Hybrid Neural Network Augmented Physics-based Models for Nonlinear Filtering

Tales Imbiriba[†*], Ahmet Demirkaya[†*], Jindřich Duník[‡], Ondřej Straka[‡], Deniz Erdoğmuş[†], and Pau Closas[†]
† Electrical & Computer Engineering Department
Northeastern University
Boston, MA, USA
Emails: {talesim, demirkaya, erdogmus, closas}@ece.neu.edu
‡ Department of Cybernetics
University of West Bohemia
Pilsen, Czech Republic
Emails: {dunikj, straka30}@kky.zcu.cz

***Abstract*—In this paper we present a hybrid neural network augmented physics-based modeling (APBM) framework for Bayesian nonlinear latent space estimation. The proposed APBM strategy allows for model adaptation when new operation conditions come into play or the physics-based model is insufficient (or incomplete) to properly describe the latent phenomenon. One advantage of the APBMs and our estimation procedure is the capability of maintaining the physical interpretability of estimated states. Furthermore, we propose a constraint filtering approach to control the neural network contributions to the overall model. We also exploit assumed density filtering techniques and cubature integration rules to present a flexible estimation strategy that can easily deal with nonlinear models and high-dimensional latent spaces. Finally, we demonstrate the efficacy of our methodology by leveraging a target tracking scenario with nonlinear and incomplete measurement and acceleration models, respectively.**

***Index Terms*—Nonlinear filtering; Target tracking; Hybrid Neural Network; Physics-based Neural Models; Gaussian filtering.**

I. Introduction

Complex nonlinear dynamic models can be found in numerous applications such as describing biological systems, weather prediction, fluid dynamics, and target tracking, to name but a few. In many such applications the "true" underlying model can be very complex and context-dependent. For instance, this is the case for sunlight interaction with materials in the Earth surface [1], [2], disease spread [3], indoor positioning [4], navigation [5], or predicting the trajectory of a target [6]. In these examples, the model governing the evolution of such signals is often either very complex or unknown (in the case of target tracking) leading to more complex optimization strategies and/or other sources of information to be able to provide accurate predictions.

In this context machine learning (ML) algorithms become appealing. This is the case especially when accurate physical-based models are too complex or unknown. One drawback of purely data-driven ML strategies relates to the interpretability and physical meaning of estimated quantities especially when one aims at recovering latent states [7], [8].

In this contribution, we are especially interested in leveraging ML strategies to improve nonlinear dynamical models. In this context, ML approaches can be classified into hybrid [9], [10], where data driven models are used in combination with physics-based models, or purely data-driven following an end-to-end learning philosophy [11]. The hybrid approaches focus on providing corrections to estimates. The algorithms [12] and [13] use back-propagation NNs to correct position estimates.The algorithm proposed in [14] predicts nonlinear velocity and acceleration corrections to linear predicted states by a recurrent NN and the algorithm [15] augments the error-state Kalman filter (KF) by an RBF NN to compensate for the lack of KF performance. The algorithms use NNs that are unaware of the system model.

In addition to providing estimate corrections, several hybrid approaches directly estimate the state. In [16] two algorithms based on deep long short-term memory (LSTM) NNs were proposed, which provide estimates either in two steps (time-update and measurement-update) or in a single step. Both algorithms are unaware of the system model and the learning is based on estimated quality optimization. The algorithm [17] uses the NN in the prediction step of the KF to provide not only the estimate but also the associate covariance matrix. In this case, the learning optimizes the negative log-likelihood of multivariate normal distribution. An alternative approach was proposed in [18], where the NN was used to learn the system model parameters such as state transition and measurement matrices, and associated noise covariance matrices under the framework of Bayesian estimation. Recently, deep Kalman filters strategies were proposed in a smoother-like approach [19] and efficient KF implementations where the Kalman gain is approximated by NNs [20]. In both works however, the hybrid APBM model component were not studied.

The data-based approaches use plain data to learn the mapping from observations to the states to avoid complications of the hybrid methods. In [21] a large amount of data was simulated to be able to achieve end-to-end learning while

This work has been partially supported by the NSF under Awards CNS-1815349 and ECCS-1845833, NSF-1935337.
* Indicates shared first authorship.

in [22] all components of the process, i.e., data generator, sliding window, centralization strategy, and the learner, have been proposed.

In this paper we are concerned about nonlinear dynamical models of the type:

$$\begin{aligned}\dot{\boldsymbol{x}}_t &= f(\boldsymbol{x}_t) + \boldsymbol{q}_t \\ \boldsymbol{y}_t &= h(\boldsymbol{x}_t) + \boldsymbol{r}_t\end{aligned} \tag{1}$$

where $\boldsymbol{x}_t \in \mathbb{R}^{d_x}$ is the state vector, f is the state transition function and $\boldsymbol{q}_t$ is an arbitrary zero-mean noise term independent of $\boldsymbol{x}_t$. More specifically, we are interested in models where the transition function f is not fully representative of the true state dynamics due to either simplistic physics-based models not being capable to explain time-varying dynamic scenarios or other unmodeled factors. In such a scenario, we propose a hybrid neural network augmentation approach capable of augmenting physics-based models. To cope with time-varying dynamics we use a continuous learning strategy consisting of augmenting the states with model parameters leading to recursive Bayesian estimation (RBE) approaches [23]. One of the challenges of incorporating neural network parameters as states in an RBE approach is related to the computational complexity inherent to high-dimensional state spaces especially when nonlinear or non-Gaussian models are in play. To partially mitigate this issue, we leverage Cubature integration strategies in association with Gaussian assumptions, i.e., the Cubature Kalman filter [24]. We also propose a strategy to control the neural network contribution to the overall dynamic model.

This work is organized as follows. In Section II we present our NN augmentation strategy. In Section III we present an augmented likelihood mechanism to control the NN's contribution. In Section IV we discuss the Cubature integration and relationship with different filtering approaches. Experiments are presented in Section V and final remarks discussed in Section VI.

II. Hybrid Neural Network Physics-based models

In this section, we aim at augmenting the ODE model in (1) using neural networks. The model mismatch might occur due to missing ODE components or inaccurate models that do not match the governing physics of the phenomenon. In this contribution, we aim at the latter. That is, we assume that the number of states is known and somehow represented by the physical model f. The discretization of (1) is

$$\begin{aligned}\boldsymbol{x}_k &= \boldsymbol{x}_{k-1} + T_s f(\boldsymbol{x}_{k-1}) + T_s \tilde{\boldsymbol{q}}_{k-1} \\ \boldsymbol{y}_k &= h(\boldsymbol{x}_k) + \boldsymbol{r}_k\end{aligned} \tag{2}$$

where T_s is the sampling period and $\tilde{\boldsymbol{q}}_{k-1}$ is the independent discretized noise term. Thus, we propose to augment the discretized dynamical model in (2) as:

$$\boldsymbol{x}_k = g\left(f(\boldsymbol{x}_{k-1}), \boldsymbol{x}_{k-1}; \boldsymbol{\theta}\right) + \boldsymbol{q}_{k-1} \tag{3}$$

where $g(\cdot) : \mathbb{R}^{d_x} \times \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_x}$ is a vector-valued function, modeled as neural network, and parametrized by $\boldsymbol{\theta} \in \mathbb{R}^P$, that we assumed to have incorporated the sampling period T_s, and $\boldsymbol{q}_{k-1} = T_s \tilde{\boldsymbol{q}}_{k-1}$. The model in Eq. (3) is flexible enough that allow for both replacing whole ODEs (e.g, $f(\boldsymbol{x}_k) = 0$), or fusing arbitrary functions of $\boldsymbol{x}_k$ and $f(\boldsymbol{x}_k)$. We call models extended in such fashion *augmented physics-based models* (APBMs).

III. Controlling neural network contributions in augmented models

One important point regarding the hybrid dynamical model is the capability of controlling the contribution of the neural augmentation. In this section we propose to introduce a regularization over the NN model parameters $\boldsymbol{\theta}_k$ by an augmentation of the likelihood model [25] as $p(\boldsymbol{y}_k, \bar{\boldsymbol{\theta}} | \boldsymbol{x}_k, \boldsymbol{\theta}_k)$, where $\bar{\boldsymbol{\theta}}$ is the P-dimensional vector designed such that $g\left(f(\boldsymbol{x}_k), \boldsymbol{x}_k; \boldsymbol{\theta} = \bar{\boldsymbol{\theta}}\right) = f(\boldsymbol{x}_k)$. This allows us to re-write the state-space model in (3) as:

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \boldsymbol{q}^{\theta}_{k-1} \tag{4}$$

$$\boldsymbol{x}_k = g(f(\boldsymbol{x}_{k-1}), \boldsymbol{x}_{k-1}; \boldsymbol{\theta}_{k-1}) + \boldsymbol{q}^{x}_{k-1} \tag{5}$$

$$\begin{bmatrix}\boldsymbol{y}_k \\ \bar{\boldsymbol{\theta}}\end{bmatrix} = \begin{bmatrix}h(\boldsymbol{x}_k) \\ \boldsymbol{\theta}_k\end{bmatrix} + \begin{bmatrix}\boldsymbol{r}^{y}_k \\ \boldsymbol{r}^{\theta}_k\end{bmatrix} \tag{6}$$

where $\boldsymbol{q}^{\theta}_{k-1} \sim \mathcal{N}(0, \boldsymbol{Q}^{\theta})$, $\boldsymbol{q}^{x}_{k-1} \sim \mathcal{N}(0, \boldsymbol{Q}^{x})$ model the dynamic model uncertainty, $\boldsymbol{r}^{y}_k \sim \mathcal{N}(0, \boldsymbol{R}^{y})$ is the measurement noise, and $\boldsymbol{r}^{\theta}_k \sim \mathcal{N}(0, \frac{1}{\lambda}\boldsymbol{I})$ defines the ball around $\bar{\boldsymbol{\theta}}$ of possible solutions for $\boldsymbol{\theta}_k$. The posterior can then be re-written as

$$\begin{aligned}p(\boldsymbol{x}_k, \boldsymbol{\theta}_k | \boldsymbol{y}_{1:k}, \bar{\boldsymbol{\theta}}) &\propto p(\boldsymbol{y}_k, \bar{\boldsymbol{\theta}} | \boldsymbol{x}_k, \boldsymbol{\theta}_k) \\ &\times \int\int p(\boldsymbol{x}_k | \boldsymbol{x}_{k-1}, \boldsymbol{\theta}_{k-1}) p(\boldsymbol{x}_{k-1} | \boldsymbol{\theta}_{k-1}, \boldsymbol{y}_{1:k-1}, \bar{\boldsymbol{\theta}}) \\ &\times p(\boldsymbol{\theta}_k | \boldsymbol{\theta}_{k-1}) p(\boldsymbol{\theta}_{k-1} | \boldsymbol{y}_{1:k-1}, \bar{\boldsymbol{\theta}}) d\boldsymbol{x}_{k-1} d\boldsymbol{\theta}_{k-1}\end{aligned} \tag{7}$$

In this Gaussian case the recursive Bayesian filtering solution for the above problem is equivalent to the solution to the following problem for the state mean at every time instant k:

$$\begin{aligned}(\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{\theta}}_k) = \underset{(\boldsymbol{x}, \boldsymbol{\theta})}{\arg\min} &\|\boldsymbol{y}_k - h(\boldsymbol{x})\|^2_{\boldsymbol{R}^{-1}} + \lambda \|\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}\|^2 \\ &+ \|\boldsymbol{x} - f(\hat{\boldsymbol{x}}_{k-1})\|^2_{[\hat{\boldsymbol{P}}^{x}_{k|k-1}]^{-1}} + \|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_{k-1}\|^2_{[\hat{\boldsymbol{P}}^{\theta}_{k|k-1}]^{-1}}\end{aligned} \tag{8}$$

where the terms correspond to data fit, neural network parameter regularization, and the two regularizations resulting from the the dynamical models for the states $\boldsymbol{x}$ and $\boldsymbol{\theta}$, respectively. We call attention to the fact that the parameter λ controls the regularization term $\|\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}\|^2$ and, thus, the contribution of the neural network model. For instance, when $\lambda \rightarrow \infty$ the parameter constraint is fully enforced in (8) completely eliminating the neural augmentation contribution to the model. When $\lambda \rightarrow 0$ the parameter constraint in (8) is turned-off leading to free neural network model contributions that can, possibly, overpower the physics-based model.

IV. Efficient Gaussian filter model learning

In this paper, we train a hybrid ODE and neural network model using recursive Bayesian state estimation. Specifically, when the predictive and observation models are linear and Gaussian, the state posterior recursion integrals can be solved analytically leading to the well-known Kalman time and measurement update equations [26]. When nonlinear models are in play, the required integrals often become intractable and numerical strategies must be sought [27]. Alternatives include linearization of nonlinear functions (extended Kalman filters, EKF) or sampling methods such as particle filters [28], unscented Kalman filters (UKF) [29], or cubature Kalman filters (CKF) [24], which assume different levels of system simplicity. For instance, EKF, UKF, and CKF assume Gaussianity of the measurement and transitional models while handling the integration exploiting this Gaussianity in different ways. While EKF linearizes the models using a first-order Taylor expansion, UKF and CKF use unscented and cubature rules to compute integrals of the form

$$I(\ell) = \int_{\mathcal{D}} \ell(\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x} \tag{9}$$

where ℓ is a nonlinear function of $\boldsymbol{x} \in \mathbb{R}^{d_x}$ and $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a Gaussian PDF with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, as a weighted sum of function evaluations of a finite number of deterministic points. For the third-degree cubature rule, the integral in (9) can be approximated as

$$I(\ell) \approx \frac{1}{2d_x} \sum_{j=1}^{2d_x} \ell(\boldsymbol{S}^\top \boldsymbol{\xi}_j + \boldsymbol{\mu}) \tag{10}$$

where $\boldsymbol{\xi}_j = [\mathbf{1}]_j \sqrt{2d_x/2}$ are deterministic points [24], and $\boldsymbol{S}$ is the lower triangular Cholesky decomposition such that $\boldsymbol{\Sigma} = \boldsymbol{S}\boldsymbol{S}^\top$. It is important to highlight that the cubature rule demands only two points per dimension of $\boldsymbol{x}$, i.e., $2d_x$, to evaluate the sum in (10), making it more suitable when working in high-dimensional state-spaces. Assuming Gaussianity of state posteriors, the CKF can solve the integrals required in the Bayesian recursion as well as the moments (mean and covariance) of the new state posterior [24].

In contrast, particle filters do not assume any particular distribution; instead, they approximate the distribution as a linear combination of Dirac deltas. Thus, moments of propagated particles can be easily computed. One drawback of particle filters is the high number of particles needed to accurately represent distributions. This issue is profoundly aggravated if the state-space dimension is large, making this filtering strategy unfeasible in such scenarios [6].

V. Experiments

In this section, we present two experiments designed to test different forms of model augmentation. In the first, see Section V-A, we consider the Lorenz Attractor [30] where we replaced a whole ODE with a neural network model. We keep the approach hybrid in the sense that we kept the remaining ODEs fixed. In the second example, see Section V-B, we exploit a target tracking example where the data was generated with a model that contains additive constant velocity and sinusoidal components. In this example, we augmented the constant velocity model with a neural network.

For both examples, we performed Monte Carlo simulations using 100 runs. To measure the filtering performance we consider the tracking root mean-squared error (RMSE):

$$\text{RMSE}_k = \sqrt{\frac{\sum_{r=1}^{N_{MC}} \|\boldsymbol{p}_k^{(r)} - \hat{\boldsymbol{p}}_k^{(r)}\|^2}{dN_{MC}}} \tag{11}$$

with d is the dimension of $\boldsymbol{p}_k$, N_{MC} is the number of Monte Carlo runs, and $\hat{\boldsymbol{p}}_k$ is the estimated states of interest, e.g., position for the target tracking case and all states for the Lorenz Attractor example.

To analyse the evolution of the weights over time with different λ, we computed the average variance of parameters as:

$$\mathbb{E}[\text{Var}(\boldsymbol{\theta}_k)] = \frac{\sum_{r=1}^{N_{MC}} \text{Var}(\boldsymbol{\theta}_k^{(r)})}{N_{MC}} \tag{12}$$

where $\text{Var}(\boldsymbol{\theta}_k^{(r)})$ is the sample variance of the neural network parameters for the r-th Monte Carlo run.

A. Application to chaotic systems

In this section we present the Lorenz Attractor [30] as an example chaotic system to test our approach. The Lorenz Attractor consists of three ordinary differential equations (ODEs):

$$\dot{x}_1 = \sigma(x_2 - x_1) \tag{13}$$

$$\dot{x}_2 = x_1(\rho - x_3) \tag{14}$$

$$\dot{x}_3 = x_1 x_2 - \beta x_3 \tag{15}$$

where x_i, $i = 1, 2, 3$, are the system states and $\sigma, \rho, \beta \in \mathbb{R}$ are model parameters.

The model can be discretized as

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + T_s f(\boldsymbol{x}_{k-1}) \tag{16}$$

where $\boldsymbol{x}_k \in \mathbb{R}^3$ is the vector of states and $f(\cdot) = [f_1(\cdot), f_2(\cdot), f_3(\cdot)]^\top$ is a vector valued function representing the dynamical model in Eqs. (13)–(15). We generate measurements according to the following measurement equation:

$$\mathbf{y}_k = \boldsymbol{x}_k + \boldsymbol{n}_k \ , \tag{17}$$

where $\boldsymbol{n}_k \sim \mathcal{N}(0, 0.001\boldsymbol{I})$. The initial state was set as $\boldsymbol{x}_0 = (1, 1, 1)^\top$, and the dynamical model parameters as $\sigma = 28$, $\rho = 10$, and $\beta = 8/3$, and $T_s = 1$s.

In this first experimental setup, we follow a procedure similar to the one in [10] where we replaced an ODE of the system was replaced by a neural network. In the example presented in this section we replaced the ODE for x_1 in Eq. (13) with a neural network $\gamma(\cdot)$. In this case, the discretized NN-based version of (13) can be written as

$$x_{1,k} = x_{1,k-1} + \gamma(\boldsymbol{x}_{k-1}; \boldsymbol{\theta}) \tag{18}$$

leading to:

$$g(f(\boldsymbol{x}_{k-1}), \boldsymbol{x}_{k-1}; \boldsymbol{\theta}) = \begin{pmatrix} x_{1,k-1} + \gamma(\boldsymbol{x}_{k-1}; \boldsymbol{\theta}) \\ f_2(x_{1,k-1}, x_{3,k-1}) \\ f_3(\boldsymbol{x}_k) \end{pmatrix} \tag{19}$$

where f_2 and f_3 represent the functions in the RHS of (14) and (15), respectively, and T_s has been incorporated into the model parameters $\boldsymbol{\theta}$. The structure of γ was consists of 3 inputs, a hidden layer with 5 hidden units and ReLu activation functions, and a single output neuron with linear activation.

The experiment results are summarized in Figures 1 and 2. Figure 1 depicts the Lorenz Attractor ground-truth and the

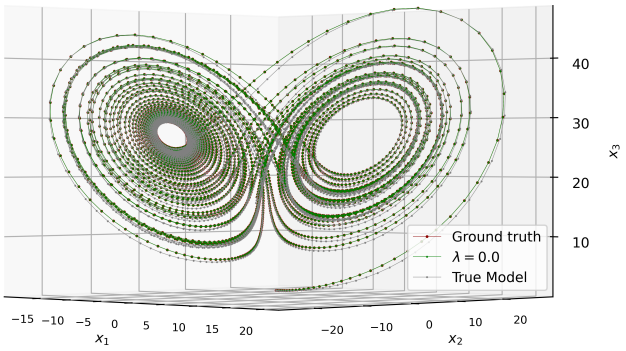

Fig. 1. Single simulation demonstrating the effectiveness of the APBM while tracking a chaotic system.

estimation using the APBM with $\lambda = 0$ and the estimation using the true model. We considered only $\lambda = 0$ for the APBM since in this case we completely replaced the ODE corresponding to the state x_1, and therefore any $\bar{\boldsymbol{\theta}}$ satisfying $g\left(f(\boldsymbol{x}_k), \boldsymbol{x}_k; \boldsymbol{\theta} = \bar{\boldsymbol{\theta}}\right) = f(\boldsymbol{x}_k)$ would be the true ODE. Analyzing Fig. 1 we can observe that, even under noisy observations, both filtering processes were able to approximate the chaotic system reasonably well. Although at first sight it seems that the APBM better approximates the Attractor's ground truth, the RMSE over MC realizations depicted in Fig. 2 shows a small decrease in performance of APBM with respect to the true model.

B. Application to target tracking

State estimation is widely used in many navigation and tracking related applications [5], which would benefit from the proposed data augmentation models. To test the discussed approach, we consider a two-dimensional target tracking application with additive constant velocity [31] and sinusoidal [24] terms. Measurements from two collocated sensors measuring received signal strength (RSS) and bearings were considered:

$$\mathbf{y}_k = \begin{pmatrix} 10\log_{10}\left(\frac{\Psi_0}{\|\mathbf{p}_0 - \mathbf{p}_k\|^q}\right) \\ \angle(\mathbf{p}_0, \mathbf{p}_k) \end{pmatrix} + \mathbf{n}_k \ , \tag{20}$$

with $\mathbf{p}_0$ being the position of the sensors, $\mathbf{p}_k$ the unknown position of the target, $10\log_{10}(\Psi_0) = 30$ dBm, $q = 2.2$

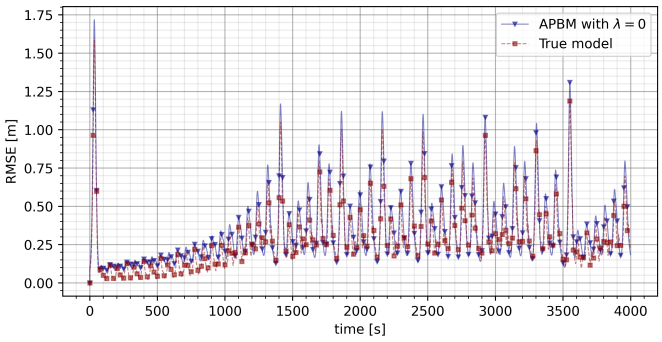

Fig. 2. Result of training on 100 Monte Carlo simulations. Average RMSE for the true model (blue triangles) is compared to the estimations of the APBM approach (orange squares). Note that the APBM is performing worse than the true model in the first 10^3 seconds but the performance is approaching the true model performance as the parameters of the APBM are being trained. RMSE across different Monte Carlo simulations are similar due to the usage of same the ρ, β, and σ parameters throughout the simulations.

the path loss exponent, $\angle(\mathbf{p}_0, \mathbf{p}_k)$ denoting the angle between locations $\mathbf{p}_0$ and $\mathbf{p}_k$ in radians, and $\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \text{diag}(1, 0.1))$ the measurement noise. Sensors were located at the origin of the coordinate system, $\mathbf{p}_0 = (0, 0)^\top$.

The dynamics of the target were simulated from

$$\begin{aligned} \mathbf{x}_k &= (\boldsymbol{F} + \boldsymbol{G}_{k-1})\mathbf{x}_{k-1} + \boldsymbol{M}\mathbf{u}_{k-1} \ , \\ \Omega_k &= \Omega_{k-1} + v_{k-1} \end{aligned} \tag{21}$$

with

$$\boldsymbol{F} = \begin{pmatrix} 1 & T_s & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T_s \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$\boldsymbol{G}_{k-1} = \begin{pmatrix} 0 & \frac{\sin\Omega_{k-1}T_s}{T_s} & 0 & -\frac{1-\cos\Omega_{k-1}T_s}{\Omega_{k-1}} \\ 0 & \cos\Omega_{k-1}T_s & 0 & -\sin\Omega_{k-1}T_s \\ 0 & \frac{1-\cos\Omega_{k-1}T_s}{\Omega_{k-1}} & 0 & \frac{\sin\Omega_{k-1}T_s}{\Omega_{k-1}} \\ 0 & \sin\Omega_{k-1}T_s & 0 & \cos\Omega_{k-1}T_s \end{pmatrix}$$

$\mathbf{x}_k = (x_k, \dot{x}_k, y_k, \dot{y}_k)^\top$ being a state vector, composed of the two-dimensional position ($\mathbf{p}_k = (x_k, y_k)^\top$) and velocity of the target ($\dot{\mathbf{p}}_k = (\dot{x}_k, \dot{y}_k)^\top$), respectively, and Ω_k being an angle state. In (21), $T_s = 1$s is the sampling period and $\mathbf{u}_k \sim \mathcal{N}(\mathbf{0}, 0.1 \cdot \mathbf{I})$ is the process noise for vector of position and velocities, while $v_k \sim \mathcal{N}(0, 0.1)$ is the process noise for the angle Ω_k . The true trajectory was initialized at

$$\mathbf{x}_0 = (100, 100, 0, 0)^\top \tag{22}$$

and the estimated $\hat{\mathbf{x}}_0$ was drawn from a Gaussian distribution with mean $\mathbf{x}_0$ and covariance diag $(0.1, 0.1, 0.01, 0.01)$.

With this setup, the trajectory described by a target moving for $T = 500$ seconds was generated. Results were averaged over 100 independent Monte Carlo runs and trajectories, and thus the results are trajectory-independent.

To test the capability of APBM models to model unknown parcels of the model, we augment a constant velocity model with a neural network. More precisely we consider:

$$g(f(\boldsymbol{x}_{k-1}), \boldsymbol{x}_{k-1}; \boldsymbol{\theta}) = \boldsymbol{F}\boldsymbol{x}_{k-1} + \gamma(\boldsymbol{x}_{k-1}; \boldsymbol{\theta}) \tag{23}$$

where $\gamma(\cdot)$ is a neural network with one hidden layer with 5 neurons and ReLu activation function, an output layer with 4 neurons and linear activation, and parameterized by $\boldsymbol{\theta}$, leading to a total of 49 parameters including bias terms. As showed in (21) the term $\boldsymbol{F}\boldsymbol{x}_{k-1}$ is a simple constant velocity model.

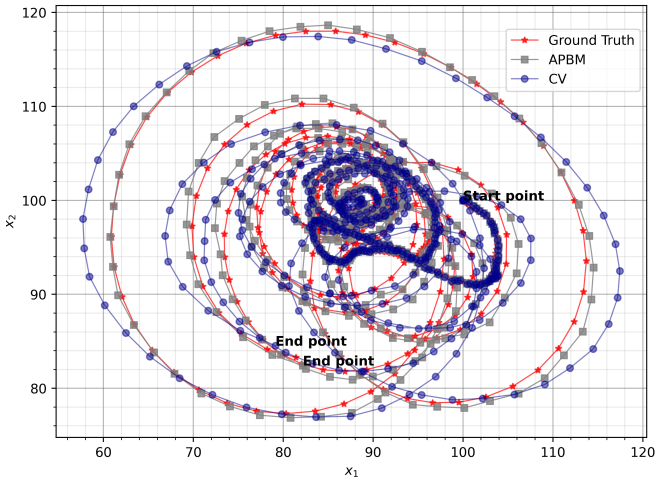

Fig. 3. Comparison of APBM and CV (constant velocity model) for a single simulation. APBM with $\lambda = 10$ (gray squares) tracks the ground truth path (red stars) with a smaller error compared to constant velocity model (CV) (blue squares).

Figure 3 presents one realization of the experiment where the ground-truth and estates estimates with APBM (with $\lambda = 10$) and constant velocity (CV) model are depicted. It can be observed that the APBM approach can track with a smaller error due to the flexibility acquired with the NN component. As expected, the improvement of the APBM with respect to the CV model can be especially noticed in sharper curves. Figure 4 depicts another realization now including APBM with different λ values. While the performance of $\lambda \in [0.01, \ldots, 10]$ can be hardily distinguished, the estimations obtained using the CV model and APBM with $\lambda = 10^6$ overlap and are clearly worse. This behavior can be more consistently demonstrated in Fig. (5) where the evolution of the RMSE computed over MC realizations for CV and APBM, with different values of λ, models are shown.

Further analyzing Fig. 5, we notice that $\lambda = 0.01$, $\lambda = 0.1$, and $\lambda = 10$ yielded to similar RMSE reaching RSMEs under 2m for $t = 500$s, that $\lambda = 0$ led to the worse result with RMSE above 5m, and, as expected, the RMSE for the APBM with very high λ and the CV models are almost identical, with RMS just under 3m. These results indicate that including the constraints over the neural network parameters may lead to improved performance over the unconstrained version and that $\lambda \to \infty$ leads to the CV model as expected.

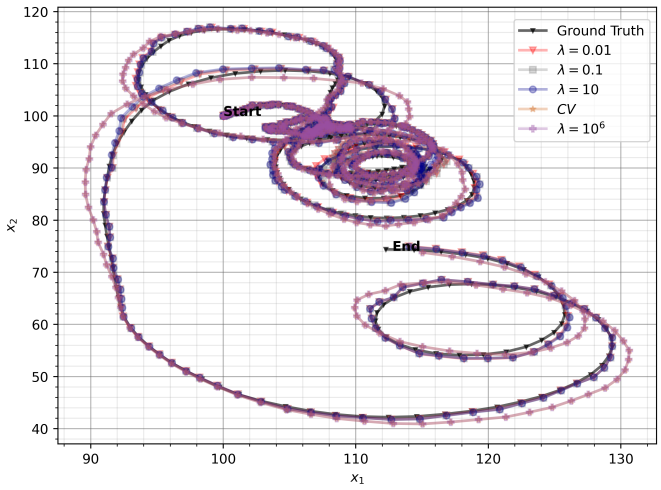

Fig. 4. Result of a single tracking simulation emphasizing the contribution of λ on the tracking performance. Black line is the ground truth. $\lambda = 0.01$ (red triangles), $\lambda = 0.1$ (gray squares), and $\lambda = 10$ (blue circles) are in a range for λ that results in low RMSE tracking. $\lambda = 10^6$ (purple pluses) and CV (orange squares) are overlapping.

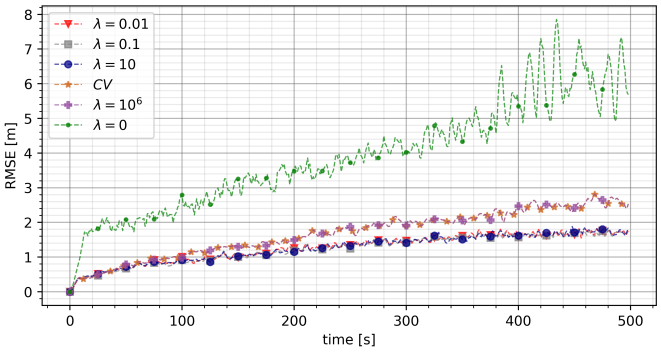

Fig. 5. Result of training on 100 Monte Carlo simulations. Average RMSE for Monte Carlo simulations using different λ values are reported. Dashed lines are representing estimated values. RMSE values for the constant velocity model (orange stars) and APBM regularized with $\lambda = 10^6$ (purple plus) are overlapping.

Finally, we observed the effect of λ in the evolution of the neural network parameters. We utilized the Eq. (12) to calculate the mean of the variance of parameters at a step k for all simulations. Figure 6 demonstrates that increasing the λ forced the parameters to remain close to 0 and we can regularize the parameters by adjusting λ accordingly.

VI. Conclusion

In this paper, we present a systematic neural augmentation approach centered on physical-based models. The motivation for such an approach is rooted in the desire to produce meaningful and interpretable results while making simplistic models more flexible and capable of adapting to different operation scenarios. Furthermore, we presented a simplistic, yet efficient, strategy to control the neural network contribution to the overall model. We showed with simulations that i) the whole of such constraint over the NN parameters is not

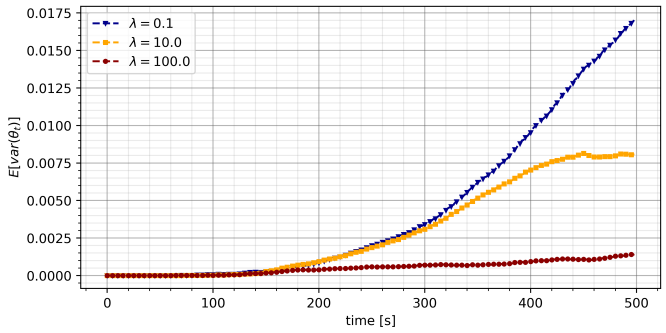

Fig. 6. Mean of variances of weights across all 100 Monte Carlo simulations. Eq. 12 is used for all lines. The x-axis demonstrates the time and the y-axis demonstrates the mean of variance of all weights at a time step. Weights are initialized as zeros, and the evolution of the weights in time for different λ values is reported. Note that weights trained with a greater regularization term λ have a smaller variance.

negligible since it led to better results when compared with the unconstrained version; and $ii)$ that our intuition regarding λ, discussed in Section III, was correct.

REFERENCES

[1] T. Imbiriba, J. C. M. Bermudez, C. Richard, and J.-Y. Tourneret, "Nonparametric detection of nonlinearly mixed pixels and endmember estimation in hyperspectral images," *IEEE Transactions on Image Processing*, vol. 25, no. 3, pp. 1136–1151, 2016.

[2] Ricardo A Borsoi, Tales Imbiriba, Pau Closas, José Carlos M Bermudez, and Cédric Richard, "Kalman filtering and expectation maximization for multitemporal spectral unmixing," *IEEE Geoscience and Remote Sensing Letters*, 2020.

[3] Parul Arora, Himanshu Kumar, and Bijaya Ketan Panigrahi, "Prediction and analysis of covid-19 positive cases using deep learning models: A descriptive case study of india," *Chaos, Solitons & Fractals*, vol. 139, pp. 110017, 2020.

[4] D. Dardari, P. Closas, and P.M. Djuric, "Indoor Tracking: Theory, Methods, and Technologies," *IEEE Trans. on Vehicular Technology*, vol. 64, no. 4, pp. 1263–1278, April 2015.

[5] Jindrich Duník, Sanat K Biswas, Andrew G Dempster, Thomas Pany, and Pau Closas, "State estimation methods in navigation: overview and application," *IEEE Aerospace and Electronic Systems Magazine*, vol. 35, no. 12, pp. 16–31, 2020.

[6] Tales Imbiriba and Pau Closas, "Enhancing particle filtering using gaussian processes," in *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*. IEEE, 2020, pp. 1–7.

[7] L. Haoqing, R. A. Borsoi, T. Imbiriba, P. Closas, J. C. M. Bermudez, and D. Erdoğmuş, "Model-based deep autoencoder networks for nonlinear hyperspectral unmixing," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2022.

[8] T. Imbiriba, P. Wu, G. LaMountain, D. Erdoğmuş, and P. Closas, "Recursive Gaussian processes and fingerprinting for indoor navigation," in *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2020, pp. 933–940.

[9] Dimitris C Psichogios and Lyle H Ungar, "A hybrid neural network-first principles approach to process modeling," *AIChE Journal*, vol. 38, no. 10, pp. 1499–1511, 1992.

[10] Ahmet Demirkaya, Tales Imbiriba, Kyle Lockwood, Sumientra Rampersad, Elie Alhajjar, Giovanna Guidoboni, Zachary Danziger, and Deniz Erdogmus, "Cubature kalman filter based training of hybrid differential equation recurrent neural network physiological dynamic models," in *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. 2021-11-1, 2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), IEEE.

[11] Simon Haykin, *Kalman filtering and neural networks*, vol. 47, John Wiley & Sons, 2004.

[12] L. Chin, "Application of neural networks in target tracking data fusion," *IEEE Trans. Aerosp. Electron. Syst*, vol. 30, no. 1, pp. 281–287,, 1994.

[13] V. Vaidehi, N. Chitra, C.N. Krishnan, and M. Chokkalingam, "Neural network aided Kalman filtering for multitarget tracking applications," in *Proc. 1999 IEEE Radar Conf*, 1999.

[14] M.W. Owen and A.R. Stubberud, "NEKF IMM tracking algorithm," in *Proc. Signal and Data Proc. Small Targets*, 2003.

[15] N. Shaukat, A. Ali, M.J. Iqbal, M. Moinuddin, and P. Otero, "Multi-sensor fusion for underwater vehicle localization by augmentation of RBF neural network and error-state Kalman filter," *Sensors*, vol. 21, pp. 1149, 2021.

[16] C. Gao, J. Yan, S. Zhou, P.K. Varshney, and H. Liu, "Long short-term memory-based deep recurrent neural networks for target tracking," *Inf. Sci*, vol. 502, pp. 279–296,, 2019.

[17] S. Jung, I. Schlangen, and A. Charlish, "A mnemonic Kalman filter for non-linear systems with extensive temporal dependencies," *IEEE Signal Processing Letters*, 2020.

[18] C. Zhao, L. Sun, Z. Yan, G. Neumann, T. Duckett, and R. Stolkin, "Learning kalman network: A deep monocular visual odometry for on-road driving," *Robot. Auton. Syst*, vol. 121, pp. 103234, 2019.

[19] Rahul G Krishnan, Uri Shalit, and David Sontag, "Deep kalman filters," *arXiv preprint arXiv:1511.05121*, 2015.

[20] Guy Revach, Nir Shlezinger, Xiaoyong Ni, Adria Lopez Escoriza, Ruud JG Van Sloun, and Yonina C Eldar, "Kalmannet: Neural network aided kalman filtering for partially known dynamics," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1532–1547, 2022.

[21] K. Thormann, F. Sigges, and M. Baum, "Learning an object tracker with a random forest and simulated measurements," in *Proc. 20th Int. Conf. Inf.* 2017, p. 1–4, Fusion.

[22] B. Zhai, W. Yi, M. Li, H. Ju, and L. Kong, "Data-driven XGBoost-based filter for target tracking," *J. Eng*, , no. 20, pp. 6683–6687,, 2019.

[23] Peng Wu, Tales Imbiriba, Gerald LaMountain, Jordi Vilà-Valls, and Pau Closas, "Wifi fingerprinting and tracking using neural networks," in *Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019)*, 2019, pp. 2314–2324.

[24] Ienkaran Arasaratnam and Simon Haykin, "Cubature Kalman filters," *IEEE Transactions on automatic control*, vol. 54, no. 6, pp. 1254–1269, 2009.

[25] Dan Simon, "Kalman filtering with state constraints: a survey of linear and nonlinear algorithms," *IET Control Theory & Applications*, vol. 4, no. 8, pp. 1303–1318, 2010.

[26] Simo Särkkä, *Bayesian filtering and smoothing*, Number 3. Cambridge University Press, 2013.

[27] Pau Closas, Jordi Vila-Valls, and Carles Fernández-Prades, "Computational complexity reduction techniques for quadrature Kalman filters," in *2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. IEEE, 2015, pp. 485–488.

[28] M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on signal processing*, vol. 50, no. 2, pp. 174–188, 2002.

[29] Eric A Wan, Rudolph Van Der Merwe, and Simon Haykin, "The unscented Kalman filter," *Kalman filtering and neural networks*, vol. 5, no. 2007, pp. 221–280, 2001.

[30] Edward N. Lorenz, "Deterministic nonperiodic flow," *Journal of Atmospheric Sciences*, vol. 20, no. 2, pp. 130 – 141, 1963.

[31] P. Closas and C. Fernández-Prades, "Particle filtering with adaptive number of particles," in *2011 Aerospace Conference*. IEEE, 2011, pp. 1–7.