Learning Hybrid Dynamics Models with Simulator-Informed Latent States

Katharina Ensinger[1,2], **Sebastian Ziesche**[1], **Sebastian Trimpe**[2]

[1] Bosch Center for Artificial Intelligence, Renningen, Germany
[2] Institute for Data Science in Mechanical Engineering, RWTH Aachen Univeristy
katharina.ensinger@bosch.com

Abstract

Dynamics model learning deals with the task of inferring unknown dynamics from measurement data and predicting the future behavior of the system. A typical approach to address this problem is to train recurrent models. However, predictions with these models are often not physically meaningful. Further, they suffer from deteriorated behavior over time due to accumulating errors. Often, simulators building on first principles are available being physically meaningful by design. However, modeling simplifications typically cause inaccuracies in these models. Consequently, hybrid modeling is an emerging trend that aims to combine the best of both worlds. In this paper, we propose a new approach to hybrid modeling, where we inform the latent states of a learned model via a black-box simulator. This allows to control the predictions via the simulator preventing them from accumulating errors. This is especially challenging since, in contrast to previous approaches, access to the simulator's latent states is not available. We tackle the task by leveraging observers, a well-known concept from control theory, inferring unknown latent states from observations and dynamics over time. In our learning-based setting, we jointly learn the dynamics and an observer that infers the latent states via the simulator. Thus, the simulator constantly corrects the latent states, compensating for modeling mismatch caused by learning. To maintain flexibility, we train an RNN-based residuum for the latent states that cannot be informed by the simulator.

1 Introduction

Physical processes $(x_n)_{n=0}^N \in \mathbb{R}^{d_x}$ can often be described via a discrete-time dynamical system

$$x_{n+1} = f(x_n). \tag{1}$$

In practice, the dynamics f are often unknown but measurements are available. Typically, it is not possible to measure the full state space but noisy measurements $\hat{y}_n$ for a function g of the states can be obtained by sensors, thus

$$\begin{aligned} y_n &= g(x_n) \\ \hat{y}_n &= y_n + \epsilon_n, \text{ with } \epsilon_n \sim \mathcal{N}(0, \sigma^2), \end{aligned} \tag{2}$$

and $g : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$. Our general motivation here is to make accurate predictions for the future behavior of the measurable states y_n in Eq. (2). One common strategy to address this problem is to train recurrent architectures on $\hat{y}_n$ in Eq. (2) (Hochreiter and Schmidhuber 1997; Oliva, Póczos, and Schneider 2017). Analogous to the unknown system, they possess internal latent states. Predictions are obtained by mapping the latent states to the measurable states y_n (cf. Eq. (2)) via an observation model. While these methods are able to accurately reflect the system's transitions, they often lack physically meaningful predictions, e.g. by violating physical laws. Further, small model mismatches lead to accumulating errors over time in the latent states and thus, also in the predictions (Zhou et al. 2018).

To address these problems, it is often beneficial to include physical prior knowledge such as energy conservation, invariants etc. in the architecture (Chen et al. 2020; Greydanus, Dzamba, and Yosinski 2019). For many systems, it is even possible to model a physics-based simulator for the system producing stand-alone predictions (Jia et al. 2021; Willard et al. 2022). By design, predictions from these models are physically meaningful. Further, they typically do not suffer as much from error accumulation. However, modeling simplifications or incomplete knowledge lead to inaccuracies in the model. In order to combine the best of both worlds, hybrid modeling (HM) fuses physics-based simulations with learning-based approaches (Takeishi and Kalousis 2021; Wehenkel et al. 2023; Yin et al. 2021). Most hybrid modeling approaches rely on analytical descriptions of physics laws. However, such simulators are often not available in practice. Instead, numerical simulators building on these laws are available that are incompatible with most HM approaches. Further, simulator software is often proprietary. As a consequence, the simulators are often black-box. More specifically, they provide approximations $\hat{s} \in \mathbb{R}^{d_s}$ to the true outputs y (cf. Eq. (2)). Any additional insight as access to the simulator's internal latent states or dynamics is not available.

We propose to extract the hidden information and dynamics of the simulator outputs. By informing the latent states of a learning-based model with this information, we prevent them from error accumulation and ensure that they are consistent with the simulator. In contrast to our approach, previous approaches in the black-box HM case, typically fuse the final predictions to one common prediction. The simplest approach is fusing them additively via a residuum model (Suhartono et al. 2017), while more sophisticated approaches e.g. leverage the spectral properties (Ensinger et al. 2023). Thus, the simulator has an influence on the final predictions

Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

but not on the evolution of the learning-based component. Instead, we directly control the root of accumulating errors in the predictions by correcting the latent states. This approach is also less restrictive than, e.g., a simple residuum model since it requires less informative simulator trajectories.

On a technical level, we inform our latent states by leveraging observers. Observers are commonly used in control systems inferring unknown latent states from observations for a *known* dynamics model over time (Bernard 2019; Bernard, Andrieu, and Astolfi 2022). Here, we propose to leverage them in a HM scenario. In particular, we learn the dynamics for latent states that can explain both, data and simulator via separate observation models. These latent states are informed and controlled via the simulator by learning an observer that receives the simulator outputs as input. Intuitively, the observer constantly minimizes the error between predicted and measured simulator outputs. Thus, it compensates for model mismatch and prevents the latent states from accumulating errors. As a consequence, it also prevents the predictions from accumulating errors, since latent states are mapped to predictions via the observation model. With the observer architecture, we ensure physical behavior to some extent. This is due to the fact that we can only reconstruct latent states that are able to explain the physics-based simulator.

To maintain the flexibility of recurrent architectures, we train an additional recurrent neural network (RNN)-based residuum to address parts of the system, where the simulator is not informative. The influence of both components can be balanced in the loss. Additionally, the learning-based residuum can be controlled, e.g. by forcing it to vanish over time modeling transient behavior. Thus, we obtain full control over all components of the model enabling the simulator to take over as much responsibility over the predictions as desired. In the experiments, we show that our model produces accurate short and long-term predictions and is on par or better than baselines. In particular, it also outperforms a recurrent architecture that receives the simulator as control input (Schön, Götte, and Timmermann 2022; Jia et al. 2021), especially when the simulator is only partially informative. We also derive the differences and advantages with respect to this architecture intuitively and mathematically and support the empirical findings.

Our model has additional advantages. Due to its architecture, system and simulator dynamics are modeled simultaneously. This information can be used to buffer missing simulator outputs. Further, the method can be extended to the pure learning-based scenario. Here, we substitute the simulator with a robust but incomplete learning-based substitute. As in the hybrid case, the substitute prevents the full system from accumulating errors. In summary, our contributions are:

- A new view on HM that aims to maximize the influence of a black-box simulator by splitting the latent dynamics into two parts. One is fully controlled by the simulator, the other can be regularized arbitrarily.
- An observer-based architecture that informs and constantly corrects the latent states of a learning-based model via the simulator. Thus, preventing error accumulation.
- We achieve higher or equal accuracy than learning-based and hybrid baselines.

2 Background: Observer Design

In this work, we leverage observers, a concept from control theory in order to infer the unknown latent states of a learning-based model via a black-box simulator. Here, we provide the necessary background on observer design (Bernard and Andrieu 2019; Brivadis and Ulysse 2019). Consider a system

$$\begin{aligned} u_{n+1} &= f_u(u_n) \\ \hat{s} &= h(u_n), \end{aligned} \tag{3}$$

with dynamics $f_u : \mathbb{R}^{d_u} \to \mathbb{R}^{d_u}$, observation model $h : \mathbb{R}^{d_u} \to \mathbb{R}^{d_s}$ and measurements $\hat{s}$. The latent states u are denoted *observable* in case they can be inferred from measurements $\hat{s}$ over time, regardless of the initial condition, given known dynamics f_u. The algorithm fulfilling this task is denoted observer. In contrast to the standard ML terminology, observable states are *latent* and cannot be measured.

Partially observable systems denote systems, where only parts of the states can be reconstructed from the outputs via an observer (4) (Tami et al. 2016; Röbenack 2006).

Observer: The goal of an observer is to reconstruct the unknown but observable latent states u from outputs $\hat{s}$ and known dynamics. Here, we refer to an observer as a mapping $\mathcal{T}$ that produces an estimate $\tilde{u}$ with $\tilde{u}_{n+1} = \mathcal{T}(\hat{s}_1, \dots, \hat{s}_n, \tilde{u}_0)$ and

$$|u_n - \tilde{u}_n| \to 0, n \to \infty. \tag{4}$$

KKL Observer

In this work, we consider the so-called Kazantzis–Kravaris/Luenberger (KKL) observers due to their beneficial properties such as robustness against model mismatch (Niazi et al. 2022). They can be designed in the continuous-time and discrete-time case (Bernard and Andrieu 2019; Brivadis and Ulysse 2019). In the following, let $\mathcal{U}_0 \subset \mathcal{U} \subset \mathbb{R}^{d_u}$ be a compact subset such that for all initial conditions $u_0 \in \mathcal{U}_0$ and all $n \in \mathbb{N}$ it holds that $u_n \in \mathcal{U}$. We will define the central properties of KKL observers here. Mathematical details are added in Appendix Sec. 1.1. The key criterion for the existence of KKL observers is the so-called backward-distinguishability.

Backward-distinguishability: Intuitively, for distinguishable trajectories in the latent space with $u_0^a \neq u_0^b$, there exists a point $t < 0$ in the past, for which the respective outputs are distinguishable as well, thus $\hat{s}_t^a \neq \hat{s}_t^b$.

Controllable pair: A matrix pair (D, F) with $D \in \mathbb{R}^{d_u \times d_u}$ and $F \in \mathbb{R}^{d_u \times d_s}$ is denoted controllable if the controllability matrix $C = [F, DF, D^2F, \dots, D^{d_u - 1}F]$ has full row rank (Ogata 1997).

KKL observer: Define $d_z = d_y(d_u + 1)$. Consider a backward-distinguishable system with dynamics f_u and observation function h. Furthermore, let (D, F) be a controllable pair, where the eigenvalues $\lambda_1, \dots, \lambda_{d_z}$ of D fulfill $\max(|\lambda_i|) < 1$. Then, under additional mild assumptions, there exists a continuous injective mapping $T : \mathbb{R}^{d_u} \to \mathbb{R}^{d_z}$ with continuous pseudo-inverse $T^\star : \mathbb{R}^{d_z} \to \mathbb{R}^{d_u}$ and

$$T(f_u(u)) = DT(u) + Fh(u) \tag{5}$$

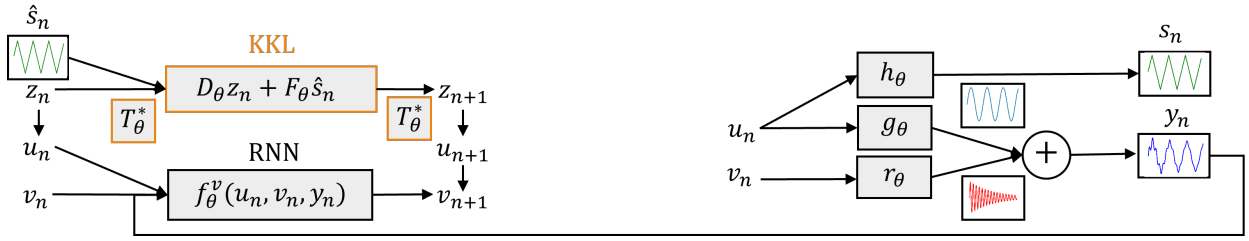

Figure 1: High-level overview of our method. Transition model (left): The simulator signal $\hat{s}$ is fed into a trainable observer to infer the OVS states u. The non-OVS states v are learned by an additional RNN. Observation model (right): The simulator is reconstructed by h_θ, while g_θ and r_θ reconstruct the measurements.

on $\mathcal{U}$ and

$$\lim_{n\to\infty} |u_n - T^\star(z_n)| = 0 \tag{6}$$

for any trajectory defined via $z_{n+1} = Dz_n + F\hat{s}_n$. Thus, $T^\star(z)$ is an observer for u.

Properties and limits: It is not clear how to obtain the transformation T though its existence is guaranteed. Ramos et al. (2020); Peralez and Nadri (2021) approach the problem by sampling trajectories for u and z and solving regression problems for T. However, there is a significant difference since we consider the case of *unknown* dynamics f_u. Still, we benefit from the properties of KKL observers as we will demonstrate in the following sections. We parametrize T with a neural network (NN) similarly to Janny et al. (2021) that consider a learning-based but not a HM scenario.

3 Problem Setting

In this section, we formulate our problem and demonstrate how the concepts in Sec. 2 are related to it. To this end, consider system (1) with outputs y_n (cf Eq.(2)). Our goal is to make accurate predictions for y_n. Here, we consider the situation, where access to an inaccurate physics-based approximation $(\hat{s}_n)_{n=0}^N \in \mathbb{R}^{d_s}$ of the outputs is provided by a black-box simulator. Thus, access to the simulator's latent states or the dynamics is not available. We propose to leverage the concepts in Sec. 2 and learn how to reconstruct latent states from the simulator that are meaningful for the prediction task, thus maximizing the simulator's influence on the predictions. Since the simulator can typically not capture all dynamics, we introduce an additional residuum r.

We model these concepts by splitting the latent state x (cf Eq. (1)) into $u \in \mathbb{R}^{d_u}$ and $v \in \mathbb{R}^{d_v}$, where $d_x = d_u + d_v$. With u, we denote the latent states that can be reconstructed from the simulator and refer to them as being **observable via the simulator (OVS)**. The non-OVS states v on the other hand cannot be reconstructed from the simulator. We propose to formulate a common dynamics model for simulator and data by extending system (7) via

$$\begin{aligned} u_{n+1} &= f_u(u_n) \\ v_{n+1} &= f_v(u_n, v_n) \\ y_n &= g(u_n) + r(u_n, v_n) \\ \hat{s}_n &= h(u_n), \end{aligned} \tag{7}$$

with $f_u : \mathbb{R}^{d_u} \to \mathbb{R}^{d_u}$ and $f_v : \mathbb{R}^{d_u} \times \mathbb{R}^{d_v} \to \mathbb{R}^{d_v}$. Here, f_u determines the propagation of the OVS latent states u, while f_v determines the propagation of the non-OVS latent states v. The observation model g maps the OVS states u to the OVS part of the predictions, while r maps the OVS states u and non-OVS states v to the non-OVS residuum. The simulator is reconstructed via the observation model h. Our goal is to learn f_u, f_v, g, h and r from measurement data $\hat{y}$ and simulator outputs $\hat{s}$.

The key insight is that we force u to be OVS by addressing f_u via a trainable observer that receives $\hat{s}$ as an input. By design, we obtain simulator-informed latent states u and corresponding outputs $g(u)$. In general, we can represent every system via Eq. (7). By setting $d_v = 0$ and removing the non-OVS residuum, we obtain the fully OVS case. By setting $d_u = 0$, our model is reduced to the pure learning-based case. We choose an additive structure for the observation model in order to obtain control over the corresponding non-OVS residuum r. However, different structures are also possible.

4 Method: Hybrid KKL-RNN

Here, we develop a hybrid model that extracts as much information as possible from black-box simulators and maximizes the simulator's influence on the predictions. Thus, addressing the problem stated in Sec. 3. The key component is a trainable KKL observer that informs the OVS latent states of a learning-based model via the simulator. Since predictions are linked to latent states via an observation model, this allows to control the predictions via the simulator. In contrast to the standard setting in control, the dynamics in Eq. (7) are unknown and have to be learned. However, we can still benefit from the properties of KKL observers, in particular, we leverage their robustness against model errors here caused by learning (Niazi et al. 2022). Fig. 1 provides an overview of our method. We model the evolution of the OVS states u (cf. Eq. (7)) by learning a KKL observer. Implicitly, this observer represents the dynamics f_u. This yields a state space that can explain the OVS parts of the data via the observation model g and the simulator via the observation model h. Intuitively, the observer ensures that the estimated simulator outputs $h(u)$ coincide with the true simulator output $\hat{s}$ by constantly correcting the OVS states u. This prevents error accumulation in u and thus, also in the OVS parts of the predictions $g(u)$. In order to obtain a flexible model, we address the non-OVS states by jointly training a residual model f_v between

measurements and the observer-based reconstructions. We balance the influence of both components in the loss.

Fig. 1 also demonstrates the concept of OVS and non-OVS states. Intuitively, the blue signal is OVS with simulator signal (green) since different points on the blue curve always correspond to different points on the green curve. Thus, the blue states are backward-distinguishable via the green states and can be addressed by the trained KKL observer. In contrast, identical outputs on the green curve yield different points on the red curve. Due to the periodicity of the green states, the red states are not backward-distuingishable via the green states. Thus, the red signal is non-OVS and has to be addressed by the RNN residuum.

Architecture: We build a flexible learning scheme based on the partially OVS system (7). The OVS states $u \in \mathbb{R}^{d_u}$ are addressed by a trainable KKL observer. To this end, we consider additional latent states z, that can be transformed into u via $T^\star$ (cf. Sec. 2). The non-OVS latent states $v \in \mathbb{R}^{d_v}$ are addressed by RNN dynamics f_θ^v. Consider training data $\hat{y}_n \in \mathbb{R}^{d_y}$ and simulator outputs $\hat{s}_n \in \mathbb{R}^{d_s}$. This yields

$$\begin{pmatrix} z_{n+1} \\ u_{n+1} \\ v_{n+1} \end{pmatrix} = \begin{pmatrix} D_\theta z_n + F_\theta \hat{s}_n \\ T_\theta^\star(z_n) \\ f_\theta^v(u_n, v_n, y_n) \end{pmatrix}. \tag{8}$$

and corresponding outputs

$$\begin{pmatrix} s_n \\ y_n^v \\ y_n \end{pmatrix} = \begin{pmatrix} h_\theta(u_n) \\ r_\theta(v_n) \\ g_\theta(u_n) + r_\theta(v_n) \end{pmatrix}. \tag{9}$$

Here, θ denotes the trainable parameters. We consider a trainable KKL observer with matrices $D_\theta \in \mathbb{R}^{d_z \times d_z}$, $F_\theta \in \mathbb{R}^{d_z \times d_s}$ and a mapping $T_\theta^\star : \mathbb{R}^{d_z} \to \mathbb{R}^{d_u}$. Furthermore, we consider trainable observation models $h_\theta : \mathbb{R}^{d_u} \to \mathbb{R}^{d_s}$ $g_\theta^u : \mathbb{R}^{d_u} \to \mathbb{R}^{d_y}$ and $r_\theta : \mathbb{R}^{d_v} \to \mathbb{R}^{d_y}$. Here, h_θ^u adresses the simulator, g_θ^u the OVS part of the predictions and r_θ the non-OVS residuum. Due to the KKL structure, we learn the dynamics f_u only implicitly by propagating z through time and mapping to u via $T_\theta^\star$. In case direct access to $f_u(u_n) = u_{n+1} = T_\theta^*(DT_\theta u_n + F\hat{s}_n)$ is required, we provide the option to jointly learn T_θ.

In the experiments, we show that our method can also be leveraged in the pure learning-based scenario. In particular, $\hat{s}$ is replaced by a robust but incomplete learning-based substitute. Similar to the HM scenario, the observer architecture allows the substitute to inform the latent states of a second high-accuracy model, preventing error accumulation.

Models: To respect the requirements in Sec. 2, the matrix D_θ is modeled as a diagonal matrix with trainable bounded eigenvalues, while F_θ is chosen as a matrix with ones as entries. This is not a restriction since a mapping T exists for every controllable pair. In the default scenario, $T_\theta^\star$ is modeled by an MLP similarly to Janny et al. (2021). However, our framework also offers the option to model $T_\theta^\star$ with an invertible NN consisting of affine coupling layers (Dinh, Sohl-Dickstein, and Bengio 2017). The inversion provides access to T_θ and thus, to f_u. The trainable observation models g_θ, h_θ and r_θ are modeled as linear layers. Here, we model the non-OVS part f_θ^v in Eq. (8) with a gated recurrent unit (GRU) (Cho et al. 2014). Details are provided in Appendix Sec. 1.2.

Loss: Consider measurement data $\hat{y}_{0:N}$ and simulator outputs $\hat{s}_{0:N}$. The first R steps $\hat{y}_{0:R}$ are used as a warmup phase for the non-OVS residuum to obtain appropriate latent states. We train our model by computing an N-step rollout $z_{0:N}, y_{0:N}^v$ and $y_{0:N}$ via Eq. (9) and minimizing the loss

$$\hat{\theta} = \arg\min_\theta \|y_{0:N} - \hat{y}_{0:N}\|_2 + \|s_{0:N} - \hat{s}_{0:N}\|_2 + \lambda \|y_{0:N}^v\|_2, \tag{10}$$

where $\|\cdot\|_2$ denotes the MSE. We introduce a regularization factor $\lambda \in \mathbb{R}$ that allows to balance the influence of the learning-based component as it is typical for hybrid models (Takeishi and Kalousis 2021; Yin et al. 2021). This can yield some additional performance. However, it is not mandatory and we do not include it in all experiments. In summary, the loss optimizes for a model that is able to represent the simulator outputs $\hat{s}$ via the observation model h_θ and the measurements $\hat{y}$ via a residuum model $g_\theta(u) + r_\theta(u, v)$.

OVS and non-OVS components: Latent states that are OVS can be informed and corrected via the simulator over time. However, we do not intend to reconstruct the unknown internal states of the simulator as they might not even be informative enough for the data. Instead, OVS states can explain both, the simulator and parts of the data via different observation models. Also, there is no direct physical interpretation of the states. However, since they are extracted by a physics-based simulator and respect backward-distinguishability, unphysical behavior is prohibited to some extent. We will also demonstrate that in the experiments.

In order to maximize the influence of the simulator, we aim to maximize the influence of the OVS part g. The residual structure of the model allows to control the non-OVS counterpart r_θ (cf. Eq. (9)). As an example, it is easily possible to simulate transient behavior via a decaying residuum. This yields a model that is fully driven by the simulator after some time (cf. Eq. (8)) via g. Thus, potential drifts or errors in the RNN do not affect the model performance in the long term. In the experiments, we apply exponential damping by bounding the RNN observation model r_θ with an appropriate activation function and multiplying it with $\exp(-at)$. However, more elaborate strategies such as stable networks (Schlaginhaufen et al. 2021) could be easily incorporated.

Distinction to other architectures: In the experiments, we compare to GRUs that receive the simulator as additional control input (Schön, Götte, and Timmermann 2022; Jia et al. 2021). Under certain conditions, GRUs and other RNNs are a contraction (Bonassi, Farina, and Scattolini 2021; Miller and Hardt 2019). In these cases, they can act as an observer with latent states being driven by the simulator as desired. However, the GRU architecture is not restricted to these favorable models. Further, there is no guarantee that indeed a GRU with the required properties exists for the specific system. For the KKL architecture, on the other hand, the existence is guaranteed if certain requirements described in Sec. 2 hold. By choosing the architecture described in Eq. (8) and Eq. (9), the states u are further OVS and the model is an observer by design under mild assumptions. Additionally, the split in OVS and non-OVS part encourages the system to maximize the influence of the physics-based component. The

GRU architecture, on the other hand, could easily underestimate or even ignore the simulator, destroying the observer property. Mathematical details for the statements are added in Appendix Sec. 1.2. In the experiments, we will show the advantage of the proposed architecture in practice. We will also demonstrate the advantages of addressing the OVS part in the partially OVS system (7) indeed with an observer in contrast to modeling all components with separate GRUs.

Our model can also be interpreted as an extension of the standard residuum model by setting $h_\theta = g_\theta$. We will show empirically that this extension allows to leverage simulator signals that are not informative enough for the standard residuum model and thus cause deteriorated behavior. Intuitively, this is due to the fact that we leverage hidden information from the simulator.

5 Related Work

HM is an emerging trend and a certain type of grey-box modeling. While general grey-box models often respect structural prior knowledge as energy-preservation, invariants etc. (Greydanus, Dzamba, and Yosinski 2019; Geist and Trimpe 2020; Rath, Geist, and Trimpe 2021), HM combines physics-based simulations with data-driven models. Many works consider HM for dynamical systems or time-series. Like us, Linial, Eytan, and Shalit (2020) approach their HM task with RNNs. In particular, they infer the initial states and the parameters of an ODE via an LSTM. However, in contrast to our setting, the system can be fully explained by the simulator with optimized parameters. Another common approach in HM is extending a physics-based dynamics model, e.g. additively, with neural ODEs (Yin et al. 2021; Qian et al. 2021; Quaghebeur, Nopens, and De Baets 2021) or modeling unknown parts of the dynamics with NNs (Su et al. 1992). Rackauckas et al. (2021) present a unified view on these types of models, allowing to jointly learn physical and NN parameters. Recently, variational autoencoders are used to decode the latent states of observations and simulator from data and encode predictions from latent states and the physical model (Takeishi and Kalousis 2021). Existing approaches are extended in (Wehenkel et al. 2023) via a data augmentation concept improving the behavior on unseen data. However, all these models assume direct access to the latent simulator states. In contrast, we consider black-box simulators that provide access only to output trajectories.

However, some approaches consider black-box simulators as well. One branch of HM with black-box simulators deals with optimizing parameters of the simulator (Ruiz, Schulter, and Chandraker 2019; Gutmann and Corander 2016). However, this is not the setting that we consider since the simulator is fully informative once the parameters are adapted. A typical approach in our setting is to learn the errors or residua of simulator predictions and data (Forssell et al. 1997; Suhartono et al. 2017). Similar to our approach, Ensinger et al. (2023) aim to control the long-term behavior of the predictions via the simulator. To this end, they propose a complementary filtering approach. However, in contrast to their approach, our method is not restricted to simulators with correct low-frequency behavior. Furthermore, none of these works informs the latent states of the learning-based component. Another possibility is to provide the simulator as control input to an RNN (Schön, Götte, and Timmermann 2022; Jia et al. 2021). As discussed in detail in Sec. 4, this can easily lead to an underestimation of the simulator.

KKL observers have been combined with learning in different ways. In (Ramos et al. 2020), nonlinear regression via NNs is performed in order to learn the nonlinear transformation of a KKL observer. Buisson-Fenet et al. (2023a) build up on the approach by optimizing the choice of the controllable pair. Peralez and Nadri (2021) propose a learning-based observer design by learning the nonlinear transformation of a KKL observer with autoencoders. In contrast to our approach, all of these works consider observer design for a dynamical system with *known* dynamics. However, some approaches consider KKL observers in the context of dynamics model learning. Buisson-Fenet et al. (2023b) propose a KKL-based recognition model for neural ODEs. The initial latent state is obtained by running a KKL observer forward or backward. In contrast to our setting, the remaining rollout is not observer-based. Furthermore, they do not consider HM. Janny et al. (2021) propose to construct an output predictor via a KKL observer. The framework can be trained similar to standard recurrent architectures. But in contrast to those architectures, mathematical guarantees for the output predictor can be obtained. However, they do not consider HM. In contrast, we leverage the properties of KKL observers in order to control the behavior of the system by informing the latent states via the simulator.

6 Experiments

In this section, we show that: (i) Our KKL-RNN achieves equal or higher accuracy than baselines, especially in the partially OVS case; (ii) We learn a plausible split in OVS and non-OVS components with our method; (iii) Our method can buffer missing simulator inputs; (iv) We can easily incorporate properties as a decaying non-OVS part; (v) The concept can also be leveraged in the pure learning-based scenario.

Baselines: We model the non-OVS residuum in our KKL-RNN with GRU-based architectures. Thus, our model can also be interpreted as a hybrid extension of a GRU. In order to obtain baselines with comparable structure, we consider learning-based and hybrid baselines with a GRU backbone. We compare to the following baselines (for architecture details see Appendix Sec. 3.2). **GRU:** State-of-the-art recurrent architecture for time-series and dynamics learning; **Hybrid GRU:** GRU with simulator trajectory as control input similar to (Jia et al. 2021); **Residual model:** trains GRU predictions r on the residual between data and simulator by minimizing $\|\hat{y} - (\hat{s} + r)\|_2$ similar to Forssell et al. (1997); **Filter:** Fuses long-term information from the simulator with short-term information from a GRU by low-pass filtering the simulator and high-pass filtering the GRU (Ensinger et al. 2023). **Sim:** Physics-based simulator. For the pure learning-based task, we replace the simulator with a learning-based substitute. **Ablation study:** In Appendix Sec. 2.2, we investigate how different aspects of the architecture affect the results. Thus, we consider several strategies to model the partially OVS system (7) with GRUs.

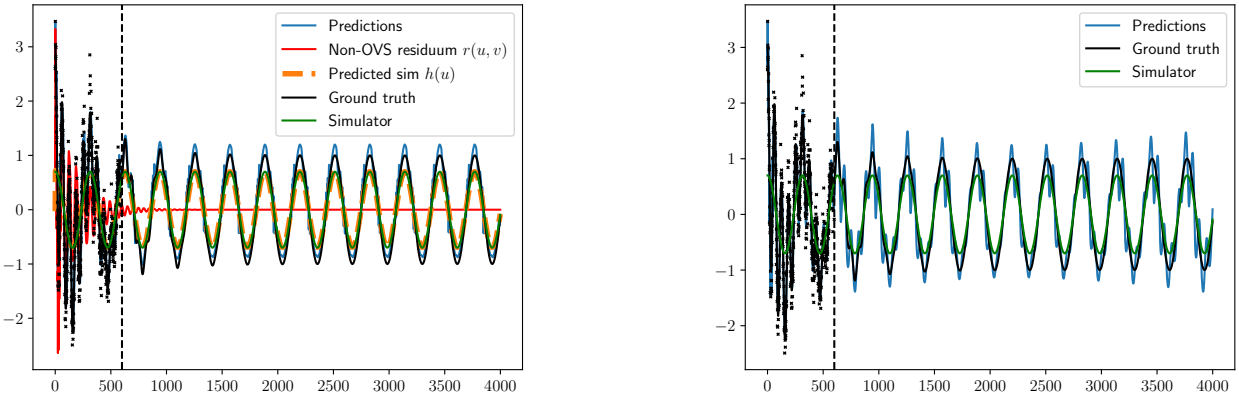

Figure 2: Rollouts over time for system i) with our hybrid KKL-RNN (left) and with the hybrid GRU (right). The training horizon is marked with dotted lines. The results demonstrate that our method reproduces all components correctly and learns a plausible split in OVS and non-OVS. The hybrid GRU shows deteriorated and unphysical long-term behavior.

Learning task: For each experiment, we observe a single trajectory. Rollouts are performed on a short part of the trajectory, while predictions are performed on the full trajectory. Either, we have access to real measurements or we consider simulated data corrupted with noise. For the real-world data, we measure the root-mean-squared error (RMSE) between predictions y and observations $\hat{y}$. For the simulated systems, we compare to the noise-free observations. All models are trained on batches of subtrajectories. Here, we learn $T_\theta^\star$ with an MLP since direct access to f_u is not required (cf. Sec. 4). See Appendix Sec. 3.2 for training details and Sec. 2.3 for invertible NN results.

Systems

For each of the four systems, we focus on different aspects of our method demonstrating that it can easily deal with different non-OVS residua (GRU, exponentially damped GRU) and missing simulator data. For equations of the simulated systems see Appendix Sec. 3.1.

i) Damped oscillations: Two superposed sine oscillations, where the second oscillation is damped and vanishes over time. The simulator is represented by a sine wave with the correct main frequency but a modeling mismatch in the amplitude (see Fig. 2). We enforce a vanishing residuum by exponentially damping the GRU observations r_θ. Thus, after some time, the predictions are solely determined by the observer. The system is trained on a 600-steps interval, predictions are performed with 1500 steps.

ii) Double-torsion pendulum: We consider the measurements and the corresponding numerical simulation from Lisowski et al. (2020) and use the first 150 steps for training. To add a transient component, we artificially add decaying sinusoidal oscillations to the data (see. Fig. 3 (right)). We model the non-OVS residuum r with a standard GRU.

iii) Drill-string system: We train on measurements provided in Aarsnes and Shor (2017) Fig. 14 and the corresponding simulator. 2000 time steps are used for training and 2000 additional time steps for predictions. To demonstrate the robustness of the method, we randomly remove 500 steps from the simulator during the prediction phase.

iv) Van-der-Pol oscillator (pure learning-based): We extend our ideas and results to the pure learning-based scenario. In particular, the observer architecture is leveraged to fuse models with contrastive strengths. Thus, we learn a simplified but robust model as a simulator substitute for the KKL-RNN. As before, we model the non-OVS residua of our KKL-RNN with GRUs. Here, we simulate a Van-der-Pol oscillator with external sine excitation. As a simulator substitute, we consider a simple sine wave with parameters that are trained jointly with the models. Here, the Sim baseline is obtained by fitting the sine wave to the data.

In Appendix Sec. 2.1, we consider a second example, that extends experiment ii) in Ensinger et al. (2023) . As in their setting, a GRU trained on a low-pass filtered signal serves as a simulator substitute. In contrast to them we inform the high-pass components via the low-pass signal.

Results: The results demonstrate that our method produces accurate long and short-term predictions. Also, the simulator is reconstructed accurately (cf. Fig. 2 and 3 (left)). This allows to buffer missing simulator information via the learned simulator signal as demonstrated for System iii). Further, the learned non-OVS residua r indeed correspond to the non-OVS parts of the system, represented e.g. by transient behavior (cf. Fig. 2 and 3 (left)). This property is leveraged for System i), where the hybrid GRU produces deteriorated and unphysical long-term behavior. Such behavior can not occur for the OVS components of our KKL-RNN since it violates backward-distinguishability. The non-OVS transient oscillations are further damped over time by design. Table 1 shows that we achieve higher accuracy than the standard

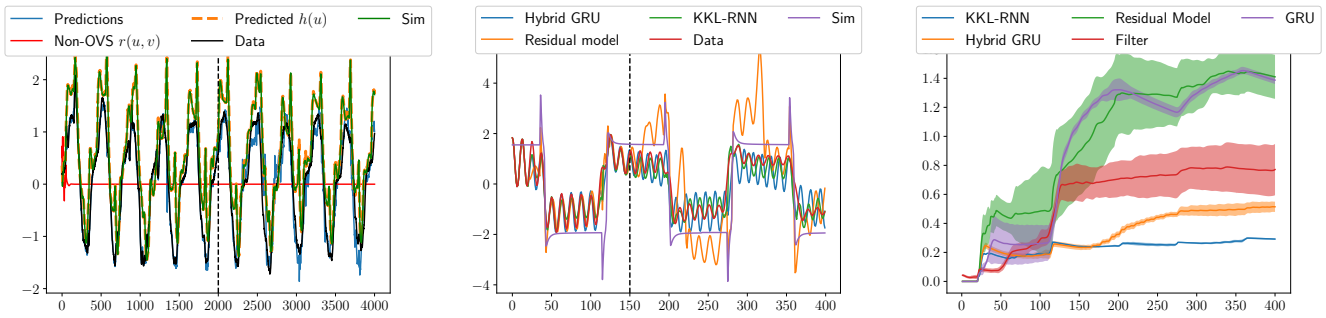

Figure 3: The rollouts with our KKL-RNN for System iii) (left) show that our method learns a plausible split into OVS and non-OVS and is able to buffer missing simulator inputs. The rollouts for System ii) (middle) demonstrate that our KKL-RNN produces more accurate results than the other HM approaches. The residual model even shows unphysical behavior. Accumulating errors in the baseline are further visible in the RMSE over time for System (ii) (right).

task	GRU	Residual Model	Hybrid GRU	Filter	KKL-RNN (ours)	Sim
i)	1.02 (0.03)	0.31 (0.04)	0.27 (0.08)	0.64 (0.21)	**0.16** (0.03)	0.36
ii)	1.39 (0.03)	1.41 (0.15)	0.51 (0.04)	0.61 (0.09)	**0.24** (0.02)	1.09
iii)	0.40 (0.19)	0.63 (0.10)	**0.26** (0.01)	0.60 (0.01)	**0.23** (0.01)	0.66
iv)	0.43 (0.19)	0.51 (0.15)	0.09 (0.065)	-	**0.02** (0.001)	0.537

Table 1: RMSEs for Systems i)-iii) (mean (std)) over 5 independent runs.

GRU, which suffers from deteriorated long-term behavior on all systems. Further, our method is also superior to the Residual model and Filter (cf. Fig. 3 (middle)) since, in contrast to these approaches, the simulator informs the latent states of the learning-based component. This prevents unphysical behavior as it occurs in the Residual model (cf. Fig. 3 (middle)) and allows to leverage less informative simulators. As explained, informing the latent states further prevents error accumulation. The Filter also prevents error accumulation to some extent by adopting the long-term behavior of the simulator. However, it relies on the assumption that the simulator provides the correct low-frequency information, which is not the case for the systems here (e.g. the simulator has the wrong amplitude in System (i)). The accumulating errors in the baselines are further clearly visible in the accumulating RMSE over time (cf. Fig. 3 (right)). Often, the accuracy of our method is similar to the hybrid GRU (cf. System iii). It can be interpreted that in these cases, the GRU acts as an observer as explained in Sec. 4. However, for Systems i), ii) and iv), our method provides higher accuracy than the hybrid GRU. A likely explanation is that, in contrast to our method, the hybrid GRU does not learn an optimal split into OVS and non-OVS components especially if the system is not fully OVS. This coincides with the findings that the GRU could ignore or underestimate the simulator input (cf. Sec. 4).

System iv) shows that the findings extend to the pure learning-based scenario. Hybrid GRU and KKL-RNN outperform the other methods. Intuitively, both learn a sine wave with matching frequency. Further, they learn to infer the Van-der-Pol oscillator via it. However, the KKL-RNN still outperforms the hybrid GRU that is not able to perfectly reproduce the oscillations. This indicates again that the hybrid GRU does not learn a fully-OVS system. However, the results for the hybrid GRU can be leveraged. They demonstrate that a standard GRU can be prevented from deteriorating long-term predictions by supporting it with a simple trainable signal. Similar findings for another pure learning-based experiment are provided in Appendix Sec. 2.1.

The ablation study in Appendix Sec. 2.2 demonstrates that the GRU architectures trained on Eq. (7) do not learn an optimal split into OVS and non-OVS and lack high accuracy. In particular, the non-OVS residuum takes over parts that are actually OVS. Since they are also trained on the partially OVS system (7), this suggests that the KKL observer is an essential component of our architecture. in Appendix Sec. 2.4, we provide additional plots, runtimes and plot the RMSE over time, indicating again accumulating errors in the baselines.

7 Conclusion

We propose a hybrid modeling scheme that allows to extract hidden information from black-box simulators by informing the latent states of a learning-based model. To this end, we train a KKL observer that infers OVS latent states via the simulator. The OVS states and corresponding predictions are thus constantly controlled by the simulator, preventing them from error accumulation. Interesting aspects for future work are the extension to the stochastic setting or to different observers and partially observable forms.

Acknowledgements

The authors thank Barbara Rakitsch and Mona Buisson-Fenet for valuable discussions.

References

Aarsnes, U. J.; and Shor, R. 2017. Torsional vibrations with bit off bottom: Modeling, characterization and field data validation. *Journal of Petroleum Science and Engineering*, 163.

Bernard, P. 2019. Observer Design for Nonlinear Systems. In *Lecture Notes in Control and Information Sciences, volume 479. Springer International Publishing*.

Bernard, P.; and Andrieu, V. 2019. Luenberger Observers for Nonautonomous Nonlinear Systems. *IEEE Transactions on Automatic Control*, 64(1): 270–281.

Bernard, P.; Andrieu, V.; and Astolfi, D. 2022. Observer design for continuous-time dynamical systems. *Annual Reviews in Control*, 53: 224–248.

Bonassi, F.; Farina, M.; and Scattolini, R. 2021. On the stability properties of Gated Recurrent Units neural networks. *Systems And Control Letters*, 157: 105049.

Brivadis, A. V., Lucas; and Ulysse, S. 2019. Luenberger observers for discrete-time nonlinear systems. In *58th IEEE Conference on Decision and Control, (CDC)*.

Buisson-Fenet, M.; Bahr, L.; Morgenthaler, V.; and Meglio, F. D. 2023a. Towards Gain Tuning for Numerical KKL Observers. *IFAC-PapersOnLine*, 56(2): 4061–4067. 22nd IFAC World Congress.

Buisson-Fenet, M.; Morgenthaler, V.; Trimpe, S.; and Meglio, F. D. 2023b. Recognition Models to Learn Dynamics from Partial Observations with Neural ODEs. *Transactions on Machine Learning Research*.

Chen, Z.; Zhang, J.; Arjovsky, M.; and Bottou, L. 2020. Symplectic Recurrent Neural Networks. In *International Conference on Learning Representations*.

Cho, K.; van Merrienboer, B.; Gülçehre, Ç.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, 1724–1734.

Dinh, L.; Sohl-Dickstein, J.; and Bengio, S. 2017. Density estimation using Real NVP. In *5th International Conference on Learning Representations, ICLR*.

Ensinger, K.; Ziesche, S.; Rakitsch, B.; Tiemann, M.; and Trimpe, S. 2023. Combining Slow and Fast: Complementary Filtering for Dynamics Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(6): 7476–7484.

Forssell, U.; Lindskog, P.; Forssell, U.; and Lindskog, P. 1997. Combining Semi-Physical and Neural Network Modeling: An Example of Its Usefulness. In *the 11th IFAC Symposium on System Identification (SYSID'97)*, 795–798.

Geist, A. R.; and Trimpe, S. 2020. Learning Constrained Dynamics with Gauss Principle adhering Gaussian Processes. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research (PMLR)*, 225–234.

Greydanus, S.; Dzamba, M.; and Yosinski, J. 2019. Hamiltonian Neural Networks. In *Advances in Neural Information Processing Systems*, volume 32.

Gutmann, M. U.; and Corander, J. 2016. Bayesian Optimization for Likelihood-Free Inference of Simulator-Based Statistical Models. *Journal of Machine Learning Research*, 17(125): 1–47.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation*, 9(8): 1735–1780.

Janny, S.; Andrieu, V.; Nadri, M.; and Wolf, C. 2021. Deep KKL: Data-driven Output Prediction for Non-Linear Systems. In *2021 60th IEEE Conference on Decision and Control (CDC)*, 4376–4381.

Jia, X.; Willard, J.; Karpatne, A.; Read, J. S.; Zwart, J. A.; Steinbach, M.; and Kumar, V. 2021. Physics-Guided Machine Learning for Scientific Discovery: An Application in Simulating Lake Temperature Profiles. *ACM/IMS Trans. Data Sci.*, 2(3).

Linial, O.; Eytan, D.; and Shalit, U. 2020. Generative ODE modeling with known unknowns. *Proceedings of the Conference on Health, Inference, and Learning*.

Lisowski, B.; Retiere, C.; Moreno, J.; and Olejnik, P. 2020. Semiempirical identification of nonlinear dynamics of a two-degree-of-freedom real torsion pendulum with a nonuniform planar stick–slip friction and elastic barriers. *Nonlinear Dynamics*, 100: 3215–3234.

Miller, J.; and Hardt, M. 2019. Stable Recurrent Models. In *International Conference on Learning Representations*.

Niazi, M. U. B.; Cao, J.; Sun, X.; Das, A.; and Johansson, K. H. 2022. Learning-based Design of Luenberger Observers for Autonomous Nonlinear Systems. *Preprint arXiv:2210.01476*.

Ogata, K. 1997. *Modern Control Engineering*. Number Bd. 1 in Prentice Hall International Editions.

Oliva, J. B.; Póczos, B.; and Schneider, J. 2017. The Statistical Recurrent Unit. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 2671–2680.

Peralez, J.; and Nadri, M. 2021. Deep Learning-based Luenberger observer design for discrete-time nonlinear systems. In *2021 60th IEEE Conference on Decision and Control (CDC)*, 4370–4375.

Qian, Z.; Zame, W. R.; Fleuren, L. M.; Elbers, P.; and van der Schaar, M. 2021. Integrating Expert ODEs into Neural ODEs: Pharmacology and Disease Progression. In *Advances in Neural Information Processing Systems*, volume 34.

Quaghebeur, W.; Nopens, I.; and De Baets, B. 2021. Incorporating Unmodeled Dynamics Into First-Principles Models Through Machine Learning. *IEEE Access*, 9: 22014–22022.

Rackauckas, C.; Ma, Y.; Martensen, J.; Warner, C.; Zubov, K.; Supekar, R.; Skinner, D.; Ramadhan, A.; and Edelman, A. 2021. Universal Differential Equations for Scientific Machine Learning. *Preprint arXiv:2001.04385*.

Ramos, L. d. C.; Di Meglio, F.; Morgenthaler, V.; da Silva, L. F. F.; and Bernard, P. 2020. Numerical design of Luenberger observers for nonlinear systems. In *2020 59th IEEE Conference on Decision and Control (CDC)*, 5435–5442.

Rath, L.; Geist, A. R.; and Trimpe, S. 2021. Using Physics Knowledge for Learning Rigid-Body Forward Dynamics with Gaussian Process Force Priors. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, 101–111.

Röbenack, L. A., Klaus. 2006. Observer design using a partial nonlinear observer canonical form. *International Journal of Applied Mathematics and Computer Science*, 16(3): 333–343.

Ruiz, N.; Schulter, S.; and Chandraker, M. 2019. Learning To Simulate. In *7th International Conference on Learning Representations, ICLR 2019*.

Schlaginhaufen, A.; Wenk, P.; Krause, A.; and Dörfler, F. 2021. Learning Stable Deep Dynamics Models for Partially Observed or Delayed Dynamical Systems. In *Neural Information Processing Systems*.

Schön, O.; Götte, R.-S.; and Timmermann, J. 2022. Multi-Objective Physics-Guided Recurrent Neural Networks for Identifying Non-Autonomous Dynamical Systems. *IFAC-PapersOnLine*, 55(12): 19–24. 14th IFAC Workshop on Adaptive and Learning Control Systems ALCOS 2022.

Su, H.-T.; Bhat, N.; Minderman, P.; and McAvoy, T. 1992. Integrating Neural Networks with First Principles Models for Dynamic Modeling. *IFAC Proceedings Volumes*, 25(5): 327–332.

Suhartono; Rahayu, S.; Prastyo, D.; Wijayanti, D.; and Juliyanto. 2017. Hybrid model for forecasting time series with trend, seasonal and salendar variation patterns. *Journal of Physics: Conference Series*, 890: 012160.

Takeishi, N.; and Kalousis, A. 2021. Physics-Integrated Variational Autoencoders for Robust and Interpretable Generative Modeling. In *Advances in Neural Information Processing Systems*.

Tami, R.; Zheng, G.; Boutat, D.; Aubry, D.; and Wang, H. 2016. Partial observer normal form for nonlinear system. *Automatica*, 64: 54–62.

Wehenkel, A.; Behrmann, J.; Hsu, H.; Sapiro, G.; Louppe, G.; and Jacobsen, J.-H. 2023. Robust Hybrid Learning With Expert Augmentation. *Transactions on Machine Learning Research*.

Willard, J.; Jia, X.; Xu, S.; Steinbach, M.; and Kumar, V. 2022. Integrating Scientific Knowledge with Machine Learning for Engineering and Environmental Systems. *ACM Comput. Surv.*, 55(4).

Yin, Y.; Guen, V. L.; Dona, J.; de Bézenac, E.; Ayed, I.; Thome, N.; and Gallinari, P. 2021. Augmenting Physical Models with Deep Networks for Complex Dynamics Forecasting. In *9th International Conference on Learning Representations, ICLR 2021*.

Zhou, Y.; Li, Z.; Xiao, S.; He, C.; Huang, Z.; and Li, H. 2018. Auto-Conditioned Recurrent Networks for Extended Complex Human Motion Synthesis. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada*.