Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Modeling System Dynamics with Physics-Informed Neural Networks Based on Lagrangian Mechanics ⋆

Manuel A. Roehrl *,** **Thomas A. Runkler** *,**
Veronika Brandtstetter * **Michel Tokic** * **Stefan Obermayer** *

* *Siemens AG, Corporate Technology, 81739 Munich, Germany (e-mail: manuel.roehrl@siemens.com; thomas.runkler@siemens.com; veronika.brandstetter@siemens.com; michel.tokic@siemens.com)*
** *Technical University of Munich, 85748 Garching, Germany (e-mail: m.roehrl@tum.de; thomas.runkler@tum.de)*

Abstract: Identifying accurate dynamic models is required for the simulation and control of various technical systems. In many important real-world applications, however, the two main modeling approaches often fail to meet requirements: first principles methods suffer from high bias, whereas data-driven modeling tends to have high variance. Additionally, purely data-based models often require large amounts of data and are often difficult to interpret.
In this paper, we present *physics-informed neural ordinary differential equations* (PINODE), a hybrid model that combines the two modeling techniques to overcome the aforementioned problems. This new approach directly incorporates the equations of motion originating from the Lagrange mechanics into a deep neural network structure. Thus, we can integrate prior physics knowledge where it is available and use function approximation—e. g., neural networks—where it is not. The method is tested with a forward model of a real-world physical system with large uncertainties. The resulting model is accurate and data-efficient while ensuring physical plausibility.
With this, we demonstrate a method that beneficially merges physical insight with real data. Our findings are of interest for model-based control and system identification of mechanical systems.

Keywords: neural network models, computer simulation, differential equations, semi-parametric identification, system identification

1. INTRODUCTION

Performant and expressive computer simulation models, able to seamlessly incorporate physical measurements, are key for *digital twins*; these, in turn, are indispensable for the digitalization of industry (Rosen et al., 2015). This artificial replica refers to a virtual description of a system that can integrate models with data in real time. The combination enables what–if analyses in an artificial environment and optimization of processes and products without inferring the real process. Recently, there has been impressive progress in the field of data-based modeling (Krizhevsky et al., 2012), especially in the area of deep learning (LeCun et al., 2015). Nevertheless, two factors are essential for this progress: (a) huge, high-quality data sets and (b) immense computing power resulting from clusters of arithmetic units. These resources have become available most recently (see Moore's law). Factor (a) allows variance reduction, and factor (b) enables larger models to be trained, which reduces bias (Fan et al., 2019). However, one problem with this type of modeling is that there are not always large amounts of usable data available. If that is the case, data-based models often show high variance. Furthermore, complex models lose their interpretability because of the high number of adaptation parameters, whereas classical physical models usually only have a few parameters and thus remain understandable. In addition, physical models only require a small amount of data for calibration. Because of their simplicity, however, they usually have a high bias. To reduce model bias and bridge the gap between both model types, various approaches have been employed, including learning of correction terms and semi-physical models based on different subsystems or multi-fidelity modeling (Fernández-Godino et al., 2016; von Stosch et al., 2014). One recent development is physics-informed neural networks (Raissi et al., 2019; Lutter et al., 2019; Greydanus et al., 2019; Zhong et al., 2020; Gupta et al., 2019; Rackauckas et al., 2020), which use mechanistic equations to endow neural networks with better prior. We follow this line and propose physics-informed neural ordinary differential equations (PINODE). Our approach uses the equations of motions to structure the neural network. The model is then integrated to obtain the final model output. Within this work, we investigate whether PINODE is applicable to a real system and is more accurate than a standard model derived from Lagrangian mechanics.

⋆ This work was sponsored by the German Federal Ministry of Education and Research (ID: 01 IS 18049 A).

Therefore, we demonstrate the procedure using a real-world mechanical benchmark system—an inverted pendulum mounted on a cart, or, in short, cart pole. We focus on only parametrizing non-conservative forces such as friction and using physical insights for the remaining parts of the differential equation.

After this introduction, we describe the proposed methodology. We then explain the conducted experiment and discuss the resulting findings. The next section overviews related approaches to modeling system dynamics in a semi-physics manner. Lastly, we summarize the results and give an outlook on future work.

2. METHODOLOGY

In model-based control, dynamics of mechanical systems are modeled by linking the system state $\mathbf{q}$ with the acting input $\mathbf{u}$. Depending on the respective contexts either the forward f or the inverse f^{-1} model is used. We want to find a forward model

$$f(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}) = \ddot{\mathbf{q}}, \tag{1}$$

which simulates the system state change for a given input $\mathbf{u}$. To obtain a favorable coupling between input and states, we suggest first deriving the equations of motion with the *Lagrange formalism* and then integrating them into a neural network structure.

2.1 Lagrangian Mechanics

Describing the trajectory of a system has been extensively studied, and various mathematical formulations exist to derive the corresponding differential equations. Within this approach, a modified form of the Lagrange mechanics is used, i. e., the Euler–Lagrange formulation with generalized coordinates and non-conservative forces. The formalism uses the energy of the system; therefore, the Lagrangian L is a function of the generalized coordinates $\mathbf{q}$, which is defined as

$$L = T - V. \tag{2}$$

T represents the entire kinetic energy and V is the total potential of the system. Applying the calculus of variations yields the *Euler–Lagrange equation* as follows:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\mathbf{q}}} - \frac{\partial L}{\partial \mathbf{q}} - \mathbf{Q}^{ncons} = 0, \tag{3}$$

where $\mathbf{Q}^{ncons}$ are the non-conservative forces. By inserting Equation (2), the formula can be described as

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{\mathbf{q}}} - \frac{\partial T}{\partial \mathbf{q}} + \frac{\partial V}{\partial \mathbf{q}} - \mathbf{Q}^{ncons} = 0, \tag{4}$$

since V is not a function of $\dot{\mathbf{q}}$. Although Equation (4) contains partial derivatives, the result of those derivatives provides *ordinary differential equations* (ODEs) of the second order. By applying the chain rule, we can write the equations of motion in the common matrix form

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \mathbf{Q}^{ncons}. \tag{5}$$

Here, $\mathbf{M}(\mathbf{q})$ represents the inertia matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the Coriolis matrix, and $\mathbf{G}(\mathbf{q})$ are the conservative forces. By using this equation, any mechanical system with holonomic constraints can be described, e. g., coupled pendulums or robotic manipulators.

2.2 Incorporating Equations into Neural Net Structure

On the basis of the equation of motion (5), a classical engineering approach would now begin with measuring or estimating the required parameters and forces. In contrast, a learning approach would abandon the equations and find a mapping between input and states directly from data. We want to combine both approaches, using the structure of the Euler–Lagrange equation and directly parametrizing parts of it.

While enforcing this structure, we do not use a direct function to map from an initial state to the next one. Instead, we learn the underlying ODE. Therefore, to find a future state, we need to integrate the differential equation. That way, the model is memory and parameter efficient (Chen et al., 2018). Furthermore, we can use position and velocity measurements for the parameter optimization and avoid the need to measure accelerations. To solve the ODE, we apply the *Runge–Kutta* method of the fourth order (Runge, 1895), which is a standard method for fixed-time-step integration, although we assume the use of other solver schemes is also possible. Given the integration method and a system's initial generalized coordinates and velocities, we can solve the initial value problem. In this manner, we find the future system state after some step size h at some time $t_{n+1} = t_n + h$. We reduce the problem and define $\mathbf{z}_n = [\mathbf{q}_n, \dot{\mathbf{q}}_n]^T$. The iterative scheme can then be described for the equations of motion as

$$\mathbf{z}_{n+1} = \mathbf{z}_n + \frac{1}{6}\left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right), \tag{6}$$

with

$$\begin{aligned} \mathbf{k}_1 &= h \cdot g\left(t_n, \mathbf{z}_n, \mathbf{u}_n\right) \\ \mathbf{k}_2 &= h \cdot g\left(t_n + h/2, \mathbf{z}_n + \mathbf{k}_1/2, \mathbf{u}_n\right) \\ \mathbf{k}_3 &= h \cdot g\left(t_n + h/2, \mathbf{z}_n + \mathbf{k}_2/2, \mathbf{u}_n\right) \\ \mathbf{k}_4 &= h \cdot g\left(t_n + h, \mathbf{z}_n + \mathbf{k}_3, \mathbf{u}_n\right). \end{aligned} \tag{7}$$

$\mathbf{g}(t, \mathbf{z}, \mathbf{u})$ is a vector of the generalized velocities and generalized accelerations. The latter can be found by solving Equation (5) for $\ddot{\mathbf{q}}$ (see Equation (9)).

We propose the use of a universal function approximator, like a neural network, for parts, that are unknown or nontrivial to model but define the physical parameters where knowledge is available. For example, parameters like gravity, lengths, masses and moments of inertia are often easy to identify, whereas non-conservative forces are more difficult to determine. It is nontrivial because the forces often represent one or more partly interfering physical phenomena, e. g., friction, air drag, or fluid interactions. Models used to describe those phenomena are often not derived from first principles and need the use of extensive empirical methods, which are sometimes either specific for an individual application or only poorly approximate the underlying physical phenomena, or both.

Generally, identifying physical parameters jointly with the neural network has only worked for a very limited parameter space. Therefore, if a certain part of the differential equation is not known, this part should be replaced by a universal approximator like e. g. a multi-layer perceptron.

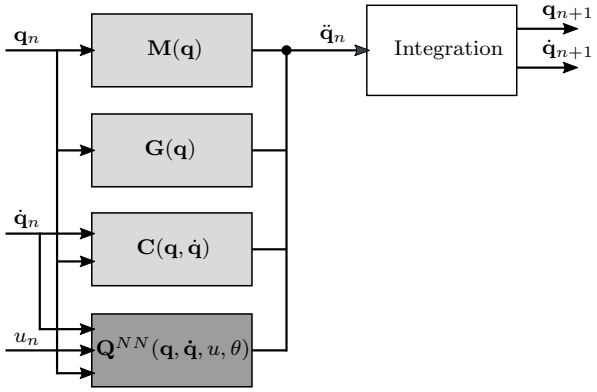


Fig. 1. Simplified computation graph for one forward pass of an exemplary PINODE model to predict $\mathbf{q}_{n+1}$ and $\dot{\mathbf{q}}_{n+1}$ at time $t_{n+1} = t_n + h$.

Figure 1 illustrates the technique described above for a single forward pass with a simplified computation graph. The diagram shows an example where the non-conservative forces are substituted by a neural network, which is described as

$$\mathbf{Q}^{ncons} = \mathbf{Q}^{NN}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}; \theta), \tag{8}$$

where θ represents the trainable weights. The remaining matrices $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{G}$ are determined by traditionally measuring or estimating the necessary parameters. All components together are solved to yield the explicit form for the acceleration:

$$\ddot{\mathbf{q}} = \mathbf{M}(\mathbf{q})^{-1}\Big[\mathbf{Q}^{NN}(\mathbf{q}, \dot{\mathbf{q}}, u, \theta) - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{G}(\mathbf{q})\Big]. \tag{9}$$

We then apply the integration method, described in Equation (6), to obtain the position $\mathbf{q}_{n+1}$ and velocity $\dot{\mathbf{q}}_{n+1}$ for the next state.

2.3 Learning Parameters from Time Series Data

Having discussed how to construct the PINODE model, we next address how to obtain the network variables θ. For this, the optimization problem is defined as follows:

$$\theta^* = \min \mathcal{L}\big(\text{ODESolve}\big(f(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}; \theta)\big), \mathbf{q}, \dot{\mathbf{q}}\big), \tag{10}$$

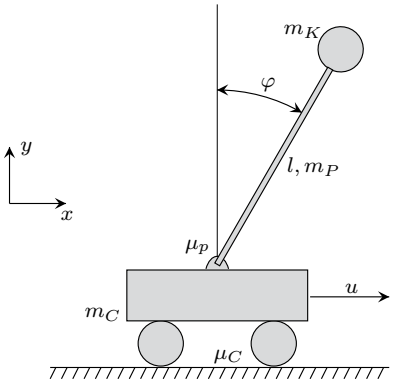
where $\mathcal{L}$ can be an arbitrary loss function. The only constraint is that it must be differentiable with respect to its parameters to enable gradient computation with *automatic differentiation*. In particular, we use the reverse-mode automatic differentiation to get the derivatives of the loss in Equation (11) toward its weights θ. However, for the experiment in the subsequent section, we used the following cost function:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{k=1}^{N} \lambda_1(\mathbf{q}_k - \mathbf{q}'_k)^2 + \lambda_2(\dot{\mathbf{q}}_k - \dot{\mathbf{q}}'_k)^2, \tag{11}$$

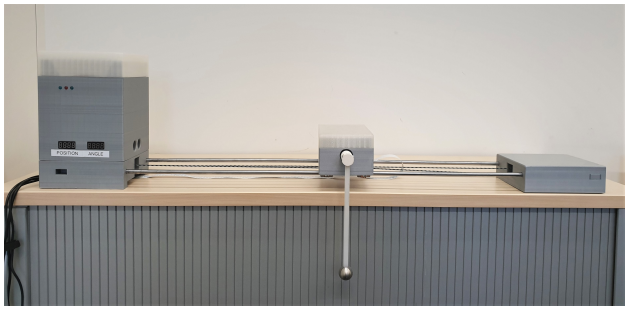
where the kth future state is found by integrating Equation (9):

$$[\mathbf{q}'_k, \dot{\mathbf{q}}'_k] = \text{ODESolve}\big(f(\mathbf{q}_k, \dot{\mathbf{q}}_k, \mathbf{u}_k; \theta)\big), \tag{12}$$

N represents the number of state transitions for which the loss is calculated. The λ-factors can be chosen to put weight on specific generalized coordinates or velocities. If a generalized degree of freedom is described with angular coordinates, the loss function must be slightly extended. This is because the fact that angles lie in a non-Euclidean space, mostly between $[0, 2\pi]$, which impedes the optimization. To address this issue, we use, instead of the direct angle φ, an embedded form $(cos(\varphi) + sin(\varphi))$ for the cost calculation. Finally, we find the parameters with the Adam optimizer with a learning rate of 10^{-3} (Kingma and Ba, 2014), although the use of other gradient descent algorithms is generally also possible.



(a) Schematic drawing of the test rig.



(b) Picture of the test rig system, presented at CeBIT 2018.

Fig. 2. Schematic drawing and picture of the test rig system—the inverted pendulum on a cart.

3. EXPERIMENTS

For the purpose of testing and evaluation, we used the method for system identification of a real-world physical cart pole. The subject of investigation is shown by a schematic sketch and a photograph in Figure 2. We want to justify the claim that the approach can learn the non-conservative forces of a real complex system. To proof that, we show that it is possible to integrate existing knowledge seamlessly and to map the physical, still unmodeled components with the neural network.

3.1 Experimental Setup

The cart is connected with a string to a motor, which can be actuated to give control impulses to the system. The control input moves the cart along the linear guidance, which has a total length of about 0.4 m. The test rig is equipped with two sensors that record the position of the cart and pole. The first is measured with an ultrasonic distance sensor, and the second by an optical hollow shaft encoder. Velocities are calculated from the difference between two measurements. Additionally, the introduced control input is logged and saved with the state measurements. Within the experiment, trajectories were generated by manually controlling the cart in a random fashion. In total, samples of 8 min measuring were used for training, this corresponds

to approximately 200 times moving from right to left and back again. The data are recorded at a sample frequency of 50 Hz. Accordingly, the time step of the integrator was set to the resulting time interval of 0.02 s. Table 1 gives an overview of information about the data used for training.

The algorithm is implemented within the *TensorFlow* framework, which enables automatic differentiation and offers various optimizers and predefined modules for building deep learning architectures. The models created this way use the same parameters, except for the friction factor of the cart, which is needed only for the pure ODE comparison model. All used parameters of both models are specified in Table 2.

Table 1. Information about the training data.

Sample rate			50 Hz		
Samples			24,271		
		Mean	STD.	Min	Max
Statistics	x (m)	0.024	0.126	−0.325	0.276
	φ (rad)	3.290	1.823	0	6.260
	$\dot{x}$ (m/s)	0	0.430	−4.398	4.141
	$\dot{\varphi}$ (rad/s)	−0.328	6.742	−23.562	18.326

Table 2. Parameters describing dynamics of cart and pendulum apparatus

Parameter	Value	Unit
mass cart m_c	0.466	kg
mass pole m_p	0.06	kg
mass sphere m_s	0.012	kg
length pole l	0.201	m
friction factor cart μ_c	0.0408	-
friction factor pole μ_p	0.0020	-
gravity g	9.81	m/s^2

3.2 PINODE Model

The parameters gravity, lengths, masses, and moments of inertia are measured for the cart pole. With this, the matrices $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{G}$ are defined. Further, we use viscous friction for the pole bearing. The residual non-conservative forces $\mathbf{Q}$ are modeled by a artificial neural network depending on the last system state and control input u. The model is composed of a multi-layer perceptron with two hidden layers each containing 50 units. All layers beside the last one use rectifier activation functions; the last applies a hyperbolic tangent function. The model has five inputs and one output, which gives a total of 5,451 trainable parameters. Before training, the samples were shuffled randomly and then combined into batches of 128 elements.

3.3 Pure ODE Model

The performance of the PINODE model is compared with that of a standard model derived with the Lagrange formalism (4). This model contains non-conservative forces to consider both friction between cart and linear guidance, and resistance in the pole bearing. For the former, Coulomb's friction with constant normal force, and for the latter, viscous friction was used. Thus, the block for the non-conservative forces in Figure 1 is described for the pure ODE model as follows:

$$\mathbf{Q}^{ncons} = \\ = \begin{bmatrix} u - (m_P + m_K + m_C) \cdot g \cdot \mu_C \cdot sign(\dot{x}) \\ -\mu_P \cdot \dot{\varphi} \end{bmatrix}. \quad (13)$$

The necessary friction factors were found by least squares optimization.

4. RESULTS AND DISCUSSION

Figure 3 compares the measured ground truth to the pure ODE model and the PINODE model by showing simulations for the generalized coordinates over hundred time steps (2 seconds), while starting from the same initial state. The cart position of the hybrid model in the top left corner of the figure is predicted very accurately—only a small error occurs. Similarly, the cart velocity is reproduced precisely, although the measurements are very noisy. On the right side of the figure, the pole positions φ and velocities $\dot{\varphi}$ of the PINODE model show that a small error is accumulating over time, but not to the same extent as in the case of the cart position x and velocity $\dot{x}$ on the left. The pure ODE curves in all four graphs show a far greater deviation from the measured trajectory than does the semi-physical model, although the principle tendency is correct.

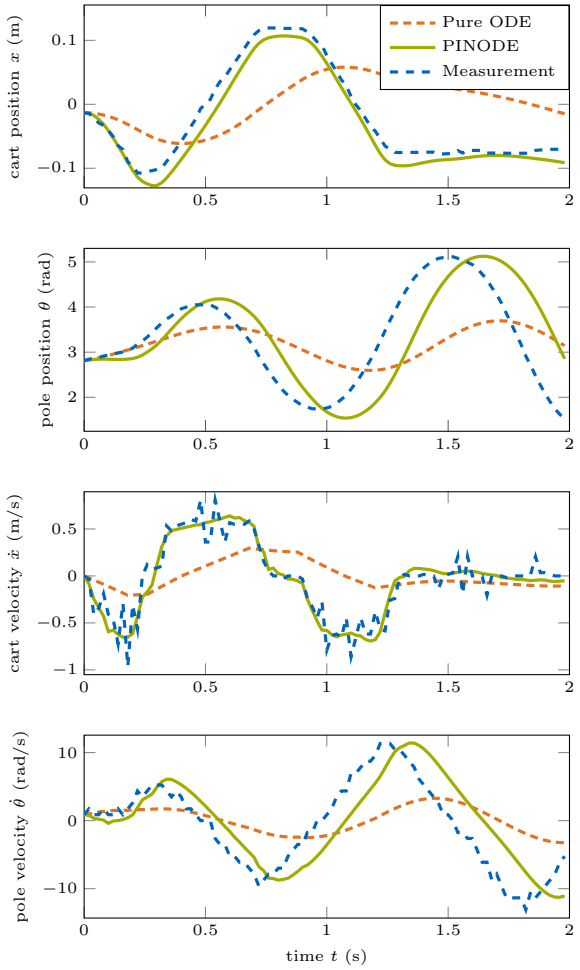


Fig. 3. Pure ODE simulation and PINODE model predictions for 2 s with the same initial conditions and control input u compared with sensor data.

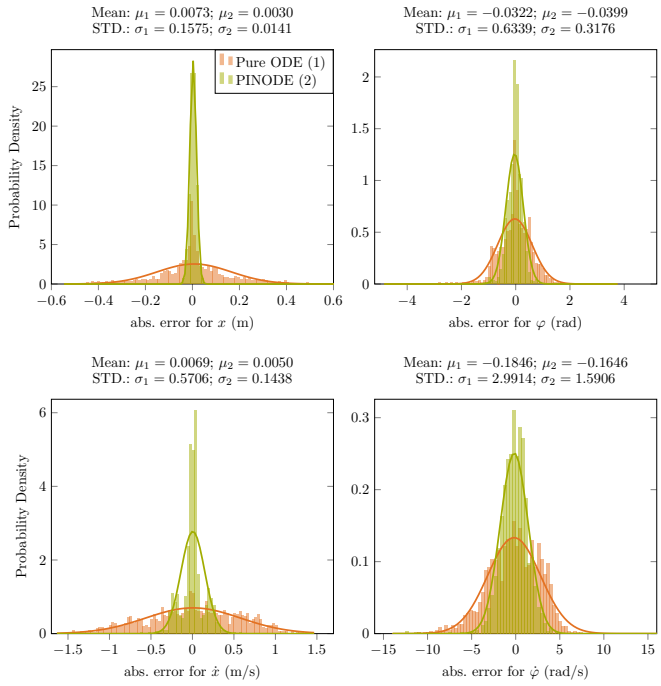


Fig. 4. Histograms showing absolute errors of pure ODE and PINODE model generalized coordinates and velocities for one thousand 0.6 seconds-long simulations. A normal distribution is fitted and plotted for each histogram, their means $\mu_{\{1,2\}}$ and standard deviations $\sigma_{\{1,2\}}$ are shown above the respective graphs.

To present the performance of the approach for a larger state space, Figure 4 illustrates the absolute errors over multiple simulated time intervals. More specifically, the models were evaluated for thirty time steps for thousand times. The initial state was set to the real measurement in every time interval. For each time frame and state, the absolute error between both models and the measured ground truth was calculated. The resulting error values are plotted as histograms. This representation confirms the observation from the time sequences in Figure 3: the kinematics of the cart are modeled more accurately than are those of the pole. A possible explanation for this might be that the pendulum has more complex dynamics than has the cart. Overall, the PINODE model completely outperforms the pure ODE model. The better performance of the composite model must be attributed to the universal approximation capability of the neural network component, because it is able to take phenomena into account, which have not been considered in the pure ODE model. A few examples that could be observed as phenomena in the real-world system are friction depending on the x-direction, elasticity in the string connecting cart and motor, elasticity in the linear guidance, time delay of the control input and mechanical clearance in the bearings.

In addition to the two described models, a standard—black box—neural network was trained to map from an initial state to the hidden state. The performance of this model is not shown in Figures 3 and 4 because it was unable to learn the dynamics with the same training data. We presume this is because the proposed structure contains fewer free parameters and therefore requires less data than a pure black box model. In addition, the PINODE model ensures more physical interpretability.

5. RELATED WORK

Research on modeling of dynamics has a long history. Models can be learned from data, derived from physics, or developed in a hybrid, i. e., semi-physical, manner (Ljung and Glad, 1994). When deriving the model from physics, the dynamics parameters are either estimated or calibrated, often by using least-squares regression. The data-driven models mostly use standard representation learning methods to fit a model in the proximity of the available data. However, this paper follows the line of hybrid modeling, namely, the combination of differential equations with neural networks.

Learning Differential Equations Methods relying on the universal function approximation ability of machine learning methods to solve ordinary or partial differential equations (PDEs) have already been discovered early (Lagaris et al., 1998), but have been rediscovered recently (Raissi et al., 2019; Long et al., 2018). These studies focus on using feed-forward networks to overcome limitations of differential equation solvers by designing the loss function according to the respective equations and taking advantage of the efficient derivative computation in neural networks. Rather than solving the ODEs, our work focuses on structuring the network and modeling the underlying change rate of the physical process.

Differentiating through ODE Solver Much recent work has proposed integrating an ODE solver into the network structure. Chen et al. (2018) propose a general method to parameterize the derivative of the hidden state and then apply an arbitrary ODE solver. Gupta et al. (2019), Greydanus et al. (2019) and Zhong et al. (2020) take such a perspective for mechanical systems and model the derivative of the desired state. An earlier example for this idea, applied in another domain, given by Al Seyab and Cao (2008). Their approach parameterizes dynamic sensitivity equations with a recurrent neural network and then integrates the ODE with Taylor series.

Structuring Learning Problems with Physical Prior A number of studies have suggested endowing neural networks with better physical prior. One concurrent work by Rackauckas et al. (2020) suggests a general semi-mechanistic approach where part of the differential equation is an universal approximator. Their approach shares a similar motivation to ours and also achieves improved data and computational efficiency in diverse examples. Two recent works aim to uncover physical laws from data in a general manner (Iten et al., 2020; Greydanus et al., 2019). More specific for mechanical systems, physics-informed neural networks were demonstrated with simulated time series data for the forward model (1) of a pendulum, double pendulum, and a cart pole system (Gupta et al., 2019; Zhong et al., 2020). Instead of using generalized coordinates, Zhong et al. (2020) use translational coordinates to avoid the problem, that comes with angle data. An example for the application to a simulated and real robot system is given by Lutter et al. (2019). They use a similar approach, but learned the inverse model. Further they also learned the forward model for a physical Furuta pendulum (Lutter

and Peters, 2019). In contrast to our setup, they had measurements for second-order derivatives, which is why it was not necessary to differentiate through an ODE solver.

6. CONCLUSION

The most obvious finding to emerge from this study is that, for the forward model of a real-world physical system, it is possible to integrate existing parts of the equation of motion and model residual physical effects, which are not trivial to capture, by a neural network. Compared with a complete black box approach, the method needs less data, and a certain physical interpretability is retained. We can answer the research question conclusively and demonstrate with PINODE the usage of equations of motion as model prior. Therefore, we can propose a further step toward bridging the gap between purely data-driven and mechanistic models. We have thus taken up the related work by applying and adapting recent methods to develop a forward model for a physical cart pendulum system.

In future works, the technique may be compared with more advanced system identification approaches and studied for its ability to extrapolate. Furthermore one could consider to use different solvers and automatically switch between implicit and explicit methods depending on whether the problem is stiff or non-stiff. Besides that, to further investigate the generality of the approach, additional experiments should be performed and the method adapted for the application to more complex systems, where ODEs or PDEs exist. Other possible extensions to this work may be to use the forward model as environment for learning reinforcement policies (Hein et al., 2018).

ACKNOWLEDGEMENTS

The contribution was supported with funds from the German Federal Ministry of Education and Research within the project "ALICE-III: Autonomous Learning in Complex Environments" under the identification number 01 IS 18049 A.

REFERENCES

Al Seyab, R. and Cao, Y. (2008). Nonlinear system identification for predictive control using continuous time recurrent neural networks and automatic differentiation. *Journal of Process Control*, 18(6), 568–581.

Chen, R.T.Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. (2018). Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 6571–6583.

Fan, J., Ma, C., and Zhong, Y. (2019). A Selective Overview of Deep Learning. *arXiv preprint arXiv:1904.05526 [cs, math, stat]*.

Fernández-Godino, M.G., Park, C., Kim, N.H., and Haftka, R.T. (2016). Review of Multi-Fidelity Models. *arXiv preprint arXiv:1609.07196 [stat]*.

Greydanus, S., Dzamba, M., and Yosinski, J. (2019). Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, 15353–15363.

Gupta, J.K., Menda, K., Manchester, Z., and Kochenderfer, M.J. (2019). A General Framework for Structured Learning of Mechanical Systems. *arXiv preprint arXiv:1902.08705 [cs]*.

Hein, D., Udluft, S., and Runkler, T.A. (2018). Interpretable Policies for Reinforcement Learning by Genetic Programming. *Engineering Applications of Artificial Intelligence*, 76, 158–169.

Iten, R., Metger, T., Wilming, H., Del Rio, L., and Renner, R. (2020). Discovering physical concepts with neural networks. *Physical Review Letters*, 124(1), 010508.

Kingma, D.P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*.

Krizhevsky, A., Sutskever, I., and Hinton, G.E. (2012). Imagenet Classification with Deep Convolutional Neural Networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 25*, 1097–1105. Curran Associates, Inc.

Lagaris, I.E., Likas, A., and Fotiadis, D.I. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5), 987–1000.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521, 436.

Ljung, L. and Glad, T. (1994). *Modeling of Dynamic Systems*. Prentice-Hall, Inc.

Long, Z., Lu, Y., Ma, X., and Dong, B. (2018). PDE-net: Learning PDEs from Data. In *International Conference on Machine Learning (ICML)*.

Lutter, M. and Peters, J. (2019). Deep Lagrangian Networks for end-to-end learning of energy-based control for under-actuated systems. In *International Conference on Intelligent Robots and Systems (IROS)*.

Lutter, M., Ritter, C., and Peters, J. (2019). Deep Lagrangian Networks: Using Physics as Model Prior for Deep Learning. In *International Conference on Learning Representations (ICLR)*.

Rackauckas, C., Ma, Y., Martensen, J., Warner, C., Zubov, K., Supekar, R., Skinner, D., and Ramadhan, A. (2020). Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385 [stat]*.

Raissi, M., Perdikaris, P., and Karniadakis, G.E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378, 686–707.

Rosen, R., von Wichert, G., Lo, G., and Bettenhausen, K.D. (2015). About The Importance of Autonomy and Digital Twins for the Future of Manufacturing. *IFAC-PapersOnLine*, 48(3), 567–572.

Runge, C. (1895). Ueber die numerische Auflösung von Differentialgleichungen. *Mathematische Annalen*, 46(2), 167–178.

von Stosch, M., Oliveira, R., Peres, J., and Feyo de Azevedo, S. (2014). Hybrid semi-parametric modeling in process systems engineering: Past, present and future. *Computers & Chemical Engineering*, 60, 86–101.

Zhong, Y.D., Dey, B., and Chakraborty, A. (2020). Symplectic ODE-Net: Learning Hamiltonian Dynamics with Control. In *International Conference on Learning Representations (ICLR)*.