MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Moving Horizon Estimation for Digital Twins using Deep Autoencoders

Chakrabarty, Ankush; Vinod, Abraham P.; Mansour, Hassan; Bortoff, Scott A.; Laughman, Christopher R.

TR2023-088 July 11, 2023

Abstract

Digital twins have emerged in recent years as black-box, software-based simulation tools that can mirror the behavior of complex dynamical systems. Digital twin simulations generate the same outputs as the target system using internal states; however, these states are not readily available online from the real system. In this paper, we develop a data-driven moving horizon estimation framework capable of using online noisy measurements of the real system in order to estimate digital twin states. Our framework combines the high expressiveness of deep autoencoders with a moving horizon state estimator that accurately predicts the internal state of the black-box digital twin without access to an analytical model of the system dynamics. We demonstrate that our approach outperforms extended and Koopman Kalman filter solutions on a benchmark reverse van der Pol oscillator example.

World Congress of the International Federation of Automatic Control (IFAC) 2023

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Moving Horizon Estimation for Digital Twins using Deep Autoencoders

Ankush Chakrabarty * **Abraham P. Vinod** * **Hassan Mansour** *
Scott A. Bortoff * **Christopher R. Laughman** *

* *Mitsubishi Electric Research Laboratories, Cambridge, MA 02139, USA. (Corresponding author e-mail: achakrabarty@ieee.org).*

Abstract: Digital twins have emerged in recent years as black-box, software-based simulation tools that can mirror the behavior of complex dynamical systems. Digital twin simulations generate the same outputs as the target system using internal states; however, these states are not readily available online from the real system. In this paper, we develop a data-driven moving horizon estimation framework capable of using online noisy measurements of the real system in order to estimate digital twin states. Our framework combines the high expressiveness of deep autoencoders with a moving horizon state estimator that accurately predicts the internal state of the black-box digital twin without access to an analytical model of the system dynamics. We demonstrate that our approach outperforms extended and Koopman Kalman filter solutions on a benchmark reverse van der Pol oscillator example.

Keywords: Learning, state observers, nonlinear systems, Koopman operator, system identification, black-box models.

1. INTRODUCTION

Nonlinear state observer design generally requires knowledge of system dynamics and uses analytical models whose mathematical structure can be exploited for synthesis (Simon, 2006; Rajamani, 1998; Chakrabarty et al., 2017). In the absence of a complete mathematical model, operational data collected from the system can be utilized to learn the unmodeled dynamics, for instance, using neural networks (Lei et al., 2021; Zhao et al., 2022; Chakrabarty et al., 2019) or Bayesian optimization (Chakrabarty and Benosman, 2021; Riva et al., 2022) and subsequently compute state estimator gains.

Advancements in modeling and numerical computing has resulted in the construction of high-fidelity simulation software, injected with physics-informed modules designed by domain-experts, with the capacity to mirror the behavior of complex dynamical systems; these software tools are often referred to as *digital twins* (Grieves and Vickers, 2017). Digital twins have proven effective virtual proxies for numerous application areas, including: mechatronics (Classens et al., 2021; Xu et al., 2021), energy systems (Zhan et al., 2022; Li et al., 2020; Bortoff and Laughman, 2019), and vehicles (Bhatti et al., 2021; Riva et al., 2022). A digital twin can be used for simulating a wide range of operating conditions (Tao et al., 2018), thereby enabling one to collect simulation data for observer synthesis. Unfortunately, there are two challenges to using digital twins for estimator design. First, a digital twin is usually composed of modules that are 'black-box', made opaque for reasons such as protection of proprietary information, reduction of code complexity, or improvement of user interfacing and experience. Therefore, a digital twin may not admit a simple model structure with which one can perform classical model-based estimator design. Second, a digital twin typically contains an internal representation of a 'state' of the dynamical system being mirrored, and subsequent controller design and closed-loop verification is performed using such an internal state. Therefore, state estimators for digital twins must be designed to estimate this specific internal state; this is in contrast to other data-driven state estimation methods where the state representation is at the discretion of the designer. Therefore, it is imperative to design state estimation algorithms that can integrate with black-box simulation environments and generate estimates of pre-defined digital twin states purely from output data online.

In this paper, we adopt a deep autoencoder (AE)-based approach to directly learn a predictive model from simulation data without requiring transparency of digital twin modules, and to preserve the internal state representation of the digital twin by training on a dataset that contains both state and output trajectories. Online, when only measured outputs are available from the true system, we adopt a moving horizon estimator framework with the AE-based predictive model to estimate the digital twin state directly using the measurements. Note that this problem setup is specific to simulator-driven design, where states and outputs are available during training since these are generated by the twin, and only system outputs are available online from which the prescribed state must be reconstructed. We reiterate that this different from standard data-driven modeling and estimation frameworks, where only output data is available during training and the concept of a state is a design choice (for example, a latent variable of an autoencoder).

AEs have recently been used to learn transformations that 'lift' the dynamics of the nonlinear system to a latent space where a finite-dimensional linear state-space model can

provide a satisfactorily accurate predictive model (Lusch et al., 2018). The benefit of AE-based approaches, therefore, is that the observer design can leverage the linear system representation and classical observer gain tuning methods can be applied. Since the decoder of the AE is an inverse of the encoder, one can tractably project the dynamics to and from the latent space, respectively. In fact, the effectiveness of learning lifting transformations in nonlinear system identification has been reported in Masti and Bemporad (2021) and Beintema et al. (2021). These lifting approaches are linked to Koopman operator-theoretic designs, where the linear state-transition operator learned by the AE represents a finite approximation of the Koopman operator (Mauroy et al., 2020). Note that the aforementioned references use AEs for model identification with no explicit investigation of state estimation. In fact, to the best of our knowledge, only (Forgione et al., 2022) describes a deadbeat observer approach for initial state estimation, but this is only for the purposes of initializing an encoder network state for forward predictions, not for sustained online state estimation. Koopman Kalman filtering (KF) is described in Surana and Banaszuk (2016a); Surana (2016) where the authors describe a method for transforming the given nonlinear system to a Koopman canonical form which consequently allows linear observer design. However, this method requires some model knowledge and specific structure to be applicable, and relies on kernel methods for the case when unmodeled dynamics exist, which does not scale well to large, high-dimensional datasets.

Our main *contributions* are: (i) the proposal of a data-driven framework for estimating digital twin states from online system outputs; (ii) using a novel combination of deep autoencoders trained on simulation data and moving horizon estimators capable of performing without access to analytical models of the system dynamics; and, (iii) demonstrating the potential of the proposed approach by comparing against extended and Koopman KFs (which assume some model knowledge) on a benchmark example. The rest of the paper is organized as follows. In Section 2, we present the problem statement, and the proposed AE-based moving horizon estimator is described in Section 3, along with a discussion about hyperparameter selection. In Section 4, we demonstrate the effectiveness of the proposed algorithm on a benchmark example, and perform an ablation study. Finally, we conclude in Section 5.

2. PRELIMINARIES

We consider black-box simulators such as digital twins with internal dynamics

$$x_{t+1} = f(x_t), \tag{1a}$$
$$y_t = h(x_t), \tag{1b}$$

where $t \in \mathbb{N}$ is the time-index, $x \in \mathbb{R}^{n_x}$ is the simulator state, $y \in \mathbb{R}^{n_y}$ is the measured output, and f, h are unmodeled state update and output functions, respectively. The dynamics (1) are unknown to the designer, but we assume f and h exist and are well-defined. It is assumed that the system response is bounded from any initial condition $x \in \mathbb{R}^{n_x}$, and that f and h are such that the state x is observable from the output y.

Unlike classical input-output system identification where the concept of a state is selected by the designer, in simulator-driven identification the state-space is fixed by the digital twin simulator, and the outputs are nonlinear functions of these prescribed states. Therefore, we assume access to a bounded training dataset $\mathbf{D}_{\text{train}} \triangleq \{(x_t, y_t)\}_{t \subset \mathbb{N}}$ comprising state-output pairs collected by offline simulations of the digital twin; the data is assumed to have been collected by persistently exciting the system (1), which is a standard assumption for model identification (Ljung, 1998). In general, this dataset may comprise trajectories from different initial conditions. State-feedback controllers designed on these digital twin simulators require the simulator states x online, but the states are typically not measured, and therefore have to be estimated from only the measurement outputs y.

The *objective* of this paper is to estimate the simulator states $\{x_t\}_{t \geq 0}$ online from the outputs $\{y_t\}_{t \geq 0}$, based on a representation of the dynamics (1) learned using the dataset $\mathbf{D}_{\text{train}}$. To this end, our proposed approach involves: (i) learning a deep autoencoder model that is capable of replicating the state and output dynamics of (1), and (ii) constructing a moving horizon estimator that leverages the autoencoder model to generate an estimate of the current state based on a window of past measurements.

3. AUTOENCODER-BASED MOVING HORIZON ESTIMATOR (AE-MHE)

This section proposes an *autoencoder-based moving horizon estimator* (AE-MHE) to achieve data-driven nonlinear state estimation. We first describe the architecture of the autoencoders used and detail the training loss used. Next, we describe the optimization problem formulation for the moving horizon estimator. We conclude with a detailed discussion of various hyperparameters associated with our approach.

3.1 Architecture

In order to construct our state estimator, we first propose a deep autoencoder-based dynamical model, illustrated in Fig. 1. The input to the autoencoder is the simulator state x_t, which is passed through an encoder to compute a latent encoding $\psi_t \in \mathbb{R}^{n_\psi}$. The latent encoding is updated using a linear state-transition operator $\mathcal{A} : n_\psi \mapsto n_\psi$. The updated latent ψ_{t+1} is passed through a state-decoder $\mathcal{D}_x$ that returns the state estimate $\bar{x}_{t+1}$. A separate decoder $\mathcal{D}_y$ generates an estimate $\bar{y}_t$ of the output y_t based on the latent variable ψ_t. That is,

$$\psi_t = \mathcal{E}_x(x_t), \tag{2a}$$
$$\psi_{t+1} = \mathcal{A}(\psi_t), \tag{2b}$$
$$\bar{x}_{t+1} = \mathcal{D}_x(\psi_{t+1}) = \mathcal{D}_x \circ \mathcal{A} \circ \mathcal{E}_x(x_t), \tag{2c}$$
$$\bar{y}_t = \mathcal{D}_y(\psi_t) \quad = \mathcal{D}_y \circ \mathcal{E}_x(x_t). \tag{2d}$$

Note that one can enforce $\mathcal{A}$ and D_y to be state-transition and output matrices (respectively) by selecting them to be linear layers with zero bias.

The state-transition operator $\mathcal{A}$ can be viewed as a Koopman operator that enables a linear description of the nonlinear dynamics (1) in the high-dimensional latent space

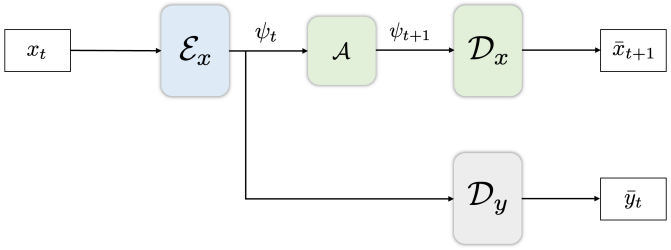

Fig. 1. Deep autoencoder for proposed state estimation.

$\mathbb{R}^{n_\psi}$ Lusch et al. (2018); Mauroy et al. (2020). A shortcoming of Koopman-based system identification is that the latent space that enables a linear description may be hard to identify in practice, incurring modeling errors. AE-based dynamical models use the training data set to define the latent space with the goal of reducing such modeling errors.

3.2 Training Loss

Training the deep autoencoder involves computing the weights of the encoder $\mathcal{E}_x$, the two decoders $\mathcal{D}_x$ and $\mathcal{D}_y$, and the state transition operator $\mathcal{A}$. To this end, we use a multi-step reconstruction loss with regularization. Formally, at index t we can select the current state x_t as the input and a window of length $\hat{H}$ of current/future states

$$X_{t+1:t+\hat{H}} = \{x_{t+1}, x_{t+2}, \cdots, x_{t+\hat{H}}\},$$

along with the future outputs

$$Y_{t:t+\hat{H}-1} = \{y_t, y_{t+1}, \cdots, y_{t+\hat{H}-1}\},$$

using the data in $\mathbf{D}_{\text{train}}$. Subsequently, with x_t, we can generate estimates $\bar{X}_{t+1:t+\hat{H}}$ and $\bar{Y}_{t:t+\hat{H}-1}$ using the autoencoder (2).

We will use the following notation to describe the training loss function. Let $Q_{t_1:t_2} = \{q_{t_1}, \cdots, q_\tau, \cdots, q_{t_2}\}$ denote any tensor with $q_\tau \in \mathbb{R}^{n_q}$, along with its estimate $\hat{Q}_{t_1:t_2}$. Then, $\mathsf{MSE}(Q_{t_1:t_2}, \hat{Q}_{t_1:t_2}) \triangleq \frac{1}{n_q(t_2-t_1)} \sum_{\tau=t_1}^{t_2} \|q_\tau - \hat{q}_\tau\|_2^2$.

For training, we minimize the loss function

$$\ell = \ell_{\text{recon}} + \ell_{\text{pred,X}} + \ell_{\text{pred,Y}} + \lambda_{\text{reg}}\ell_{\text{reg}}, \tag{3}$$

where the first three terms are mean-squared-error (MSE) losses, with

$$\ell_{\text{recon}} = \mathsf{MSE}\left(x_t, \mathcal{D}_x \circ \mathcal{E}_x(x_t)\right),$$

denoting the reconstruction loss of the autoencoder without a state update,

$$\ell_{\text{pred,X}} = \mathsf{MSE}\left(X_{t+1:t+\hat{H}}, \bar{X}_{t+1:t+\hat{H}}\right),$$

denoting the multi-step prediction error on the states of the system, and

$$\ell_{\text{pred,Y}} = \mathsf{MSE}\left(Y_{t:t+\hat{H}-1}, \bar{Y}_{t:t+\hat{H}-1}\right)$$

is the multi-step prediction error computed over the outputs. As suggested in Lusch et al. (2018), we also penalize the $\mathcal{A}$ operator via the $\mathcal{L}_1$ regularization loss

$$\ell_{\text{reg}} = \|\mathcal{A}\|_2$$

with regularization coefficient $\lambda_{\text{reg}} > 0$ to inject stability to the learned $\mathcal{A}$.

3.3 Moving Horizon Estimation

We propose a moving horizon estimator for state estimation using the autoencoder model equations (2).

At each time index t, given a H-length window of measurements $Y_{t:t+H}$, we generate an estimate of the state x_t by solving the following optimization problem,

$$\min_{X_{t-H:t}} \quad \sum_{\tau=0}^{H-1} \|\delta_\tau^x\|_{Q_f}^2 + \sum_{\tau=0}^{H} \|\delta_\tau^y\|_{R_f}^2 \tag{4a}$$

subject to:

$$\bar{y}_{t-\tau} = \mathcal{D}_y \circ \mathcal{E}_x(x_{t-\tau}), \qquad \forall\, \tau \in \mathbb{N}_{0:H} \tag{4b}$$

$$\bar{x}_{t-\tau+1} = \mathcal{D}_x \circ \mathcal{A} \circ \mathcal{E}_x(x_{t-\tau}), \quad \forall\, \tau \in \mathbb{N}_{0:H} \tag{4c}$$

$$\delta_\tau^x = \bar{x}_{t-\tau} - x_{t-\tau}, \tag{4d}$$

$$\delta_\tau^y = \bar{y}_{t-\tau} - y_{t-\tau}, \tag{4e}$$

where $\mathbb{N}_{a:b}$ is the set of natural numbers between and including $a, b \in \mathbb{N}$. To solve (4), we generate a sequence of state estimates $\{x_{t-\tau}\}_{\tau=0}^{H}$ that minimizes a weighted sum of the state and output prediction errors. We define the state prediction error in (4d) using the deviations between the generated state estimates and the one-step predictions using (4c). We define the output prediction error in (4e) using the deviation between the supplied output measurements and the outputs estimated using (4b). We use $Q_f \succeq 0$ and $R_f \succ 0$ in (4a) are weighting matrices to enforce preferences over specific state dimensions.

Since the encoder and decoder transformations may be nonlinear, (4) is a nonlinear least squares problem. We solve (4) using limited memory BFGS (Liu and Nocedal, 1989) implemented using automatic differentiation that leverages the `PyTorch` library (Paszke et al., 2019a) and `Pytorch-minimize` (Feinman, 2021). Alternatively, one could utilize standard nonlinear least squares techniques like Levenberg–Marquardt algorithms (Boyd and Vandenberghe, 2018) or nonlinear moving horizon estimators using the identified nonlinear dynamics (Rawlings et al., 2020).

3.4 Remarks on hyperparameter selection

The main hyperparameters of the AE are the number of layers in the encoder and the decoder, the type of state-transition operator, and the latent dimension. Since the encoder represents a 'lifting' transformation, it usually needs a depth of at least three layers in order to be expressive enough to represent complex nonlinear transformations; this implies the decoder(s) (which behaves as the inverse transformation of the encoder) must also be deep enough to be expressive. Classically the encoder and decoder(s) have the same depth, but this is not mandatory.

The state-transition operator $\mathcal{A}$ is defined as a linear layer with no bias, as we want to represent the dynamics by a linear system, following Koopman-theoretic arguments (Mauroy et al., 2020). The selection of the latent variable dimension is a trade-off: since the state-transition operator is a finite-dimensional approximation of the Koopman operator, one wants n_ψ to be large, so that the approximation error is small. Conversely, choosing n_ψ too large can result in increased computational expenditure during training and will result in a large $\mathcal{A}$ matrix

which may be unwieldy for linear systems design methods. Prior art such as Masti and Bemporad (2021) has shown that multi-step prediction in training is necessary for good generalization at inference, indicating a need to choose a large $\hat{H}$. We recommend choosing $\hat{H} \geq 5$.

We also briefly discuss the design choices for the parameters associated with solving (4). The optimization problem (4) has three parameters — the weighting matrices Q_f and R_f, and the window length H. The choice of weighting matrices are governed by the preference over the individual state dimensions, and the accuracy requirements over state and output predictions. We chose $Q_f = 10I_{n_x}$ and $R_f = I_{n_y}$ to emphasize higher accuracy in the moving horizon estimator. Solving (4) with longer window lengths typically improves the estimation quality at the cost of higher data requirements. Additionally, since (4) is a nonlinear optimization problem, the computational effort to solve (4) increases considerably with longer window lengths. Empirically, we observed that the choice of $H \geq 7$ provided a good compromise between these contending desiderata. We also note that the deep autoencoder can generate gradients for the objective and constraints of (4) by well-studied automatic differentiation tools. One can, therefore, utilize these gradients to accelerate the solver used to solve (4).

4. SIMULATION RESULTS

To illustrate our algorithm, we consider the reverse van der Pol oscillator

$$\begin{bmatrix} x_{1,t+1} \\ x_{2,t+1} \end{bmatrix} = \begin{bmatrix} x_{1,t} - \tau x_{2,t} \\ x_{2,t} + \tau\big(x_{1,t} - x_{2,t} + x_{1,t}^2 x_{2,t}\big) \end{bmatrix},$$
$$y_t = x_{1,t}^2 + x_{2,t},$$

as the unknown dynamical system inside a digital twin, where $\tau = 0.1$ is the sampling time. Note that this system has been previously studied in (Surana and Banaszuk, 2016b), and therefore gives us an opportunity to use the authors' proposed Koopman Kalman filter (KKF) and extended Kalman filter as baselines for comparison.

4.1 Implementation and data collection

We use an autoencoder with the architecture described in Fig. 1, with the following hyperparameters. The input layer is 2-dimensional to admit the state of the oscillator, so is the output layer; the latent dimension is selected to be one of $n_\psi = 16$, 32, and 64, to study the effect of the latent variable size on estimation performance. The encoder has a depth of 5 layers, each of which is activated by rectified linear unit (ReLU) functions, and initialized by uniform Xavier initialization; see Goodfellow et al. (2016) for more details. The number of neurons from input to latent is: 128-256-256-256-128. The state-transition layer is a linear layer with no bias, so effectively $\mathcal{A}$ is a matrix of size $n_\psi \times n_\psi$, which depends on the n_ψ we use. The state and output decoders are both five layers deep, with ReLU activations, and follow the same neuronal pattern as the encoder. While we do not provide an explicit comparison of predictive performance with the number of layers, we can report that reducing the number of layers to below four in the encoder (and therefore, both decoders to maintain parity) results in severely degraded training performance.

In order to collect data for training, we sample 200 unique, random initial conditions within $[-1, 1]^2$ and simulate the oscillator for $T = 51$ time samples, as reported in Surana and Banaszuk (2016b). The states and outputs are stored for training, and we use $\hat{H} = 10$ (i.e. ten future states and outputs are predicted) for computing the autoencoder loss function. The regularization coefficient is set to $\lambda_{\text{reg}} = 10^{-3}$. We use the Adamax solver (Kingma and Ba, 2015) for training with a step-size of 0.001, and learning rate scheduling which cuts the step size by half after every 500 training iterations with plateaued loss. Implementation is entirely in `Python 3.10` and `PyTorch` (Paszke et al., 2019b). For estimation, we select a window of length $H = 15$, (we will justify this later in this section) with $Q_f = 10I$ and $R_f = I$. The estimator optimization problem is solved using automatic differentiation using `Pytorch-minimize` (Feinman, 2021). We reiterate that the entire training process is based solely on simulation data; no model information is assumed.

4.2 Effect of latent dimension, n_ψ

For testing the moving horizon estimator, we select an initial condition distinct from any initial condition in the training set, and simulate forward on $[0, 20]$ s, with only outputs measured every 0.1 s. The output is corrupted by zero-mean, 0.01-variance Gaussian noise. Fig. 2 shows the performance of the proposed approach. The top row of subplots illustrate the evolution of the states of the system over time, with the black continuous line being the true state, and the blue, orange, and green lines indicating state estimates with an autoencoder with latent dimension 16, 32, and 64, respectively. In the lower row, the left subplot shows the noisy measured output, and the right subplot shows the norm of the state estimation error evolving over time. We observe that despite the selection of different latent dimensions and measurement noise, the proposed method works satisfactorily well and the estimation error reduces to a small neighborhood around the origin. As n_ψ decreases, the condition number of $\mathcal{A}_\psi$ increases from 1.8×10^4 for $n_\psi = 16$ to 6.0×10^4 for $n_\psi = 32$ to 1.2×10^5 for $n_\psi = 64$.

4.3 Effect of window length, H

To understand the effect of estimation performance on the length of the moving horizon window, we refer the reader to Fig. 3, where the top plot shows the sum-of-squared state estimation error with varying window lengths from $H = 3$ to $H = 20$. Corresponding statistics of time taken to compute a state estimate is provided in the lower subplot. For this numerical example, we observe that the error decreases rapidly up to $H = 7$, and then gradually till $H = 15$, after which the decrease is unremarkable. As expected, the corresponding time to compute a state estimate increases linearly in this small range of $H \in \{3, 4, \ldots, 20\}$. Therefore, a suitable trade-off can be made between computational speed and error to select H; we choose $H = 15$ to that end.

4.4 Comparison to other baselines and an ablation

We also compare our approach to the KKF and EKF reported in Surana and Banaszuk (2016b). Here, $n_\psi = 32$

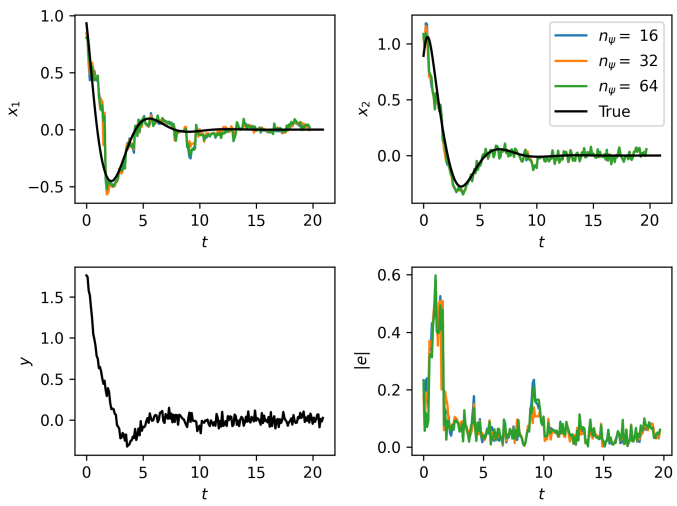

Fig. 2. Estimation of states x from noisy output y, along with estimation error norm $\|e\|$.

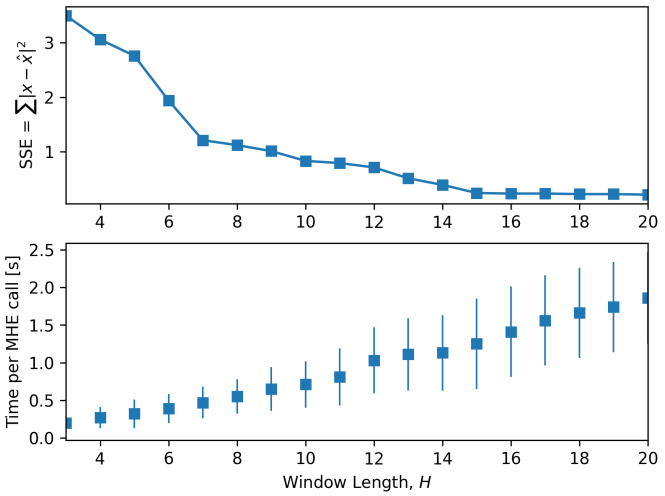

Fig. 3. Comparison of sum-squared-error and time per estimator call with varying estimator window length.

and $H = 15$. We admit that while we have tried to keep the comparison fair by maintaining the same measurement noise level, identical initial conditions and forward simulation time for testing, there are some differences. To our understanding, the prior art assumes some model knowledge/structure, which we do not. Conversely, we do not optimize code for speed, and therefore, our estimator is most likely slower than the prior art.

The results of the comparison is shown in Fig. 4, where the red and black lines are the median state estimation error norm of the EKF and the KKF, respectively; no statistics were reported. Our proposed moving horizon estimator is run 100 times to compute median and 95% confidence intervals of the state estimation error (all from the same initial condition as the KKF/EKF, but with varying noise seeds). The plots in Fig 4 show that the proposed AE-based MHE approach consistently outperforms the competitors despite the lack of a model of the dynamics.

A further comparison is made to try to understand the importance of $\mathcal{A}_\psi$, whose linearity is not exploited in the MHE optimization problem (4). To this end, we remove the $\mathcal{A}_\psi$ layer, and redo the training and estimation with this ablated network, which we refer to in Fig. 4 as AE-MHE-No-A. An important difference in this training loop is that since there is no state-transition operator $\mathcal{A}_\psi$, $\hat{H} = 1$ by default, which can limit the predictive performance. The corresponding MHE problem therefore reduces to a similar framework to the neural deadbeat estimator proposed in §2.5 of Forgione et al. (2022). While this ablated network is promising, and at times outperforms KKF and EKF, our proposed AE-MHE is superior to AE-MHE-No-A everywhere except for a small subset of time near the 40 and 60-second marks.

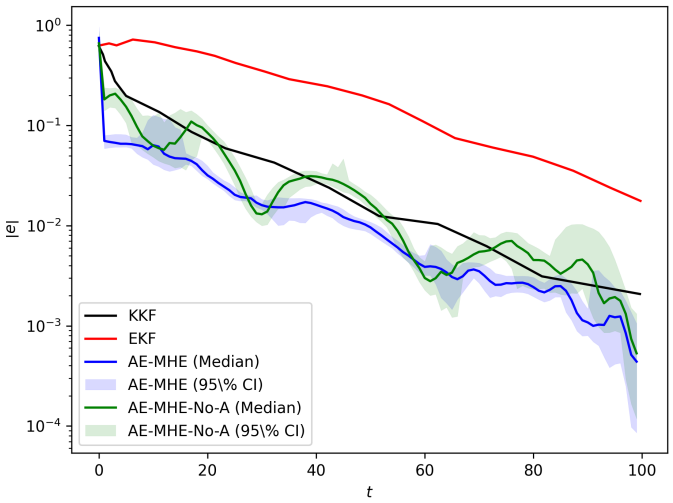

Fig. 4. Comparison of estimation error norm of AE-MHE against Koopman KF (KKF), extended KF (EKF), and ablation method AE-MHE-No-A.

5. CONCLUSIONS

We demonstrated in this work that digital twin states of complex dynamical systems can be accurately estimated from online measurements using a combination of a deep autoencoder and moving horizon state estimation. The deep autoencoder can directly learn from simulation data the mapping from the state space to the observations along with identifying a latent space where a linear, finite-dimensional, state-space model can provide a satisfactorily accurate predictive model. We demonstrate through numerical simulations that the proposed framework delivers smaller estimation error compared to state of the art extended and Koopman Kalman filtering approaches. We will investigate stability properties of the estimator in future.

REFERENCES

Beintema, G., Toth, R., and Schoukens, M. (2021). Nonlinear state-space identification using deep encoder networks. In *Learning for Dynamics and Control*, 241–250. PMLR.

Bhatti, G., Mohan, H., and Singh, R.R. (2021). Towards the future of smart electric vehicles: Digital twin technology. *Renewable and Sustainable Energy Reviews*, 141, 110801.

Bortoff, S.A. and Laughman, C.R. (2019). An extended Luenberger observer for HVAC application using FMI. In *Modelica*, 157–015.

Boyd, S. and Vandenberghe, L. (2018). *Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares*. Cambridge University Press, 1 edition.

Chakrabarty, A. and Benosman, M. (2021). Safe learning-based observers for unknown nonlinear systems using Bayesian optimization. *Automatica*, 133, 109860.

Chakrabarty, A., Corless, M.J., Buzzard, G.T., Żak, S.H., and Rundell, A.E. (2017). State and unknown input observers for nonlinear systems with bounded exogenous inputs. *IEEE Transactions on Automatic Control*, 62(11), 5497–5510.

Chakrabarty, A., Zemouche, A., Rajamani, R., and Benosman, M. (2019). Robust data-driven neuro-adaptive observers with Lipschitz activation functions. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, 2862–2867. IEEE.

Classens, K., Heemels, W.M., and Oomen, T. (2021). Digital twins in mechatronics: From model-based control to predictive maintenance. In *2021 IEEE 1st International Conference on Digital Twins and Parallel Intelligence (DTPI)*, 336–339. IEEE.

Feinman, R. (2021). Pytorch-minimize: a library for numerical optimization with Autograd. URL `https://github.com/rfeinman/pytorch-minimize`.

Forgione, M., Mejari, M., and Piga, D. (2022). Learning neural state-space models: Do we need a state estimator? *arXiv preprint arXiv:2206.12928*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT press.

Grieves, M. and Vickers, J. (2017). Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. In *Transdisciplinary perspectives on complex systems*, 85–113. Springer.

Kingma, D.P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Lei, Q., Ma, Y., Liu, J., and Yu, J. (2021). Neuroadaptive observer-based discrete-time command filtered fault-tolerant control for induction motors with load disturbances. *Neurocomputing*, 423, 435–443.

Li, W., Rentemeister, M., Badeda, J., Jöst, D., Schulte, D., and Sauer, D.U. (2020). Digital twin for battery systems: Cloud battery management system with online state-of-charge and state-of-health estimation. *Journal of energy storage*, 30, 101557.

Liu, D.C. and Nocedal, J. (1989). On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1), 503–528.

Ljung, L. (1998). System Identification. In *Signal Analysis And Prediction*, 163–173. Springer.

Lusch, B., Kutz, J.N., and Brunton, S.L. (2018). Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1), 1–10.

Masti, D. and Bemporad, A. (2021). Learning nonlinear state–space models using autoencoders. *Automatica*, 129, 109666.

Mauroy, A., Susuki, Y., and Mezić, I. (2020). *Koopman Operator In Systems And Control*. Springer.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019a). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 8024–8035.

Paszke, A., Gross, S., et al. (2019b). PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, 8024–8035. Curran Associates, Inc.

Rajamani, R. (1998). Observers for Lipschitz nonlinear systems. *IEEE transactions on Automatic Control*, 43(3), 397–401.

Rawlings, J., Mayne, D., and Diehl, M. (2020). *Model predictive control: theory, computation and design*. Nob Hill Publishing, LLC, Santa Barbara, 2nd edition edition.

Riva, G., Formentin, S., Corno, M., and Savaresi, S.M. (2022). Simulator-in-the-loop state estimation for vehicle dynamics control: theory and experiments. *arXiv preprint arXiv:2204.06259*.

Simon, D. (2006). *Optimal State Estimation: Kalman, H-infinity, and Nonlinear Approaches*. John Wiley & Sons.

Surana, A. (2016). Koopman operator based observer synthesis for control-affine nonlinear systems. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, 6492–6499. IEEE.

Surana, A. and Banaszuk, A. (2016a). Linear observer synthesis for nonlinear systems using Koopman operator framework. *IFAC-PapersOnLine*, 49(18), 716–723.

Surana, A. and Banaszuk, A. (2016b). Linear observer synthesis for nonlinear systems using koopman operator framework. *IFAC-PapersOnLine*, 49, 716–723. doi: 10.1016/j.ifacol.2016.10.250.

Tao, F., Zhang, H., Liu, A., and Nee, A.Y. (2018). Digital twin in industry: State-of-the-art. *IEEE Transactions on industrial informatics*, 15(4), 2405–2415.

Xu, W., Cui, J., Li, L., Yao, B., Tian, S., and Zhou, Z. (2021). Digital twin-based industrial cloud robotics: Framework, control approach and implementation. *Journal of Manufacturing Systems*, 58, 196–209.

Zhan, S., Wichern, G., Laughman, C., Chong, A., and Chakrabarty, A. (2022). Calibrating building simulation models using multi-source datasets and meta-learned Bayesian optimization. *Energy and Buildings*, 270, 112278.

Zhao, E., Yu, J., Liu, J., and Ma, Y. (2022). Neuroadaptive dynamic surface control for induction motors stochastic system based on reduced-order observer. *ISA transactions*, 128, 318–328.