbf{x}$ for given control input $\mathbf{u}$. The known physical laws are encapsulated within $\mathbf{f}$ and characterized by physical parameters $\mathbf{p}$. The lack of knowledge about the unknown phenomena $\mathcal{P}$ is compensated for by introducing an additional input $\mathbf{z}$.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{z}; \mathbf{p}) \quad (2)$$

The NNAP model here includes an artificial neural network $\boldsymbol{\eta}$ with the weights and biases of the neural nodes as learnable parameters $\boldsymbol{\alpha}$ to compensate for these unknown phenomena $\mathcal{P}$. The state measurements $\mathbf{x}$ and control input $\mathbf{u}$ are used to simultaneously update the neural network variables $\boldsymbol{\alpha}$ and physical parameters $\mathbf{p}$ to match the actual physical dynamical system behavior. Note that here, contrary to [24], the data-driven model is not learned on the basis of (additional) measurement data that relate directly to the unknown phenomena $\mathcal{P}$. A feedforward and a recurrent physics-based neural network model formalism are presented in the two next subsections, respectively. Subsequently, the optimization process is explained, followed by the implementation details on the slider-crank mechanism.

A. Feedforward NNAP Model

A feedforward model predicts the next state $\mathbf{x}_{k+1}$ for given control input $\mathbf{u}_k$ and current state $\mathbf{x}_k$. By repeatedly feeding the predicted state back as input during the subsequent time instance, the model becomes suitable for multistep predictions. The universal approximation theorem states that feedforward
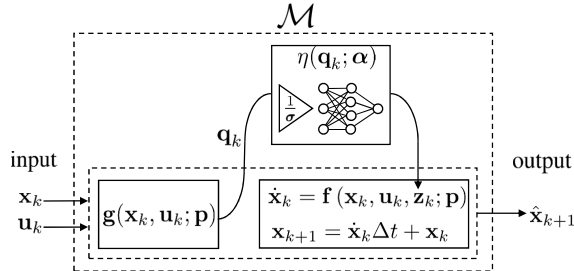

Fig. 7: Feedforward NNAP modeling formalism.

neural networks with one hidden layer can approximate any continuous function for inputs within a specific range [31]. More specifically, we define $\boldsymbol{\eta}$, a one-hidden-layer rectified linear unit (ReLU) network of n_h hidden units predicting a n_o dimensional output. A ReLU model is chosen due to its demonstrated usefulness in various regression tasks [11], [32]. The n_i dimensional input $\mathbf{q} \in \mathbb{R}^{n_i}$ to the neural network is defined by a physics-inspired network layer $\mathbf{g}$.

$$\mathbf{q} = \mathbf{g}(\mathbf{x}, \mathbf{u}; \mathbf{p}) \qquad (3)$$

The input vector $\mathbf{q}$ is scaled by the standard deviation $(\sigma_j, \ j = 1, \cdots, n_i)$ of the corresponding input elements to eliminate magnitude differences and consequently avoid dominance of particular dimensions. The corresponding transformation matrix $\Sigma = \text{diag}\left(\sigma_1^{-1}, \ldots, \sigma_{n_i}^{-1}\right)$ is determined in advance directly from the measurement data. Subsequently, the input undergoes a linear transformation by the weights $W_h \in \mathbb{R}^{n_i \times n_h}$ of the hidden layer. The hidden unit activation function Υ will, after adding the bias vector $\mathbf{b}_h \in \mathbb{R}^{n_h}$, include the required nonlinearity in the function $\boldsymbol{\eta}$. Subsequently, the output is obtained via a linear output layer, defined by the weights $W_o \in \mathbb{R}^{n_h \times n_o}$ and $\mathbf{b}_o \in \mathbb{R}^{n_o}$. We denote the ReLU model as being

$$\begin{aligned} \boldsymbol{\eta}(\mathbf{q}; \boldsymbol{\alpha}) &= W_o^T \Upsilon \left(W_h^T \Sigma \mathbf{q} + \mathbf{b}_h \right) + \mathbf{b}_o \\ \Upsilon(\cdot) &= \max\left(0, \cdot\right) \end{aligned} \qquad (4)$$

The architecture of the feedforward NNAP models is given in Fig. 7. The output of the ReLU network $\boldsymbol{\eta}$, initialized by random weights and biases included in $\boldsymbol{\alpha}$, replaces the variable $\mathbf{z}$ in the ODE layer (2). Consequently, the assumption is that $\mathbf{q}$ contains sufficient input information to estimate $\mathbf{z}$, by approximating $\mathcal{P}$ via a neural network $\boldsymbol{\eta}$. We distill the following derivative function at each time instant k.

$$\dot{\mathbf{x}}_k = \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\eta}(\mathbf{q}_k; \boldsymbol{\alpha}); \mathbf{p}\right) \qquad (5)$$

The derivative function now has a hybrid nature and is further used to propagate to the next state via Euler's method by choosing Δt equal to the fixed sampling time of the measurement data. The combination of interconnected layers adds up

to an overall feedforward network function $\mathcal{M}\left(\mathbf{x}_k, \mathbf{u}_k; \mathbf{p}, \boldsymbol{\alpha}\right)$ that estimates the state $\mathbf{x}_{k+1}$ given the prior state $\mathbf{x}_k$ and input $\mathbf{u}_k$.

$$\hat{\mathbf{x}}_{k+1} = \mathcal{M}\left(\mathbf{x}_k, \mathbf{u}_k; \mathbf{p}, \boldsymbol{\alpha}\right) \qquad (6)$$

The feedforward NNAP model $\mathcal{M}$ predicts only the state of the subsequent timestep. The model $\mathcal{M}$ encompasses a static relation that can describe dynamical behavior once prior state predictions are fed back as model input.

B. Recurrent NNAP model

Recurrent neural networks (RNN) can cope with timeseries due to their ingrained dynamic nature [11], [33]. These models are penalized for estimation errors over a larger time horizon, aiming for improved multistep prediction capabilities. Fig. 8 illustrates a schematic overview of a basic RNN. The operator $\mathcal{D}$ induces a one-step time delay of the internal state $\mathbf{h}$. If we unfold the RNN in time, we obtain a computational graph that contains shared weights $\boldsymbol{\psi}$ at each time instance j. The total prediction error is an accumulation of the error at each timestep. The gradients of this virtual multilayer network are required to update $\boldsymbol{\psi}$ and can be derived via backpropagation through time (BPTT), which calculates the gradients based on the chain rule of differentiation [33]. A RNN model accepts an
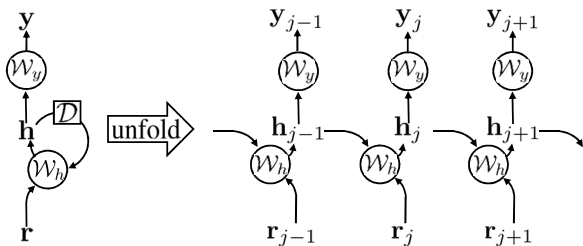

Fig. 8: Recurrent neural network.

input sequence of N steps by processing each input element $\mathbf{r}_j$ for $j \in \{1, \ldots, N\}$ at a time. These models contain information about prior inputs by updating an internal state vector $\mathbf{h}_j$ via a (nonlinear) function $\mathcal{W}_h$. Thereafter, an output relation $\mathcal{W}_y$ maps the internal state to the model output $\mathbf{y}$. The functions $\mathcal{W}_h$ and $\mathcal{W}_y$ typically contain network parameters $\boldsymbol{\psi}$ that are optimized via gradient-based algorithms.

$$\begin{aligned} \mathbf{h}_j &= \mathcal{W}_h(\mathbf{r}_j, \mathbf{h}_{j-1}; \boldsymbol{\psi}) \\ \mathbf{y}_j &= \mathcal{W}_y(\mathbf{h}_j; \boldsymbol{\psi}) \end{aligned} \qquad (7)$$

A recurrent NNAP model $\mathcal{R}$ is constructed to predict a state trajectory $\{\mathbf{x}_{k+1}, \ldots, \mathbf{x}_{k+N}\}$ for given input sequence $\{\mathbf{u}_k, \ldots, \mathbf{u}_{k+N-1}\}$, starting from state $\mathbf{x}_k$. The dimension of the internal state vector $\mathbf{h}$ is typically a hyperparameter that needs to be tuned depending on the complexity of the problem. Here, we assume that the dimension of the state of $\mathbf{h}$ equals the dimension of the state vector $\mathbf{x}$. The corresponding initial value $\mathbf{h}_0$, typically initialized as zero, is chosen to equal $\mathbf{x}_k$. This allows to use the feedforward NNAP function $\mathcal{M}$, detailed in Fig. 7, as an update function $\mathcal{W}_h$ of the internal state $\mathbf{h}$ in (7). This forces the hidden state $\mathbf{h}$ to mimic the behavior of the system state $\mathbf{x}$. Consequently, the output

function $\mathcal{W}_y$ in (7) includes an identity transformation because we can consider $\mathbf{y}_j = \mathbf{h}_j = \mathbf{x}_{k+j}$ for $j \in \{1, \ldots, N\}$. Fig. 9 illustrates the architecture of $\mathcal{R}$. The identification of the recurrent NNAP model $\mathcal{R}$ parameters ψ comes down to optimizing the variables $\mathbf{p}$ and $\boldsymbol{\alpha}$ included in the feedforward NNAP model $\mathcal{M}$ in (6).

$$[\hat{\mathbf{x}}_{k+1}, \ldots, \hat{\mathbf{x}}_{k+N}] = \mathcal{R}(\mathbf{x}_k, \mathbf{u}_k, \ldots, \mathbf{u}_{k+N-1}; \mathbf{p}, \boldsymbol{\alpha}) \quad (8)$$
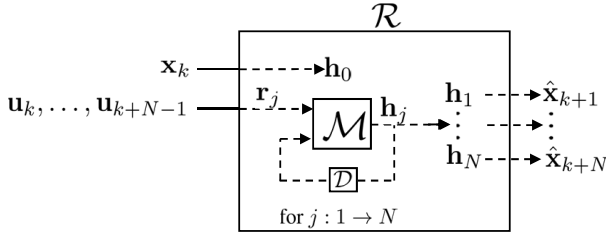

Fig. 9: Recurrent NNAP modeling formalism.

C. Optimization

The identification of the NNAP model requires the simultaneous optimization of the neural ($\boldsymbol{\alpha}$) and physical ($\mathbf{p}$) parameters by minimizing the cost function $\mathcal{L}$.

$$\mathcal{L}(\boldsymbol{\alpha}, \mathbf{p}) = \frac{1}{S} \frac{1}{L} \frac{1}{N} \sum_{s=1}^{S} \sum_{i=0}^{L-N} \sum_{k=i+1}^{i+N} (\mathbf{x}_k^s - \hat{\mathbf{x}}_k^s)^T C (\mathbf{x}_k^s - \hat{\mathbf{x}}_k^s) \quad (9)$$

The function $\mathcal{L}$ penalizes the discrepancies between the measured ($\mathbf{x}$) and predicted ($\hat{\mathbf{x}}$) states, scaled by a diagonal matrix C to accommodate for magnitude differences between the state variables. The loss function $\mathcal{L}$ incorporates S trajectory signals, for which each signal contains L samples of the state $\mathbf{x}$ and control input $\mathbf{u}$. Figure 10a illustrates the evaluation
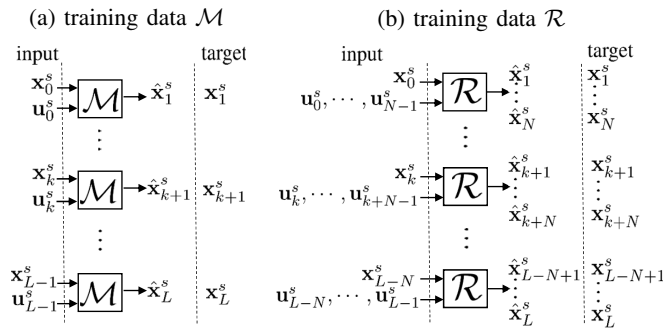

Fig. 10: Schematic overview of training data specifications for trajectory $s \in \{1, \ldots, S\}$ existing of L training samples.

of each trajectory sample by $\mathcal{M}$, to match the subsequent state value (we have here $N = 1$). The recurrent model $\mathcal{R}$ is constructed on the basis of an initial state value and control sequence of N steps, as depicted in Fig. 10b. The derivation of the analytical gradients $\nabla_p \mathcal{L}$ and $\nabla_\alpha \mathcal{L}$ imposes that, apart from the neural network $\boldsymbol{\eta}$, the physics layers also need to be differentiable. The gradient information is used to simultaneously update $\boldsymbol{\alpha}$ and $\mathbf{p}$ based on a gradient descent approach with step size δ.

$$\begin{aligned} \mathbf{p}_{i+1} &= \mathbf{p}_i - \delta_i \nabla_p \mathcal{L} \\ \boldsymbol{\alpha}_{i+1} &= \boldsymbol{\alpha}_i - \delta_i \nabla_\alpha \mathcal{L} \end{aligned} \quad (10)$$

Note that the difference between the feedforward model $\mathcal{M}$ and recurrent architecture $\mathcal{R}$ lies in the different training process as the gradients of the latter are derived by BPTT to allow robust predictions of longer time series. Nevertheless, once the recurrent model $\mathcal{R}$ converges, it can be reworked to a more simple feedforward structure $\mathcal{M}$ because both architectures ingrain the same parameters $\mathbf{p}$ and $\boldsymbol{\alpha}$.

D. Implementation

The presented NNAP modeling formalism is applied on the slider-crank mechanism detailed in Section II. The general workflow is illustrated in Fig. 11. First, the known dynamics of the system are distilled and described by their physical laws. As mentioned before, the dynamics of the slider-crank mechanism are known, except for the state-dependent load interactions $\mathcal{P}$ acting on the slider. The incomplete physics-based relations are combined into an ODE $\mathbf{f}$ that accepts an additional input $z = F$, as explained in Appendix A. Subsequently, the uncertain physical parameters $\mathbf{p}$ are defined. In the slider-crank case, we consider the parameter set $\mathbf{p} = \{J_1, B_m, m_3\}$ as unknown. Other possibilities are discussed in Section IV-D.

Next, the unknown load interactions $\mathcal{P}$ are replaced by a randomly initialized ReLU network η. This predictor of the load force F contains one hidden layer of $n_h = 32$ hidden units, determined via hyperparameter tuning. The network input $\mathbf{q}$ needs to provide sufficient information to the network η. Therefore, expert knowledge is used to define $\mathbf{q} = \begin{bmatrix} d & v \end{bmatrix}^T$ by choosing the input function $\mathbf{g}$ equal to the geometrical relations $\boldsymbol{\gamma}$ in (1). Other options for $\mathbf{g}$ and their effect are discussed in Section IV-A. The relations defined in the functions $\mathbf{g}(., \mathbf{p})$, $\eta(., \boldsymbol{\alpha})$ and $\mathbf{f}(., \mathbf{p})$ are implemented as successive custom network layers, that result into the overall NNAP model $\mathcal{M}(., \mathbf{p}, \boldsymbol{\alpha})$, as shown in Fig. 7. Moreover, this feedforward model can be implemented as a recurrent NNAP model $\mathcal{R}(., \mathbf{p}, \boldsymbol{\alpha})$ by adding the feedback loops shown in Fig. 9. In practice, these network architectures are implemented by using the Keras API [34] with TensorFlow [35] backend in Python. The physical laws are implemented as custom (differentiable) network layers for which the parameters $\mathbf{p}$ are defined as trainable variables. This library employs automatic
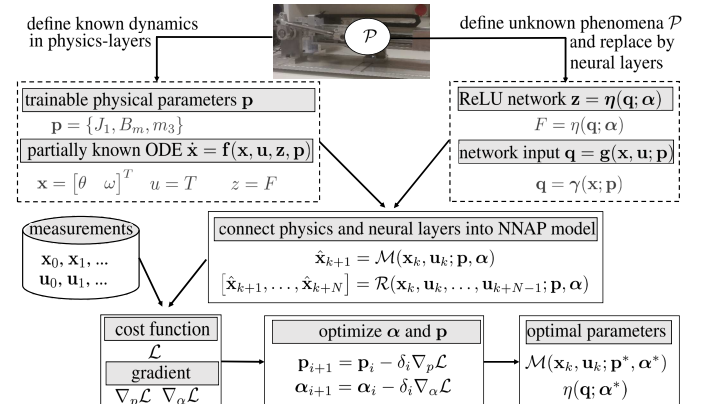

Fig. 11: Practical implementation of the NNAP model.

differentiation, which provides the analytical expressions for the gradients $\nabla_p \mathcal{L}$ and $\nabla_\alpha \mathcal{L}$. The availability of analytical gradient information enables a joint identification of both $\mathbf{p}$ and $\boldsymbol{\alpha}$ by using a gradient based Adam optimizer that processes the data in mini-batches of 200 samples [36].

IV. RESULTS AND DISCUSSION

The objective of the joint optimization of $\mathbf{p}$ and $\boldsymbol{\alpha}$ is twofold. First, the prediction performances of the optimized NNAP model are assessed by analyzing the root mean squared error (RMSE) of the trajectory prediction of the key variable ω. The presented results are validated via leave-one-out-cross-validation (LOOCV). This implies that the model is trained by $S = 19$ signals and validated by the trajectory excluded from the training set. This is repeated for all 20 signals in order to develop a fair performance benchmark. Unless otherwise stated, we will perform the experiments on the feedforward architecture $\mathcal{M}$ because inherently, both architectures $\mathcal{M}$ and $\mathcal{R}$ include the same parameters to be identified. Secondly, the convergence and identifiability of the physical parameters $\mathbf{p}$ and the neural network η will be analyzed to validate if new physical insights can be gained about the physical parameters and the unknown load interactions captured by η.

A. Influence of the Neural Network Inputs

The prediction performance of the NNAP model $\mathcal{M}$ highly relies on the information that can be captured by the neural network η during the joint optimization of the parameters in $\boldsymbol{\alpha}$ and $\mathbf{p}$. The ability of η to compensate for the unmodeled physical relations in $\mathbf{f}$ depends on the input $\mathbf{q}$, defined by the function $\mathbf{g}$. For this case, expert knowledge is used to define $\mathbf{g}$ as the geometrical relations $\boldsymbol{\gamma}$ in (1), to provide $\mathbf{q} = \begin{bmatrix} d & v \end{bmatrix}^T$ as input to η. However, in practice, the dependencies of the unknown phenomena $\mathcal{P}$ are not known. Therefore, the required information to the neural network can be determined by analyzing the obtained prediction accuracy of $\mathcal{M}$ for varying function candidates of $\mathbf{g}$. Variables such as $\{\theta, \omega\}$ and $\{T\}$ can be obtained by choosing $\mathbf{g}$ equal to an identity transformation because they are directly included in the state $\mathbf{x}$ or control input u, respectively. Figure 12b illustrates the time evolution of the angular velocity ω on the basis of the discrete model $\mathcal{M}$ for varying inputs $\mathbf{q}$. We perform a multistep prediction as the state $\hat{\mathbf{x}}_{k+1}$ in (6) is the subsequent state input $\mathbf{x}_k$ to $\mathcal{M}$ in the next time instant. Errors are propagated each time when $\mathcal{M}$ is evaluated, explaining the diverging trajectories with respect to the reference signal. The lowest average RMSE is obtained for $\mathbf{q} = \begin{bmatrix} d & v \end{bmatrix}^T$. This insight makes sense knowing that the spring load and possible additional friction phenomena, both contained in the overall force F, are not included in the physics equations and, thus, should be approximated by the network η. The set of experiments for which the required input information (i.e. set $\{d, v\}$ or the equivalently $\{\theta, \omega\}$) is not fed to the neural network η clearly lead to inferior trajectory prediction accuracies. This experiment revealed the importance of the input information to the neural network η, at the same time showing that the NNAP model $\mathcal{M}$ converges if indeed sufficient information is fed to η.
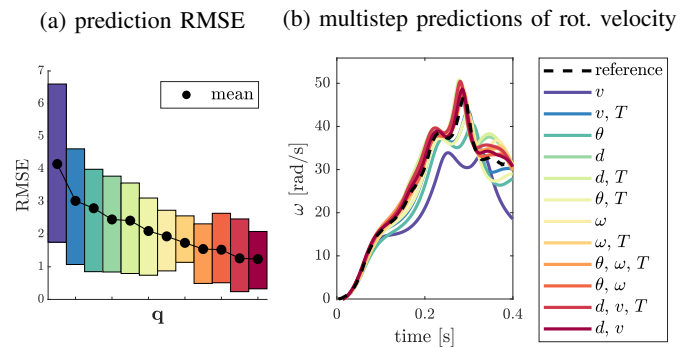

Fig. 12: Influence of the input information included in $\mathbf{q}$. The left plot represents 90% of the most accurate results obtained on test data via LOOCV. A prediction of test signal T_9 is illustrated on the right.

B. Region of Operations

Neural networks can capture complex relations. However, as is generally known, in their current form, they fail to extrapolate well outside their training region. Physics-based relations, on the other hand, can cover a wide range of operating points because they are substantiated by general physical laws. By using physics-inspired custom layers, we alleviate the modeling complexity a neural network of the complete system needs to capture. To validate the robustness and stability of the overall NNAP model, we define the set $\mathcal{I}_1$ that includes 10 trajectories with low operating speed. The central plot in Fig. 13 shows the predictions of the signals operating at the highest speed, by training the NNAP model only with trajectories included in $\mathcal{I}_1$. The right plot in Fig. 13 illustrates the prediction performances obtained by a LOOCV experiment, covering the entire region of operations in the training set. As expected, the average RMSE indicates that the prediction accuracy deteriorates once the test signals exceed the training region. The NNAP model, however, still follows the general dynamic behavior outside the training region. These results indicate the benefit of having a solid physics-
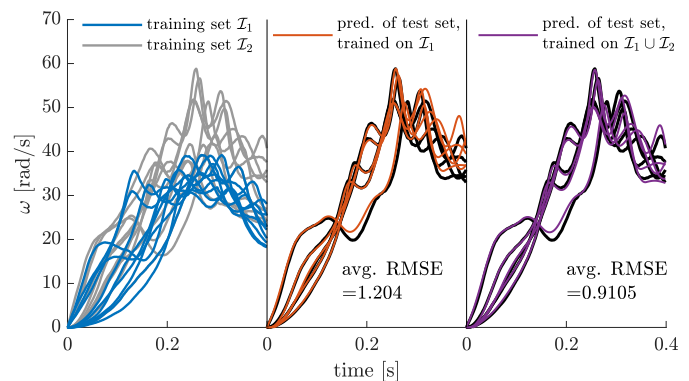

Fig. 13: Influence of having training data in the neighborhood of test signals. The left plot indicates the subdivision of the training set. The central and right plots show the prediction result on test data of a model trained on halved (i.e. limited region of operation) and full training set respectively.

inspired basis within the dynamic NNAP model.

C. Recurrent NNAP Model Multistep Prediction

The training process of a feedforward NNAP model $\mathcal{M}$ only penalizes the prediction error of the next timestep. A recurrent NNAP model $\mathcal{R}$, on the other hand, penalizes the average accuracy of a trajectory sample. Because the model learns from the general dynamics this way, multistep predictions can be potentially improved. The influence of the length of the trajectory sequence N of the recurrent model $\mathcal{R}$ of the slider-crank system is depicted in Fig. 14a. The aforementioned converged model $\mathcal{M}$ is used to initialize $\mathcal{R}$. Each model $\mathcal{R}$ is trained for 300 epochs in order to check whether these recurrent structures can improve the prediction capabilities. The average accuracy of the feedforward model $\mathcal{M}$ after 300 additional training epochs serves as a reference. The LOOCV reveals that the recurrent model $\mathcal{R}$, in general, does not perform better for unseen trajectories for small values of N. However, the accuracy on test data clearly outperforms the feedforward model $\mathcal{M}$ for training on larger time sequences N. The average RMSE of the test data was reduced by 33.4% for a model with $N = 300$ compared to the feedforward model $\mathcal{M}$. This improved prediction accuracy, however, comes with increased computational costs as recurrent neural networks virtually unfold into more complex multilayer networks in order to apply BPTT (as illustrated in Fig. 8). The required computational time (training time per epoch) almost follows a power-law relation w.r.t. the sequences length N used during the training, as can be observed in Fig. 14b. This experiment was performed on a processing unit including a 6-core Intel i5-8400 CPU with 16 GB RAM. Fig. 15a illustrates an example of a multistep prediction of a test data trajectory of $L = 800$ time instances by a NNAP model. The recurrent neural network $\mathcal{R}$ indicates better prediction performances compared to $\mathcal{M}$. The higher the length N of the time sequences used during training, the more the NNAP models are able to capture the global dynamics of the slider-crank mechanism. A FFNN architecture only incorporates the prediction accuracy of one timestep within the cost function (9). Naturally, drift occurs
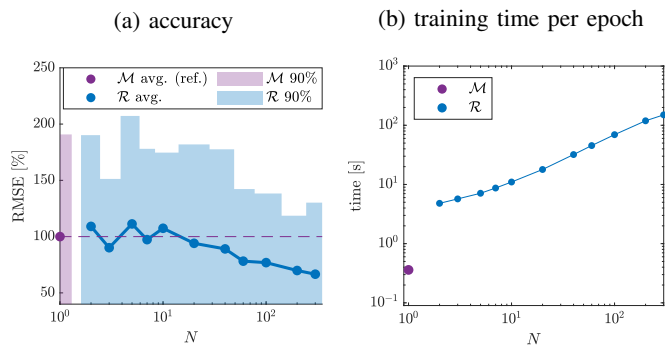

Fig. 14: Influence of step size N on the accuracy and required training time per epoch. The accuracy of the multistep predictions is tested by LOOCV. The left plot shows the average accuracy and the interval that includes 90% of the most accurate test signals. The right plot illustrates the time required to train one epoch for different number of timesteps N of the recurrent NNAP model $\mathcal{R}$.

because the prediction error at a specific time instance is propagated through the subsequent timestep. A RNN structure accommodates for these drift phenomena by incorporating the prediction accuracy for an entire trajectory segment within the cost function. Therefore, the RNN model learns to capture the dynamics for a longer time horizon, which leads to more robust prediction capabilities that better track the measured trajectories, as shown in Fig. 15b.
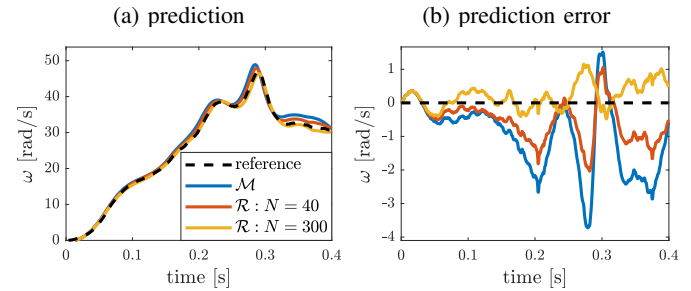

Fig. 15: Multistep prediction of test signal T_9.

D. Identifiability of the Physical Parameters

Besides having an accurate NNAP model for multistep predictions, we aim to discover new physical insights. Because the physical parameters $\mathbf{p}$ are updated simultaneously with the identification of η, research should be devoted to assess the identifiability of these parameters. The physical layers, including the model described in Appendix A, contain 7 parameters $\mathbf{p} = \{m_1, m_2, m_3, l_1, l_2, J_1, B_m\}$. In practice, the ability to uniquely identify a parameter relates both to its sensitivity to the model output and its interference with other parameters during the optimization process. For the presented NNAP model in particular, one should also analyze whether the influence of the neural network η is sufficiently uncorrelated with parameters $\mathbf{p}$ so that the neural network does not completely take over in the optimization process. Therefore, in Appendix B, a sensitivity analysis is applied to select the identifiable parameter sets, indicating that l_1 and m_3 need to be separated. Figure 16 illustrates the convergence history of parameter sets[1] $\mathbf{p} = \{m_3, J_1, B_m\}$ and $\mathbf{p} = \{l_1, J_1, B_m\}$ during the gradient-based optimization process of both $\mathbf{p}$ and η. The initial guess is randomly chosen $\pm$ 50% around the nominal value for each optimization run during the training procedure (LOOCV). The remaining physical parameters are held fixed during the optimization process. Nevertheless, we let them vary $\pm$ 10% around their nominal value (see Table I) in each experiment to indicate their insignificance when identifying the NNAP model. The orange line indicates the median, and the optimization boundaries are defined by the black dotted lines. The actual value of the geometrical parameter l_1 can easily be measured and is indicated by the purple line. The presented results indicate convergence of the proposed parameter sets. However, note that convergence cannot be obtained for all parameter combinations, as is further explained in Appendix B.

---

[1]Because these parameters are implemented as trainable weights within the custom physics layers, the optimization process requires parameter scaling to cope with the differences in order of magnitude between the variables in $\mathbf{p}$.
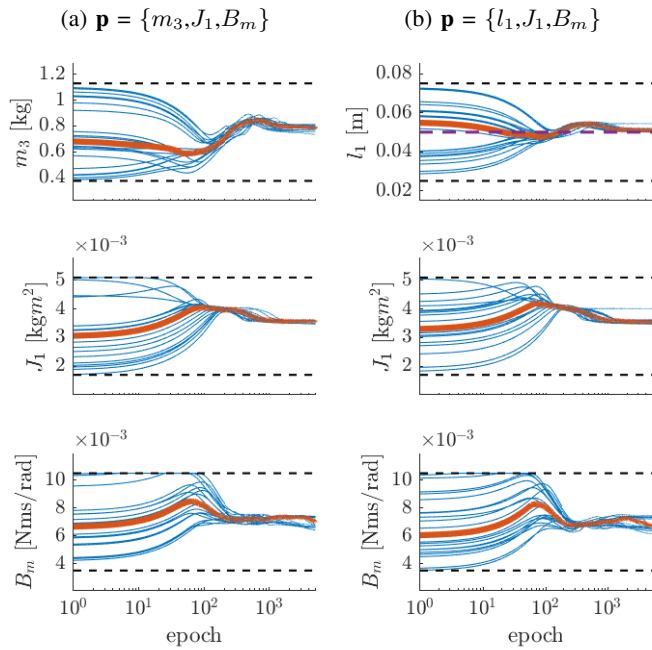
Fig. 16: Convergence history of physical parameters $\mathbf{p}$ during simultaneous optimization of $\mathbf{p}$ and neural network parameters $\boldsymbol{\alpha}$. The left and right plots indicate the convergence history of an optimization process of parameters $\mathbf{p} = \{m_3, J_1, B_m\}$ and $\mathbf{p} = \{l_1, J_1, B_m\}$, respectively.

E. Retrieving Physical Insights from the Neural Network

The identification of the physical parameters in $\mathbf{p} = \{m_3, J_1, B_m\}$ are indicative for the convergence of η because both parameters $\mathbf{p}$ and $\boldsymbol{\alpha}$ are simultaneously optimized. Therefore, we analyze the information captured by η after convergence of the NNAP model $\mathcal{M}$ to determine if the unknown load interactions $\mathcal{P}$ can be discoverd. At initialization, the neural network has no physical meaning (no resemblance with the actual $\mathcal{P}$) because the neural network is defined by random learnable parameter values $\boldsymbol{\alpha}_0$. The extracted neural network having as inputs d and v and as output $z = F$ is depicted in Fig. 17a. In that phase of training, completely wrong estimations of z are used as the input to (2), resulting in high initial loss. After convergence to a predictive NNAP model $\mathcal{M}$ (or $\mathcal{R}$), the ReLU network η with optimized parameter values $\boldsymbol{\alpha}^*$ can be extracted, see Fig. 17b.
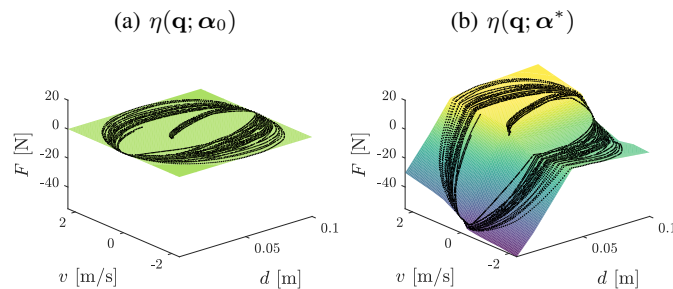
Fig. 17: Extracted ReLU network η for different model parameters $\boldsymbol{\alpha}$. The black dots indicate the data that were passed through the network during training.

The optimized ReLU network η is evaluated on the training dataset, resulting in Fig. 18a. The model η is only precise locally around the area because data-driven models are inherently difficult in extrapolation capabilities. In [37], we show for numerical simulation data that the ReLU network η indeed converges towards the real force relation F at these explored areas. The corresponding side view, depicted in Fig. 18b, illustrates the similarity between the empirically validated compression spring force F_s (see also Fig. 4) and the identified ReLU network η. From Fig. 18b, we can deduce a spring law together with friction phenomena.
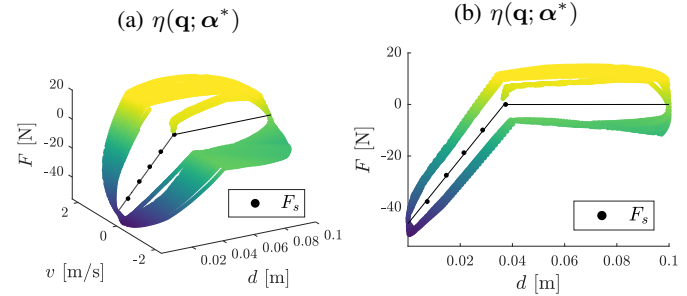

Fig. 18: Extracted ReLU network η evaluated on the training dataset.

The discrepancy between the discovered force relation η and the spring force F_s raises the interest to elaborate further on the interpretability of η. The overall horizontal force acting on the slider is, therefore, considered as a summation $F = F_c + F_{nc}$ of a conservative (F_c) and non-conservative (F_{nc}) force. A conservative force is characterized by the property that the work done by moving the object between two points is independent of the taken path. In practice, we consider a conservative force $F_c(d)$, being only dependent on the position d of the slider. Furthermore, the dissipative force $F_{nc}(d, v)$ relies on both the position d and linear speed v. Therefore, we replaced the ReLU network $z = \eta(d, v)$ with a new relation $z = \eta_c(d) + \eta_{nc}(d, v)$ deploying a summation of two separate feedforward ReLU networks $\eta_c(d)$ and $\eta_{nc}(d, v)$ that are trained in parallel within the NNAP model $\mathcal{M}$. The solution of $\eta_c(d)$ and $\eta_{nc}(d, v)$ is, however, not unique because $\eta_{nc}(d, v)$ includes the inputs of $\eta_c(d)$. In practice, an additional L_2 regularization term ($c \cdot \eta_{nc}^2$), with regularization parameter c, was added to the loss function. Fig. 19 illustrates the identified network components for an experimentally tuned regularization parameter $c = 10^{-6}$. The obtained results are the average predictions of η_c and η_{nc} evaluated by the training data for 10 converged NNAP models $\mathcal{M}$. The ReLU network η_c is able to accurately predict the behavior of the conservative spring force F_s, as can be observed in Fig. 19a. The regularization was able to successfully address the spring force to the ReLU network η_c. The remaining unknown dynamics attributed to the η_{nc} are shown in Fig. 19c. From the corresponding side view in Fig. 19e, we are able to discover the friction pattern that is dependent on the direction of the slider.

To validate the methodology that was able to retrieve physical insights, the aforementioned experiment was repeated

on the experimental setup having no compression spring. Hence, the state-dependent conservative force is removed. From Fig. 19b, it can be seen that the discovered conservative force model η_c trends towards zero. The non-conservative force can be extracted from η_{nc} and is depicted in Figs. 19d and 19f. The non-conservative forces exhibit similar behavior as those shown in 19c and 19e. These results suggest the ability of NNAP models to provide insights into the on-going conservative and non-conservative forces of a mechatronic system interacting with its environment.
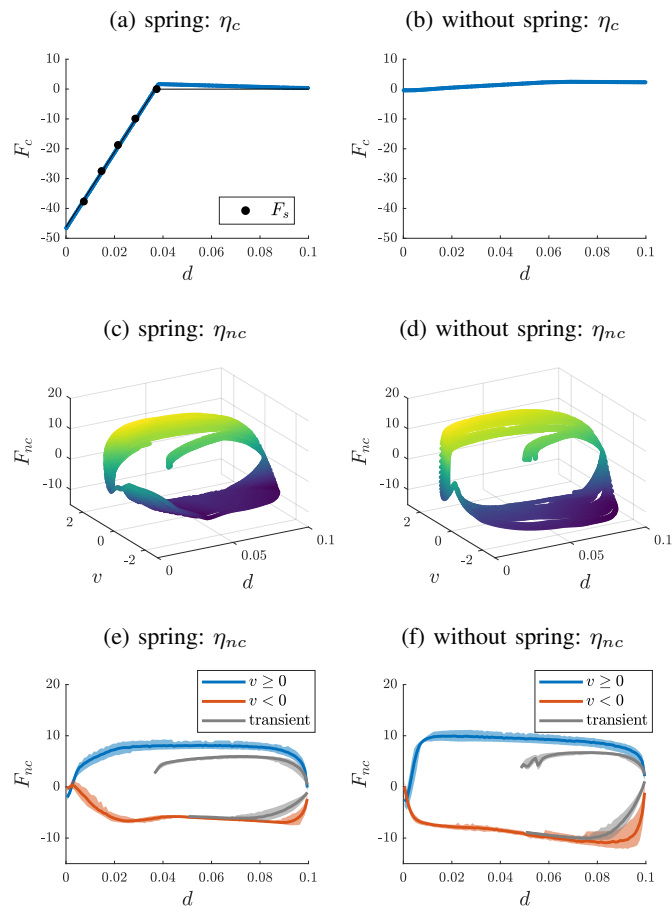

Fig. 19: Extracted ReLU networks η_c and η_{nc} evaluated on the training dataset, capturing respectively conservative and nonconservative forces.

V. Conclusion

This paper presents neural network augmented physics (NNAP) models that can learn the behavior of systems for which the dynamics are only partially known. Data-driven modeling techniques, here neural networks, are inserted in the model to compensate for the unmodeled interactions, without requiring direct measurements of these unknown phenomena. The method is validated on a slider-crank mechanism that exhibited unknown load interactions, induced by a spring and additional friction phenomena. The NNAP model is identified by simultaneously optimizing the neural network and the physical parameters, solely by using state and control input measurements. Therefore, automatic differentiation is used to calculate the gradient information through both the custom physics-inspired and neural layers. The obtained results of

the NNAP model are twofold. First, the modeling formalism enabled robust and accurate multistep predictions of a system that was a priori only partially known. Furthermore, both a feedforward and recurrent implementation of the modeling formalism are benchmarked, for which the latter was able to reduce the prediction error by up to 33.4% on test data. Secondly, the methodology provided a means to discover new insights of the unknown system properties. Results have shown that the NNAP methodology applied to a slider-crank mechanism can uniquely identify uncorrelated sets of physical parameters, while optimizing the neural network. In turn, the neural network, which was extracted from the NNAP model after convergence, accurately captured the state dependent load interactions. The presented methodology, however, heavily relies on the availability of accurate physical models describing the known dynamics. Furthermore, a good understanding of the influence of the unknown phenomena within the overall model is essential to assure physically interpretable results. It should be noted that for more complex cases, the flexibility of the neural network can take over the identification process, resulting in non-unique solutions. Further research is, therefore, required for discovering unknown phenomena using more elaborate regularization strategies.

APPENDIX A
PHYSICS-BASED MODEL

The physics-based dynamical model of the slider-crank mechanism is derived by considering the simplified multibody system in Fig. 20. The variable J_1 includes the rotational inertia of both motor and crank with respect to the rotation axis. The rotational inertia J_2 of the connection rod is approximated by $J_2 = \frac{1}{3} l_2 m_2^2$ in its center of mass (CM). Subsequently, this implies that we take the $r_2 = \frac{l_2}{2}$ in Fig. 2. By analogy, we approximate the CM of the crank by $r_1 = \frac{l_1}{2}$. Furthermore, we define $r' = l - r$ and let the subscripts r_x and r_y indicate the projection of r on, respectively, the x-axis and y-axis.

$$
\begin{aligned}
m_1 \ddot{x}_1 &= F_{ax} + F_{bx} \\
m_1 \ddot{y}_1 &= -m_1 g + F_{ay} + F_{by} \\
J_1 \dot{\omega} &= T - B_m \omega - r_{1x} m_1 g + l_{1x} F_{by} - l_{1y} F_{bx} \\
m_2 \ddot{x}_2 &= -F_{bx} + F_{cx} \\
m_2 \ddot{y}_2 &= -F_{by} + F_{cy} - m_2 g \\
J_2 \ddot{\phi} &= -r'_{2x} F_{by} - r'_{2y} F_{bx} - r_{2x} F_{cy} - r_{2y} F_{cx} \\
0 &= -F_{cy} - m_3 g + F_N \\
m_3 \dot{v} &= -F_{cx} - F
\end{aligned}
\tag{11}
$$

The aforementioned equations are combined with the geometrical relations, illustrated in Fig. 2, towards a physics-inspired model $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, u, F)$ with state $\mathbf{x} = \begin{bmatrix} \theta & \omega \end{bmatrix}^T$ and control input $u = T$. The highly nonlinear expression of $\mathbf{f}$ is derived via symbolic solvers and is due to its complexity not mentioned in this paper.

APPENDIX B
SENSITIVITY ANALYSIS

The identified physical parameters $\mathbf{p}$ of the slider-crank mechanism are depicted in Table I. The parameters that
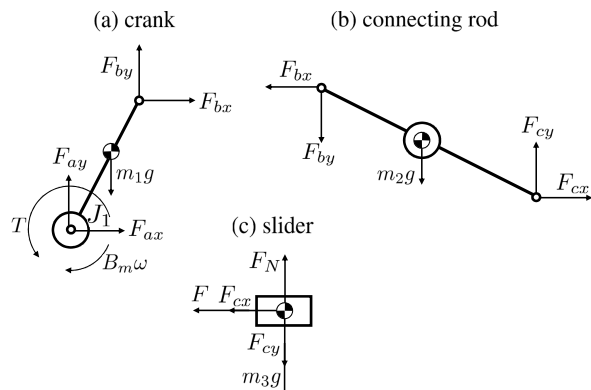
Fig. 20: Multibody scheme of slider-crank mechanism.

TABLE I: Identified parameters of slider-crank setup.

m_1	mass of crank	0.23	[kg]
m_2	mass of connection rod	0.35	[kg]
m_3	mass of slider	0.77	[kg]
l_1	length of crank	0.05	[m]
l_2	length of connection rod	0.29	[m]
J_1	inertia motor + crank	0.0034	[kg m^2]
B_m	friction coefficient of motor	0.007	[$\frac{\text{N m}}{\text{rad/s}}$]

could not be determined by optimizing the NNAP model are measured experimentally or determined via CAD drawings. Because some parameter combinations may lead to the same model output, non-unique solutions may, however, appear. The highly nonlinear nature of the proposed slider-crank model makes a structural identifiability analysis non trivial. However, based on [38], we implemented approximation techniques indicative for local identifiability around the optimal solution. We elaborate on the derivative function f evaluated for a sample k as $\dot{\omega}_k = f(\theta_k, \omega_k, T_k, \hat{F}_k; \mathbf{p})$. In practice, all measurement trajectories are aggregated into one dataset of K samples. The converged neural network η is deployed on each measurement sample $k \in \{1, \ldots, K\}$ to generate the corresponding estimations $\hat{F}_k$ of the unknown force interaction. Next, we derive the sensitivity matrix S in which we quantify the impact on the model output for modifications on F and physical parameters $\mathbf{p} \in \mathbb{R}^r$.

$$
\text{S} = \begin{bmatrix} \mathbf{s}_1 \cdots \mathbf{s}_{r+1} \end{bmatrix} = \begin{bmatrix} \left.\frac{\partial f}{\partial F}\right|_{k=1} & \left.\frac{\partial f}{\partial p_1}\right|_{k=1} & \cdots & \left.\frac{\partial f}{\partial p_r}\right|_{k=1} \\ \vdots & \vdots & \ddots & \vdots \\ \left.\frac{\partial f}{\partial F}\right|_{k=K} & \left.\frac{\partial f}{\partial p_1}\right|_{k=K} & \cdots & \left.\frac{\partial f}{\partial p_r}\right|_{k=K} \end{bmatrix}
$$

Again, we consider normalized parameters in order to develop a fair scaling to cope with the different sizes of magnitude. The L_2 norm of each sensitivity vector $\mathbf{s}_j$ measures the influence of the corresponding feature j to the model output. Figure 21 illustrates the high sensitivity of the parameters l_1, J_1 and m_3, justifying the choice to include them in the optimization set $\mathbf{p}$. In addition, the parameter B_m is added because this variable is unknown and cannot be measured. The high sensitivity of F, which is representative for the output of the neural network η, is related to the convergence of η. Although a set of variables can be highly sensitive, if parameter combinations have the same influence on the model output, they cannot be uniquely determined. Therefore, we construct the normalized
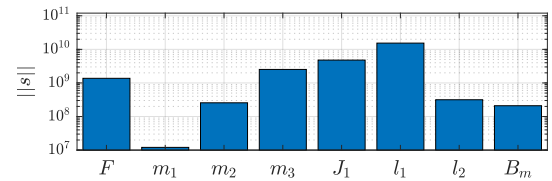

Fig. 21: Local sensitivity analysis of unknown force interaction F and physical parameters $\mathbf{p}$.

sensitivity matrix S_N that contains the normalized column vectors $\mathbf{s}_j^N = \mathbf{s}_j/||\mathbf{s}_j||$ for $j \in \{1, .., r+1\}$. The corresponding correlation matrix $\text{Q} = \text{S}_N^T \text{S}_N$ exhibits diagonal elements equal to one and results in the graph shown in Fig. 22. This result demonstrates that the physical parameters are barely correlated with the output of the neural network represented by F. However, we can observe, for instance, a high correlation of parameters l_1 with m_3 (and m_2) on the model output. When considering parameter set $\mathbf{p} = \{l_1, m_3, J_1, B_m\}$, we can observe from Fig. 23 that they indeed do not converge well in the optimization process.
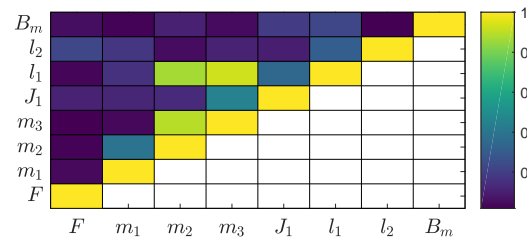

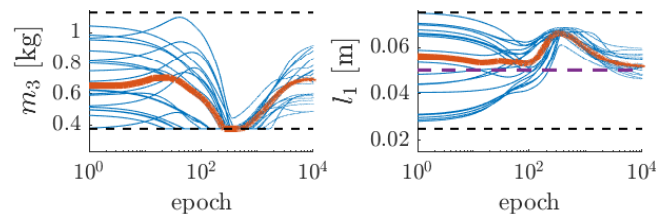Fig. 22: Correlation matrix Q of the sensitivity vectors $\mathbf{s}_j^N$.

Fig. 23: Convergence history of l_1 and m_3 when including both in the parameter set $\mathbf{p} = \{l_1, m_3, J_1, B_m\}$, simultaneously optimized with the neural network parameters $\boldsymbol{\alpha}$.

REFERENCES

[1] S. Kim, "Moment of inertia and friction torque coefficient identification in a servo drive system," *IEEE Trans. Ind. Electron.*, vol. 66, no. 1, pp. 60–70, 2019.

[2] D. Papageorgiou, M. Blanke, H. H. Niemann, and J. H. Richter, "Robust backlash estimation for industrial drive-train systems–theory and validation," *IEEE Trans. Control Syst. Technol.*, 2018.

[3] X. Wang, W. Wang, L. Li, J. Shi, and B. Xie, "Adaptive control of dc motor servo system with application to vehicle active steering," *IEEE/ASME Trans. Mechatronics*, 2019.

[4] A. Kamadan, G. Kiziltas, and V. Patoglu, "Co-design strategies for optimal variable stiffness actuation," *IEEE/ASME Trans. Mechatronics*, vol. 22, no. 6, pp. 2768–2779, 2017.

[5] S. J. Kim, K.-S. Kim, and D. Kum, "Feasibility assessment and design optimization of a clutchless multimode parallel hybrid electric powertrain," *IEEE/ASME Trans. Mechatronics*, vol. 21, no. 2, pp. 774–786, 2015.

[6] H. Ziyuan, H. Bangcheng, and L. Yun, "An explicit nonlinear model predictive abs controller for electro-hydraulic braking systems," *IEEE Trans. Ind. Electron.*, 2019.

[7] T. B. Schön, A. Wills, and B. Ninness, "System identification of nonlinear state-space models," *Automatica*, vol. 47, no. 1, pp. 39–49, 2011.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMECH.2021.3058536, IEEE/ASME Transactions on Mechatronics

11

[8] J. Kocijan, *Modelling and control of dynamic systems using Gaussian process models*. Springer, 2016.

[9] T. B. Wills, A.and Schön, L. Ljung, and B. Ninness, "Identification of hammerstein–wiener models," *Automatica*, vol. 49, no. 1, pp. 70–81, 2013.

[10] S. A. Billings, *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons, 2013.

[11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[12] O. Lakhal, A. Melingui, and R. Merzouki, "Hybrid approach for modeling and solving of kinematics of a compact bionic handling assistant manipulator," *IEEE/ASME Trans. Mechatronics*, vol. 21, no. 3, pp. 1326–1335, 2015.

[13] O. Ogunmolu, X. Gu, S. Jiang, and N. Gans, "Nonlinear systems identification using deep dynamic neural networks," *arXiv preprint arXiv:1610.01439*, 2016.

[14] N. Mohajerin and S. L. Waslander, "Modular deep recurrent neural network: Application to quadrotors," in *IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC 2014)*. IEEE, 2014, pp. 1374–1379.

[15] ——, "Multistep prediction of dynamic systems with recurrent neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, 2019.

[16] Z. Liu, H. Fang, and J. Xu, "Identification of piecewise linear dynamical systems using physically-interpretable neural-fuzzy networks: Methods and applications to origami structures," *Neural Networks*, vol. 116, pp. 74–87, 2019.

[17] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Multistep neural networks for data-driven discovery of nonlinear dynamical systems," *arXiv preprint arXiv:1801.01236*, 2018.

[18] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *Science*, vol. 324, no. 5923, pp. 81–85, 2009.

[19] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proc. of the Nat. Academy of Sciences*, p. 201517384, 2016.

[20] M. Lutter, C. Ritter, and J. Peters, "Deep lagrangian networks: Using physics as model prior for deep learning," *Int. Conf. on Learning Representations, 2019.*, 2019.

[21] S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 15 353–15 363.

[22] X. Jia, J. Willard, A. Karpatne, J. Read, J. Zwart, M. Steinbach, and V. Kumar, "Physics guided rnns for modeling dynamical systems: A case study in simulating lake temperature profiles," in *Proc. of the 2019 SIAM Int. Conf. on Data Mining*. SIAM, 2019, pp. 558–566.

[23] A. Ajay, M. Bauza, J. Wu, N. Fazeli, J. B. Tenenbaum, A. Rodriguez, and L. P. Kaelbling, "Combining physical simulators and object-based networks for control," *arXiv preprint arXiv:1904.06580*, 2019.

[24] A. Punjani and P. Abbeel, "Deep learning helicopter dynamics models," in *IEEE Int. Conf. on Robotics and Automation*. IEEE, 2015, pp. 3223–3230.

[25] I. Boldea, L. N. Tutelea, W. Xu, and M. Pucci, "Linear electric machines, drives, and maglevs: an overview," *IEEE Trans. Ind. Electron.*, vol. 65, no. 9, pp. 7504–7515, 2018.

[26] C. Sharma and K. Purohit, *Theory of mechanisms and machines*. PHI Learning, 2006.

[27] Z. Gao, R. S. Colby, L. Turner, and B. Leprettre, "Filter design for estimating parameters of induction motors with time-varying loads," *IEEE Trans. Ind. Electron.*, vol. 58, no. 5, pp. 1518–1529, 2011.

[28] M. Li, R. Foss, K. Stelson, J. Van de Ven, and E. J. Barth, "Design, dynamic modelling and experiment validation of a novel alternating flow variable displacement hydraulic pump," *IEEE/ASME Trans. Mechatronics*, 2019.

[29] G. Eren, R.and Ozkan and M. Karahan, "Comparison of heald frame motion generated by rotary dobby and crank & cam shedding motions," *Fibres and textiles in eastern Europe*, vol. 13, no. 4, p. 78, 2005.

[30] E. Zheng and X. Zhou, "Modeling and simulation of flexible slider-crank mechanism with clearance for a closed high speed press system," *Mechanism and Machine theory*, vol. 74, pp. 10–30, 2014.

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

[32] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[33] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Int. Conf. on Machine Learning*, 2013, pp. 1310–1318.

[34] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[35] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symp. on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[37] W. De Groote, E. Kikken, S. Goyal, S. Van Hoecke, E. Hostens, and G. Crevecoeur, "Hybrid derivative functions for identification of unknown loads and physical parameters with application on slider-crank mechanism," in *2019 IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2019.

[38] I. Ioslovich, M. Moran, and P. Gutman, "Identification of a nonlinear dynamic biological model using the dominant parameter selection method," *Journal of the Franklin Institute*, vol. 347, no. 6, pp. 1001–1014, 2010.

Wannes De Groote Wannes De Groote received a M.Sc. degree in electromechanical and control engineering from Ghent University, Ghent, Belgium, in 2017. In August 2017, he joined the Department of Electromechanical, Systems and Metal Engineering, Ghent University, as a Ph.D. Student. He received a personal grant strategic basic research (SB) from Research Foundation - Flanders (FWO) in 2019. His research interests mainly concern simulation and modeling of dynamic systems by combining physical and data-driven approaches.

Edward Kikken Edward Kikken obtained his master's degree in mechanical engineering - focused on control theory - at the Technical University of Eindhoven in 2013. His professional career began as an R&D engineer focused on advanced motion control for offshore application at Allseas. Since 2017, he has been working for the Flemish strategic research institute Flanders Make, currently as a senior researcher in systems and control, signal processing, optimization, and machine learning.

Erik Hostens Erik Hostens earned a master's degree magna cum laude in electromechanical engineering (datamining and automation) in 2003, and a PhD in quantum information theory in 2007, both at KULeuven. He worked for a few years in R&D on advanced process control for Siemens. Since 2010, he has been employed at Flanders Make, currently as a senior researcher in systems and control, estimation, optimization, and machine learning.

Sofie Van Hoecke Sofie Van Hoecke is an Associate Professor at IDLab, Ghent University-imec. She and her team are active on the study and development of innovative hybrid machine learning solutions combining machine learning with semantics, expert knowledge and/or physical knowledge, efficient and accurate characterization, modeling and training, and testing and optimization, with applications in predictive maintenance and predictive healthcare.

Guillaume Crevecoeur Guillaume Crevecoeur (°1981) received a MSc and a PhD degree in Engineering Physics from Ghent University in 2004 and 2009, respectively. In 2009, he became a postdoctoral fellow of the Research Foundation Flanders. In 2014, he was appointed Associate Professor at the Faculty of Engineering and Architecture of Ghent University. His research interests are the modeling, optimization, and control of dynamical mechatronic and industrial robotic systems.