2024 European Control Conference (ECC)
June 25-28, 2024. Stockholm, Sweden

Recursive Least Squares-Based Identification for Multi-Step Koopman Operators

Omar Sayed and Sergio Lucia

Abstract— **This paper proposes a generalized algorithmic approach for learning linear model representations for nonlinear systems within the Koopman framework. We focus on schemes that rely on learning the nonlinear transformation functions using deep neural networks. Beyond achieving dynamical accuracy, our primary objective is to develop models capable of simulating nonlinear systems across multiple time steps in the linear space. An algorithm that is based on recursive least squares is proposed to address the optimization complexities inherent in learning such models. In addition, we leverage the learned linear representation to design a linear quadratic regulator to control the original nonlinear system. The effectiveness of the proposed algorithm is demonstrated in two numerical examples.**

I. Introduction

With the increasing availability of data and the growing complexity of modern control systems, more interest has shifted toward learning mathematical models using system identification [1], [2]. Since the majority of systems exhibit nonlinear behaviour, nonlinear models are often employed to capture dynamics from input-output datasets. However, a drawback of these models is the computational challenges they impose when used as prediction models in optimal control [3], motivating research on linearization techniques [4].

Koopman theory, as articulated by Koopman in his seminal works [5] and [6], states that nonlinear systems can be represented by an infinite-dimensional linear systems. In this transformed coordinate system, often referred to as the embedding space, the dynamics are represented by the Koopman operator. In the field of modelling and control, this is appealing, since it facilitates the use of linear theory analysis and optimal control techniques [7], [8], [9] on complex nonlinear systems. To make this theory practically applicable, it is necessary to seek finite approximations of the Koopman operator and the embedding space.

A prominent approach to learning the Koopman operator in the context of autonomous systems is Dynamic Mode Decomposition (DMD) [10]. This approach learns a finite Koopman operator by using linear transformation matrices. To increase the versatility of DMD, modifications have been implemented, enabling nonlinear transformations through the use of predefined basis functions. This extension is commonly referred to as Extended Dynamic Mode Decomposition (EDMD) [11]. Furthermore, various algorithms have been proposed to extend the aforementioned schemes to incorporate exogenous inputs, making the transformed linear models suitable for use as prediction models in optimal control [12], [13], [14].

However, finding the appropriate basis functions requires knowledge about the system which can sometimes be a limiting factor [15]. To alleviate this issue, Deep Neural Networks (DNNs) can be used to learn the nonlinear transformation [16], [17], [15]. Consequently, this idea has also been extended to control problems in [18], [19], [20].

Koopman-based algorithms trained with single-step loss, show poor performance when used for multi-step predictions [21], where the model's output is used in a closed-loop fashion to predict states at further time steps. To improve the multi-step prediction, a common approach is to use a multi-step loss during training [22], [16]. However, optimizing for multi-step loss is challenging, even if the dynamic model is linear, as this turns the optimization problem of finding the optimal model parameters into a non-convex problem with various local minima [23].

The use of commonly employed first-order optimization methods can fail depending on the initial parametrization [24]. To address this issue, this work reformulates the non-convex problem of multi-step training into a convex recursive least-squares (RLS) identification [25] for learning the Koopman operator. RLS has been used in the Koopman framework in [26] to update the model parameters online. In this work, we use the RLS to solve the problem of multi-step predictions using a convex optimization during offline training.

Another challenge arises when using the linearized model in conjunction with an LQR controller. This stems from the difficulty of designing the cost function in the lifted Koopman space, as it is a priori not known how achieving a certain control goal on the lifted Koopman space can translate to achieving the desired control goal in the original space. In this work, we mitigate this issue by using a linear transformation of the cost function as in [18], [27].

The main contribution of this work is as follows. First, we propose a training scheme that utilizes RLS estimation to improve the training of the Koopman operator for multi-step prediction during training. Second, we highlight the improved performance of our proposed scheme over the baseline methods through two numerical example. We show how our approach effectively overcomes local optima that can hinder the training of models using first-order optimization techniques. Finally, we design an LQR controller using the models trained using the proposed and baseline methods,

*This work was not supported by any organization

Omar Sayed and Sergio Lucia are with the Chair of Process Automation Systems, TU Dortmund University, Dortmund, Germany {omar.sayed, sergio.lucia}@tu-dortmund.de

and we evaluate their effectiveness in controlling the original nonlinear system.

This paper is structured as follows: Section II provides a brief introduction to the Koopman theory. Section III presents the proposed algorithms. Section IV shows how the learned linear models can be used to design LQR controllers. In Section V we evaluate and compare the performance of our proposed algorithms with baseline methods. Finally, we conclude the work in Section VI.

II. Background

A. Koopman theory for autonomous systems

We consider the following autonomous nonlinear system with discrete dynamics

$$x_{k+1} = F(x_k), \tag{1}$$

where $x_k, x_{k+1} \in \mathbb{R}^{n_x}$ is the state vector of the system with n_x states. $F : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ is a nonlinear function that simulates the states forward by one time step of length t_s. To analyse the system in (1) with the tools of linear control theory, the states of the original system are transformed into a larger infinite dimension where the following relation holds:

$$\mathcal{K}\psi(x_k) = \psi \circ F(x_k) = \psi(x_{k+1}). \tag{2}$$

In the context of the Koopman theory, $\psi : \mathbb{R}^{n_x} \to \mathbb{R}^{n_z}$ is the lifting function, i.e. a nonlinear transformation function that maps the nonlinear original states to the infinite linear Koopman space ($n_z = \infty$). $\mathcal{K} \in \mathbb{R}^{n_z \times n_z}$ is the Koopman operator, linear and infinite in dimension, which simulates the embedding states by one sampling time.

To approximate $\mathcal{K}$ and ψ, finite-dimensional approximations are derived using time-series data sampled from the original nonlinear system. The goal is to find a finite embedding space that is typically larger than the true state space of the original system. For a given data-set $D := [x_0, x_1, \ldots, x_n]$ this approximation can be obtained by solving an optimization problem given by

$$\underset{\psi, \mathcal{K}, n_z}{\operatorname{argmin}} \sum_{k=0}^{n-1} \|\psi(x_{k+1}) - \mathcal{K}\psi(x_k)\|. \tag{3}$$

It is important to note that this optimization problem has a trivial solution ($\psi(x \in \mathbb{R}^{n_x}) = 0$), which can be avoided by employing appropriate measures as we will also describe in the next sections.

B. Koopman operator with exogenous inputs

In this work, we consider a controlled system represented as $x_{k+1} = F(x_k, u_k)$. To extend the Koopman theory to this type of system, similar to previous studies [18], [13] and [12], we redefine (2) as

$$\mathcal{K}(\psi(x_k), u_k) = \psi(F(x_k, u_k)) = \psi(x_{k+1}), \tag{4}$$

where $F : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ is a nonlinear function representing the dynamic equation with both the state x_k and the control input $u_k \in \mathbb{R}^{n_u}$. The Koopman operator, $\mathcal{K}$, is decomposed into two parts, $\mathcal{K} = [\mathcal{K}_x \in \mathbb{R}^{n_z \times n_z}, \mathcal{K}_u \in \mathbb{R}^{n_z \times n_u}]$, which are linear functions learned for the state and control input, respectively.

The lifting function and the Koopman operator can be obtained by solving the following optimization problem:

$$\underset{\psi, \mathcal{K}_x, \mathcal{K}_u, n_z}{\operatorname{argmin}} \sum_{k=0}^{n-1} \|\psi(x_{k+1}) - (\mathcal{K}_x\psi(x_k) + \mathcal{K}_u u_k)\|. \tag{5}$$

Following from the definition of the Koopman operator for controlled systems, a discrete nonlinear system can be approximated by the following linear system

$$\psi(x_{k+1}) = \mathcal{K}_x\psi(x_k) + \mathcal{K}_u u_k, \tag{6}$$

$$z_{k+1} = \mathcal{K}_x z_k + \mathcal{K}_u u_k, \tag{7}$$

where $\mathcal{K}_x$ and $\mathcal{K}_u$ are equivalent to the state matrix A and the input matrix B. z_k, z_{k+1} are the embedding states (Koopman states), and defined as $z_i := \psi(x_i)$. For more details on the Koopman operator, readers can refer to [28].

III. Deep Koopman with Recursive Least Squares

Deep Koopman schemes typically consist of two key components: the first is the learning of the transformation function, which lifts the original states to a higher-dimensional coordinate system, while the second is the learning of the Koopman operator, which represents the dynamics of the lifted linear system. This work considers learning both the transformation function and the Koopman operator simultaneously.

In Subsection III-A we demonstrate how recursive least squares (RLS) can be used with an offline dataset to improve multi-step prediction in linear system identification replacing the nominal multi-step loss. Then we show in Subsection III-B how this approach can be integrated into the Koopman framework.

A. RLS for Linear System Identification

Consider a linear discrete system in the following form:

$$x_{k+1} = Ax_k + Bu_k + \epsilon \tag{8}$$

where $\epsilon \sim \mathcal{N}(0, \sigma)$ is white Gaussian noise resembling model uncertainties due to the finite approximation of the Koopman operator. Given a state trajectory $X_{0:n} := [x_0, x_1, \ldots, x_n]$ and input trajectory $U_{0:n-1} := [u_0, u_1, \ldots, u_{n-1}]$, where n is the length of the trajectory, the following single-step optimization problem for system identification can be formulated as

$$\min_{\theta} \sum_{k=0}^{n-1} \|x_{k+1} - A(\theta)x_k - B(\theta)u_k\|^2, \tag{9}$$

where $A(\theta)$ and $B(\theta)$ are parametrized state and input matrices. The optimal parameters θ^* have the closed-form solution

$$\theta^* = X_{1:n}[X_{0:n-1}|U_{0:n-1}]^\dagger, \tag{10}$$

where $\dagger$ is the Moore-Penrose pseudoinverse. However, in the context of optimal control, it is necessary to compute a

multi-step trajectory of the states given an input trajectory U and initial condition x_0. We define the multi-step predicted trajectory $\hat{X}$ as

$$\hat{X} := \{\hat{x}_{[0,1]}, \ldots, \hat{x}_{[p,m]}, \ldots, \hat{x}_{[n-k,n]}\}, \quad (11)$$
$$p = \begin{cases} m-k & \text{if } m \geq k \\ 0 & \text{otherwise} \end{cases}, \; m \in [0, n],$$

where

$$\hat{x}_{[p,m]} = A^{m-p}x_p + \sum_{i=p}^{m-1} A^{m-i-1}Bu_i. \quad (12)$$

In words, $\hat{x}_{[p,m]}$ denotes the state at time step m predicted using (12) with an initial state at time p and the inputs from p to $m-1$. In this case, the number of multi-steps in the prediction for $\hat{x}_{p,m}$ is $(k = m - p)$. Using $A(\theta)$ and $B(\theta)$ yielded from the one-step training (9) and the forward propagation (12) can lead to a poor performance [22], [16], [23], due to the accumulation of the approximation errors $(\sum_{i=p}^{m-1} A^{m-i-1}\epsilon_i)$ along the predictions. Alternatively, one can use a multi-step loss, which finds the model parameters by minimizing the error not only for the next step but over a trajectory of k steps. The corresponding optimization problem can be defined as:

$$\min_{A(\theta),B(\theta)} \|X_{1:n} - \hat{X}\|^2. \quad (13)$$

Problem (13) is non-convex and its complexity increases as the number of states n_x and multi-steps k increases [23]. In this work, we seek to mitigate this issue by reformulating (13) into an RLS formulation defined by

$$[\tilde{A}(\theta)|\tilde{B}(\theta)] = X_{2:n}[\hat{X}_{0:n-2}|U_{1:n-1}]^\dagger, \quad (14)$$
$$[A(\theta)|B(\theta)] = (1-\gamma)[A(\theta)|B(\theta)] + \gamma[\tilde{A}(\theta)|\tilde{B}(\theta)], \quad (15)$$

where γ is the forgetting factor, $X_{2:n} = [x_2, \ldots, x_n]$ and $\hat{X}_{0:n-2} = [\hat{x}_0, \ldots, \hat{x}_{n-2}]$. The RLS loop starts with the initialization of $A(\theta)$ and $B(\theta)$ by the single-step loss (10). Then, the predicted trajectory $(\hat{X})$ is iteratively computed as outlined in (11). $\tilde{A}(\theta)$ and $\tilde{B}(\theta)$ are computed using least squares (14) to find the system dynamics that can shift the predicted trajectory one step further. In an optimal scenario, $\tilde{A}(\theta)$ and $\tilde{B}(\theta)$ would be equal to $A(\theta)$ and $B(\theta)$.

Using (15), the computed $\tilde{A}(\theta)$ and $\tilde{B}(\theta)$ are used to update the current $A(\theta)$ and $B(\theta)$ with a step size of γ. This iterative process continues until specific termination criteria for (13) are satisfied. For numerical stability the initial $A(\theta)$ and $B(\theta)$ matrices must be be Schur stable.

B. Deep Learning Framework

This work leverages DNNs to learn the forward ψ and inverse ψ^{-1} transformation functions, in order to recover the transformed states back to their original form. These requirements make the use of deep autoencoders [29] a natural choice. A schematic of the overall architecture shown in Fig. 1.

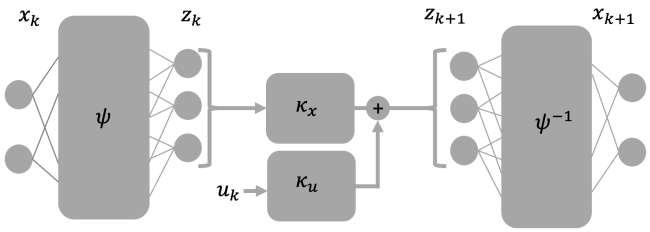

Fig. 1. Deep Koopman Scheme: ψ and ψ^{-1} are parametrized neural networks denoting the forward and inverse transformation functions. $\mathcal{K}_x$ and $\mathcal{K}_u$ are linear matrices similar to the system (A) and input (B) matrices respectively.

The encoder ψ has the task of lifting the states to a higher dimension while linearizing the dynamics. This can be achieved by the following single-step loss function $\mathcal{L}_{\text{ss}}$

$$\mathcal{L}_{\text{ss}} = \sum_{k=0}^{n-1} \|\psi(x_{k+1};\theta) - \mathcal{K}_x\psi(x_k;\theta) + \mathcal{K}_u u_k)\|, \quad (16)$$

where $\mathcal{K}_x$ and $\mathcal{K}_u$ are learned by the RLS formulation in III-A, and θ are the trainable parameters of the encoder network. As mentioned earlier, the use of (16) can lead to the trivial solution of adjusting the weights such that $\psi(x) = 0$. This is avoided by the reconstruction loss $\mathcal{L}_{\text{rec}}$ defined as

$$\mathcal{L}_{\text{rec}} = \sum_{k=0}^{n-1} \|x_k - \psi^{-1}(\psi(x_k;\theta);\theta)\|, \quad (17)$$

where ψ^{-1} is the decoder with trainable parameters θ. We propose to add a third loss function for the multi-step loss $\mathcal{L}_{\text{ms}}$ as

$$\mathcal{L}_{\text{ms}} = \sum_{p=0}^{n-k} \sum_{m=p+1}^{p+k} \|\mathbf{X}_{p,m} - \hat{\mathbf{X}}_{p,m}\|, \quad (18)$$

$$\text{where, } \mathbf{X}_{p,m} = [x_p, x_{p+1}, \ldots, x_m], \quad (19)$$

$$\hat{\mathbf{X}}_{p,m} = [\hat{x}_{[p-1,p+1]}, \hat{x}_{[p-1,p+2]}, \ldots, \hat{x}_{[p-1,m]}], \quad (20)$$

$$\hat{x}_{[p,m]} = \psi^{-1}(\mathcal{K}_x{}^{m-p}\psi(x_p) + \sum_{i=p}^{m-1} \mathcal{K}_x^{m-i-1}\mathcal{K}_u u_i;\theta). \quad (21)$$

The total loss function is the weighted sum of the $\mathcal{L}_{\text{ss}}$, $\mathcal{L}_{\text{ms}}$ and $\mathcal{L}_{\text{rec}}$. While it can be argued that $\mathcal{L}_{\text{rec}}$ is implicitly fulfilled by $\mathcal{L}_{\text{ms}}$, the inclusion of $\mathcal{L}_{\text{rec}}$ serves to improve the overall training process. Algorithm 1 summarizes the training approach, that integrates DNN training and RLS.

IV. LQR IN THE KOOPMAN EMBEDDING SPACE

The objective of this section is to design a controller for the nonlinear system using the transformed linear dynamics. For this purpose an LQR set-point tracking controller can be used, which is defined as follows:

$$\min_{u_k,x_k} \sum_{k=0}^{\infty} \|z_k - z_{\text{target}}\|_Q^2 + \|u_k - u_{\text{ss}}\|_R^2, \quad (22)$$
$$\text{s.t. } z_{k+1} = \mathcal{K}_x z_k + \mathcal{K}_u u_k,$$
$$z_{\text{target}} := \psi(x_{\text{target}}), \quad z_0 = \psi(x_0),$$

Algorithm 1 Deep Koopman with RLS

Input: Set forgetting factor γ
Initialize $[A(\theta)|B(\theta)] \leftarrow \psi(X_{1:n})[\psi(X_{0:n-1}|U_{0:n-1})]^{\dagger}$
while termination criteria not met **do**
  Compute $\hat{X}$ for $\psi(x)$ using (11) ▷ Prediction step
  $[\tilde{A}(\theta)|\tilde{B}(\theta)] \leftarrow \psi(X_{2:n})[\hat{X}_{0:n-2}|U_{1:n-1}]^{\dagger}$
  Update $[A(\theta)|B(\theta)]$ using (15) ▷ Update step
  Update θ for ψ and ψ^{-1} minimizing (16), (17), (18)
return θ, $A(\theta)$ and $B(\theta)$

where Q and R are the cost matrices for the states and inputs respectively. u_{ss} is the steady state input. The set-point in the original nonlinear state space is x_{target}. Solving (22) using the algebraic Riccati equation yields the optimal gain matrix K^* and the optimal control input becomes $u_t^* = K^* z_t$. To apply the LQR to an arbitrary steady state, the following control input can be used:

$$u^* = u_{\text{ss}} - K^*(z_k - z_{\text{target}}), \tag{23}$$
$$u_{\text{ss}} = -(B^T B)^{-1} B^T A z_{\text{target}},$$

Designing the state cost matrix Q for embedding states presents a challenge due to the increased dimensionality of the state space, making it less manageable to establish the relationship between embedded states and their original counterparts. One solution is to linearly approximate the nonlinear relationship between the two spaces [21], [27]. The new state cost matrix Q^* can be calculated as

$$Q^* = T^T \hat{Q} T, \tag{24}$$
$$T := (\psi(X)^{\dagger} X)^T, \tag{25}$$

where $T \in \mathbb{R}^{n_x \times n_z}$ is the linear transformation matrix. $\hat{Q} \in \mathbb{R}^{n_x \times n_x}$ is the state matrix designed for the original state space. Since the the input is not lifted, the R matrix does not need to be transformed.

V. Result

In this section, we evaluate the proposed RLS identification method within the Koopman framework, comparing it to a baseline method that uses first-order optimization for linear system identification using the multi-step loss problem (13). First, we compare the performance of our proposed algorithm with the baseline in learning the linear dynamics of a simple two-state nonlinear system. We then test the learned models by simulating the dynamics of the system over 200 time steps, using only the initial state $\psi(x_0)$ and the control inputs. In the second case study, we evaluate the proposed approach against the baseline to learn a model for a quadruple process and use the learned models with LQR to control the original nonlinear system.

A. RLS with Deep Koopman

Consider the following two-state nonlinear system taken from [28],

$$\dot{x}_1 = \mu x_1,$$
$$\dot{x}_2 = \lambda(x_2 - x_1^2) + u, \tag{26}$$

where μ and λ are constants chosen to determine the behavior of the system, set to $\lambda = 1$ and $\mu = 0.5$, resulting in an unstable system. A data set of 200 trajectories was used, each consisting of $n = 200$ steps with a step size of $t_s = 0.2$. The inputs in the dataset are chosen so that the closed-loop dynamics are stable.

Similar to [28] and [20], our goal is to obtain a representation of a three-state linear system in which the nonlinear dynamics can be represented linearly. To achieve this, we use an encoder and decoder network, each consisting of an input layer, a single hidden layer with 16 neurons and an output layer. We use ReLU activations between the input and hidden layers, and between the hidden and the output layers. The model is trained using 100 trajectories from the dataset. As a baseline, we use parametrized matrices $[A(\theta)|B(\theta)]$ and use first-order optimization, namely Adams, to learn the system dynamics simultaneously using (13), along with learning the weights of the encoder and decoder.

In this case study, we perform a comparison with six baseline models with fixed neural network structure trained with different multi-step losses specifically, with k-steps = [1, 4, 8, 16, 32, 64]. To ensure a fair evaluation, we initialize the $A(\theta)$ matrix with a diagonal matrix of 0.5, while setting the $B(\theta)$ matrix is set to 0 for both the proposed and baseline methods. Thus, the only difference is that the linear dynamics for the proposed method is learned using our RLS scheme. For the RLS setting, we set the k-steps = 200 and γ was fixed at 0.005. The results of this comparison are shown in Fig. 2.

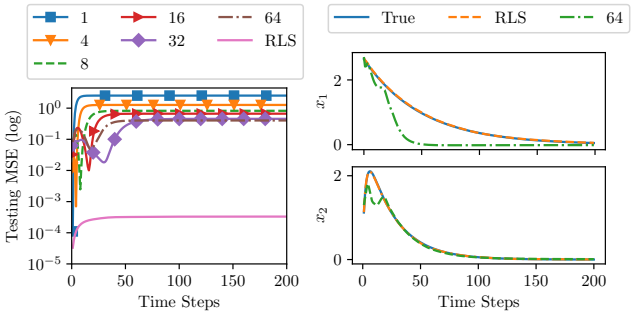

Fig. 2. Deep Koopman for a two-state nonlinear system: The left plot shows the test Mean Squared Error (MSE) over 100 test trajectories. These models were evaluated for their ability to linearly simulate the system dynamics over 200 time steps, using only $\psi(x_0)$ and the control inputs. On the right, we present a qualitative evaluation of the predictions made by the proposed and baseline methods on one of the test trajectories.

The results, as shown in Fig.2, indicate that increasing the number of steps for the baseline leads to an overall improvement in the simulation for 200 steps. However, going beyond k-steps = 64 does not improve performance. This is probably due to the model reaching a local minimum during training.

In contrast, the performance of the RLS model appears to be unaffected by this issue, as it relies on optimizing the multi-step prediction using a recursive convex operation. It is important to note that the identification of the nonlinear

transformation function remains non-convex and is optimized using Adams for both the baseline and proposed methods, as these are neural networks with ReLU activation functions.

B. LQR in the embedding space

Consider the following four-state nonlinear system taken from [30]:

$$\begin{aligned}
\dot{h}_1 &= -\frac{a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{\gamma_1 k_1}{A_1}u_1, \\
\dot{h}_2 &= -\frac{a_2}{A_2}\sqrt{2gh_2} + \frac{a_4}{A_2}\sqrt{2gh_4} + \frac{\gamma_2 k_2}{A_2}u_2, \\
\dot{h}_3 &= -\frac{a_3}{A_3}\sqrt{2gh_3} + \frac{(1-\gamma_2)k_2}{A_3}u_2, \\
\dot{h}_4 &= -\frac{a_4}{A_4}\sqrt{2gh_4} + \frac{(1-\gamma_1)k_1}{A_4}u_1,
\end{aligned} \tag{27}$$

where $[h_1, h_2, h_3, h_4] \in \mathbb{R}^4$ represent the heights (in meters) of the liquid in each of the four tanks, while $[u_1, u_2] \in \mathbb{R}^2$ denote the system inputs in volts (V). The parameters γ_1 and γ_2 refer to the valve constants, that regulate the flow in all four tanks and are set to 0.75. The remaining variables are constants, as defined in [30].

We generated a dataset of 1100 trajectories by simulating (27) with control inputs from an MPC controller. The sampling time was set to $t_s = 1s$, and the trajectory length was fixed to $n = 40$ for all trajectories. The initial states were randomly sampled from a uniform distribution in the range $[0, 0, 0, 0] \leq [h_1, h_2, h_3, h_4] \leq [10, 10, 10, 10]$, while the controller set-point was sampled from a uniform distribution within the range $[10, 10] \leq [h_1, h_2] \leq [20, 20]$. 800 trajectories were used for training, while the remaining 300 trajectories were reserved for testing.

Different latent space sizes were tested to find the optimal balance between prediction accuracy and embedding space dimensionality. An embedding space of 8 states showed the best compromise. The encoder and decoder are designed with two hidden layers of 32, 16 neurons each and ReLU activation functions. For the baseline, six models were trained with k-steps = [1, 4, 8, 16, 32]. The results can be seen in Fig. 3.

The results in Fig. 3 show a similar trend to the previous case study. Increasing the number of k-steps improves the accuracy of the simulation up to a point where the learning converges to a local minimum. In the given example it can be seen that between [8, 16, 32] steps an increase in k-steps did not lead to any improvement in the performance.

Moving on, we apply an LQR tracking controller to the learned linear models to control the nonlinear quadruple tank. In this example, we set the state cost matrix as $Q = [1, 1, 0, 0]$, since our goal is to control the levels of the lower tanks, namely h_1 and h_2. The $Q \in \mathbb{R}^{n_x}$ matrix can be transformed into the Koopman space as $\tilde{Q} \in \mathbb{R}^{n_z}$ using (24). To compute the linear transformation matrix T, we apply (25) to 300 randomly selected trajectories from the training dataset. Using the learned $A(\theta)$ and $B(\theta)$ matrices, we determine the control gain matrix K^* using (22). Finally,

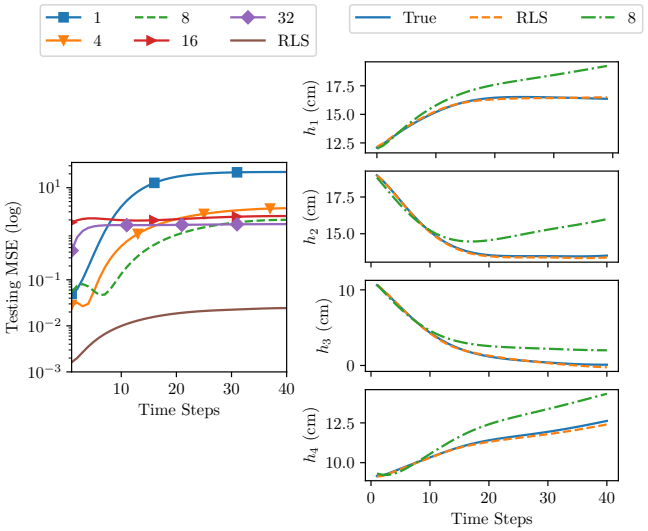

Fig. 3. Deep-Koopman for the quadruple tank: The left plot displays the testing Mean Squared Error (MSE) across 300 trajectories. These models were evaluated for their ability to linearly simulate system dynamics over 40 time steps, using only $\psi(x_0)$ and the control inputs. On the right, we present a qualitative evaluation of the predictions made by the proposed and baseline methods on one of the test trajectories.

for a given set-point, we compute the optimal inputs using (23).

In this example, we compare the controllers designed by both the proposed scheme and the baseline, which was trained with an 8-step loss. For each controller, we compute the open-loop control inputs for a span of 80 time steps. We achieve this by initializing the initial point as $z_0 = \psi(x_0)$ and setting the desired target points as $z_{\text{target},1} = \psi(x_{\text{target},1})$ between $0 \leq k < 40$ and $z_{\text{target},2} = \psi(x_{\text{target},2})$ between $40 \leq k \leq 80$. Here, $\psi(\cdot)$ denotes the encoder, which is different for both the proposed and baseline models. Finally, we apply these calculated inputs to the original systems. The results are shown in Fig. 4.

The results show that open-loop inputs computed using the proposed model (RLS (OL)) effectively guide the original system to the set-point with minimal steady-state error. In contrast, the open-loop performance of the baseline model (8-step (OL)) exhibits compounding errors that diverge beyond the limits of the training dataset. Moving to the closed-loop scenario (8-step (CL)), the baseline shows improved performance with reduced steady-state error, although it remains inferior to that of the controller designed using the proposed model in terms of meeting the control objective.

VI. Conclusion and Future Work

This work proposes a deep learning Koopman framework that uses recursive least squares to effectively address the optimization complexities associated with multi-step system identification. This scheme replaces the use of multi-step loss by solving a recursive convex problem. The result is linear models that can be reliably used to perform long multi-step predictions in the embedding space with increased accuracy. Furthermore, this work utilizes the learned linear model to

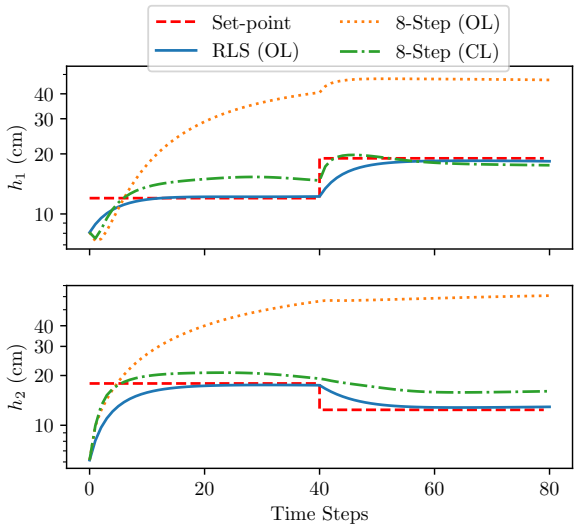

Fig. 4. Koopman LQR: Comparison of the performance of the linear controller on the original non linear system. We evaluate the control inputs of the linear model obtained by the proposed scheme, operated in an open-loop manner. This is compared with the performance of the 8-step model, which is operated in both open-loop and closed-loop modes.

control the original nonlinear system using LQR through a linear transformation of the state cost matrix. The results show that the controllers designed using the models learned by the proposed scheme outperform those of the baseline in terms of meeting the control objectives.

Our future work will evaluate the performance of the proposed algorithms in more realistic scenarios where some states are not measurable, and with nonlinear systems with more complex dynamics.

References

[1] Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022.

[2] Rajesh Kumar, Smriti Srivastava, JRP Gupta, and Amit Mohindru. Comparative study of neural networks for dynamic nonlinear systems identification. *Soft Computing*, 23:101–114, 2019.

[3] S Joe Qin and Thomas A Badgwell. An overview of nonlinear model predictive control applications. *Nonlinear model predictive control*, pages 369–392, 2000.

[4] Yusuke Igarashi, Masaki Yamakita, Jerry Ng, and H Harry Asada. Mpc performances for nonlinear systems using several linearization models. In *2020 American Control Conference (ACC)*, pages 2426–2431. IEEE, 2020.

[5] Bernard O Koopman. Hamiltonian systems and transformation in hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.

[6] Bernard O Koopman and J v Neumann. Dynamical systems of continuous spectra. *Proceedings of the National Academy of Sciences*, 18(3):255–263, 1932.

[7] Katsuhiko Ogata. *Modern control engineering fifth edition*. Prentice Hall PTR, 2010.

[8] Kenneth R Muske and James B Rawlings. Model predictive control with linear models. *AIChE Journal*, 39(2):262–287, 1993.

[9] Brian DO Anderson and John B Moore. *Optimal control: linear quadratic methods*. Courier Corporation, 2007.

[10] Clarence W Rowley, Igor Mezić, Shervin Bagheri, Philipp Schlatter, and Dan S Henningson. Spectral analysis of nonlinear flows. *Journal of fluid mechanics*, 641:115–127, 2009.

[11] Matthew O Williams, Maziar S Hemati, Scott TM Dawson, Ioannis G Kevrekidis, and Clarence W Rowley. Extending data-driven koopman analysis to actuated systems. *IFAC-PapersOnLine*, 49(18):704–709, 2016.

[12] Xu Ma, Bowen Huang, and Umesh Vaidya. Optimal quadratic regulation of nonlinear system using koopman operator. In *2019 American Control Conference (ACC)*, pages 4911–4916. IEEE, 2019.

[13] Milan Korda and Igor Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, 2018.

[14] Joshua L Proctor, Steven L Brunton, and J Nathan Kutz. Generalizing koopman theory to allow for inputs and control. *SIAM Journal on Applied Dynamical Systems*, 17(1):909–930, 2018.

[15] Qianxiao Li, Felix Dietrich, Erik M Bollt, and Ioannis G Kevrekidis. Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the koopman operator. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(10), 2017.

[16] Enoch Yeung, Soumya Kundu, and Nathan Hodas. Learning deep neural network representations for koopman operators of nonlinear dynamical systems. In *2019 American Control Conference (ACC)*, pages 4832–4839. IEEE, 2019.

[17] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):4950, 2018.

[18] Yiqiang Han, Wenjian Hao, and Umesh Vaidya. Deep learning of koopman representation for control. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 1890–1895. IEEE, 2020.

[19] Jan C Schulze and Alexander Mitsos. Data-driven nonlinear model reduction using koopman theory: Integrated control form and nmpc case study. *IEEE Control Systems Letters*, 6:2978–2983, 2022.

[20] Akhil Ahmed, Ehecatl Antonio del Rio-Chanona, and Mehmet Mercangöz. Linearizing nonlinear dynamics using deep learning. *Computers & Chemical Engineering*, 170:108104, 2023.

[21] Haojie Shi and Max Q-H Meng. Deep koopman operator with control for nonlinear systems. *IEEE Robotics and Automation Letters*, 7(3):7700–7707, 2022.

[22] H-T Su and TJ McAvoy. Neural model predictive control of nonlinear chemical processes. In *Proceedings of 8th IEEE International Symposium on Intelligent Control*, pages 358–363. IEEE, 1993.

[23] Adrian Wills, Chengpu Yu, Lennart Ljung, and Michel Verhaegen. Affinely parametrized state-space models: Ways to maximize the likelihood function. *IFAC-PapersOnLine*, 51(15):718–723, 2018.

[24] Pablo A Parrilo and Lennart Ljung. Initialization of physical parameter estimates. *IFAC Proceedings Volumes*, 36(16):1483–1488, 2003.

[25] Torsten Söderström, Lennart Ljung, and Ivar Gustavsson. *A comparative study of recursive identification methods*. Lund Institute of Technology Sweden, 1974.

[26] Horacio M Calderón, Erik Schulz, Thimo Oehlschlägel, and Herbert Werner. Koopman operator-based model predictive control with recursive online update. In *2021 European Control Conference (ECC)*, pages 1543–1549. IEEE, 2021.

[27] Zuowei Ping, Zhun Yin, Xiuting Li, Yefeng Liu, and Tao Yang. Deep koopman model predictive control for enhancing transient stability in power grids. *International Journal of Robust and Nonlinear Control*, 31(6):1964–1978, 2021.

[28] Steven L Brunton, Marko Budišić, Eurika Kaiser, and J Nathan Kutz. Modern koopman theory for dynamical systems. *arXiv preprint arXiv:2102.12086*, 2021.

[29] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49. JMLR Workshop and Conference Proceedings, 2012.

[30] Karl Henrik Johansson. The quadruple-tank process: A multivariable laboratory process with an adjustable zero. *IEEE Transactions on control systems technology*, 8(3):456–465, 2000.