IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Real-Time Constrained Tracking Control of Redundant Manipulators Using a Koopman-Zeroing Neural Network Framework

Chandan Kumar Sah*, Rajpal Singh* *Student Member, IEEE* and Jishnu Keshavan, *Member, IEEE*

***Abstract*—This study proposes a combined Koopman-ZNN (Zeroing Neural Network) architecture for real-time control of redundant manipulators subject to input constraints. An autoencoder-based neural architecture is employed to discover the bilinear Koopman model for manipulator dynamics in joint space using input-output data, which is subsequently integrated with a feed-forward neural network that maps the joint coordinates to end-effector Cartesian coordinates. The proposed architecture allows for efficient learning of highly accurate models using a significantly lower number of observable states compared to the previous studies. This learning architecture is then coupled with a ZNN controller, which offers a computationally inexpensive alternative to state-of-the-art Nonlinear Model Predictive Control (NMPC) controllers whose computational burden might render real-time control infeasible. The low-dimensional nature of the learned model, combined with a computationally inexpensive ZNN controller, facilitates real-time control applications with improved tracking accuracy. Simulation and experimental studies of trajectory tracking, including performance comparisons with leading alternative designs, are used to verify the efficacy of the proposed scheme.**

***Index Terms*—Redundant Robots, Model Learning for Control, Machine Learning for Robot Control.**

I. Introduction

THE development of a generally applicable and scalable framework for the control of nonlinear systems remains an elusive problem. The nonlinearity and high dimensionality of these systems make the design of a simple and tractable model-based controller challenging. The application of model-based control techniques requires computational feasibility, which is often achieved via local/global linearization. Local linearization techniques like first-order Taylor series linearization are often used. However, they cannot accurately model a nonlinear system outside the vicinity of its equilibrium point.

In contrast, global linearization techniques, such as Koopman theory, provide linear models that accurately describe the behavior of the nonlinear system across their entire workspace [1] and hence, offer better prediction performance compared to the local linearization techniques [2]. Koopman theory employs observable functions that evolve linearly in the observable space via the Koopman operator [3]. However, a complete description of the system dynamics requires an infinite number of observable functions, rendering it infeasible for practical applications. Thus, several data-driven techniques such as DMD [4], DMDc [5], EDMD [6], and SINDy [7] have been proposed for the construction of finite-dimensional Koopman linear models. However, the problem of selecting a finite set of basis functions that are capable of fully representing the original dynamics in the lifted space remains open. While prior knowledge about rudimentary systems may be leveraged to carefully select observable functions, the choice of an optimal set is rendered difficult for more complex systems. To address this issue, several learning-based techniques have been devised. For instance, dictionary learning is used to implement EDMD in [8] whereas autoencoders are used to realize SINDy in [9] and Koopman eigenvalue decomposition in [10].

A wide range of robotic systems is described by control-affine systems, which are characterized by state variant control vector fields. The dynamics for these systems in lifted space include nonlinear actuation terms that could make control policy implementation difficult. Bilinear Koopman models offer a good balance between model accuracy and ease of control implementation. Hence, they are commonly used for control applications, often being paired with the model predictive control (MPC) framework. MPC-based bilinear Koopman frameworks have been used for trajectory tracking in planar serial manipulators [11], quadrotors [12], and soft robots [13], [14]. However, MPC is inherently computationally expensive, thus, real-time control may be infeasible without a reduction in the efficacy of MPC to ease the computational burden, which is well observed in studies [11], [13], [14] where real-time experiments often result in poor tracking behavior.

To overcome these drawbacks, this paper proposes a novel autoencoder-based learning architecture for learning serial manipulator dynamics. The proposed architecture utilizes an autoencoder to capture the bilinear relationship between torque inputs and joint angle/velocity outputs, building on the Koopman Canonical Transform (KCT) [15], which provides a mathematical framework to obtain accurate bilinear models with eigenfunctions as observables. The autoencoder architecture is subsequently paired with a feed-forward neural network, which generates the forward map from the joint coordinates to the Cartesian end-effector coordinates. As a result, the proposed framework can efficiently learn the dynamics through a

* These authors have contributed equally to the present work.

Manuscript received: October, 12, 2023; Accepted: December, 17, 2023.

This paper was recommended for publication by Editor Jaydev P. Desai upon evaluation of the Associate Editor and Reviewers' comments.

This research was supported in part by the Institute of Eminence grant No. IE/RERE-21-0537.08. The authors would also like to acknowledge AI and Robotics Park (ARTPark), I-Hub, Bengaluru for access to their facilities.

The authors are with the Department of Mechanical Engineering, Indian Institute of Sciences, Bangalore, Karnataka 560012, India (email: chandanks@iisc.ac.in, rajpalsingh@iisc.ac.in, kjishnu@iisc.ac.in).

Digital Object Identifier (DOI): see top of this page.

Copyright ©2024 IEEE

relatively low-dimensional Koopman model compared to the previous studies such as [11]. In addition, this paper proposes using zeroing neural network (ZNN)-based controllers, which incur a lower computational burden (compared to the MPC policy) while still utilizing the Koopman bilinear model to achieve efficient control with consideration for input constraints. In contrast to previous studies that employ Koopman-MPC architectures for the control of serial manipulators [11], the Koopman-ZNN architectures allow for efficient real-time tracking with control loop frequencies of up to 800 Hz. The benefits of the proposed scheme are demonstrated through simulation and experimental studies, including comparison with leading alternative Koopman [11] and non-Koopman-based approaches [16]. In particular, in contrast with these studies which have only demonstrated planar (2D) trajectory tracking [11] [16], the proposed model is extended to achieve trajectory tracking with serial manipulators with a 3D workspace.

The rest of the paper is organized as follows. Section II includes the preliminaries to the Koopman theory. Section III and Section IV present the details of the neural architecture employed for learning the Koopman dynamics and the associated ZNN controller, respectively. Section V includes the simulation and the experimental results. The conclusions are encompassed in Section VI. The major contributions of this paper are as follows:

1) A novel neural network architecture is proposed that enables efficient learning for serial manipulators via a low-dimensional bilinear Koopman model.
2) This learning framework is then coupled with a novel ZNN controller that is capable of stringently constraining the inputs in a computationally efficient manner.
3) Compared to prior studies that showcase model learning of manipulators with a 2D Cartesian workspace [11] [16], the proposed work extends this capability to enable model learning of manipulators with a 3D workspace.
4) The cost-effective algorithm allows for high-frequency control loop execution, reaching up to 800 Hz during experiments. Simulation and experiments confirm the superior performance of our proposed schemes.

II. Preliminaries

This section provides the mathematical framework for the modeling of control-affine nonlinear systems by leveraging the Koopman Canonical Transform (KCT).

A. Koopman Theory

Consider a continuous-time nonlinear autonomous dynamical system defined by $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathbb{X} \subseteq \mathbb{R}^d$ and $\boldsymbol{f} : \mathbb{X} \to \mathbb{X}$ are considered to be Lipschitz continuous on $\mathbb{X}$. The solution of the aforementioned ODE with the initial condition $\boldsymbol{x_0}$ is the flow map, $\boldsymbol{\Phi}^t(\boldsymbol{x_0})$. Consider a set of complex-valued scalar observable functions $\vartheta : \mathbb{X} \to \mathbb{C}$ which reside in Banach space $\mathcal{F}$ i.e. $\vartheta \in \mathcal{F}$, then the continuous time Koopman (semi-)group of operators, $\mathcal{K}^t : \mathcal{F} \to \mathcal{F}$, associated with flow $\boldsymbol{\Phi}^t$, describes the evolution of the observables through the composition

$$\mathcal{K}^t\vartheta(\boldsymbol{x_0}) = \vartheta \circ \boldsymbol{\Phi}^t(\boldsymbol{x_0}),\ \forall\ \vartheta \in \mathcal{F} \tag{1}$$

The Koopman operator possesses an intriguing characteristic of linearity on $\mathcal{F}$; However, this advantage is offset by its infinite-dimensional nature.

B. Koopman Canonical Transform (KCT)

Consider a general nonlinear system in a control-affine form

$$\dot{\boldsymbol{x}} = \boldsymbol{f_0}(\boldsymbol{x}) + \sum_{i=1}^{m} \boldsymbol{f}_i(\boldsymbol{x})u_i,\ \boldsymbol{\theta} = \boldsymbol{h}(\boldsymbol{x}), \tag{2}$$

where, $u_i \in \mathbb{R}\ \forall\ i = 1, ..., m$ are the control inputs, $\boldsymbol{\theta} \in \mathbb{R}^p$ is the output vector, $\boldsymbol{f_0} : \mathbb{X} \to \mathbb{R}^d$ is the drift vector field, $\boldsymbol{f}_i : \mathbb{X} \to \mathbb{R}^d\ \forall\ i = 1, ..., m$ are state dependent control vector fields and $\boldsymbol{h} : \mathbb{X} \to \mathbb{R}^p$ is the output function. The PDE governing the evolution of the time-varying observable, $\boldsymbol{\psi}(t, \boldsymbol{x})$, for the control-affine system (2) is given by [17]

$$\frac{\partial \psi}{\partial t} = \mathcal{L}_{\boldsymbol{f_0}}\psi + \sum_{i=1}^{m} u_i\mathcal{L}_{\boldsymbol{f}_i}\psi,\ \ \psi(0, \boldsymbol{x}) = \vartheta(\boldsymbol{x_0}), \tag{3}$$

where $\mathcal{L}_{\boldsymbol{f}_i} = \boldsymbol{f}_i.\nabla\, \forall\, i{=}0, ..., m$ are the corresponding Lie derivatives. While the system dynamics presented in (3) may resemble general bilinear dynamics, it is important to note that the Lie derivatives $\mathcal{L}_{\boldsymbol{f}_i}$ are infinite-dimensional operators that operate within the Hilbert space of observables. Nonetheless, these infinite-dimensional operators can be represented in a finite-dimensional space by selecting a suitable basis for the observable space and projecting the operators onto it. The KEFs, which establish coordinates evolving linearly along the flow, serve as a natural choice of basis functions for this purpose, allowing the infinite-dimensional operators to be effectively projected onto a finite-dimensional space.

The mathematical transformation of the state space basis to KEFs, known as the Koopman Canonical Transform (KCT) [15], allows for the bilinearization of control-affine dynamics. With a sufficiently large number of basis functions ($n \to \infty$), any observable function can be expressed in terms of KEFs within the observable space. Therefore, the state vector $\boldsymbol{x}$ can be written as $\boldsymbol{x} = \sum_{i=1}^{n} \phi_i(\boldsymbol{x})\boldsymbol{v}_i^{\boldsymbol{x}}$; likewise, the output function can be expressed as $\boldsymbol{h}(\boldsymbol{x}) = \sum_{i=1}^{n} \phi_i(\boldsymbol{x})\boldsymbol{v}_i^{\boldsymbol{h}}$, where $\boldsymbol{v}_i^{\boldsymbol{x}} \in \mathbb{C}^d$ and $\boldsymbol{v}_i^{\boldsymbol{h}} \in \mathbb{C}^p$ are the respective Koopman eigenmodes. Invoking the transformation $\boldsymbol{z} = \boldsymbol{T}(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), ..., \phi_n(\boldsymbol{x})]^T$, the system (2) in the new coordinates becomes:

$$\dot{\boldsymbol{z}}{=}\boldsymbol{D_c}\boldsymbol{z} + \sum_{i=1}^{m} \mathcal{L}_{\boldsymbol{f}_i}\boldsymbol{T}\ (\boldsymbol{x})u_i|_{\boldsymbol{x}=\boldsymbol{C^x}\boldsymbol{z}},\ \boldsymbol{x}{=}\boldsymbol{C^x}\boldsymbol{z},\ \boldsymbol{\theta}{=}\boldsymbol{C^h}\boldsymbol{z}, \tag{4}$$

where $\boldsymbol{C^x} = [\boldsymbol{v_1^x}|...|\boldsymbol{v_n^x}]$ and $\boldsymbol{C^h} = [\boldsymbol{v_1^h}|...|\boldsymbol{v_n^h}]$, $\boldsymbol{D_c} \in \mathbb{R}^{n\times n}$ is a diagonal matrix with diagonal entries $\boldsymbol{D}_{\boldsymbol{c}i,i} = \lambda_i$. Note that the KCT transformation defined above is valid only for real eigenfunctions and a more comprehensive analysis pertaining to complex eigenfunctions can be found in [17].

The dynamics expressed in canonical coordinates (4) remain nonlinear with respect to the control term, but they can be linearized through the application of the following theorems.

Theorem 1 [17]: The system dynamics (4) is bilinearizable with an infinite number of basis functions if the span of the KEFs corresponding to the drift vector field is an invariant

subspace of $\mathcal{L}_{\boldsymbol{f}_i}, i = 1, ..., m$, *i.e.,* $\mathcal{L}_{\boldsymbol{f}_i}\phi_k \in$ *span* $\{\phi_j : j = 1, 2, ..., \infty\}, \forall i = 1, ..., m;\ k = 1, 2, ..., \infty$.
This implies that $\forall k = 1, 2, ..., \infty$, we have $\mathcal{L}_{\boldsymbol{f}_i}\phi_k = \sum_{j=1}^{\infty} v_j^{\boldsymbol{f}_i}\phi_j$, where $v_j{}^{\boldsymbol{f}_i} \in \mathbb{R}$. Then, the system dynamics in an infinite basis can be written as:

$$\dot{\boldsymbol{z}} = \boldsymbol{D_c z} + \sum_{i=1}^{m} \boldsymbol{B}_{\boldsymbol{c},i}\boldsymbol{z}u_i, \tag{5}$$

where, $\boldsymbol{B}_{\boldsymbol{c},i} = [\boldsymbol{v}_1^{\boldsymbol{f}_i}|\boldsymbol{v}_2^{\boldsymbol{f}_i}...]$. Though system (5) is the bilinear transformation of the original nonlinear dynamics, it is infinite dimensional consisting of the KEFs $\boldsymbol{z} \in \mathbb{R}^{\infty}$. However, for a finite-dimensional bilinearization of (4), the invariant eigensubspace of $\mathcal{L}_{\boldsymbol{f_0}}$ must be spanned by a finite number of KEFs [17]. To this end, we have the following theorem.

Theorem 2: [17] Assume that there exist KEFs of the drift vector field, f_o *such that* $\{\phi_j : j = 1, ..., n\}, n \in \mathbb{N}, n < \infty$ *and span*$\{\phi_1, ..., \phi_n\}$ *forms an invariant subspace of* $\mathcal{L}_{f_i}, i = 1, ..., m$. *Then the system dynamics can be represented using a finite set of KEFs as basis so that the bilinear dynamics of (5) can be written in finite-dimensional transformed coordinates,* $\boldsymbol{z} \in \mathbb{R}^n, n < \infty$.
While achieving bilinearization through a finite number of KEFs can be challenging, selecting an appropriate number of basis functions can help reduce approximation errors to an acceptable level. We enable this by automating the choice of the basis functions using the autoencoder framework.

III. Learning the Koopman Bilinear Dynamics

This section presents the architecture of the neural network structure used for learning the finite-dimensional Koopman operator from data.

A. Neural Architecture for Learning Koopman Embeddings

Koopman operator theory provides the mathematical foundation to facilitate the simplification of complex nonlinear systems by transforming them into linear or bilinear forms. However, constructing the Koopman operator that accurately captures and represents the system's dynamics can pose a significant challenge. Dynamic Mode Decomposition (DMD) [4] and Extended Dynamic Mode Decomposition (EDMD) [6] are data-driven techniques that have emerged as powerful methods for learning the Koopman operator. A detailed overview of the EDMD-based methods for learning the bilinear model of control-affine systems is available in [11].

The classical EDMD approach is limited by its dependence on the manual selection of the basis functions of the observable space via trial and error. This may lead to inaccurate bilinear models as there is no assurance that the bilinearization assumptions invoked in Theorem 2 would hold. To mitigate this, we present a neural network-based approach that can simultaneously learn the Koopman eigenfunctions and bilinear dynamics matrices directly from data, eliminating the need for the manual selection of dictionary functions. The architecture of the neural network is shown in Fig. 1. It includes an encoder that maps states to an observable space, a fully connected layer of linear neurons, and the projection layer representing the $\boldsymbol{C}^{\boldsymbol{x}} \in \mathbb{R}^{n\times d}$ matrix that projects from observable space to the original states. The neural network is designed to learn both the mapping to lifted space and model matrices concurrently by minimizing a relevant loss function that can be formulated to satisfy the specific requirements of various sections of the network. These loss functions are defined below [12]:

1) Reconstruction Loss: The network is expected to lift the dynamics from the state space to the Koopman Invariant subspace, followed by projecting the lifted state $\boldsymbol{z}_k$ at time instant t_k back to the original state $\boldsymbol{x}_k$. This is done through the use of an encoder network, $\boldsymbol{\xi}(\boldsymbol{x}_k, \boldsymbol{W}_E)$, where $\boldsymbol{W}_E$ represents the parameters of the network, and the linear layer, $\boldsymbol{C}^{\boldsymbol{x}}$, respectively. To minimize the difference between the original states and the states obtained through the projection of the lifted states, a loss function is defined as $L_{rec} = \|\boldsymbol{x}_k - \boldsymbol{C}^{\boldsymbol{x}}\boldsymbol{\xi}(\boldsymbol{x}_k, \boldsymbol{W}_E)\|_2$.

2) Prediction Loss: : The layer with linear neurons, representing the bilinear dynamics, advances the lifted states, $\boldsymbol{z}_k$, one time-step ahead to $\hat{\boldsymbol{z}}_{k+1}$, which is then projected back to the state space to obtain the predicted base state $\hat{\boldsymbol{x}}_{k+1}$. The prediction loss quantifies the discrepancy between the actual state $\boldsymbol{x}_{k+1}$ and the predicted state $\hat{\boldsymbol{x}}_{k+1}$ at the $(k+1)^{th}$ time-step, which is defined as $L_{pred} = \|\boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}_{k+1}\|_2 = \|\boldsymbol{x}_{k+1} - \boldsymbol{C}^{\boldsymbol{x}}(\boldsymbol{D}\boldsymbol{\xi}(\boldsymbol{x}_k, \boldsymbol{W}_E) - \sum_{i=1,..,m} \boldsymbol{B}_i\boldsymbol{\xi}(\boldsymbol{x}_k, \boldsymbol{W}_E)u_{i,k})\|_2$, where $\boldsymbol{D} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{B}_i \in \mathbb{R}^{n\times n}$, are the matrix representation of the weights of the fully-connected layer of linear neurons with inputs $\boldsymbol{z}_k$ and $\boldsymbol{z}_k u_{i,k}$, respectively.

3) Lifted State Prediction Loss: : It encompasses the loss associated with the prediction of the lifted states at $(k+1)^{th}$ time-step and the actual lifted state obtained by encoding state $\boldsymbol{x}_{k+1}$, i.e, $L_{lift} = \|\boldsymbol{z}_{k+1} - \hat{\boldsymbol{z}}_{k+1}\|_2 = \|\boldsymbol{\xi}(\boldsymbol{x}_{k+1}, \boldsymbol{W}_E) - (\boldsymbol{D}\boldsymbol{\xi}(\boldsymbol{x}_k, \boldsymbol{W}_E) - \sum_{i=1,..,m} \boldsymbol{B}_i\boldsymbol{\xi}(\boldsymbol{x}_k, \boldsymbol{W}_E)u_{i,k})\|_2$.

To sum up, the overarching objective is to minimize the combined loss, $L(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}, \boldsymbol{u}_k) = \alpha_1 L_{pred} + \alpha_2 L_{Rec} + \alpha_3 L_{lift} + \alpha_4\|\boldsymbol{W}\|_2$ over the entire input-output dataset iteratively, where $\alpha_1, \alpha_2, \alpha_3$ and α_4 are weighting scalars. Note that the total loss also includes the L_2 regularization term on the total weights of the network, to prevent overfitting.
Remark 1: In [11], the learning algorithm uses the Cartesian position and velocity of the various joints as the states of the system, for modeling the serial manipulator's dynamics. However, comparatively, the use of joint coordinates as the system states requires a lower dimensional observable space to achieve similar accuracy. Therefore, in this study, the states are comprised of joint angles and joint angular velocities. However, tracking control with such a model would require the reference trajectory to be defined in terms of joint angles instead of Cartesian positions. To overcome this issue, a fully connected multi-layer neural network is employed to map joint angle, $\boldsymbol{\theta}_k$, to the Cartesian position of the end-effector, $\boldsymbol{y}_k$ at time instant, t_k. As a result, the learned dynamics consist of a bilinear model along with a nonlinear feed-forward neural network, as shown in Fig. 1.

To maintain consistency and facilitate communication, the neural network architecture responsible for learning the bilinear dynamics is referred to as the Koopman autoencoder network, while the neural network to learn the forward map is referred to as the forward NN and the entire neural network structure is referred to as Koopman neural network.

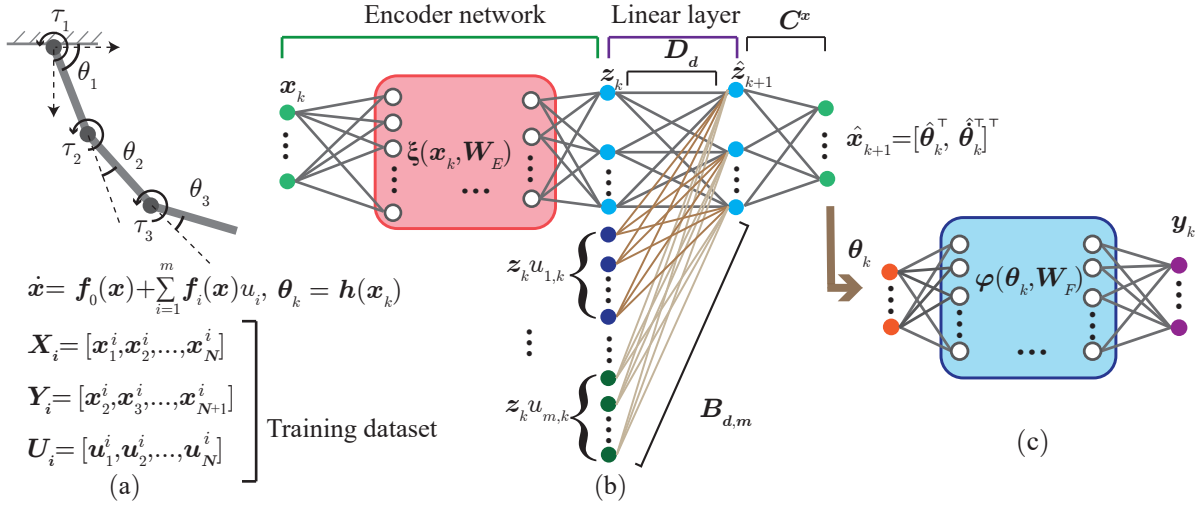


Fig. 1. Neural network architecture for Koopman-based learning. a) Data generation. b) Neural network architecture for Koopman bilinearization c) Neural network architecture for forward map (forward NN).

IV. Zeroing Neural Network-Based Controller

This section provides the details of the ZNN controller design for the bilinear Koopman model. Even though the dynamic equation is bilinear, the forward map from joint angles to the Cartesian position of the end-effector is a fully connected neural network, which is nonlinear and influences the design of the control policy. The learned bilinear dynamics in continuous time can be written as:

$$\begin{aligned} \dot{\boldsymbol{z}}_k &= \boldsymbol{D_c} \boldsymbol{z}_k + \boldsymbol{B_c}(\boldsymbol{z}_k \otimes \boldsymbol{u}_k),\ \boldsymbol{z}_k = \boldsymbol{\xi}(\boldsymbol{x}_k, \boldsymbol{W}_E), \\ \boldsymbol{\theta}_k &= \boldsymbol{C^x} \boldsymbol{z}_k,\ \boldsymbol{y}_k = \boldsymbol{\varphi}(\boldsymbol{\theta}_k, \boldsymbol{W}_F), \end{aligned} \tag{6}$$

where $\boldsymbol{y}_k \in \mathbb{R}^p$ is the position of end-effector at time instant t_k, $\boldsymbol{\varphi}(\boldsymbol{\theta}, \boldsymbol{W}_F)$ denotes a multi-layer fully connected neural network, and $\boldsymbol{D_c}$ and $\boldsymbol{B_c}$ are bilinear model matrices in continuous time. Defining tracking error as, $\boldsymbol{e}_k = \boldsymbol{r}_k - \boldsymbol{r}_{ref,k}$, where $\boldsymbol{r}_k = [\boldsymbol{y}_k^\top,\ \dot{\boldsymbol{y}}_k^\top]^\top$ and $\boldsymbol{r}_{ref,k} = [\boldsymbol{y}_{ref,k}^\top,\ \dot{\boldsymbol{y}}_{ref,k}^\top]^\top$ represent the actual and desired Cartesian state of the end-effector at k^{th} time step, the zeroing dynamics [18] $\dot{\boldsymbol{e}}_k = \dot{\boldsymbol{y}}_k - \dot{\boldsymbol{y}}_{ref,k} = -\boldsymbol{\gamma} \boldsymbol{e}_k$ can be employed to drive tracking error to zero. The learned bilinear dynamics (6) in discrete time $\boldsymbol{z}_k = \boldsymbol{D_d} \boldsymbol{z}_{k-1} + \boldsymbol{B_d}(\boldsymbol{z}_{k-1} \otimes \boldsymbol{u}_{k-1})$ upon differentiation gives, $\dot{\boldsymbol{z}}_k = \boldsymbol{D_d} \dot{\boldsymbol{z}}_{k-1} + \boldsymbol{B_d}(\dot{\boldsymbol{z}}_{k-1} \otimes \boldsymbol{u}_{k-1}) + \boldsymbol{B_d}(\boldsymbol{z}_{k-1} \otimes \dot{\boldsymbol{u}}_{k-1})$, where $\boldsymbol{D_d}$ and $\boldsymbol{B_d}$ are corresponding discrete model matrices. The zeroing dynamics can be written as $\dot{\boldsymbol{e}} = \begin{bmatrix} \dot{\boldsymbol{y}}_k - \dot{\boldsymbol{y}}_{ref,k} \\ \ddot{\boldsymbol{y}}_k - \ddot{\boldsymbol{y}}_{ref,k} \end{bmatrix} = -\gamma \boldsymbol{e}_k$, or equivalently as,

$$\begin{aligned} \tilde{\boldsymbol{J}} \boldsymbol{C^x} (\boldsymbol{D_d} \dot{\boldsymbol{z}}_{k-1} + \boldsymbol{B_d}(\dot{\boldsymbol{z}}_{k-1} \otimes \boldsymbol{u}_{k-1}) + \boldsymbol{B_d}(\boldsymbol{z}_{k-1} \\ \otimes \dot{\boldsymbol{u}}_{k-1})) = \dot{\boldsymbol{r}}_{ref,k} - \boldsymbol{\gamma} \boldsymbol{e}_k \end{aligned} \tag{7}$$

where $\tilde{\boldsymbol{J}} = \begin{bmatrix} \boldsymbol{J} & \boldsymbol{0} \\ \dot{\boldsymbol{J}} & \boldsymbol{J} \end{bmatrix}$. $\boldsymbol{J} = \partial \boldsymbol{\varphi} / \partial \boldsymbol{\theta}_k$ and $\dot{\boldsymbol{J}} = d\boldsymbol{J}/dt$, which can be exactly computed in closed form by differentiating the mathematical expression of the forward map. On simplification, the required control input to drive tracking error is given by,

$$\begin{aligned} \dot{\boldsymbol{u}}_{k-1} = (\boldsymbol{B}_d(\boldsymbol{z}_{k-1} \otimes \boldsymbol{I}))^\dagger (-\boldsymbol{D_d} \dot{\boldsymbol{z}}_{k-1} - \boldsymbol{B}_d(\dot{\boldsymbol{z}}_{k-1} \\ \otimes \boldsymbol{u}_{k-1}) + (\tilde{\boldsymbol{J}} \boldsymbol{C^x})^\dagger (\dot{\boldsymbol{r}}_{ref,k} - \boldsymbol{\gamma} \boldsymbol{e}_k)) \end{aligned} \tag{8}$$

where $(\boldsymbol{J}\boldsymbol{C^x})^\dagger$ is the left Moore-Penrose inverse given by $(\boldsymbol{J}\boldsymbol{C^x})^\dagger = [(\boldsymbol{J}\boldsymbol{C^x})^\top (\boldsymbol{J}\boldsymbol{C^x})]^{-1} (\boldsymbol{J}\boldsymbol{C^x})^\top$. The control input computed from (8) can effectively drive the traced position of the end-effector to the reference position, but a major caveat is that the computed control effort is not guaranteed to remain constrained within predefined bounds, which can be detrimental to both the path tracking task as well as the safety of the robot. To alleviate this problem, we leverage the use of the following nonlinear map that transforms from constrained control input $\boldsymbol{u}$ to the unconstrained variable $\boldsymbol{\chi} \in \mathbb{R}^m$ as,

$$\frac{u_{i,k} - u_i^-}{u_i^+ - u_i^-} = \frac{e^{\chi_{i,k}}}{1 + e^{\chi_{i,k}}} = \omega(\chi_{i,k}),\ \forall\ i = 1, ..., m, \tag{9}$$

where u_i^+ and u_i^- are the corresponding upper and lower limits respectively. It is evident from (9) that map $\omega(\cdot)$ is monotonically increasing and invertible. The motivation for invoking this nonlinear map is evident from the fact that transforming (8) in terms of the unconstrained variable χ_i and solving the control equation in terms of χ_i allows it to remain unconstrained within $(-\infty, \infty)$, which in turn ensures that the control input u_i remains bounded within (u_i^-, u_i^+). From (9), it follows that $u_{i,k} = u_i^- + (u_i^+ - u_i^-)\omega(\chi_{i,k})$, which upon differentiation can be written in vector form as $\dot{\boldsymbol{u}}_k = \boldsymbol{\mathcal{U}} \boldsymbol{\Omega} \dot{\boldsymbol{\chi}}_k$, where $\boldsymbol{\mathcal{U}} = \text{diag}\ [\boldsymbol{u}^+ - \boldsymbol{u}^-] \in \mathbb{R}^{m \times m}$ and $\boldsymbol{\Omega} = \text{diag}[\Omega_{ii}] = \text{diag}\ \left[e^{\chi_{i,k}} / (1 + e^{\chi_{i,k}})^2\right]$, $i = 1, 2, .., m$. By substituting $\dot{\boldsymbol{u}}_k$ in (8), the necessary update equation is obtained in terms of $\boldsymbol{\chi}$ as,

$$\begin{aligned} \dot{\boldsymbol{\chi}}_{k-1} = (\boldsymbol{B}_d(\boldsymbol{z}_{k-1} \otimes \boldsymbol{I}) \boldsymbol{\mathcal{U}} \boldsymbol{\Omega})^\dagger (-\boldsymbol{D_d} \dot{\boldsymbol{z}}_{k-1} - \boldsymbol{B}_d(\dot{\boldsymbol{z}}_{k-1} \\ \otimes \boldsymbol{u}_{k-1}) + (\tilde{\boldsymbol{J}} \boldsymbol{C^x})^\dagger (\dot{\boldsymbol{r}}_{ref,k} - \boldsymbol{\gamma} \boldsymbol{e}_k)) \end{aligned} \tag{10}$$

Owing to the invertible map defined by (9), the control input computed in terms of the unconstrained variable can be mapped back to the constrained variable $\boldsymbol{u}$ and the map ensures that it remains component-wise bounded within $[\boldsymbol{u}^-, \boldsymbol{u}^+]$. The global convergence property of the actual trajectory $\boldsymbol{y}_k$ to the desired trajectory $\boldsymbol{y}_{ref,k}$ can be demonstrated in a straightforward manner as that considered in [19], [20].

V. RESULTS

In this section, simulation and experimental studies have been utilized to demonstrate the efficacy of the proposed algorithms for learning the dynamics and the associated tracking control of serial manipulators.

A. Simulation Results

[1] In this subsection, we present the performance evaluation of the Koopman neural network for learning bilinear Koopman models of manipulator dynamics via simulation studies. In particular, the study involves two types of manipulators: a planar 3R serial manipulator with a 2D workspace and a 5R manipulator with a 3D workspace. To assess the effectiveness of the learned models, we conducted a performance comparison study involving coupling the bilinear models for each manipulator separately with two different controllers: the proposed ZNN controller (10) and NMPC. In addition, the performance of the proposed Koopman-ZNN architecture is compared with the classical EDMD-based approach presented in a prior study [11] and reservoir computing-based model-free method [16] for planar 3R manipulator to substantiate the superior performance of the proposed algorithm.

The model parameters for the planar 3R and the 5R serial robot dynamics used in data collection are set as: the mass of link i is $m_{li} = 0.1$ kg, the length is $l_i = 0.33$ m, the moment of inertia is $I_i = 1$ kgm^2, the distance of the mass center from its respective joint axis is $a_i = l_i/2$.

1) Bilinear Model Validation: The proposed Koopman framework is employed to learn bilinear models, leveraging data gathered from 200 randomly generated trajectories spanning the workspace of both manipulators. All the trajectories consist of 1000 data points sampled at intervals of 5 ms. The required neural networks are implemented using PyTorch.

The dimensionality of the lifted state is a hyperparameter that significantly influences model accuracy and the computational cost of training the neural networks and computing control inputs. An increase in the lifted state dimension enhances the accuracy of the learned model at the cost of higher computational complexity. To balance the trade-off between accuracy and computational cost, we plot the mean normalized error against the dimension of lifted states in Figs. 2a and 2b for planar 3R and 3D-5R manipulators, respectively. It is evident that the optimal lifted state dimensions for the planar 3R manipulator and 5R manipulator are 30 and 41 respectively, while that for the EDMD algorithm in [13] is far higher (Fig. 2c). The slightly higher number of lifted states required for the 3D-5R manipulator may be attributed to its more intricate dynamics.

Table I presents the hyperparameters for the Koopman neural network utilized to learn bilinear dynamics for the planar 3R and 3D-5R manipulators. The table demonstrates that the networks were effectively trained, as indicated by the low validation losses for both the Koopman autoencoder network and forward NN. The network architecture is represented as an array, where each element indicates the number of nodes in the corresponding layer. For instance, in the case of a planar 3R manipulator, the Koopman autoencoder network's architecture is denoted as $[6, 30, 30, 30]$. This signifies that the network has 6 input neurons for the 6 states, two fully connected hidden layers with 30 neurons each, and an output dimension of 30, which represents the required lifted state. Further, LR refers to the learning rate.

[1]https://github.com/Rajpal9/Koopman_ZNN

Table I: Hyperparameters of Koopman Neural Network.

	Koopman autoencoder network		Forward NN	
Robot	**3R 2D**	**5R 3D**	**3R 2D**	**5R 3D**
Architecture	$[6, 30, 30, 30]$	$[10, 30, 30, 41]$	$[3, 20, 20, 2]$	$[5, 20, 20, 3]$
Training loss	$2.73e-6$	$1.9e-7$	$1.75e-5$	$7.5e-5$
Validation loss	$3.24e-6$	$2.4e-7$	$7.58e-5$	$1.58e-4$
Dataset	10^3 snaps, 150 training and 50 validation trajectories			
LR	$4e{-}4$		$1e{-}3$	

Table II: Tracking performance comparison of the proposed ZNN (10) and NMPC controllers.

			Horizon (p, m)	RMSE [m]	Computation time/iteration [s]	Total control effort [Nms]
Planar 3R	Cardioid	NMPC	(10, 10)	0.0019	0.5704	0.7705
			(15, 15)	0.0012	1.335	0.7047
			(20, 20)	0.0017	2.352	**0.6902**
		ZNN (10)	-	**0.00042**	$\mathbf{1.4 \times 10^{-3}}$	0.722
	Epicycloid	NMPC	(10, 10)	0.0341	1.391	4.368
			(15, 15)	0.0038	1.991	2.855
			(20, 20)	0.0025	2.9722	2.3176
		ZNN (10)	-	**0.0014**	$\mathbf{1.7 \times 10^{-3}}$	**2.28**
3D 5R	Cardioid	NMPC	(10, 10)	0.0325	1.192	1.435
			(15, 15)	0.0325	2.503	1.066
			(20, 20)	0.0324	4.4258	**0.9982**
		ZNN (10)	-	**0.0022**	$\mathbf{1.5 \times 10^{-3}}$	1.4944
	Lissajous	NMPC	(10, 10)	0.0564	3.2411	4.489
			(15, 15)	0.0435	8.1868	4.697
			(20, 20)	0.0258	14.30	4.219
		ZNN (10)	-	**0.0035**	$\mathbf{1.5 \times 10^{-3}}$	**1.8448**

2) ZNN-Based Controller: The performance of the ZNN-based controller (10) is evaluated by quantitatively assessing the tracking accuracy of predefined paths with the learned bilinear models for both serial manipulators. For the planar 3R manipulator, Cardioid and Epicycloid paths are traced, where the control input bounds were set to $[-0.1, 0.1]$ Nm and $[-0.3, 0.3]$ Nm, respectively. Table II shows that the tracking errors for both paths are of the order of 10^{-3} m, demonstrating the combined accuracy of the learned bilinear model and the effectiveness of the ZNN-based controller (10). Furthermore, the control inputs remain within the specified bounds as shown in Figs. 3c and 3d, illustrating the benefits of utilizing an invertible nonlinear map (9) for constraining control inputs. For the 5R manipulator, Cardioid and Lissajous paths are traced, where the control input bounds were set to $[-0.25, 0.25]$ Nm for both paths. The tracking error for the Cardioid and Lissajous path is of the order of 10^{-3} m (Table II), substantiating the generalizability of the proposed algorithm to the 3D workspace. Further, the control inputs again remain within the specified bounds (Figs. 4c, and 4d).

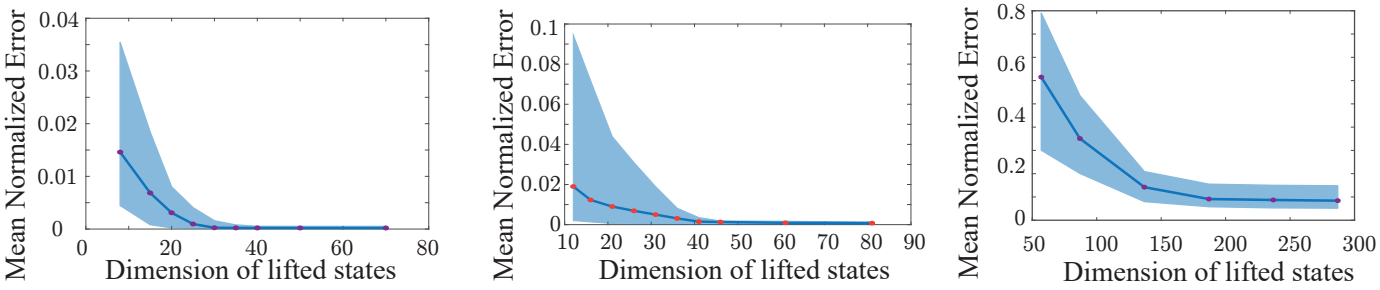


Fig. 2. Mean normalized error (with upper and lower quartiles) vs. lifted state dimension. a) Koopman Neural Network for planar 3R. b) Koopman Neural Network for 5R. c) EDMD-based learning [13] for planar 3R.

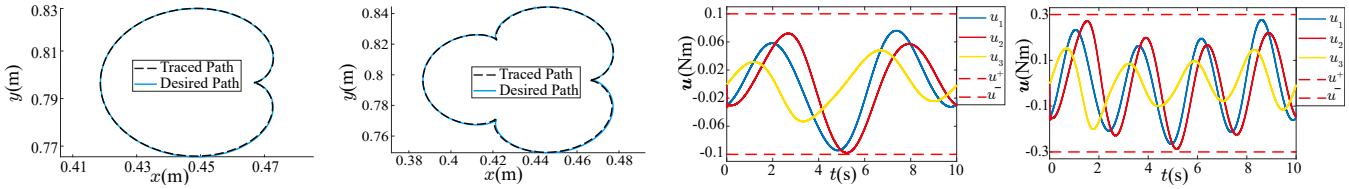


Fig. 3. Simulation results of ZNN-based controller for tracking a planar Cardioid and an Epicycloid path using the learned bilinear model for 3R manipulator. a,b) Desired and the actual traced paths. c,d) Control inputs evolution.

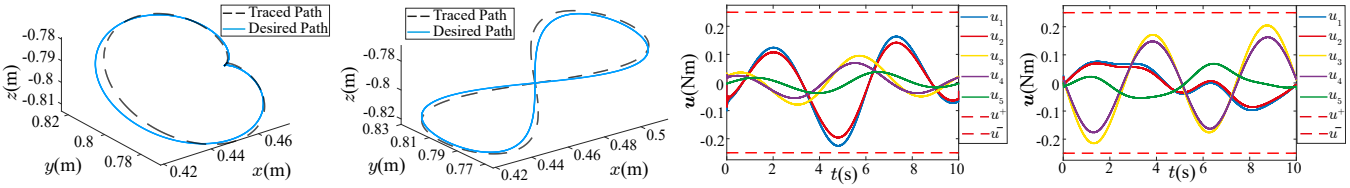


Fig. 4. Simulation results of ZNN-based controller for tracking a 3D Cardioid and a Lissajous path using the learned bilinear model of the 5R manipulator. a,b) Desired and the actual traced paths. c,d) Control inputs evolution.

3) Comparison with NMPC: This section provides an exhaustive comparative analysis between the ZNN-based controller (10) and NMPC to accomplish tracking based on the bilinear model learned using the Koopman neural network. The metrics of average tracking error, total control effort, and computational time are chosen as a basis for comparison. The total control effort is defined as $\int_0^T \|\boldsymbol{u}\|_2 dt$, where T is the total time taken to trace the path. Since the performance of NMPC is highly influenced by the choice of control and prediction horizons [11], simulation studies are carried out for different values of the prediction and control horizons (denoted p and m respectively), and the results are presented in Table II. Even though NMPC calculates control inputs that are optimal over the control horizon, the total control effort obtained by ZNN (10), which does not incorporate a similar optimization process, remains comparable to NMPC. Moreover, the ZNN-based controller demonstrates a lower average tracking error compared to NMPC due to the latter's need to linearize the bilinear dynamics at each time step. On the other hand, the ZNN-based controller (10) does not rely on such linearization, leading to improved tracking performance. Additionally, the ZNN-based controller (10) offers a significant advantage in terms of computational efficiency compared to NMPC. In fact, the reduced computational cost makes it particularly suitable for online applications that require high-frequency control updates.

4) Comparison with Previous Studies: In this section, we evaluate our proposed Koopman-ZNN framework, against two Koopman-based methods [11] and a non-Koopman-based algorithm [16]. The first two methods use the classical EDMD-based learning paradigm with a polynomial basis, coupled with BMPC and NMPC controllers. The non-Koopman-based algorithm is a model-free control framework based on reservoir computing which directly learns control policies from data.

Fig. (2c) indicates that the classical EDMD-based method requires approximately 137 basis functions for learning the dynamics of a 3R planar manipulator, whereas our Koopman Neural network-based approach achieves lower model error with far fewer basis functions for both the 3R and the 5R manipulators (Figs. 2a and 2b). In Fig. 5, we compare the performance of these algorithms in path-tracking tasks for a 3R planar manipulator in terms of tracking accuracy, computation time, and control effort. The K-BMPC algorithm uses linear MPC for control, which is faster compared to K-NMPC but

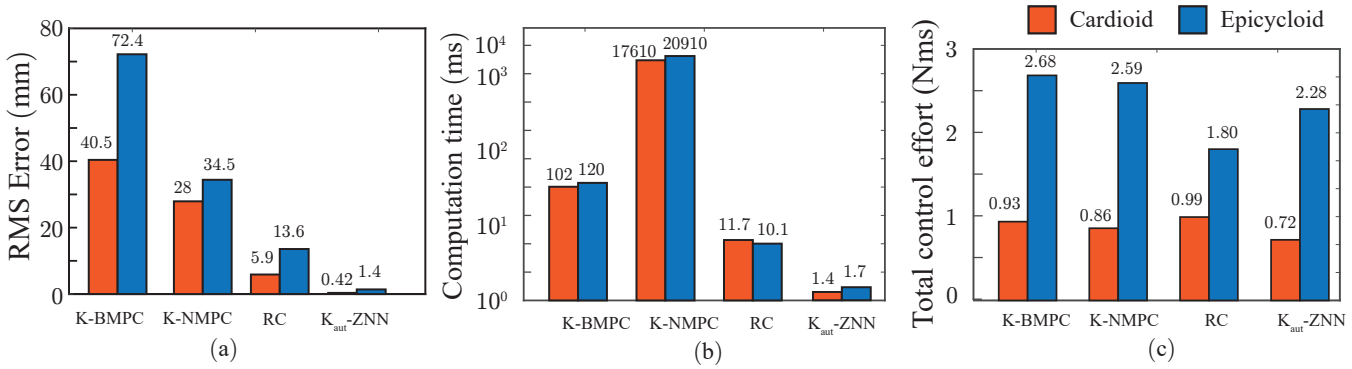


Fig. 5. Performance comparison of the proposed Koopman-ZNN algorithm (K_{aut}-ZNN), K-BMPC and K-NMPC [11], and reservoir computing-based (RC) approach [16] for tracking a planar 3R manipulator. a) RMS error. b) Computation time per iteration. c) Total control effort.

leads to larger tracking errors due to the linearization of the bilinear dynamics. Despite both BMPC and NMPC using optimization over the control horizon, observe that ZNN policy (10) results in a comparable control effort. Furthermore, our proposed algorithm, K_{aut}-ZNN significantly outperforms both K-BMPC and K-NMPC in terms of tracking error and computation time. The computational cost of solving optimization problems is increased due to the high dimensionality of model matrices in lifted states, making both BMPC and NMPC less suitable for experimental implementation.

The model-free reservoir computing-based algorithm [16] learns a nonlinear map to generate control signals $\boldsymbol{u}(t)$ based on partially observed states $\boldsymbol{y}(t) = [\boldsymbol{x}^\intercal,\ \dot{\boldsymbol{\theta}}^\intercal]^\intercal$ and reference states $\boldsymbol{y}_{ref}(t) = [\boldsymbol{x}_{ref}^\intercal,\ \dot{\boldsymbol{\theta}}_{ref}^\intercal]^\intercal$, where $\boldsymbol{x}$ and $\boldsymbol{x}_{ref}$ are the traced and reference Cartesian coordinates of the end-effector, respectively. To effectively train the reservoir computing machine, 125 trajectories with 8000 snaps each are required which is substantially higher than that needed for the proposed Koopman-based approach. Notably, the reservoir network only needs weight updates in its output layer, possibly resulting in lower training costs than the Koopman neural network. Fig. 5 shows that the proposed K_{aut}-ZNN framework outperforms the reservoir computing-based approach, which in turn outperforms K-BMPC and K-NMPC in tracking accuracy and computation time. Further, unlike the ZNN policy (10), the reservoir computing approach doesn't offer any guarantees regarding control input constraints during tracking. However, the reservoir computing method, like the ZNN policy (10), doesn't involve optimization of control inputs, resulting in a similar control effort. Thus, our proposed Koopman-ZNN-based approach provides superior tracking performance and lower computational demands for control input computation, making it more applicable to real-world applications.

Remark 2: Among the literature on Koopman learning, the proposed algorithm stands out as the only work that successfully learns the dynamics of a serial manipulator within a 3D Euclidean workspace. In contrast, the recent studies in [11] and [16] only showcase tracking control of manipulators with a 2D Cartesian workspace. The effectiveness of the proposed algorithm in learning the dynamics of a serial manipulator with a 3D workspace can be attributed to a careful choice of states,

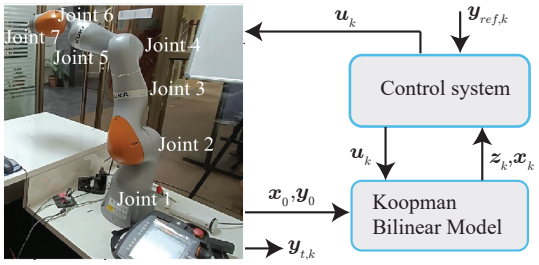
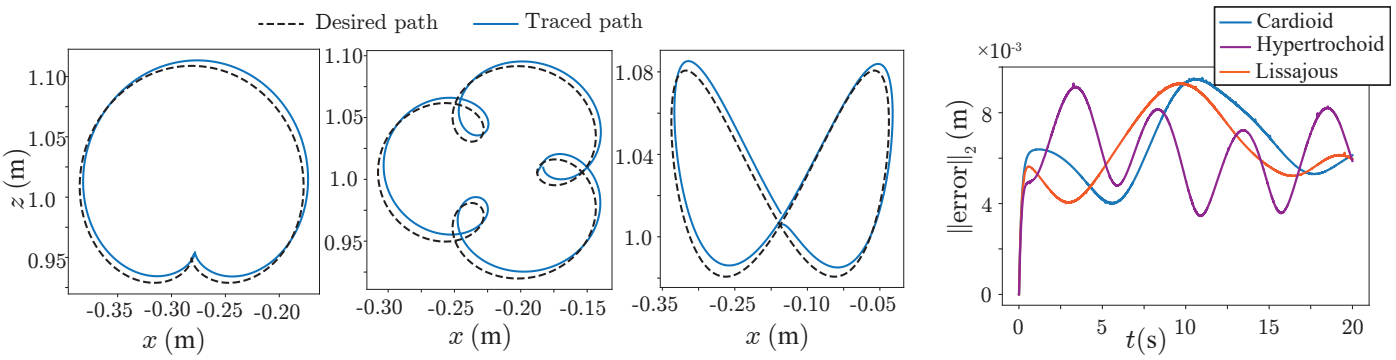


Fig. 6. Experimental setup and information flow diagram

which not only reduces the dimension of required lifted states but also yields a significantly more accurate bilinear model.

B. Experimental Results

In this section, the experimental results with the proposed Koopman-ZNN architecture are presented for trajectory tracking the KUKA LBR IIWA 14 R820 robot (Fig. 6). The hardware capabilities of the setup used include the Intel® Core™ i7-7700 processor with 32GB RAM and 1GB Intel® HD Graphics 630 graphics card. For simplicity, four joints-joints 1, 3, 5 and 7 were locked, reducing the 7 DOF robot to a 3R serial robot in a planar configuration. For learning the bilinear model, 200 trajectories are generated randomly within the workspace of the robot, each consisting of 1000 data points sampled at 500 Hz. The effectiveness of the proposed algorithms is evaluated by implementing the control architecture to trace Cardioid, Lissajous, and Hypertrochoid paths. As seen from Fig. 7, the robot can successfully trace the desired paths with a tracking error of the order of 10^{-3} m consistent with the simulation results. The resulting RMSE errors for the Cardioid, Hypertrochoid, and Lissajous shapes are 0.0067 m, 0.0064 m, and 0.0066 m respectively. The ZNN algorithm (10) is computationally inexpensive, allowing for its implementation at a frequency of 800 Hz, resulting in more robust tracking. On the other hand, the K-BMPC and K-NMPC algorithms require a significant amount of computational time, making them impractical for real-time applications. Therefore, the experimental results show that the proposed architecture is effective in achieving real-time trajectory tracking with



(a) (b)

Fig. 7. Experimental results of KUKA LBR IIWA 14 R820 robot for path tracking. a) Traced and desired paths (Cardioid, Hypertrochoid, and Lissajous, respectively). b) Norm of tracking error.

the bilinear model learned from data while considering input constraints.

VI. Conclusion

This study proposes a Koopman-ZNN architecture for the real-time control of redundant manipulators subject to input constraints. The architecture integrates an autoencoder-based architecture with a feed-forward neural network to learn the manipulator dynamics in the joint space and the associated forward map to Cartesian end-effector coordinates and allows for the discovery of Koopman bilinear models with fewer states compared to previous studies. The learned model is combined with a computationally inexpensive ZNN controller, that explicitly accounts for input constraints and facilitates real-time control. Simulation studies for trajectory tracking with serial manipulators are conducted to demonstrate the efficacy of the proposed architecture. In particular, comparisons with EDMD-based NMPC/BMPC and reservoir computing-based algorithms are carried out to show the superior performance of the proposed scheme for real-time control. Furthermore, experimental studies are carried out to verify the efficacy of the proposed schemes in achieving precise real-time control. Future work would involve extending the proposed scheme to unmanned aerial vehicles that would rely on an optimal zeroing dynamics controller for achieving improved tracking performance compared to leading alternative designs.

References

[1] M. Budišic, R. Mohr, and I. Mezic, "The koopman operator in systems and control: Concepts, methodologies, and applications," 2020.
[2] M. Korda and I. Mezić, "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control," *Automatica*, vol. 93, pp. 149–160, 2018.
[3] B. O. Koopman, "Hamiltonian systems and transformation in hilbert space," *Proceedings of the National Academy of Sciences*, vol. 17, no. 5, pp. 315–318, 1931.
[4] J. H. Tu, *Dynamic mode decomposition: Theory and applications*. PhD thesis, Princeton University, 2013.
[5] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Dynamic mode decomposition with control," *SIAM Journal on Applied Dynamical Systems*, vol. 15, no. 1, pp. 142–161, 2016.
[6] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A data–driven approximation of the koopman operator: Extending dynamic mode decomposition," *Journal of Nonlinear Science*, vol. 25, pp. 1307–1346, 2015.
[7] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the national academy of sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.
[8] Q. Li, F. Dietrich, E. M. Bollt, and I. 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*, vol. 27, no. 10, p. 103111, 2017.
[9] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of coordinates and governing equations," *Proceedings of the National Academy of Sciences*, vol. 116, no. 45, pp. 22445–22451, 2019.
[10] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature communications*, vol. 9, no. 1, p. 4950, 2018.
[11] D. Bruder, X. Fu, and R. Vasudevan, "Advantages of bilinear koopman realizations for the modeling and control of systems with unknown dynamics," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4369–4376, 2021.
[12] C. Folkestad, S. X. Wei, and J. W. Burdick, "Koopnet: Joint learning of koopman bilinear models and function dictionaries with application to quadrotor trajectory tracking," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 1344–1350, 2022.
[13] D. Bruder, X. Fu, R. B. Gillespie, C. D. Remy, and R. Vasudevan, "Data-driven control of soft robots using koopman operator theory," *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 948–961, 2020.
[14] D. Bruder, X. Fu, R. B. Gillespie, C. D. Remy, and R. Vasudevan, "Koopman-based control of a soft continuum manipulator under variable loading conditions," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6852–6859, 2021.
[15] A. Surana, "Koopman operator based observer synthesis for control-affine nonlinear systems," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 6492–6499, IEEE, 2016.
[16] Z.-M. Zhai, M. Moradi, L.-W. Kong, B. Glaz, M. Haile, and Y.-C. Lai, "Model-free tracking control of complex dynamical trajectories with machine learning," *Nature Communications*, vol. 14, no. 1, p. 5698, 2023.
[17] D. Goswami and D. A. Paley, "Global bilinearization and controllability of control-affine nonlinear systems: A koopman spectral approach," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 6107–6112, IEEE, 2017.
[18] Y. Zhang, D. Jiang, and J. Wang, "A recurrent neural network for solving sylvester equation with time-varying coefficients," *IEEE Transactions on Neural Networks*, vol. 13, no. 5, pp. 1053–1063, 2002.
[19] R. Singh and J. Keshavan, "A provably constrained neural control architecture with prescribed performance for fault-tolerant redundant manipulators," *IEEE Access*, vol. 10, pp. 97719–97732, 2022.
[20] C. K. Sah and J. Keshavan, "Prescribed performance control for solving time-varying underdetermined linear systems with bounds on states and their derivatives," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 3, pp. 3166–3177, 2023.