Physics-Constrained Deep Autoencoded Kalman Filters for Estimating Vapor Compression System States

Deshpande, Vedang M.; Chakrabarty, Ankush; Vinod, Abraham P.; Laughman, Christopher R.

TR2023-138 November 28, 2023

Abstract

Physics-based computational models of vapor compression systems (VCSs) enable high-fidelity simulations but require high-dimensional state representations. The underlying VCS dynamics are stiff, constrained by conservation laws, and only a small fraction of states can be measured. While recent advances on constrained extended Kalman filtering (EKF) have provided a systematic framework for estimating VCS states via simulation models, two major bottlenecks to efficient implementation include: (i) expensive forward predictions requiring customized stiff solvers; and, (ii) frequent and computation- ally expensive linearization operations on high-dimensional nonlinear models. In this paper, we circumvent these bottlenecks by constructing deep autoencoder (AE)-based state-space models (SSMs) from simulation data for which both forward predictions and linearization operations via automatic differentiation can be performed efficiently. In addition, we incorporate physical constraints based on pressure gradients explicitly into the autoencoder, and demonstrate, on a Julia-based high-fidelity simulator, that the physics-constrained model improves the estimation performance compared to a AE-based SSM that does not enforce physics.

IEEE Control Systems Letters 2023

Physics-Constrained Deep Autoencoded Kalman Filters for Estimating Vapor Compression System States

Vedang M. Deshpande, *Member, IEEE*, Ankush Chakrabarty[†], *Senior Member, IEEE*, Abraham P. Vinod, *Member, IEEE*, and Christopher R. Laughman, *Member, IEEE*

***Abstract*— Physics-based computational models of vapor compression systems (VCSs) enable high-fidelity simulations but require high-dimensional state representations. The underlying VCS dynamics are stiff, constrained by conservation laws, and only a small fraction of states can be measured. While recent advances on constrained extended Kalman filtering (EKF) have provided a systematic framework for estimating VCS states via simulation models, two major bottlenecks to efficient implementation include: (i) expensive forward predictions requiring customized stiff solvers; and, (ii) frequent and computationally expensive linearization operations on high-dimensional nonlinear models. In this paper, we circumvent these bottlenecks by constructing deep autoencoder (AE)-based state-space models (SSMs) from simulation data for which both forward predictions and linearization operations via automatic differentiation can be performed efficiently. In addition, we incorporate physical constraints based on pressure gradients explicitly into the autoencoder, and demonstrate, on a Julia-based high-fidelity simulator, that the physics-constrained model improves the estimation performance compared to a AE-based SSM that does not enforce physics.**

***Index Terms*— Physics-informed machine learning, constrained systems, Koopman operators, energy systems, Kalman filters.**

I. Introduction

VAPOR compression systems (VCSs) are crucial in various heat transfer applications due to their versatility and wide range of operating temperature. Technologies that can improve operation and overall performance of these systems are critical for mankind to achieve sustainable and habitable living conditions. Many technological solutions for VCSs use simulation models and digital twins [1], [2] for predicting their behavior. Accurate predictions of VCSs dynamics are needed for control [3], estimation [4]–[6], and performance optimization [7], [8]. High-fidelity models of VCSs are generally derived based on the physics of the individual components in VCS and the overall system [9]. Developing physics-based models of these systems require equation-oriented programming environment and face several computational challenges, even when simplified and combined with efficient data-driven learning modules [10].

While physics-oriented models are necessary for accurate system predictions, their computational complexity becomes a bottleneck for efficient deployment, where rapid forward simulation and frequent model linearization is often required. Therefore, *computationally efficient surrogates* of high-fidelity physics-based models of VCSs which mirror the behavior of underlying complex physical system are critical for analysis and design.

Recently, an autoencoder (AE)-based approach was proposed to directly learn a predictive surrogate using the simulation data collected from a high-fidelity model [11]. 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 [12]. 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 performs an inverse transformation 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 [13]–[17]. However, they do not explicitly investigate the problem of estimating the simulator states from measured outputs, which is necessary for simulator-based prediction. Recent studies have identified and proposed methods to attack the problem of estimating hidden/latent states; c.f. [11], [18], [19] with deep neural networks used as surrogate models and *post-hoc* estimation frameworks for estimating states. While they have exhibited good performance on benchmark dynamical systems, this paper extends these ideas to a specific energy systems application leveraging physics-constrained neural networks as surrogate models. These neural surrogate models inform constrained Kalman filters for physically consistent state estimation in a vapor compression system for downstream applications such as heat exchange capacity estimation or refrigerant mass estimation for leak detection.

In particular, we explicitly include physics-informed constraints into the neural SSM architecture proposed in [11], which enforces model predictions to be consistent with the fundamental physical laws governing vapor compression cy-

[†]Corresponding author. Phone: +1 (617) 758-6175. Email: `achakrabarty@ieee.org`.

All authors are with Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA.

cles. A fundamental constraint which VCS states must satisfy is the decreasing pressure gradient in the direction of refrigerant flow. Since high-fidelity VCSs models are based on physics, they inherently satisfy this constraint. On the other hand, neural SSMs which are learned only from the data do not guarantee satisfaction of such physical constraints unless they are explicitly embedded in the SSM architecture. While a post-hoc projection can be employed to enforce this constraint, it often leads to degraded prediction accuracy as we show in the numerical results. To this end, we explicitly incorporate pressure gradient constraint in the SSM architecture using a neural operator. This physically-constrained neural SSM can be used as the core predictive model in an extended Kalman filtering (EKF) framework: we refer to this combination of a deep neural network and the filter as a *deep autoencoded EKF*. In this paper, we seamlessly integrate the neural SSM for enforcing the constraint on filter estimates in a computationally efficient manner. In particular, we constrain the filter estimate by simple forward passes through neural layers rather than using a more expensive projection-based approach with quadratic programming (QP).

Some *contributions* of this work include: (i) incorporating physically motivated pressure constraints explicitly into neural state-space surrogate models of VCS dynamics; (ii) leveraging these physics-constrained neural networks for constrained state estimation via a deep autoencoded EKF; (iii) replacing QP-based constraint enforcement with a more efficient transformation that involves the pre-trained encoder-decoder pair of the trained neural SSM; and (iv) demonstration of the proposed framework on a high-fidelity Julia-based simulator of an industrial VCS system with complex nonlinear dynamics.

II. Preliminaries

We can abstract the VCS in the general form

$$x_{k+1} = f(x_k, u_k) + w_k, \quad y_k = h(x_k) + \eta_k, \tag{1}$$

where $x_k \in \mathbb{X} \subset \mathbb{R}^{n_x}$ denotes the state of the system at time $k \in \mathbb{N}$ with x_0 being the initial state, $u \in \mathbb{R}^{n_u}$ denotes the control inputs, $y \in \mathbb{R}^{n_y}$ denotes the measured outputs, f denotes the dynamics, h denotes the measurement model, and w_k and η_k denote the zero mean Gaussian uncertainties with covariance matrices Q_k^w and Q_k^η, respectively. The functions f and h are often not accessible to the user, as they may be represented in high-fidelity simulation software such as 'digital twins' that contain black-box components. However, we assume that one has access to simulation data. In particular, we are capable of generating trajectories $\{(x_k, y_k)\}_{k=0}^N$ of finite length $N \in \mathbb{N}$ for specified x_0 and $\{u_k\}_{k=0}^N$. Subsequently, we use this simulation data to learn a neural state-space representation of the black-box simulator utilizing the training dataset $\mathsf{D}_{\text{train}} \triangleq \{(x_k, y_k, u_k)\}_{k=0}^N$. Since the simulator has its own specific representation of the states, and we wish to use the simulator for prediction and eventually, control, we need to estimate the state of the simulator online using only measurements $\{y_k\}$. Therefore, our objective is to learn the autoencoder-based predictive model to design a 'deep autoencoded EKF' in order to estimate the simulator states $\{x_k\}$ in a model-based manner.

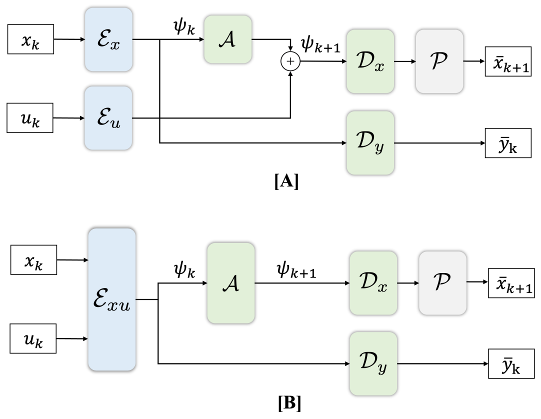

Fig. 1. Neural architecture of deep state-space model. [A] With split $(\boldsymbol{x}, \boldsymbol{u})$ encoders. [B] With mixed $(\boldsymbol{x}, \boldsymbol{u})$ encoder.

We consider an autoencoder-like deep neural SSM; see Fig. 1[A]. The key operations in this architecture are:

$$\psi_k = \mathcal{E}_x(x_k), \tag{2a}$$
$$\psi_{k+1} = \mathcal{A}(\psi_k) + \mathcal{E}_u(u_k), \tag{2b}$$
$$\bar{x}_k = \mathcal{P} \circ \mathcal{D}_x(\psi_k), \tag{2c}$$
$$\bar{y}_k = \mathcal{D}_y(\psi_k), \tag{2d}$$

where $\psi_k \in \mathbb{R}^{n_\psi}$ denotes the latent encoding of the state vector x_k obtained using the encoder $\mathcal{E}_x : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_\psi}$. Similarly, the control vector u_k is mapped to the latent space by the encoder $\mathcal{E}_u : \mathbb{R}^{n_u} \mapsto \mathbb{R}^{n_\psi}$. The latent state update ψ_{k+1} based on ψ_k and u_k is determined by (2b) using the state transition operator $\mathcal{A} : \mathbb{R}^{n_\psi} \to \mathbb{R}^{n_\psi}$. The latent encoding ψ_k is mapped back to the original state-space by the decoder $\mathcal{D}_x : \mathbb{R}^{n_\psi} \to \mathbb{R}^{n_x}$. The operator $\mathcal{P} : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_x}$ is used for enforcing physics-informed constraints described later. A separate decoder $\mathcal{D}_y : \mathbb{R}^{n_\psi} \mapsto \mathbb{R}^{n_y}$ maps the latent to the reconstructed output vector $\bar{y}_k$.

We can now describe how to train the neural SSM (2) without the constraint layer $\mathcal{P}$. Ignoring $\mathcal{P}$, training the neural SSM involves optimizing the weights of $\mathcal{E}_x$, $\mathcal{E}_u$, $\mathcal{A}$, $\mathcal{D}_x$, and $\mathcal{D}_y$ by minimizing a suitable loss function. To this end, we construct the training loss*:

$$\mathsf{L} = \mathsf{L}_{\text{recon}} + \mathsf{L}_{\text{pred},x} + \mathsf{L}_{\text{pred},y} \tag{3}$$

where $\mathsf{L}_{\text{recon}} = \mathsf{MSE}_{0:N}(x_k, \mathcal{D}_x \circ \mathcal{E}_x(x_k))$ denotes the reconstruction loss of the auto-encoder pair $(\mathcal{E}_x, \mathcal{D}_x)$. The next term $\mathsf{L}_{\text{pred},x}$ in (3) denotes one-step prediction error loss in the state vector, i.e., $\mathsf{L}_{\text{pred},x} = \mathsf{MSE}_{0:N-1}\big(x_{k+1}, \mathcal{D}_x\left(\mathcal{A} \circ \mathcal{E}_x(x_k) + \mathcal{E}_u(u_k)\right)\big)$, and $\mathsf{L}_{\text{pred},y}$ denotes the prediction error loss in outputs $\mathsf{L}_{\text{pred},y} = \mathsf{MSE}_{0:N}(y_k, \mathcal{D}_y \circ \mathcal{E}_x(x_k))$. Since these losses are differentiable, we use standard optimization tools such as stochastic gradient descent or its variants to train the neural SSM.

An alternative to splitting the encoder into $\mathcal{E}_x$ and $\mathcal{E}_u$ is to use a single mixed encoder $\mathcal{E}_{xu}$ that encodes

*Note that the mean-squared-error (MSE) between any two vectors of $v_k, \hat{v}_k \in \mathbb{R}^{n_v}$ for $k \in \{0 \dots N\}$ is defined by $\mathsf{MSE}_{0:N}(v_k, \hat{v}_k) \triangleq \frac{1}{n_v(N+1)} \sum_{k=0}^N \|v_k - \hat{v}_k\|^2$.

$(x,u) \mapsto \psi$ by mixing the state and control inputs; see Fig. 1[B]. The latent state update (2b) would become $\psi_{k+1} = \mathcal{A}(\psi_k)$. The decoders would remain the same, but the losses would be: $\mathsf{L}_{\text{recon}} = \mathsf{MSE}_{0:N}(x_k, \mathcal{D}_x \circ \mathcal{E}_{xu}(x_k, u_k))$, $\mathsf{L}_{\text{pred},x} = \mathsf{MSE}_{0:N-1}\big(x_{k+1}, \mathcal{D}_x\left(\mathcal{A} \circ \mathcal{E}_{xu}(x_k, u_k)\right)\big)$, and $\mathsf{L}_{\text{pred},y} = \mathsf{MSE}_{0:N}(y_k, \mathcal{D}_y \circ \mathcal{E}_{xu}(x_k, u_k))$.

III. Physics-Constrained Deep Autoencoded Kalman Filtering

High-fidelity simulators (1) used in VCS applications are typically developed based on physics of the system [6], therefore, they inherently satisfy any fundamental constraints imposed by the physics. An example of such a constraint relevant to VCSs is the pressure gradient constraint, i.e., the non-increasing pressure in the direction of refrigerant flow. In order to ensure consistency with the physical laws, it is imperative that state estimates obtained using a learned neural proxy model and EKF must also satisfy these physics-informed constraints. We explicitly incorporate these constraints into the autoencoder and the filtering algorithm.

A. Equipping physical constraints to the predictive model

Formally, we can write these physical constraints as a linear inequality $Gx \le 0$, and define its feasible set

$$\mathcal{G} := \{x \in \mathbb{X} : Gx \le 0\}, \tag{4}$$

which is a polytope for our pressure gradient matrix G. We know from the Minkowski-Weyl Theorem that the polytope $\mathcal{G}$ admits an equivalent description using rays [20][†]. That is, for some finite set of rays $\mathcal{R} := \begin{bmatrix}\mathcal{R}_1 & \cdots & \mathcal{R}_r\end{bmatrix} \in \mathbb{R}^{n_x \times r}$ we can write $\mathcal{G} = \mathsf{cone}(\mathcal{R})$, where $\mathsf{cone}(\mathcal{R}) := \sum_{j=1}^{r} \mu_j \mathcal{R}_j$ for some $\mu_j \ge 0$. In order to incorporate the physics constraint into the network, we add a constraint layer after the decoder $\mathcal{D}_x$, given by

$$\mathcal{P} := \mathcal{R}\mu, \text{ where } \mu := \mathsf{ReLU} \circ \mathsf{FC}_r \circ \mathcal{D}_x(\psi),$$

with $\mathsf{ReLU}(\cdot)$ denoting the rectified linear unit activation function, and FC_r denoting a fully connected layer with output dimension r. That is, we pass the decoder output $\mathcal{D}_x(\psi)$ through a fully connected layer activated by a ReLU function to make the vector element-wise non-negative, and therefore fulfil the requirement to act as $\mu \ge 0$ in the conic constraint. This fully connected layer also maps $\mathcal{D}_x(\psi)$ to a vector μ that is compatible in dimensions with the ray matrix $\mathcal{R}$. This is because we take the product of $\mathcal{R}$ and μ, which by construction, results in a feasible predicted state $\bar{x}$. Note that all operations added in the constraint layer maintain differentiability, so any gradient-based solver is still compatible for training the proposed network.

The total loss function for the physics-constrained neural SSM is

$$\mathsf{L} = \bar{\mathsf{L}}_{\text{recon}} + \bar{\mathsf{L}}_{\text{pred},x} + \mathsf{L}_{\text{pred},y} \tag{5}$$

[†]To be accurate, Minkowski-Weyl states that the feasible polytope of an inhomogeneous linear inequality $Gx \le g$ can be equivalently described by a conic combination of rays and a convex combination of vertices. When $g = 0$, only the conic combination remains.

with reconstruction and prediction losses

$$\begin{aligned}\bar{\mathsf{L}}_{\text{recon}} &= \mathsf{MSE}_{0:N}(x_k, \mathcal{P} \circ \mathcal{D}_x \circ \mathcal{E}_x(x_k)),\\ \bar{\mathsf{L}}_{\text{pred},x} &= \mathsf{MSE}_{0:N-1}\big(x_{k+1}, \mathcal{P} \circ \mathcal{D}_x\left(\mathcal{A} \circ \mathcal{E}_x(x_k) + \mathcal{E}_u(u_k)\right)\big)\end{aligned}$$

for the split encoder, and

$$\begin{aligned}\bar{\mathsf{L}}_{\text{recon}} &= \mathsf{MSE}_{0:N}(x_k, \mathcal{P} \circ \mathcal{D}_x \circ \mathcal{E}_{xu}(x_k, u_k)),\\ \bar{\mathsf{L}}_{\text{pred},x} &= \mathsf{MSE}_{0:N-1}\big(x_{k+1}, \mathcal{P} \circ \mathcal{D}_x\left(\mathcal{A} \circ \mathcal{E}_{xu}(x_k, u_k)\right)\big)\end{aligned}$$

for the mixed encoder.

Remark 1. *Our neural SSM architecture does not impose any specific structure on $\mathcal{A}$ and $\mathcal{D}_y$. One may choose them to be linear operators by implementing them as linear layers with zero bias. Such a choice can be defended by Koopman operator theory; c.f. [15]. If the dimension of x is large (e.g., in refrigerant flow dynamics), an encoder could be used to induce a low-dimensional latent space, wherein a nonlinear $\mathcal{A}$ may prove more expressive.* ∎

Remark 2. *Note that (5) can be extended to multi-step prediction by recursively generating a sequence of latents $\psi_{k+1:k+N_S}$ for N_S steps starting from ψ_k using the state-transition operator $\mathcal{A}$ and inputs $u_{k:k+N_S-1}$. Decoding these latents would result in multi-step state and output prediction, with which MSE losses could be computed.* ∎

B. Enabling constrained state estimation

After the SSM (2) is learned by minimizing the loss (5), it is used for state estimation. We adopt an EKF framework in this paper, but it is noted that it can be readily extended to other filtering approaches. We rewrite (2) in a form similar to (1) that is amenable to filtering formalism, as follows.

$$x_{k+1} = \mathcal{P} \circ \mathcal{D}_x\big(\mathcal{A} \circ \mathcal{E}_x(x_k) + \mathcal{E}_u(u_k)\big) + w_k, \tag{6a}$$

$$y_k = \mathcal{D}_y \circ \mathcal{E}_x(x_k) + \eta_k, \tag{6b}$$

We use the following notations for deep autoencoded EKF (DA-EKF) estimates. (x_k^-, P_k^-) and (x_k^+, P_k^+) respectively denote the prior (measurements assimilated up to time $k-1$) and posterior (measurements assimilated up to time k) mean-covariance pairs of the state vector at time k, i.e.,

$$x_{k|k-1} \sim \mathcal{N}(x_k^-, P_k^-) \text{ and } x_{k|k} \sim \mathcal{N}(x_k^+, P_k^+), \tag{7}$$

where $\mathcal{N}(\cdot)$ denotes the multivariate normal distribution.

For a given (x_0^-, P_0^-) and neural predictive model (6), the DA-EKF equations are given by the time update:

$$P_{k+1}^- = F_k P_k^+ F_k^\top + Q_k^w \tag{8a}$$

$$x_{k+1}^- = \mathcal{P} \circ \mathcal{D}_x\big(\mathcal{A} \circ \mathcal{E}_x(x_k^+) + \mathcal{E}_u(u_k)\big) \tag{8b}$$

and the measurement update:

$$K_k = (P_k^- H_k^\top)(H_k P_k^- H_k^\top + Q_k^\eta)^{-1} \tag{8c}$$

$$P_k^+ = (I - K_k H_k) P_k^- \tag{8d}$$

$$x_k^+ = x_k^- + K_k\big(y_k - \mathcal{D}_y \circ \mathcal{E}_x(x_k^-)\big) \tag{8e}$$

where F_k and H_k are Jacobian matrices defined as

$$F_k \triangleq \frac{\partial}{\partial x}\mathcal{P} \circ \mathcal{D}_x \circ \mathcal{A} \circ \mathcal{E}_x(x_k^+), \tag{9a}$$

$$H_k \triangleq \frac{\partial}{\partial x}\mathcal{D}_y \circ \mathcal{E}_x(x_k^-). \tag{9b}$$

Note that computing the Jacobian matrices in (9) requires taking gradients of the operators, i.e., differentiating the outputs of the deep neural networks with respect to their inputs. To this end, we utilize auto-differentiation (AD) capabilities available in standard deep learning libraries such as `PyTorch`.

Despite designing the neural SSM (6) to satisfy the state constraints (4), the filter updates using (8) are not designed to explicitly satisfy the constraints, which could yield physics-inconsistent state estimates. Therefore, we deem it necessary to explicitly incorporate state constraints during the filtering process as well [6], [21].

There are several approaches [22] such as density truncation [21], estimate projection [6] and pseudo or perfect measurements [5], available in the literature that can be adapted for enforcing constraint in state estimation algorithms. However, unlike previous works, here we leverage the inherent constraint-satisfying nature of the SSMs in order to enforce the same constraints on state vectors estimated by the EKF. In particular, we propose the use of auto-encoder pair $(\mathcal{E}_x\,, \mathcal{P} \circ \mathcal{D}_x)$ for projecting the filter estimate obtained in (8e) on the feasible set to enforce constraints. The constraint layer $\mathcal{P}$ ensures that the decoded state vector satisfies the gradient constraint. Therefore, a constrained state estimate can be obtained by the following reconstruction using the auto-encoder

$$x_k^+ \leftarrow \mathcal{P} \circ \mathcal{D}_x \circ \mathcal{E}_x(x_k^+).$$

The constraint-projection operation in (III-B) is expected to computationally efficient as it involves only forward evaluation of neural networks. This is in contrast to other prevalent constraint enforcement approaches which typically requires solving a constrained quadratic program (QP) [6], [21].

IV. Simulation Results

A. Data Generation from VCS Simulator

A VCS consists of a variable-speed compressor, a variable-position electronic expansion valve (EEV), two heat exchangers (condenser and evaporator), and two fans that force the air to flow through the heat exchangers. This system transfers heat from the air passing through the evaporator to the air passing condenser via the refrigerant flowing through the system. We use a physics-based simulator of the VCS [6] for generating the training dataset.

The model implements two heat exchangers which dominate the overall system dynamics by discretizing them in four finite volumes each, while algebraic models are used for the compressor and expansion valve. Furthermore, a heat exchanger is comprised of interconnecting components that describe the thermal behavior of the pipe wall, the flow of air across the heat exchanger, and the one-dimensional refrigerant pipe-flow, forming an index-1 system of differential algebraic equations (DAEs). The one-dimensional flow of refrigerant through the pipe is modeled using fundamental physical laws, namely, the conservation of mass, momentum, and energy. After order reduction via Pantelides' algorithm, the DAEs are transformed to a set of ordinary differential equations (ODEs). The state vector contains the pressure $\mathsf{P}[i]$, specific enthalpy $\mathsf{h}[i]$, and temperature $\theta[i]$ for each finite volume $i \in \{1 \cdots 8\}$ and the control inputs to the model are compressor frequency and expansion valve position, thus, $n_x = 24$ and $n_u = 2$. The measurements include temperatures and pressures at select volume locations in the heat exchangers. In particular, the measurement vector is given by

$$y = \begin{bmatrix}\mathsf{P}[1] & \theta[2] & \theta[4] & \theta[6] & \mathsf{P}[8] & \theta[8]\end{bmatrix} \in \mathbb{R}^6, \tag{10}$$

that is, only 6 of the 24 states are directly measured.

This model implements an initialization routine in which all variables are set to their default values. A set of 21 initial conditions to be used for generating the training dataset was constructed by driving the model (with default initialization) to a steady state over a simulation of 50 s with constant control inputs. While the EEV position was set to 300 counts for all runs, a set of uniformly spaced 21 values of compressor frequency was selected within the interval [40, 80] Hz. This eliminates effects of default initialization to some extent and allows construction of a meaningful, controlled set of initial conditions to test our proposed methods.

The model was simulated to generate a set of 21 trajectories over 500 seconds with the afore-mentioned set of initial conditions. The control inputs, compressor frequency (Hz), and expansion valve position were randomly sampled from the uniform distributions $\mathcal{U}(40, 80)$ and $\mathcal{U}(282, 312)$, respectively, to generate a rich dataset with persistently excited inputs. These trajectories were sampled at 0.1 second intervals with sample-and-hold control inputs. We emphasize that the trajectories generated from the specified initial conditions are not steady-state trajectories and they do exhibit transient behavior due to time varying control inputs. Since the model enforces fundamental physical laws, the data generated using this model inherently satisfies the pressure gradient constraint (4).

B. Performance of Autoencoded SSM

The set of 21 trajectories is split into train, test, and validation sets of 11, 5, 5 trajectories respectively, such that the underlying compressor frequencies associated with the initial conditions of the trajectories in each set are uniformly spaced in the interval interval [40, 80] Hz.

After a preliminary neural architecture search, the dimension of the latent space was selected to be $n_\psi = 32$. Each of the operators $\mathcal{E}_x$, $\mathcal{E}_u$, $\mathcal{D}_x$, and $\mathcal{E}_{xu}$ was implemented as a fully connected neural network with 4 layers and the hidden dimension of 64. Activation functions used in the input and hidden layers were ReLU and the output layer was configured as a linear transformation. Furthermore, the operators $\mathcal{A}$ and $\mathcal{D}_y$ were implemented as linear layers with no biases.

We consider different neural SSM architectures tabulated in Table I. The 'Constrained' column indicates whether an SSM is equipped with a constraint layer $\mathcal{P}$ as discussed in §III-A, whereas the 'Encoder' column indicates whether the SSM employs a split or mixed encoder.

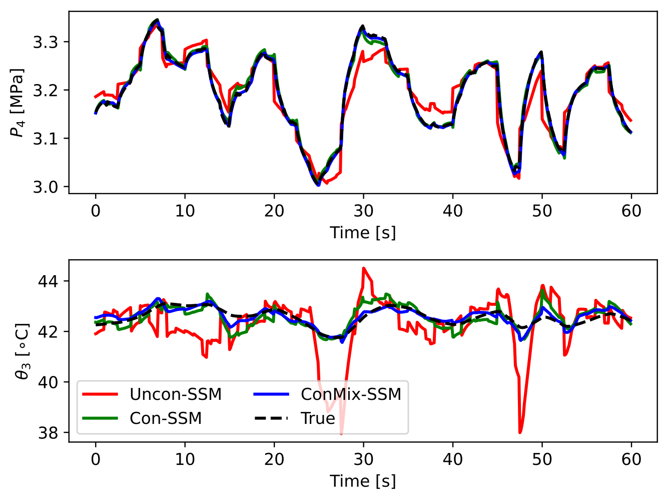

Fig. 2. True state trajectories and those predicted by the learned neural SSMs.

TABLE I
DIFFERENT SSM ARCHITECTURES AND THEIR PREDICTION SSE.

SSM Architecture	Constrained	Encoder	SSE
`Uncon-SSM`	-	Split	323.75
`Con-SSM`	Yes	Split	68.42
`ConMix-SSM`	Yes	Mixed	**58.37**

All SSMs were trained for 2000 iterations on identical training datasets using an Adam optimizer with a learning rate of 10^{-4} with weight decay of 0.999 and learning rate scheduling. Weights of the neural SSM were saved when the validation loss decreased. A holdout test dataset was used to compare the prediction accuracy of these two SSMs. The SSM predictions for select states and the corresponding true trajectories are shown in Fig. 2. The prediction accuracy is quantified in terms of sum squared error (SSE). The SSE between a predicted or estimated x_k and the true trajectory x_k^* is defined as follows: $\mathsf{SSE}(x_k, x_k^*) \triangleq \sum_{k=1}^{N} \|x_k - x_k^*\|^2$.

The cumulative SSEs computed over five test trajectories for different SSMs are shown in Table I. The `Uncon-SSM` yielded the largest SSE making it the least accurate SSM. Comparison of `Con-SSM` against `Uncon-SSM` implies that explicitly incorporating the physics-informed pressure gradient constraints into the SSM architecture significantly improves the prediction accuracy. The accuracy is further improved by employing a mixed (x, u) encoder in the architecture as `ConMix-SSM` is found to demonstrate the smallest SSE. All neural SSMs took approximately 1 s of CPU time to simulate the VCS behavior over 5000 time steps whereas Julia-based high-fidelity VCS simulator took more than 110 s CPU time for predictions over the same time horizon.

C. Performance of DA-EKF

We consider different implementations of DA-EKF based on the type of SSM used for predictions from Table I; and the type of projection method (QP or AE) used for constraining filter estimates. We use the following convention for referring to different implementations of DA-EKF: the implementation which uses `ConMix-SSM` for predictions and AE-based projection is referred to by `ConMix-AE`; the implementation which uses `Uncon-SSM` for predictions and QP-based projection is referred to by `Uncon-QP`, etc.

We performed 100 Monte-Carlo runs of the DA-EKFs to gather statistical metrics of performance. For every filter run, a reference or true trajectory was randomly sampled with replacement from the test dataset. The initial filter estimate x_0^- was obtained by perturbing the true initial condition x_0^* using a randomly drawn sample from $\mathcal{N}(0, P_0^-)$. A set of measurements to be used in filter updates was constructed by perturbing the true outputs using randomly drawn samples from $\mathcal{N}(0, Q_k^\eta)$. Different DA-EKFs were subject to identical reference trajectories, identical perturbed measurements and identical initialization during each run. Further, we set the scaled covariances to be $P_0^- = 10^{-3} I_{n_x}$, $Q_k^w = 10^{-4} \cdot \mathbf{1}_8 \otimes [2, 1, 0.1]$, and $Q_k^\eta = 10^{-4} \cdot [2, 0.1, 0.1, 0.1, 2, 0.1]$ for all runs of all filters, where I_n denotes the identity matrix of dimension n, $\mathbf{1}_n$ denotes an n-vector of all ones, and $\otimes$ is the Kronecker product.

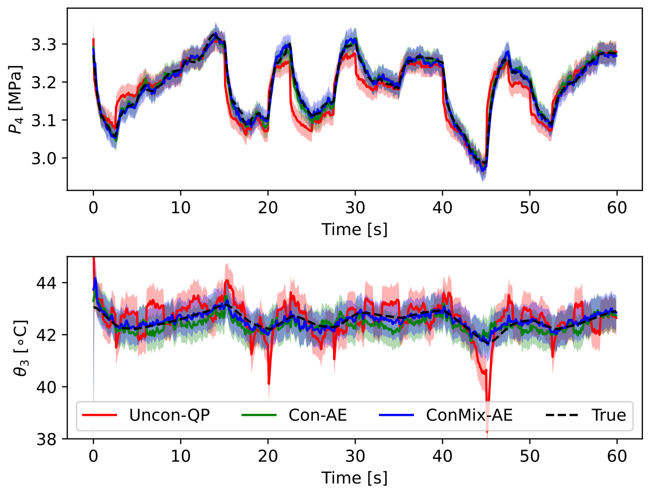

Fig. 3. True state trajectories and those estimated by DA-EKFs. Shaded area around the estimated trajectories shows the 95% confidence interval.

Fig. 3 shows the true and estimated trajectories for selected states (temperatures and pressures) along with the 95% confidence interval for an arbitrary filter run. While the `Con-AE` and `ConMix-AE` demonstrated better estimation accuracy, the confidence intervals for different DA-EKFs were found to be comparable. Fig. 4 shows the performance of different DA-EKFs using box plots for estimation SSE calculated for each individual filter run, as well as the cumulative time required during each run for projecting filter estimates onto the feasible set. The DA-EKFs, which use constrained SSMs as prediction models demonstrate improved estimation accuracy over `Uncon-QP` which uses an unconstrained neural SSM as the prediction model. The DA-EKFs, `ConMix-AE` and `ConMix-QP`, which employ both constrained SSMs and mixed (x, u) encoders, exhibit the smallest estimation SSEs that are approximately 50% of those observed in the naive implementation of DA-EKF `Uncon-QP`. This indicates the

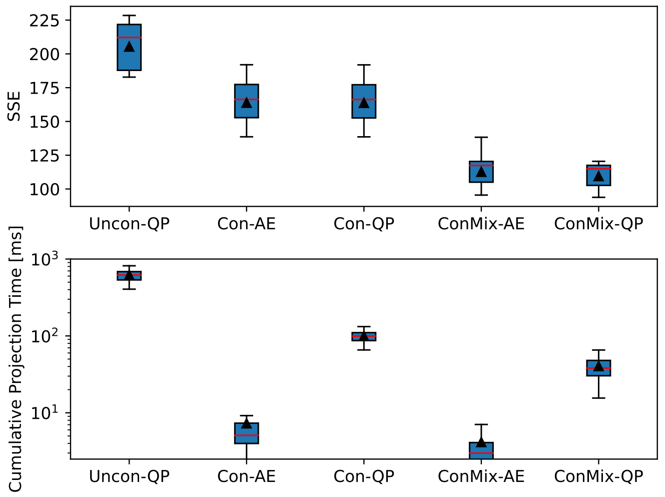

Fig. 4. Box plots of SSE and (logarithm scaled) cumulative projection time for 100 filter runs. While a horizontal line in the box shows the median value, the mean value is shown by a triangular marker.

beneficial effect of directly incorporating physics-based constraints into the neural predictive model and mixed (x, u) encoders, and this benefit is likely due to an improved prediction accuracy on the test data. The execution times required to project states onto the constraint set for DA-EKFs which enforce constraints by solving a quadratic program, are orders of magnitude greater than those using AE-based projections. It is evident that `Con-AE` and `ConMix-AE`, both of which enforce constraints by a simple forward evaluation of the autoencoder with a median projection times of less than 5 ms, are one of the most efficient DA-EKFs in terms of enforcing constraints on the filter estimates.

Comparing projection times for `Uncon-QP`, `Con-QP`, and `ConMix-QP` also provides an interesting insight into the effectiveness of constrained-SSMs; see Fig. 4. Although all of these three DA-EKFs use QP-based projection, the cumulative projection times over an entire run for `Con-QP` and `ConMix-QP` are significantly smaller than that of `Uncon-QP`. Since these DA-EKFs solve QPs of identical size for enforcing constraints, the larger cumulative projection time incurred by `Uncon-QP` is an indication that it solves the QP more number of times than `Con-QP` and `ConMix-QP`. In other words, using a constrained-SSM for predictions in DA-EKFs leads to less frequent constraint-violating filter estimates than the unconstrained-SSM. The improved constraint-enforcement time without compromising the estimation accuracy is a particular benefit of our approach that leverages the neural SSM architecture and obviates the need to explicitly add QP solvers for our particular physical constraint.

V. Conclusions

We demonstrated the effectiveness of neural surrogate state-space models for informing state estimators for complex physical systems such as vapor compression systems. These neural surrogate models, which explicitly enforce physics-based constraints in their architecture, are directly learned using the data obtained from a physics-based high-fidelity simulator. The simulation models allow rapid forward simulations that can be useful to evaluate various scenarios in a digital twin. We show via high-fidelity simulations that the proposed physics-constrained SSMs can improve the quality of downstream state estimation.

References

[1] C. Vering, S. Borges, D. Coakley, H. Kruetzfeldt, P. Mehrfeld, and D. Müller, “Digital twin design with on-line calibration for HVAC systems in buildings,” in *Proceedings of Building Simulation 2021: 17th Conference of IBPSA*, ser. Building Simulation, vol. 17. Bruges, Belgium: IBPSA, September 2021, pp. 2938–2945.

[2] C. R. Laughman, V. M. Deshpande, H. Qiao, S. A. Bortoff, and A. Chakrabarty, “Digital Twins of Vapor Compression Cycles: Challenges and Opportunities,” in *Proc. Int. Congress of Refrigeration (ICR)*, 2023.

[3] S. Bortoff, P. Schwerdtner, C. Danielson, and S. Di Cairano, “H_∞ loop-shaped model predictive control with heat pump application,” in *18th European Control Conference*, 2019, pp. 2386–2393.

[4] A. Chakrabarty, E. Maddalena, H. Qiao, and C. Laughman, “Scalable Bayesian optimization for model calibration: Case study on coupled building and HVAC dynamics,” *Energy and Buildings*, vol. 253, p. 111460, 2021.

[5] V. M. Deshpande and C. R. Laughman, “Multi-pass extended kalman smoother with partially-known constraints for estimation of vapor compression cycles,” in *22nd IFAC World Congress*, 2023, pp. 7333–7340.

[6] V. M. Deshpande, C. R. Laughman, Y. Ma, and C. Rackauckas, “Constrained smoothers for state estimation of vapor compression cycles,” in *Proc. American Control Conference*, 2022, pp. 2333–2340.

[7] M. Khosravi, A. Eichler, N. Schmid, R. S. Smith, and P. Heer, “Controller tuning by Bayesian optimization an application to a heat pump,” in *Proc. 18th European Control Conference*. IEEE, 2019, pp. 1467–1472.

[8] A. Chakrabarty, C. Danielson, S. A. Bortoff, and C. R. Laughman, “Accelerating self-optimization control of refrigerant cycles with bayesian optimization and adaptive moment estimation,” *Applied Thermal Engineering*, vol. 197, p. 117335, 2021.

[9] H. Qiao, V. Aute, and R. Radermacher, “Transient modeling of a flash tank vapor injection heat pump system–Part I: Model development,” *International Journal of Refrigeration*, vol. 49, pp. 169–182, 2015.

[10] R. Chinchilla, V. M. Deshpande, A. Chakrabarty, and C. R. Laughman, “Learning residual dynamics via physics-augmented neural networks: Application to vapor compression cycles,” in *2023 American Control Conference (ACC)*, 2023, pp. 4069–4076.

[11] A. Chakrabarty, A. P. Vinod, H. Mansour, S. A. Bortoff, and C. R. Laughman, “Moving horizon estimation for digital twins using deep autoencoders,” in *Proc. IFAC World Congress*, 2023, pp. 6005–6010.

[12] B. Lusch, J. N. Kutz, and S. L. Brunton, “Deep learning for universal linear embeddings of nonlinear dynamics,” *Nature communications*, vol. 9, no. 1, pp. 1–10, 2018.

[13] D. Masti and A. Bemporad, “Learning nonlinear state–space models using autoencoders,” *Automatica*, vol. 129, p. 109666, 2021.

[14] G. Beintema, R. Toth, and M. Schoukens, “Nonlinear state-space identification using deep encoder networks,” in *Learning for Dynamics and Control*. PMLR, 2021, pp. 241–250.

[15] A. Mauroy, Y. Susuki, and I. Mezić, *Koopman Operator In Systems And Control*. Springer, 2020.

[16] A. Surana, “Koopman operator based observer synthesis for control-affine nonlinear systems,” in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 6492–6499.

[17] A. Surana and A. Banaszuk, “Linear observer synthesis for nonlinear systems using Koopman operator framework,” *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 716–723, 2016.

[18] M. Forgione, M. Mejari, and D. Piga, “Learning neural state-space models: Do we need a state estimator?” *arXiv:2206.12928*, 2022.

[19] K. Ensinger, S. Ziesche, and S. Trimpe, “Learning hybrid dynamics models with simulator-informed latent states,” *arXiv preprint arXiv:2309.02873*, 2023.

[20] K. Fukuda and A. Prodon, “Double description method revisited,” in *Franco-Japanese and Franco-Chinese conference on combinatorics and computer science*. Springer, 1995, pp. 91–111.

[21] D. Simon and D. L. Simon, “Constrained Kalman filtering via density function truncation for turbofan engine health estimation,” *International Journal of Systems Science*, vol. 41, no. 2, pp. 159–171, 2010.

[22] N. Amor, G. Rasool, and N. C. Bouaynaya, “Constrained state estimation – a review,” 2022, arXiv:1807.03463.