Combining Slow and Fast: Complementary Filtering for Dynamics Learning

Katharina Ensinger[1,2], Sebastian Ziesche[1], Barbara Rakitsch[1], Michael Tiemann[1], Sebastian Trimpe[2]

[1] Bosch Center for Artificial Intelligence, Renningen, Germany
[2] Institute for Data Science in Mechanical Engineering, RWTH Aachen University
katharina.ensinger@bosch.com, firstname.surname@de.bosch.com, trimpe@dsme.rwth-aachen.de

Abstract

Modeling an unknown dynamical system is crucial in order to predict the future behavior of the system. A standard approach is training recurrent models on measurement data. While these models typically provide exact short-term predictions, accumulating errors yield deteriorated long-term behavior. In contrast, models with reliable long-term predictions can often be obtained, either by training a robust but less detailed model, or by leveraging physics-based simulations. In both cases, inaccuracies in the models yield a lack of short-time details. Thus, different models with contrastive properties on different time horizons are available. This observation immediately raises the question: *Can we obtain predictions that combine the best of both worlds?* Inspired by sensor fusion tasks, we interpret the problem in the frequency domain and leverage classical methods from signal processing, in particular complementary filters. This filtering technique combines two signals by applying a high-pass filter to one signal, and low-pass filtering the other. Essentially, the high-pass filter extracts high-frequencies, whereas the low-pass filter extracts low frequencies. Applying this concept to dynamics model learning enables the construction of models that yield accurate long- and short-term predictions. Here, we propose two methods, one being purely learning-based and the other one being a hybrid model that requires an additional physics-based simulator.

1 Introduction

Many physical processes $(x_n)_{n=0}^N$ with $x_n \in \mathbb{R}^{D_x}$ can be described via a discrete-time dynamical system

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

Typically, it is not possible to measure the whole state-space of the system (1), but a function of the states corrupted by noise $\hat{y}_n$ can, for example, be measured by sensors

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

and $C \in \mathbb{R}^{D_y \times D_x}$. Our general interest is to make accurate predictions for the observable components y_n in Eq. (2). One possible way to address this problem is training a recurrent model on the noisy measurements $\hat{y}_n$ in Eq. (2). Learning-based methods are often able to accurately reflect the system's behavior and therefore produce accurate short-term predictions. However, the errors accumulate over time leading to deteriorated long-term behavior (Zhou et al. 2018).

To obtain reliable prediction behavior on each time scale, we propose to decompose the problem into two components. In particular, we aim to combine two separate models, where one component reliably predicts the long-term behavior, while the other adds short-term details, thus combining the strengths of each component. Interpreted in the frequency domain, one model tackles the low-frequency components while the other tackles the high-frequency parts.

Combining high and low-frequency information from different signals or models is well-known from control engineering or signal processing tasks. One typical example is tilt estimation in robotics, where accelerometer and gyroscope data are often available simultaneously (Trimpe and D'Andrea 2010; Geist et al. 2022). On one hand, the gyroscope provides position estimates that are precise on the short-term but due to integration in each time step, accumulating errors cause a drift on the long-term. On the other hand, the accelerometer-based position estimates are long-term stable, but considerably noisy and thus not reliable on the short-term. Interpreted in the frequency domain, the gyroscope is more reliable on high frequencies, whereas the accelerometer is more reliable on low frequencies. Therefore, a high-pass filter is applied to the gyroscope measurements, whereas a low-pass filter is applied to the accelerometer measurements. Both filtered components are subsequently combined in a new complementary filtered signal that is able to approximate the actual position more accurately.

Here, we adopt the concept of complementary filter pairs to our task to fuse models with contrastive properties. In general, a complementary filter pair consists of a high-pass filter H and a low-pass filter L, where the filters map signals to signals. Depending on the specific filter, certain frequencies are eliminated while others pass. Intuitively the joint information of both filters in a complementary filter pair covers the whole frequency domain. Thus, the key concept that we leverage here is the decomposition of a signal $y = (y_n)_{n=0}^N$ into a high-pass filter component $H(y)$ and a low-pass filter component $L(y)$ via

$$y = H(y) + L(y). \tag{3}$$

Based on the decomposition, we propose to address $H(y)$ and $L(y)$ by different models that are reliable on their specific time scale. *In particular, we propose two methods, one being purely-learning based and one being a hybrid method that leverages an additional physics-based simulation.* Both concepts are visualized in Figure 1. In the purely learning-based scenario, we train seperate networks that represent $H(y)$ and $L(y)$ in Eq. (3). In order to obtain a low-frequency model that indeed provides accurate long-term predictions, we apply a downsampling technique to the training data, thus reducing the number of integration steps. During inference, the predictions are upsampled up to the original sampling rate. Applying the low-pass filter allows lossless downsampling of the signal depending on the downsampling ratio.

In the hybrid scenario, only a single model is trained. Hybrid modeling addresses the problem of producing predictions by mixing different models that are either learning-based or obtained from first principles, e.g. physics (Yin et al. 2021; Suhartono et al. 2017). Here, we consider the case where access to predictions y^{s} for the system (1) is provided by a physics-based simulator. Additional insights, such as access to the simulator's latent space or differentiability are not given. While physics-based approaches are typically robust and provide reliable long-term behavior, incomplete knowledge of the underlying physics leads to short-term errors in the model. Hence, we consider the case where $L(y^{\mathrm{s}}) \approx L(y)$ holds. By training a model for $H(y)$, the decomposition (3) becomes a hybrid model that combines the strengths of both components. The filter pair (L, H) is integrated into the training process, assuring that the long-term behavior is indeed solely addressed by the simulator. In both scenarios, the learning-based and the hybrid, recurrent neural networks (RNNs) are trained on whole trajectories.

In summary, the main contributions of this paper are:

- By leveraging complementary filters, we propose a new view on dynamics model learning;
- we propose a purely learning-based and a hybrid method that decompose the learning problem into a long-term and a short-term component; and
- we show that this decomposition allows for training models that provide accurate long and short-term predictions.

2 Related Work

In this section, we give an overview of related literature.

Several works point out parallels between classical signal-theoretic concepts and neural network architectures. In particular, connections to finite-impulse response (FIR) and infinite-impulse response (IIR) filters have been drawn. The relations between these filters and feedforward-models have been investigated in Back and Tsoi (1991). Precisely, they construct different feedforward architectures by building synapses from different filters. Depending on the specific type, locally recurrent but globally feedforward structure can be obtained. These models are revisited in Campolucci, Uncini, and Piazza (1996) by introducing a novel backpropagation technique. More recently, feedforward Sequential Memory Networks are introduced, which can be interpreted as FIR filters (Zhang et al. 2016). Relations between fully recurrent models and filters have been drawn as well. The hidden structure of many recurrent networks can be identified with classical filters. Kuznetsov, Parker, and Esqueda (2020) point out the relation between Elman-Networks and filters and introduce trainable IIR structures that are applied to sound signals in the experiments section. Precisely, an Elman network can be interpreted as simple first-order IIR filter. In Oliva, Póczos, and Schneider (2017), long-term dependencies are modeled via a moving average in the hidden units. Moving averages can again be interpreted as special FIR filters. Stepleton et al. (2018) recover long-term dependencies via a hidden structure of memory pools that consist of first-order IIR filters. However, none of this works leverages complementary filters in order to capture effects on multiple time scales. Additionally, none of these approaches addresses hybrid dynamics models. Narkhede et al. (2019); Ćertić and Milić (2011); Milić and Saramaki (2003) combine learning techniques and in particular gradient-descent with complementary filters. However, they consider the automatical adaption of the filter parameters. In contrast, we leverage complementary filters for learning, in particular dynamics learning.

Filters manipulate signals on the frequency domain and thus address spectral properties. In Proctor, Brunton, and Kutz (2016) and Lange, Brunton, and Kutz (2021), a signal is identified via spectral methods that are transformed into a linear model. Koopman theory is then leveraged to lift the system to the nonlinear space again. However, in our work, we use filters in order to separate the predictions on different time-horizons. Thus, in contrast to these works, our methods can be combined with different (recurrent) architectures and therefore allow for computing predictions via state-of-the-art techniques.

Combining physics-based simulators with learning-based models is an emerging trend. Hybrid models produce predictions by taking both models into account. Typically, the simulator is extended or parts of the simulator are replaced. There is a vast literature that deals with hybrid models for dynamical systems or time-series data. A traditional approach is learning the errors or residua of simulator predictions and data (Forssell et al. 1997; Suhartono et al. 2017). Another common approach in hybrid modeling is extending a physics-based dynamics model with neural ODEs (Yin et al. 2021; Qian et al. 2021). However, in contrast to our approach, these hybrid architectures do not explicitly exploit characteristics of the simulator, in particular the long-term behavior. Paolucci et al. (2018) construct a hybrid model for the prediction of seismic behavior. Similar to our setting, they consider the case where a physics-based simulation provides reliable predictions for low frequencies, whereas lacking of a reliable model for high frequencies. However, the approach differs significantly from ours since a neural network is trained on a mapping from low to high frequencies. Furthermore, they do not consider dynamics models. Therefore, it is unclear how to apply the approach to our problem setting.

3 Background

In this section, we provide the necessary background on signal processing and filtering. For a more detailed introduction, we refer the reader to Oppenheim et al. (1999).

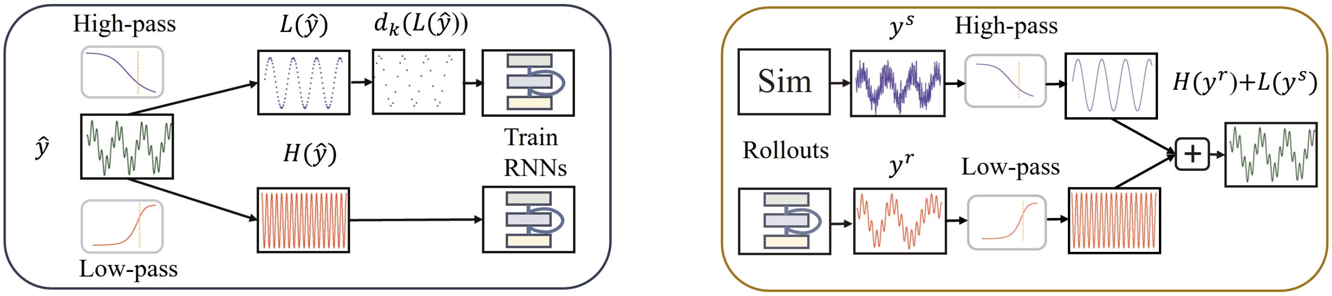

Figure 1: A high-level overview of our methods. Purely-learning-based scheme (left): a training signal is filtered into complementary components. The low-pass filtered signal is downsampled. Two seperate RNNs are trained on the decomposed signal. Hybrid model (right): The predictions of simulator and RNN are fed into the complementary filter. The resulting signal is trained end-to-end on the noisy observations by minimizing the root mean-squared error (RMSE). This structure is also applied to obtain predictions from the model.

3.1 Motivation

Filters are linear time-invariant systems that aim to extract specific frequency components from a signal. Standard types are high-pass and low-pass filters. Low-pass filters extract low frequencies and attenuate high frequencies, whereas high-pass filters extract high frequencies and attenuate low frequencies. Frequencies that are allowed to pass are determined by a desired cutoff frequency. Further, additional specifications play a principal role in filter design, such as pass- and stop-band fluctuations and width of the transition band (Oppenheim et al. 1999).

Technically, a filter F is a mapping in the time domain $F : l^\infty \to l^\infty : y \mapsto Z^{-1}(\mathcal{F}(Z(y)))$, where $\mathcal{F} : \mathbb{C} \to \mathbb{C}$ is the so-called transfer function in the frequency domain, l^∞ is the signal space of bounded sequences and $Z : l^\infty \to \mathbb{C}$ is the well-known z-transform. Hence, a filter is obtained by designing a transfer function $\mathcal{F}$ in the frequency domain. For the type of filters considered here, the structure of $\mathcal{F}$ allows to directly compute $F(y)$ via a recurrence equation in the time domain (see the appendix for more details). A typical application of filters is, for example, the denoising of signals. Noise adds a high-frequency component to the signal and can therefore be tackled by applying a low-pass filter.

3.2 IIR-Filter

Typical filter types are finite-impulse response (FIR) and infinite-impulse response (IIR) (Oppenheim et al. 1999). Here, we consider IIR filters. In contrast to FIR filters, IIR filters possess internal feedback. Filtering a signal y via an IIR-filter yields a recurrence equation for the filtered signal $\tilde{y} = (\tilde{y}_n)_{n=0}^N$ given by

$$\tilde{y}_n = \frac{1}{a_0}\left(\sum_{k=1}^{P} a_k \tilde{y}_{n-k} + \sum_{k=0}^{P} b_k y_{n-k}\right), \tag{4}$$

where P describes the filter order. The filter coefficients a_k and b_k are obtained from filter design with respect to the desired properties in the frequency domain. A detailed derivation is given in the appendix. There are different strategies to initialize the first P values $\tilde{y}_0, \dots, \tilde{y}_{P-1}$ (Chornoboy 1992; Gustafsson 1996).

3.3 Complementary Filter Pairs

A complementary filter pair consists of a high-pass filter transfer function $\mathcal{H}$ and a low-pass filter transfer function $\mathcal{L}$ (Higgins 1975), chosen in a way that they cover the whole frequency domain, thus

$$y \approx L(y) + H(y) \tag{5}$$

for any signal $y \in l^\infty$. Applying the complementary filter pair to two different signals y^{h} and y^{l} via $\tilde{y} = L(y^{\mathrm{l}}) + H(y^{\mathrm{h}})$ directly yields a recurrence equation for the complementary filtered signal $\tilde{y}$ given by

$$\tilde{y}_n = \frac{1}{a_0}\left(\sum_{k=1}^{P} a_k \tilde{y}_{n-k} + \sum_{k=0}^{P} b_k y^{\mathrm{h}}_{n-k} + \sum_{k=0}^{P} \tilde{b}_k y^{\mathrm{l}}_{n-k}\right), \tag{6}$$

where a_k, b_k describe the high-pass filter parameters and $a_k, \tilde{b}_k$ describe the low-pass filter parameters. To obtain a joint recurrence equation, the filters are forced to share the parameters a_k. However, this can be done without loss of generality.

Perfect Complement: There are different strategies to express the decomposition (5) mathematically. One way is to construct the perfect complement in the frequency domain such that $\mathcal{H} + \mathcal{L} = 1$ (Narkhede et al. 2021). Applying the perfect complementary filter to two identical signals $y^{\mathrm{h}} = y^{\mathrm{l}}$ results in the same signal as output. For the IIR complementary filter (6) this holds if $\tilde{b}_k = a_k - b_k$. A detailed derivation is moved to the appendix. However, depending on the desired behavior of the filters, perfectly complementary filters are not always favorable. Different approaches have been investigated in Vaidyanathan, Regalia, and Mitra (1987); Johansson and Saramäki (1999).

4 Method

We present two methods that leverage the idea of complementary filters for dynamics model learning in order to produce accurate short- and long-term predictions. Our first approach is applicable to general dynamics model learning, whereas our second approach is a hybrid modeling technique. In the second case, access to trajectory data produced by a physics-based simulator is required. The key ingredient of both models is a complementary filter pair (H, L) with parameters $a_k, b_k, \tilde{a}_k$ and $\tilde{b}_k$ (cf. Sec. 3.3). While in the hybrid case reliable long-term predictions are already provided by the simulator, the long-term predictions have to be addressed by an additional model in the purely learning-based scenario.

4.1 Recurrent Dynamics Model Learning

First, we give an overview of the recurrent dynamics model learning structure that serves as a backbone for our method. Here, we consider a recurrent multilayer perceptron (MLP) and a gated recurrent unit (GRU) model (Cho et al. 2014). However, the method is not restricted to that choice and could be combined with other recurrent architectures such as Hochreiter and Schmidhuber (1997); Doerr et al. (2018). Consider a trainable neural network transition function $f_\theta : \mathbb{R}^{D_h \times D_y} \to \mathbb{R}^{D_h}$ and a linear observation model $C_\theta \in \mathbb{R}^{D_y \times D_h}$. Here, θ defines the trainable parameters and h the latent states with corresponding latent dimension D_h. Predictions are computed via

$$\begin{aligned} h_{n+1} &= f_\theta(h_n, y_n) \\ y_n &= C_\theta h_n, \end{aligned} \tag{7}$$

where the initial hidden state h_0 can be obtained from the past trajectory by training a recognition model similar to Doerr et al. (2018) or by performing a warmup phase. Details are provided in the appendix. The mapping $F_\theta : \mathbb{R}^{D_h \times D_y} \times \mathbb{N} \to \mathbb{R}^{D_y \times N}$ that computes an N-step rollout via Eq. (7) reads

$$F_\theta(h_0, y_0, N) = y_{0:N}, \tag{8}$$

where $y_{0:N} \in \mathbb{R}^{D_y \times N}$ defines a trajectory with N steps.

4.2 Purely Learning-Based Model

Next, we dive into the details of constructing complementary filter-based learning schemes and introduce our methods. In the purely learning-based scenario, two different models are trained, wherein one model addresses the high-frequency parts and the other addresses the low-frequency parts (see Figure 1 (left).) To this end, the training signal $\hat{y}$ is decomposed into a high-frequency component $H(\hat{y})$ and a low-frequency component $L(\hat{y})$ via the complementary filter pair (cf. Sec. 3.3). The models are trained separately on the decomposition. In order to obtain a model that indeed provides stable long-term behavior, the low-frequency training data is downsampled. During inference, the predicted signal is upsampled again. Downsampling yields a model that performs less integration steps and thus, produces less error accumulation. As an additional advantage, backpropagation through less integration steps is computationally more efficient. Applying the low-pass filter allows lossless downsampling up to a specific ratio that is determined by the Nyquist frequency. Intuitively, only high-frequency information is removed that is addressed by the second network during training and inference. Details are provided in the appendix. Splitting the training signal and training the models separately ensures that one model indeed addresses the low-frequency part of the signal and thus, the long-term behavior. End-to-end training on the other hand might yield deteriorated long-term behavior since it generally allows a single network to tackle both- short, and long-term behavior.

Up and Downsampling: The downsampling operation $d_k : \mathbb{R}^{D_y \times N} \to \mathbb{R}^{D_y \times \lfloor N/k \rfloor}$ maps a signal to a lower resolution by considering every k^{th} step of the signal via

$$d_k(y_{0:N}) = (y_0, y_k, \ldots, y_{k\lfloor N/k \rfloor}). \tag{9}$$

The reverse upsampling operation $u_k : \mathbb{R}^{D_y \times N} \to \mathbb{R}^{D_y \times kN}$ maps a signal to a higher resolution by filling in the missing data without adding high-frequency artifacts to the signal. Mathematically, this corresponds to an interpolation problem (Oppenheim et al. 1999). Here, we consider lossless downsampling, where tolerable downsampling ratios are determined by the cutoff frequency of the low-pass filter.

Training: Consider training data $\hat{y}_{0:N} \in \mathbb{R}^{D_y \times N}$ from which the first $R < N$ steps $\hat{y}_{0:R} \in \mathbb{R}^{D_y \times R}$ are used to obtain an appropriate initial hidden state. We consider trainable models $f_\theta^{\mathrm{h}}, C_\theta^{\mathrm{h}}, f_\nu^{\mathrm{l}}, C_\nu^{\mathrm{l}}$ with corresponding rollout mappings F_θ^{h} and F_ν^{l} (cf. Eq. (8)), an up/downsampling ratio k and a complementary filter pair L, H (cf. Sec. 3.3). The weights θ and ν are trained by minimizing the root-mean-squared error (RMSE) $\|y - \hat{y}\|_2$ via

$$\begin{aligned} \hat{\theta} &= \arg\min_\theta \|H(\hat{y})_{R:N} - F_\theta^{\mathrm{h}}(\hat{y}_R^{\mathrm{h}}, h_R^{\mathrm{h}}, N - R)\|_2 \\ \hat{\nu} &= \arg\min_\nu \|d_k(L(\hat{y})_{R:N}) - F_\nu^{\mathrm{l}}(\hat{y}_R^{\mathrm{l}}, h_R^{\mathrm{l}}, \tilde{N})\|_2, \end{aligned} \tag{10}$$

with $\hat{y}_R^{\mathrm{h}} = H(\hat{y})_R$, $\hat{y}_R^{\mathrm{l}} = L(\hat{y})_R$, $N - R$ steps $H(\hat{y})_{R:N}$ and $L(\hat{y})_{R:N}$ from the filtered signals $H(\hat{y})$ and $L(\hat{y})$ and $\tilde{N} = \lfloor (N' - R)/k \rfloor$. The hidden states h_R^{h} and h_R^{l} are obtained from a warmup phase that we specify in the appendix.

Predictions: A prediction with $N' - R$ steps $\tilde{y}_{R:N'}$ is obtained by adding the high-frequency predictions and the upsampled low-frequency predictions

$$\tilde{y}_{R:N'} = F_\theta^{\mathrm{h}}(\tilde{y}_R^{\mathrm{h}}, h_R^{\mathrm{h}}, N' - R) + u_k(F_\nu^{\mathrm{l}}(\tilde{y}_R^{\mathrm{l}}, h_R^{\mathrm{l}}, \tilde{N})), \tag{11}$$

where $\tilde{y}_R^{\mathrm{h}} = H(\tilde{y}_{0:R})$ and $\tilde{y}_R^{\mathrm{l}} = L(\tilde{y}_{0:R})$ have to be provided and $\tilde{N} = \lfloor (N' - R)/k \rfloor$. The hidden states h_R^{h} and h_R^{l} can, for example, be obtained from a short warmup phase.

A slight modification of the method is obtained by wrapping an additional high-pass filter around the predictions via $H(F_\theta^{\mathrm{h}}(\tilde{y}_R^{\mathrm{h}}, h_R^{\mathrm{h}}, N - R))$ in Eq. (10) during training and via $H(F_\theta^{\mathrm{h}}(\tilde{y}_R^{\mathrm{h}}, h_R^{\mathrm{h}}, N' - R))$ in Eq. (11) during predictions. This adds an additional guarantee preventing the high-frequency model from producing low-frequency errors. We provide a numerical comparison of both variants in our experiments.

4.3 Hybrid Modeling

In the hybrid case, we assume access to a simulator that produces predictions $y_{0:N}^{\mathrm{s}}$. Thus, reliable long-term predictions are already available. In this case, we can directly train a single recurrent model in an end-to-end fashion (see Figure 1 (right)). In particular, the low-pass filter is applied to the simulator, whereas the high-pass filter is applied to the learning-based trajectory. Training directly with the complementary filter ensures that each model indeed stays on its time scale. By decoupling the propagation of latent states and the filtered simulator states, the method is technically applicable to a large class of simulators. It is solely required that the simulator is able to produce time-series predictions of the system given initial conditions. Differentiating through the simulator or any insight into the simulator's hidden states is not required.

Training and Predictions: Consider training data $\hat{y}_{0:N} \in \mathbb{R}^{D_y \times N}$, a trainable model f_θ, C_θ with corresponding rollout mapping F_θ (cf. Eq. (8)) and a complementary filter pair (L, H). Again, the first R steps $\hat{y}_{0:R}$ are used for providing the intial hidden state h_R. The weights θ are trained via

$$\hat{\theta} = \arg\min_\theta \|H(y^{\mathrm{r}}) + L(y^{\mathrm{s}}_{R:N}) - \hat{y}_{R:N}\|_2, \tag{12}$$

with $y^{\mathrm{r}} = F_\theta(\hat{y}_R, h_R, N - R)$. The calculation of $H(y^{\mathrm{r}}) + L(y^{\mathrm{s}}_{R:N})$ can directly be obtained by Eq. (6).

4.4 Filter Design

We design filters H and L (cf. Eq. (10) and (12)) before training. In the purely learning-based scenario, a broad range of cutoff frequencies is possible, which we demonstrate empirically in the appendix. In the hybrid case, we aim to use as much correct long-term information as possible from the simulator without including short term errors. In general, suitable cutoff frequencies can often be derived from domain knowledge. Here, we analyze the frequency spectra of ground truth and simulator in order to find a suitable cutoff frequency. For a specific filter design, we test the plausibility of the complementary filter by applying the high-pass component to the measurements and the low-pass component to the simulator. Calculating the RMSE between the combined signal and ground truth indicates whether the filters are appropriate. For a more detailed introduction to general filter design, we refer the reader to Oppenheim et al. (1999).

5 Experiments

In this section, we demonstrate that our complementary filter-based methods yield accurate long and short-term predictions on simulated and real world data. In the hybrid setting, we consider additional access to a physics-based simulation that is able to predict the long-term behavior of the system but is not capable of accommodating all short-term details due to e.g., modeling simplifications.

5.1 Baselines

We consider four systems. For each system, we have access to measurement data. Either real measurements are available, or we simulate trajectories from the ground truth system and corrupt them with noise.

We consider the following baselines.

RNN: RNN structure that corresponds to an MLP that is propagated through time.

GRU: State-of-the-art recurrent architecture for time-series learning (Cho et al. 2014).

Simulator: In the hybrid setting, access to simulator predictions y^{s} is required.

Residual GRU/ RNN: In the hybrid case, we consider a residual model that combines RNN or GRU predictions y^{r} with simulator predictions y^{s} via $y = y^{\mathrm{r}} + y^{\mathrm{s}}$.

5.2 Constructing the Filters

We use the tools for IIR filter design provided by `Scipy` (Virtanen et al. 2020) and apply Butterworth filters. We construct the coefficients b_k and a_k for the low-pass filter and coefficients $\tilde{b}_k$ and $\tilde{a}_k$ for the high-pass filter as described in Sec. 3, where both filters share the cutoff frequency. An example of a frequency spectrum and choice of the cutoff frequency is shown in Figure 2 (left). In the appendix, we add information on the specific design of the complementary filter pairs for each experiment. Further, we add frequency spectra for each system.

5.3 Learning Task and Comparison

For each system, we observe a single trajectory. The models are trained on a fixed subtrajectory of the full trajectory. Predictions are performed by computing a rollout of the model over the full trajectory. We evaluate the model accuracy by computing the RMSE along the full trajectory. On the simulated systems, the RMSE between predictions and ground truth is computed. On real world data, the RMSE between predictions and measurements is computed. Runtimes are reported in the appendix.

5.4 Purely Learning-Based Model

We apply the strategy derived in Sec. 4 to GRU models (referred to by "split GRU") and compare to a single GRU model trained on the entire bandwidth. In order to draw a fair comparison, we choose an equal number of total hidden units for the baseline GRU and the sum of hidden states in our approach. We provide architecture details in the appendix. Furthermore, we optionally wrap an additional high-pass filter around the predictions $F^h_\theta(\hat{y}^h_R, h^h_R, N-R)$ during training and inference (cf. Eq. (10) and (11)), and denote this by the suffix "+HP". In order to demonstrate the flexibility of our method, we add results with varying cutoff frequencies and downsampling ratios in the appendix. We train our model on the following systems:

(i) Double-Mass Spring System: We simulate a double-mass spring system that consists of two sinusoidal waves with different frequencies and corrupt the simulation with additional observation noise. Training is performed on an interval of 250 steps, while predictions are computed on 1000 steps (further details can be found in the appendix).

(ii) - (iv) Double-Torsion Pendulum: In the second set of experiments, we consider real measurements from the double-torsion pendulum system introduced in Lisowski et al. (2020). Data are obtained by exciting the system with different inputs. In particular, we consider 4 different excitations with varying frequencies. Training is performed on the first 600 measurements, while predictions are performed on an 2000-steps interval.

5.5 Hybrid Model

For the hybrid model, we train our complementary filtering method with GRU and RNN and compare against the

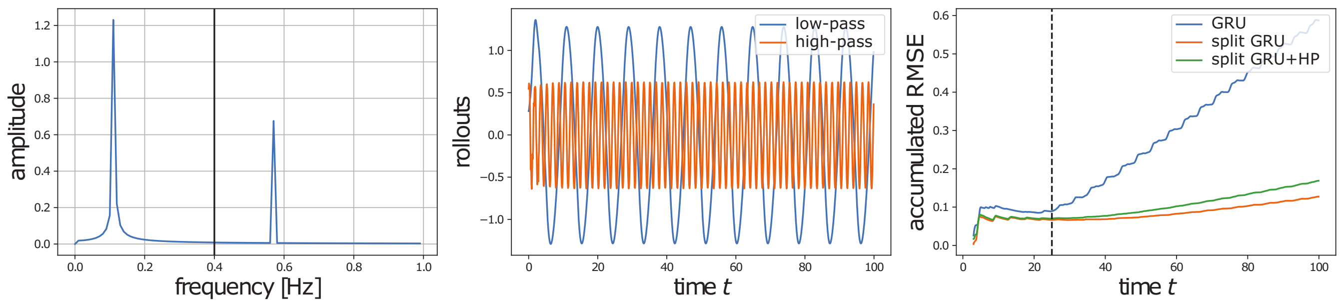

Figure 2: Demonstrating the method with the double-mass spring system (i). Shown is the analysis of the frequency spectrum with marked cutoff frequency (left). This yields the predictions of the two seperate GRUs (middle). The accumulated RMSE over time indicates good short-and-long term behavior, while the baseline method accumulates errors (right).

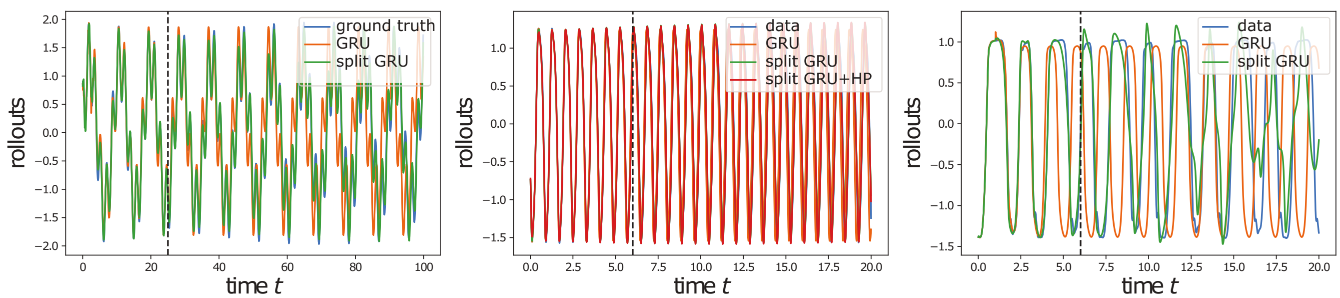

Figure 3: Rollouts for the purely learning-based scenario with all three methods for Systems (i) (left), (iv) (middle) and (ii) (right). The training horizon is marked with dotted lines. The results show accumulating errors of the baseline method in contrast to our approach.

corresponding non-hybrid models (GRU and RNN), the corresponding residual models (residual GRU/ RNN), and the simulator. We consider the following systems:

(v) Van-der-Pol Oscillator: Data from a Van-der-Pol oscillator with external force is simulated from the four-dimensional ground truth system (Grasman, Veling, and Willems (1976)). It is assumed that only the first dimension, corresponding to the position, is observed. Simulator data are obtained from an unforced Van-der-Pol oscillator. For the corresponding equations, we refer to the appendix.

(vi) Drill-String: We consider measurement data from the drill string experiment provided in Aarsnes and Shor (2018) Figure 14 as training data and the corresponding simulated signal as simulator.

5.6 Results

The results indicate the advantage of leveraging complementary filters for dynamics model learning. In particular, the resulting predictions show stable short and long-term behavior, while especially the GRU and RNN baselines tend to drift on the long-term due to accumulating errors. For both scenarios, we provide additional plots showing the accumulated RMSE over time for each system in the appendix.

Purely Learning-Based: The results in Table 1 indicate the advantage of our approach due to accumulating errors for the baseline method. Integrating a small model error in each time-step leads to a long-term drift that can also be directly observed in the rollouts (cf. Figure 3). Our approach on the other hand does not suffer from this drift due to the specific architecture and therefore outperforms the baseline method on every task. The findings are also supported by the RSME over time $(e_n)_{n=0}^N$ with $e_n = \sqrt{\sum_{k=0}^n \frac{1}{n+1} \|y_k - \hat{y}_k\|^2}$ shown in Figure 2 (right). In some cases our methods yields faster convergence than the baseline method. For System i) we report the results after 300 training epochs for our method, while the GRU was trained on 2000 epochs. To provide more insights, we demonstrate the functionality of our method with the double-mass spring system (i) (cf. Figure 2). Designing the filters shown in Figure 2 (left) yields seperate predictions from the two GRUs in Figure 2 (middle). Similar results of our split GRU and our split GRU+HP indicate that the most effective part is already contained in the split GRU (cf. Table 1). Here, the high-frequency model already stays on the desired time scale and the additional high-pass filter rather introduces a small distortion. Further, our split GRU+HP shows a higher error in the beginning due to transient behavior of the filter, which can be seen in Figure 2 (right). However, the additional high-pass filter guarantees that the high-frequency predictions are indeed affecting the correct time scale.

Hybrid Model: We report the RMSEs for the hybrid setting with RNNs in Table 2 and with GRUs in Table 3. The results demonstrate that our method is beneficial for different types of models, here MLP-based RNNs and GRUs. Again,

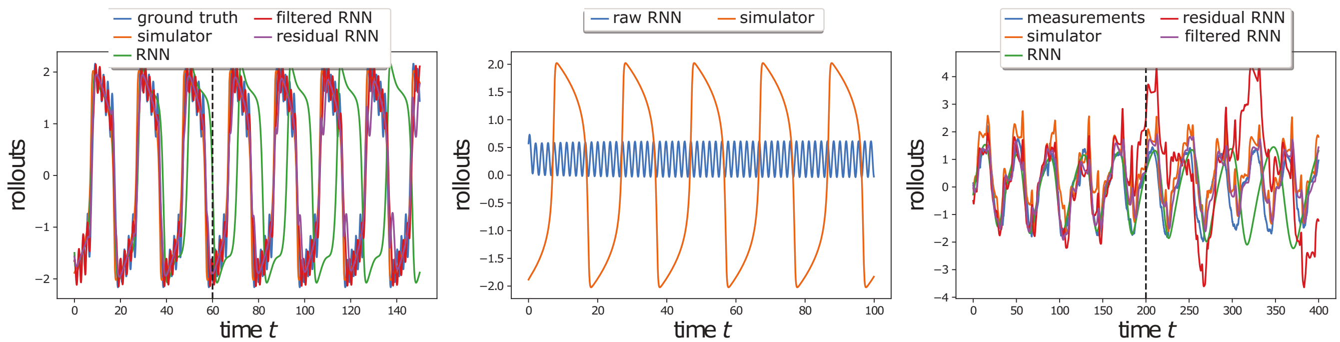

Figure 4: Rollouts for the hybrid setting. Shown are the results for the Van-der-Pol oscillator (v) with RNN (left) and rollouts of the single components before combining them via the complementary filter (middle). Further, the rollouts of the drill-string system (vi) with RNN are shown (right). The training horizon is marked with dotted lines. The results demonstrate accumulating errors in the baseline methods, while our approach provides accurate short and long-term predictions.

System	GRU	split GRU (ours)	split GRU + HP (ours)
(i)	0.587 (0.002)	**0.127** (0.008)	0.168 (0.03)
(ii)	1.124 (0.485)	0.331 (0.065)	**0.318** (0.089)
(iii)	0.287 (0.15)	0.159 (0.051)	**0.13** (0.02)
(iv)	0.262 (0.17)	**0.201** (0.07)	0.18 (0.06)

Table 1: Total RMSEs (mean (std)) over 5 indep. runs with purely learning-based scheme.

System	RNN	residual RNN	simulator	filtered RNN (ours)
(v)	1.29 (0.63)	0.417 (0.03)	0.418	**0.347** (0.041)
(vi)	1.1 (1.26)	3.60 (1.62)	0.729	**0.487** (0.381)

Table 2: Total RMSEs for the hybrid model with RNN (mean (std)) over 5 indep. runs.

System	GRU	residual GRU	simulator	filtered GRU (ours)
(v)	0.463 (0.305)	0.476 (0.096)	0.418	**0.387** (0.026)
(vi)	1.140 (0.258)	**0.681** (0.055)	0.729	0.765 (0.008)

Table 3: Total RMSEs for the hybrid model with GRU (mean (std)) over 5 indep. runs.

the standard training with single GRU or single RNN shows some drift causing bad long-term behavior. The unstable long-term behavior is demonstrated particularly clearly by the RNN results shown in Figure 4 (left and right). While the residual RNN baseline does not suffer from the typical drift that is observed for the RNN baseline, it still shows instabilities in the long-term behavior. In particular, the results for System (vi) in Figure 4 (right) demonstrate that low-frequency errors occur for the residual model as well. Our method, in contrast, eliminates these errors by design. However, on System (vi), our filtered GRU is outperformed by the residual GRU since our predictions stay close to the simulator predictions. We provide additional insights into our method by depicting the RNN and simulator predictions before combining them via the complementary filter in Figure 4 (middle). Additional plots are provided in the appendix.

6 Conclusion

In this paper, we propose to combine complementary filtering with dynamics model learning. In particular, we fuse the predictions of different models, where one models provides reliable long-term predictions and the other reliable short-term predictions. Leveraging the concept of complementary filter pairs yields a model that combines the best of both worlds. Based on this idea, we propose a purely learning-based model and a hybrid model. In the hybrid scenario, the long-term predictions are addressed by a simulator, whereas in the purely learning-based scenario an additional model has to be trained. The experimental results demonstrate that our approach yields predictions with accurate long and short-term behavior. An interesting topic for future research is an extension of the hybrid scenario learning the relationship between simulator predictions and learning-based predictions.

Acknowledgements

The authors thank Karim Barsim, Alexander Gräfe and Andreas René Geist for valuable discussions and feedback. Furthermore, we thank Paweł Olejnik for providing measurement data from the double-torsion pendulum for the systems (ii)-(iv) that we consider in our experiments.

References

Aarsnes, U. J. F.; and Shor, R. J. 2018. Torsional vibrations with bit off bottom: Modeling, characterization and field data validation. *Journal of Petroleum Science and Engineering*, 163: 712–721.

Back, A. D.; and Tsoi, A. C. 1991. FIR and IIR Synapses, a New Neural Network Architecture for Time Series Modeling. *Neural Computation*, 3(3): 375–385.

Campolucci, P.; Uncini, A.; and Piazza, F. 1996. Fast Adaptive IIR-MLP Neural Networks for Signal Processing Application. *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, 6: 3529–3532.

Cho, K.; van Merriënboer, B.; Gulcehre, C.; 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)*, 1724–1734. Doha, Qatar: Association for Computational Linguistics.

Chornoboy, E. 1992. Initialization for Improved IIR Filter Performance. *IEEE Transactions on Signal Processing*, 40: 543 – 550.

Doerr, A.; Daniel, C.; Schiegg, M.; Duy, N.-T.; Schaal, S.; Toussaint, M.; and Trimpe, S. 2018. Probabilistic Recurrent State-Space Models. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 1280–1289. PMLR.

Ćertić, J. D.; and Milić, L. D. 2011. Investigation of computationally efficient complementary IIR filter pairs with tunable crossover frequency. *AEU - International Journal of Electronics and Communications*, 65(5): 419–428.

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.; Fiene, J.; Tashiro, N.; Jia, Z.; and Trimpe, S. 2022. The Wheelbot: A Jumping Reaction Wheel Unicycle. *IEEE Robotics and Automation Letters*, 7(4): 9683–9690.

Grasman, J.; Veling, E. J. M.; and Willems, G. M. 1976. Relaxation Oscillations Governed by a Van Der Pol Equation with Periodic Forcing Term. *SIAM Journal on Applied Mathematics*, 31(4): 667–676.

Gustafsson, F. 1996. Determining the initial states in forward-backward filtering. *IEEE Transactions on Signal Processing*, 44(4): 988–992.

Higgins, W. T. 1975. A Comparison of Complementary and Kalman Filtering. *IEEE Transactions on Aerospace and Electronic Systems*, AES-11(3): 321–325.

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

Johansson, H.; and Saramäki, T. 1999. A Class of Complementary IIR Filters. In *ISCAS'99, Proceedings of the 1999 IEEE International Symposium on Circuits and Systems, May 30-June 2, 1999, Orlando, Florida, USA*, 299–302.

Kuznetsov, B.; Parker, J.; and Esqueda, F. 2020. Differentiable IIR filters for machine learning applications. In *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx-20)*, 297–303.

Lange, H.; Brunton, S. L.; and Kutz, J. N. 2021. From Fourier to Koopman: Spectral Methods for Long-term Time Series Prediction. *J. Mach. Learn. Res.*, 22: 41:1–41:38.

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.

Milić, L.; and Saramaki, T. 2003. Power-Complementary IIR Filter Pairs with an Adjustable Crossover Frequency. *Facta Universitatis, Series Electronics and Energetics*, 16: 295–304.

Narkhede, P.; Poddar, S.; Walambe, R.; Ghinea, G.; and Kotecha, K. 2021. Cascaded Complementary Filter Architecture for Sensor Fusion in Attitude Estimation. *Sensors*, 21(6).

Narkhede, P.; Raj, A. N. J.; Kumar, V.; Karar, V.; and Poddar, S. 2019. Least square estimation-based adaptive complimentary filter for attitude estimation. *Transactions of the Institute of Measurement and Control*, 41(1): 235–245.

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. PMLR.

Oppenheim, A.; Schafer, R.; Buck, J.; and Lee, L. 1999. *Discrete-time Signal Processing*. Prentice Hall international editions. Prentice Hall.

Paolucci, R.; Gatti, F.; Infantino, M.; Smerzini, C.; Güney Özcebe, A.; and Stupazzini, M. 2018. Broadband Ground Motions from 3D Physics-Based Numerical Simulations Using Artificial Neural Networks. *Bulletin of the Seismological Society of America*, 108(3A): 1272–1286.

Proctor, J. L.; Brunton, S. L.; and Kutz, J. N. 2016. Dynamic Mode Decomposition with Control. *SIAM Journal on Applied Dynamical Systems*, 15(1): 142–161.

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.

Stepleton, T.; Pascanu, R.; Dabney, W.; Jayakumar, S. M.; Soyer, H.; and Munos, R. 2018. Low-pass Recurrent Neural Networks - A memory architecture for longer-term correlation discovery. arXiv:1805.04955.

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.

Trimpe, S.; and D'Andrea, R. 2010. Accelerometer-based tilt estimation of a rigid body with only rotational degrees of freedom. In *2010 IEEE International Conference on Robotics and Automation*, 2630–2636.

Vaidyanathan, P.; Regalia, P.; and Mitra, S. 1987. Design of Doubly-Complementary IIR Digital Filters Using a Single Complex Allpass Filter, with Multirate Applications. *IEEE Transactions on Circuits and Systems*, 34: 378 – 389.

Virtanen, P.; Gommers, R.; Oliphant, T. E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; van der Walt, S. J.; Brett, M.; Wilson, J.; Millman, K. J.; Mayorov, N.; Nelson, A. R. J.; Jones, E.; Kern, R.; Larson, E.; Carey, C. J.; Polat, İ.; Feng, Y.; Moore, E. W.; VanderPlas, J.; Laxalde, D.; Perktold, J.; Cimrman, R.; Henriksen, I.; Quintero, E. A.; Harris, C. R.; Archibald, A. M.; Ribeiro, A. H.; Pedregosa, F.; van Mulbregt, P.; and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17: 261–272.

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*.

Zhang, S.; Liu, C.; Jiang, H.; Wei, S.; Dai, L.; and Hu, Y. 2016. Feedforward Sequential Memory Networks: A New Structure to Learn Long-term Dependency. arXiv:1512.08301.

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*.