

Deep Koopman with Control: Spectral Analysis of Soft Robot Dynamics

Naoto Komeno^{1,*}, Brendan Michael^{1,*}, Katharina K uchler^{1,2}, Edgar Anarossi¹ and Takamitsu Matsubara¹

Abstract—Soft robots are challenging to model and control as inherent non-linearities (*e.g.*, elasticity and deformation), often requires complex explicit physics-based analytical modelling (*e.g.*, a priori geometric definitions). While machine learning can be used to learn non-linear control models in a data-driven approach, these models often lack an intuitive internal physical interpretation and representation, limiting dynamical analysis. To address this, this paper presents an approach using Koopman operator theory and deep neural networks to provide a global linear description of the non-linear control systems. Specifically, by globally linearising dynamics, the Koopman operator is analyzed using spectral decomposition to characterise important physics-based interpretations, such as functional growths and oscillations. Experiments in this paper demonstrate this approach for controlling non-linear soft robotics, and shows model outputs are interpretable in the context of spectral analysis.

I. INTRODUCTION

Linear control theory is well suited to developing interpretable control frameworks, through exploration of the spectral components, *i.e.*, eigenvectors and eigenvalues, of the associated dynamical system. Spectral analysis can help determine system stability [1], or provide additional insight for techniques such as filtering [2]. However, application to *non-linear* systems is ill-suited, due to the absence of a linear evolution of the dynamics, resulting in sub-optimal solutions and poor control applications.

In particular, soft robots with non-linear properties (*e.g.*, elasticity) suffer from difficulties in both modelling and control (*e.g.*, unpredictable behaviour due to the high degree of freedom). As such, predictive control often requires physics based modelling [3], including analytical descriptions of the geometries [4]. However, dynamical analysis of non-linear systems remains challenging, and is generally limited to systems with closed-form derivations (*e.g.*, double pendulums [5]).

As an alternative to analytical modelling, machine learning can be employed to learn predictive models of the non-linear dynamical system, solely through observations of the environment. While there exists a large body of work exploring machine learning methods for soft robots [6], a common limitation is the lack of interpretability and explainability of models [7]. Specifically, in the context of learning predictive models for control, black-box machine learning systems [8] may only locally linearise the dynamics [9], thereby not

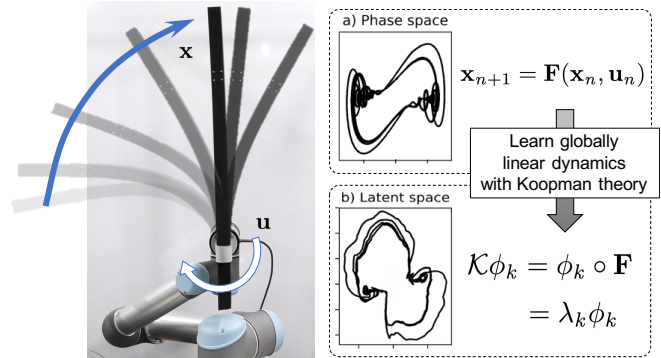


Fig. 1: Stabilization of a flexible polyurethane arm. (a) Complex non-linear robot dynamics are mapped to set of (b) linearised latent dynamics (polar co-ordinates), via Koopman operator theory, becoming amenable to spectral analysis.

capturing intrinsic important global physical properties of the system. This both limits dynamics analysis of the learnt model, and reduces confidence in model generalisability.

To address this, this paper proposes an approach to controlling and interpreting non-linear soft robotics by learning *globally linearized dynamics models* via *Koopman operator theory* [10], and applying this to model predictive controllers. Specifically, *Deep Koopman Networks* (DKN) [11] is used to learn parsimonious dynamics models with control, that expresses linear dynamics interpretations in the context of spectral analysis. Prior work applying Koopman theory to robotics [12]–[15] is limited to improving prediction and control performance, without considering the interpretability of the model. This paper presents the first evidence of DKN in soft robotics for non-linear control with dynamics analysis contextualised within the linear control domain.

Experiments apply the approach to a soft robot system (soft inverted pendulum) for both modeling the dynamics, and stabilisation control. Results show an improved control performance over standard deep learning for a soft flexible stabilisation task, and models display clear physical interpretations of the dynamic system.

II. BACKGROUND

Koopman operator theory [10] is a dynamical systems formulation, that provides a global description of non-linear dynamics, in terms of the *linear evolution* of a set of *observable functions*. Specifically, instead of attempting to model the non-linear dynamical state space (*e.g.*, positions and velocities), Koopman operator theory alternatively describes the dynamical system in terms of the linear evolution of a

¹Robot Learning Lab, Nara Institute of Science and Technology, Japan

²Institute of Knowledge Based Systems Group, RWTH Aachen University, Germany

*These authors contributed equally to this work

This work was supported by JSPS KAKENHI Grant Numbers JP19H01124 and JP22J11687.

set of state-dependent observable functions, forming a latent linear dynamical system [16].

A. Koopman operator theory

To present the role of Koopman operator theory in dynamical systems analysis, a discrete-time dynamical system is first formulated:

$$\mathbf{x}_{n+1} = \mathbf{F}(\mathbf{x}_n), \quad (1)$$

where n is the time-index, $\mathbf{F} : \mathcal{X} \mapsto \mathcal{X}$ is the flow map, and \mathcal{X} is a finite-dimensional metric state-space [16], often assumed to be a smooth manifold [17] (e.g., Euclidean, $\mathcal{X} \subseteq \mathbb{R}^M$).

Analysis and modelling of this non-linear \mathbf{F} can be challenging. Instead, a reasonable approach is to find coordinate transformations to map from the non-linear dynamics to a latent linear dynamical system. Koopman operator theory does this by describing the linear evolution of measurement functions of the non-linear state [17]:

$$\mathcal{K}g(\mathbf{x}_n) = g(\mathbf{F}(\mathbf{x}_n)) = g(\mathbf{x}_{n+1}), \quad (2)$$

where \mathcal{K}_t is the *Koopman operator*, an infinite-dimensional linear operator, acting on a measurement function g of the system (also known as an *observable*). This observable is a function of the state, i.e., $g : \mathcal{X} \mapsto \mathbb{C}$, and an infinite set of observable functions is defined on a Hilbert space \mathcal{H} [17]. This formulation is often referred to as *lifting* the state variables from the finite non-linear space, to an infinite-dimensional linear one. Given all observables, \mathcal{K} controls their evolution, i.e., $\mathcal{K} : \mathcal{H} \mapsto \mathcal{H}$. The Koopman operator is associated with the non-linear state transformation \mathbf{F} through the composition $\mathcal{K}g = g \circ \mathbf{F}, \forall g \in \mathcal{H}$ [16].

B. Koopman theory with control

Koopman operator theory is also applicable to systems with control inputs, by describing the dynamics of an extended state space of the product of \mathcal{X} and the space of all control sequences $\ell(\mathcal{U})$ [18]. By defining the operator on the extended space, the Koopman operator remains autonomous and is equivalent to the operator associated with unforced dynamics [19]. Specifically, considering a non-linear dynamical system with control $\mathbf{u} \in \mathcal{U}$:

$$\mathbf{x}_{n+1} = \mathbf{F}(\mathbf{x}_n, \mathbf{u}_n), \quad (3)$$

a common generalization is to define the associated Koopman operator as evolving an *uncontrolled* dynamic system defined by the product $\mathbf{F} : \mathcal{X} \otimes \mathcal{U} \mapsto \mathcal{X}$ [18].

As the evolution of \mathcal{X} only depends on \mathbf{u}_n , the control is considered as an additional state variable [16], instead of requiring all control sequences $\ell(\mathcal{U})$. As such, similar to Eq. (2), the Koopman operator is described as [16]:

$$\mathcal{K}g(\mathbf{x}_n, \mathbf{u}_n) = g(\mathbf{F}(\mathbf{x}_n, \mathbf{u}_n), \mathbf{u}_{n+1}). \quad (4)$$

In contrast to linear predictors [18], \mathbf{u} is also lifted through observables g alongside the state \mathbf{x} .

Applications of Koopman operator theory to domains with control are fast appearing in the literature, and generally

focus on strategies for choosing suitable observable functions [20] or optimal control [18], [21]. For a general review of Koopman based control frameworks, please see [22]. With specific regard to robotics control applications, prior work has investigated finding observables [23], LQR [12], [15], [24] or MPC [25]–[27] control, active learning [13], and shared human-robot control [28].

C. Spectral analysis and finite approximations of the Koopman operator

Analysis of this infinite-dimensional linear operator is challenging. However, finite spectral properties of the operator are of great importance, as they can outline global properties of the dynamics [29]. Specifically, \mathcal{K} can be spectrally decomposed into Koopman eigenfunctions $\phi_k(\cdot) \in \mathcal{H} \setminus \{0\}$ [16], with corresponding Koopman eigenvalue $\lambda_k \in \mathbb{C}$, which satisfies:

$$\mathcal{K}\phi_k = \phi_k \circ \mathbf{F} = \lambda_k \phi_k. \quad (5)$$

This Koopman eigenfunction is a linear intrinsic measurement coordinate on which measurements are evolved with a linear dynamical system [17]. As such, spectral analysis of this can provide physical intuition of the dynamical system under investigation.

While there exists a large body of work on finding analytical representations of eigenfunctions with knowledge of the dynamical system [16], [20], data-driven approximations are increasing prevalent. This is due to the complexity and uncertainty in finding eigenfunctions, and the modern increase in computational power for modelling, and availability of large datasets.

Generally, approximations are often made using the *dynamic mode decomposition* algorithm [17], [30], whereby spectral components of a linear transition matrix are estimated via matrices of state measurements. Specific implementations that incorporate Koopman operator theory to handle non-linear transitions include dictionary [31] or deep learning [32] for finding observables or eigenfunctions [11], or utilizing approaches such as time-delay embeddings [33].

III. APPROACH

A. Deep Koopman Network

A promising approach to learning the linearisation via eigenfunctions and the latent linear dynamics Eq. (5) for autonomous uncontrolled systems, is the *Deep Koopman Network* (DKN) approach [11]. In this, a deep autoencoder framework is used to find intrinsic latent coordinates $\mathbf{y} = \phi(\mathbf{x})$ approximating the Koopman eigenfunctions $\phi : \mathbb{R}^M \mapsto \mathbb{R}^P$, and associated linear dynamical system $\mathbf{y}_{n+1} = \mathbf{K}\mathbf{y}_n$. This is achieved by using an autoencoder to learn an encoder ϕ and decoder ϕ^{-1} , and inner layers to learn the linear dynamics \mathbf{K} . To learn parsimonious models, continuous spectra dynamics are captured by parameterising \mathbf{K} by an auxiliary layer, predicting eigenvalues as a function of \mathbf{y} .

An intuitive design constraint for DKN, is to learn latent coordinates \mathbf{y} which have complex radial symmetry

[11], as exponentials can be seen as eigenfunctions of a differential operator [34]. As such, for each eigenfunction ϕ_k , the associated eigenvalue is given as a complex pair, representing circular motion in the latent space. Specifically, $\lambda_k = \mu_k \pm i\omega_k$, where μ_k denotes the growth/decay, and ω_k the oscillation, and for each complex eigenvalue pair an associated linear operator \mathbf{K} with sampling time Δt is formed as:

$$\mathbf{K}(\mu_k, \omega_k) = \exp(\mu_k \Delta t) \begin{bmatrix} \cos(\omega_k \Delta t) & -\sin(\omega_k \Delta t) \\ \sin(\omega_k \Delta t) & \cos(\omega_k \Delta t) \end{bmatrix}. \quad (6)$$

Given this design constraint, ϕ , ϕ^{-1} , and \mathbf{K} are learnt using measurement data.

Here, we regard control input \mathbf{u} as one of the state variables for non-linear control systems so that DKN learns non-linear control dynamics. Previous studies using deep-learning-based learning of Koopman operators [14], [15] sacrifice interpretability by assuming bi-linear systems for Eq. (3) to achieve optimal control, but our approach realizes both interpretability and controllability. For the latent coordinates $\mathbf{y} = \phi([\mathbf{x}, \mathbf{u}])$, datasets are given in the DMD format of delayed snapshot matrices [17] $\mathbf{X} = [[\mathbf{x}_0, \mathbf{u}_0], \dots, [\mathbf{x}_{N-1}, \mathbf{u}_{N-1}]]$ and outputs $\mathbf{X}' = [[\mathbf{x}_1, \mathbf{u}_1], \dots, [\mathbf{x}_N, \mathbf{u}_N]]$. Eigenfunctions and linear dynamics are learnt by minimising a set of loss functions: (i) Reconstruction loss: $\|\mathbf{x}_n - \phi^{-1}(\phi([\mathbf{x}_n, \mathbf{u}_n]))\|$, (ii) Linear dynamics loss: $\|\phi([\mathbf{x}_{n+1}, \mathbf{u}_{n+1}]) - \mathbf{K}(\phi([\mathbf{x}_n, \mathbf{u}_n]))\|$, (iii) Next-step prediction loss: $\|\mathbf{x}_{n+1} - \phi^{-1}(\mathbf{K}(\phi([\mathbf{x}_n, \mathbf{u}_n])))\|$. In this paper, all components in \mathbf{x}_n are assumed to be measurable. For specific details regarding implementation and learning (including initialisations, regularisations and hyper-parameters), see [11].

B. Sampling-based model predictive control

Given this predictive model of the dynamics, a reasonable approach to controlling a system to a desired state is via a model-based controller. In this study, even in linear latent space, control inputs are given as non-linear system state variables Eq. (4). As such, standard linear optimal control (*e.g.*, LQR) applied in previous studies [12], [14], [15], [24] is inapplicable. Instead, sampling-based nonlinear model predictive control (MPC) is employed to overcome the control non-linearities. This is applied as a model-based control method in nonlinear observation space.

Specifically, MPC uses the approximation of Eq. (3) from learnt DKN components §III-A, in combination with a given control input at each time step, to predict the future state. To determine the optimal control to reach the desired target state, a minimization problem is formulated with a cost c computed as the difference between the predicted state and the target state \mathbf{x}_{target} . Given this cost, sampling-based optimization using the cross-entropy method (CEM) [35] is performed. The optimization problem for MPC is formulated:

$$u_t = \arg \min_{u_t} \sum_{i=t+1, \dots, t_p} c_i(\mathbf{x}_i, \mathbf{x}_{target}), \quad (7)$$

where t_p is prediction horizon to determine the action that will bring the prediction result closest to the target state.

IV. SIMULATION

To evaluate the proposed approach for learning non-linear dynamics models that explain intuitive physical aspects, experiments are presented to explore the application of DKN with control. This evaluation is comprised of three constituent parts: (i) learning dynamics of a non-linear system under control, (ii) analysing the learnt Koopman outputs to elucidate an intuitive understanding of the system, and (iii) applying predictive control.

A. Simulation experiments

Initially, an experiment is presented to learn and analyse the dynamics of a non-linear rigid pendulum, similar to that proposed in [11]. While a relatively mundane system, the non-linear pendulum is challenging to model within a parsimonious framework, due to it exhibiting a continuous eigenvalue spectrum (*e.g.*, differing oscillation frequency dependant on dynamical state) which often requires a harmonic expansion estimation.

The dynamics of the pendulum under control [36] is entirely described by the three dimensional state $\mathbf{x} = [q, \dot{q}, u]^T$, where q is the joint angle, \dot{q} the joint velocity, and u the applied torque. The equations of motion for the rigid pendulum is given as:

$$\dot{\mathbf{x}} = [\dot{q}, \ddot{q}]^T, \quad (8)$$

where angular acceleration \ddot{q} is:

$$\ddot{q} = \frac{g}{l} \sin(q) + \frac{(u - v\dot{q})}{(ml)^2}, \quad (9)$$

with gravity g , rod length l , mass m and viscous friction v . The aim of this experiment is to learn a predictive model of Eq. (8) and Eq. (9), through observations of q , \dot{q} , and u .

B. Generating data

Trajectories of motion are generated as samples from which to learn dynamics models. For this, Eq. (9) is solved for \mathcal{T}_f time-steps with a forth-order Runge-Kutta method. From these \mathcal{T}_f timesteps, T are taken to form a trajectory matrix $\mathbf{X}_n \in \mathbb{R}^{3 \times T} = [\mathbf{x}_0^T, \dots, \mathbf{x}_T^T]$. Trajectory matrices are reshaped into a delay-embedding vector [37], resulting in a sample $\mathbf{x}_n \in \mathbb{R}^{3T}$. To generate datasets, this process is repeated with random initialisations of q, \dot{q} and u , to form the training dataset $\mathbf{X}^A \in \mathbb{R}^{3T \times A}$, validation dataset $\mathbf{X}^B \in \mathbb{R}^{3T \times B}$ and evaluation dataset $\mathbf{X}^C \in \mathbb{R}^{3T \times C}$. In the simulation experiment $v = 0$, $g = -1$ and $m, l = 1$.

Using the Deep Koopman framework §III-A, a neural network architecture is learnt. For specific details on generating data, and architecture implementation details, please see Appendix.

C. Rigid pendulum (no control)

For an initial experiment, the dynamics of a rigid pendulum without any control input is learnt. This is similar to

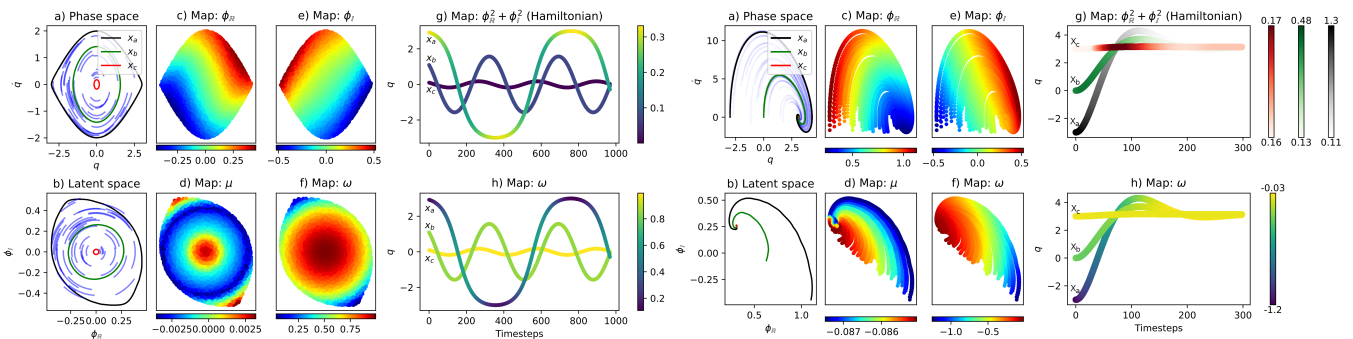


Fig. 2: DKN: Rigid pendulum without control (left), and with PD control (right). a) Phase space with exemplar trajectories with initial states (No control: $x_a = [\pi - 0.1, 0]$, $x_b = [\frac{\pi}{2}, 0]$, and $x_c = [\frac{\pi}{18}, 0]$, PD Control: $x_a = [-\pi + 0.1, 0]$, $x_b = [0, 0]$, and $x_c = [\pi - 0.1, 0]$), b) Latent space, c,e) Single eigenfunction with real and imaginary axes mapped independently, d) Eigenfunction growth μ , f) Eigenvalue frequency ω , g) Hamiltonian energy and, h) Eigenvalue frequency over time.

the experiments described in [11], and aims to elucidate the interpretation process of the Koopman outputs.

The results for learning the dynamics with DKN are shown in Fig. 2 (left). In this, it is seen that as in [11], trajectories in the input phase space Fig. 2 (left) (a) are mapped via a single complex conjugate pair of learnt eigenfunctions, to a latent-space that is linear in polar coordinates Fig. 2 (left) (b). Specifically, this demonstrates that the system is globally linearised within the Koopman framework. Additionally, by using the auxiliary network to parameterise the learnt dynamics by a continuous spectra of eigenvalues, a continuous range of oscillatory frequencies is also captured by the model, thereby capturing the continuous eigenvalue spectrum. This is seen in the latent space (Fig. 2 (left) (f)), where the continuous spectra captures an intuitive physical characteristic of the pendulum, that being the frequency of oscillation is dependent on the position in the phase space. Specifically, this shows the eigenvalue frequency decreases the further from the centre, *i.e.*, the period of the swing increases the further from the stable equilibrium position. In the context of the pendulum, this expressed as a high frequency oscillation at small angles, and low frequency oscillation at higher angles (Fig. 2 (left) (h)). Additionally, characteristics such as the Hamiltonian energy [11] can be expressed function of the phase space (Fig. 2 (left) (g)). In the context of the pendulum this captures the kinetic and potential energy of the pendulum at top of the swing. As such, the model captures interesting inherent physical properties of the underlying dynamical system.

D. Rigid pendulum (PD control)

A subsequent experiment evaluates the performance of dynamics learning with DKN, for systems with control input. In this experiment, trajectories of motion are generated according to §IV-B, with control given by a standard PD controller:

$$u = -K_p(q_{target} - q) + K_d\dot{q}, \quad (10)$$

where $q_{target} = \pi$ is the target, and $K_p = 10$, $K_d = 3$ are coefficients for the proportional and derivative terms.

In this experiment, trajectories of motion are generated for $T_f = 300$ timesteps, to reach the target. Trajectories are then split into segments of length $T = 50$.

Results for this experiment are seen in Fig. 2 (right). In this it is seen similar to §IV-C, a single complex eigenfunction characterises the behaviour of the dynamical system. Specifically, the frequency increases around the unstable equilibrium position Fig. 2 (right) (f), resulting in high frequency oscillations when the pendulum stabilises Fig. 2 (right) (h) due to the PD gain terms. As such, it is clear that the learnt model both captures dynamics under control, and the induced PD controller dynamics.

V. EXPERIMENT

A. Soft inverted pendulum

Experiments in §IV demonstrate that DKN can be applied to non-linear dynamical systems under control, to obtain intuitive outputs that can be used to help explain characteristic behaviours.

Given this, a further evaluation is performed exploring the application of this method to a more challenging scenario, stabilisation of a soft inverted pendulum in a real-environment. In the context of dynamical systems, modelling this introduces the additional challenges of both system complexity (*e.g.*, modelling non-linear elasticity and hysteresis), as well as accounting for real-world environmental noise and variations (*e.g.*, air-flow, vibrations).

An overview of soft inverted pendulum elasticity is shown in Fig. 1, using an environment as shown in Fig. 3. In this, a soft inverted pendulum made from polyurethane foam (JIS ASKER C < 1, 30 × 100 × 700 mm) is rotated at the base by a robot arm (Universal Robots Inc.: UR5). The softness induces non-linear dynamics in the system, and can be represented as a type of non-linear spring with dynamics given by the Duffing equation [38], [39]. This system has a dual-well potential, with one stable equilibrium point on either side (*e.g.*, Fig. 3 demonstrates left-sided rest), and requires nonlinear control to reach a balanced unstable equilibrium point.

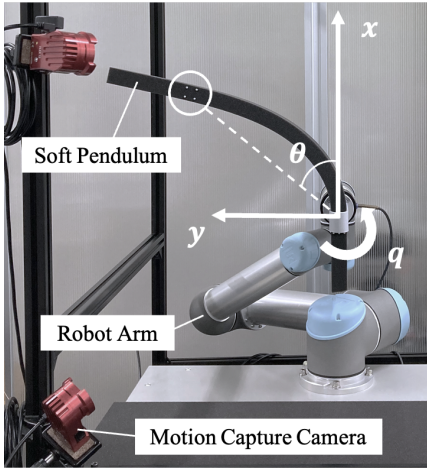


Fig. 3: The environment of the soft inverted pendulum

In these experiments, it is assumed that the dynamics can be described by the state variable $\mathbf{x} = [\theta, \dot{\theta}, q]^T$, where θ is the angle of pendulum, $\dot{\theta}$ the angular velocity of pendulum and q is the joint angle of robot. The control input $u = \dot{q}$ is the joint velocity of robot. Due to the hysteresis of elasticity, it is assumed that this experimental setup requires memory. As such, models use $T = 50$ time-steps as historical input, to learn the embedding and its associated linear dynamical system.

B. Experimental setting

The experimental setup is shown in Fig. 3, where the polyurethane foam prism is attached to the robot end-effector, with the center of the end-effector defined as the origin and the distance from the end of the pendulum given as 580mm. To measure the system state, recurrent reflection markers are attached to the end of the pendulum, and its position $p(x_t, y_t)$ is measured with a motion capture camera (OptiTrack: Flex13, sampling frequency 120 Hz) to obtain the pendulum angle θ . The angular velocity $\dot{\theta}$ is calculated by approximate differentiation.

$$\theta_t = \arctan \frac{y_t}{x_t} \quad (11)$$

$$\dot{\theta}_t = \frac{\theta_t - \theta_{t-1}}{dt} \quad (12)$$

The state of the robot $[q, \dot{q}]$ is obtained and controlled directly through the robot's internal controller.

C. Data collection

To generate trajectories of motion for dynamics learning, time series data of the soft inverted pendulum system is collected using a PD control:

$$u = -K_p(\theta_{target} - \theta) + K_d\dot{\theta}. \quad (13)$$

This controller acts as an exciter, to capture the dynamics of the soft pendulum, which is a non-autonomous system. Since PD control is a linear controller, the gain K_P, K_D and target position θ_{target} are varied to explore the nonlinear state space, as detailed in Table I. Specifically, these values

TABLE I: Details of PD controller for data collection

Controller	PD1	PD2	PD3	PD4
K_p	0.3	0.3	0.1	0.1
K_d	0.1	0.2	0.2	0.3
θ_{target}	$0, \pm 0.8$	$0, \pm 0.1$	$0, \pm 0.8$	$0, \pm 0.8$

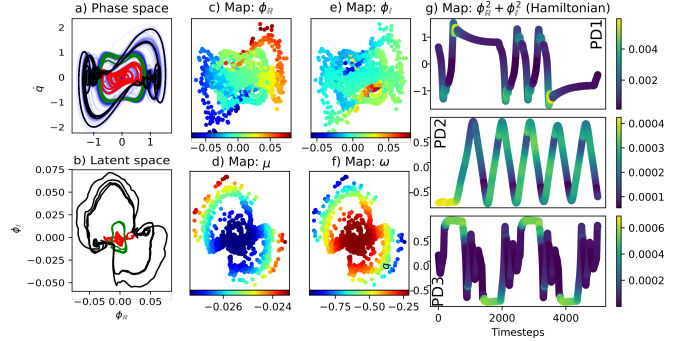


Fig. 4: DKN one complex eigenfunction of the soft inverted pendulum: a) Phase space with different behavioural trajectories (for gains in Table I), b) Latent space, c,e) Single eigenfunction with real and imaginary axes mapped independently, d) Eigenfunction growth μ , f) Eigenvalue frequency ω , g) Hamiltonian energy.

are chosen to generate a range of trajectories that vary between conservative trajectories that are stationary at the target point over time and unstable trajectories that oscillate by overshooting to the stable equilibrium point.

Data is collected for 30 minutes for each gain and target setting, with a control frequency of 20 Hz.

D. Dynamics analysis

To learn and analyse the dynamics of the soft pendulum, a DKN network is learnt following the procedure described in §III-A, with training details given in Appendix.

1) *One Complex Eigenfunction*: As an initial experiment, the dynamics are learnt using DKN with a single complex eigenfunction (as in §IV). The results for this are seen in Fig. 4. In this, it is seen that the input phase space, Fig. 4(a), is comprised of different behavioural trajectories, due to the variations of the controller gains inducing differing oscillation periods.

As a first observation, it is seen that this variation in behaviour is similarly observed in the latent complex eigenfunction space (Fig. 4(b)). Specifically, input trajectories from PD1 (black), which have a longer travelling distance and contains oscillation at the dual wells, are successful *unwound* and mapped to the polar latent space. This highlights that complex behaviours in the phase space can be mapped via to the eigenfunctions to a linear space. Similarly, trajectories from the other gains are also mapped, with decreasing radius from the centre. As such, the different behaviours have been discretised into disparate regions of the eigenfunction latent space.

To explore this in the context of gaining intuitive understanding of the system, Fig. 4(f), shows the frequency of

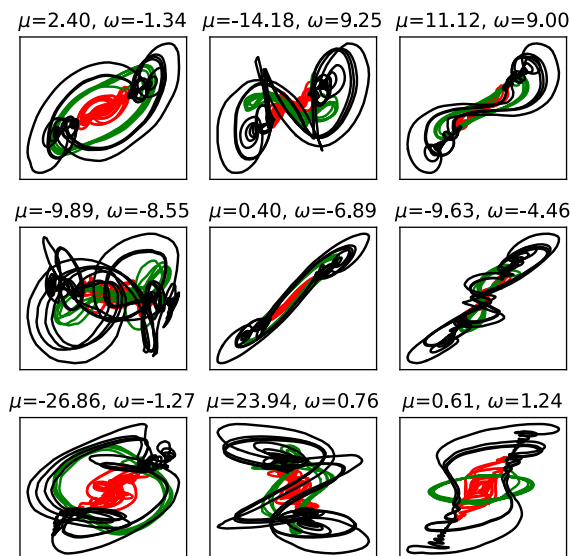


Fig. 5: DKN nine complex eigenfunctions of the soft inverted pendulum: Latent coordinates for each complex eigenfunction, for exemplar trajectories given by gains in Table I

oscillation in the latent space. In this, the frequency varies as a function of the phase space, oscillating at high frequencies as the pendulum converges to the stabilisation point (due to the gain parameters). Additionally, as seen in Fig. 4(h), the energy of the system can be extracted from the model via the magnitude of the eigenfunctions. In this, it is seen that for the stabilisation trajectory (PD controller with strongest gains, red) the highest energy corresponds to the initial change in velocity.

On the other hand, μ , which indicates the divergence and convergence of the oscillations in Fig. 4(d), takes a very small negative value, just like the rigid pendulum in Fig. 2. This suggests that DKN with a single eigenfunction learns only the steady oscillations caused by the PD controller, and the stabilization dynamics of the soft pendulum up to the equilibrium point is not captured.

2) *Nine Complex Eigenfunctions*: In order to learn the full dynamics, up to and including equilibrium stabilization, the experiment in §V-D.1 is repeated with a greater number of complex eigenfunctions. Specifically, nine eigenfunctions are chosen to capture a more expansive range of dynamic components.

The results for this increased number of eigenfunctions is seen in Fig. 5. In this, it is seen that unique configurations are captured for each pair. However, in contrast to the experiments in Fig. 5, both μ and ω are learnt as discrete, non-continuous values for each eigenfunction. In the context of dynamical systems analysis, this means that the learnt dynamical system can be expressed as a combination of nine fixed oscillatory behaviours, possibly corresponding to mechanical properties of the system.

E. Model predictive control

A further investigation is performed to assess the applicability of the learnt DKN models to control problems. In this, the performance is tested through MPC with CEM Eq. (7). CEM is performed with the cost function:

$$c(\mathbf{x}, \mathbf{x}_{target}) = w_1 \theta^2 + w_2 \dot{\theta}^2, \quad (14)$$

where w_1 and w_2 are cost weights for pendulum angle and angular velocity, and the target state \mathbf{x}_{target} is set as $[\theta_{target}, \dot{\theta}_{target}] = [0, 0]$.

Trajectories of data are obtained for 30 seconds, for 5 samples of each PD model. The control frequency is fixed at 20 Hz, and the predictive horizon t_p and the cost weights $[w_1, w_2]$ are hand-tuned. They are evaluated for their transition and summation of the error (the cost when the weights are all one) as the achievement of control. For video results, please see our YouTube channel ¹.

In comparison to the DKN method, a fully connected network (FCN) is simultaneously learnt using the same dataset. The FCN is formed of an encoder, decoder and latent inner hidden layers of the same network size as DKN. As such, this is a reasonable equivalent network, that lacks only the inherent linearisation of DKN.

Results for reaching the target with MPC are given in Fig. 6 (top), for the case of $T = 1$ (*i.e.*, no history). In this, it is seen that for DKN, the error between the tip position and target oscillates immediately from the start and is unable to stabilize for most of the time. In comparison, FCN maintains a reduced error during initial movement, but likewise destabilises and exhibits oscillatory behaviour. This is not unexpected, as due to the hysteresis of the system, historical measurements are imperative for uniquely defining the state, and failure to include this input information results in overall poor controllers. The average error and variation of each modelling approach is shown in Fig. 7.

In comparison, by including $T = 50$ historical measurements as part of the learning and prediction process, Fig. 6 (bottom) shows that DKN displays a small error during runtime, immediately moving the soft pendulum tip close to the target, and maintaining stabilisation with only small error. However, including these historical measurements in FCN, does not significantly improve performance, and displays a worse stabilization performance than the $T = 1$ case. One possible reason for this, is that the complexity of modelling the relationship between such a large history, and the next step, is too high-dimensional for such a small network. However, as DKN learns both the embedding and the associated linear dynamical system, it can exploit the implicit linear bias to learn small, parsimonious models that capture the actual physics-based dynamics.

VI. DISCUSSION

This paper presents an approach to controlling and interpreting non-linear soft robotics, by learning globally linearized dynamics models. While previous studies have

¹<https://youtu.be/BeXtHMoSSwM>

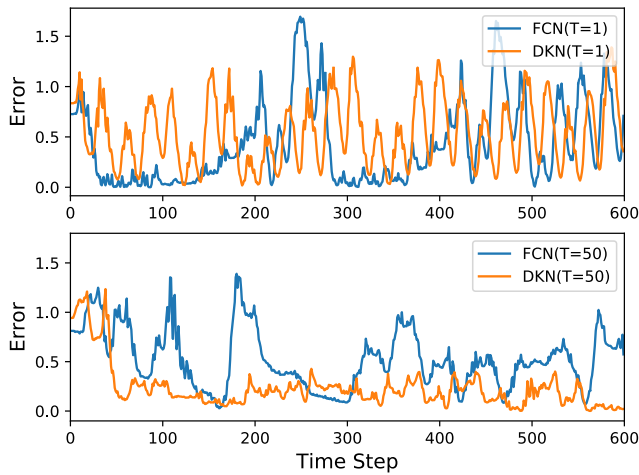


Fig. 6: Error in soft pendulum tip position during stabilization.

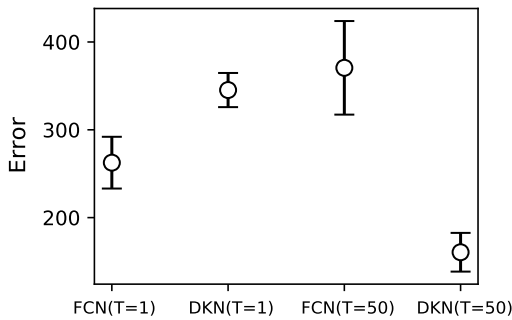


Fig. 7: Mean sum of errors between soft pendulum position and target, with standard deviation of each method over timesteps.

applied Koopman operator theory to control soft robotics, the approach in this paper explicitly analyses learnt spectral outputs to gain intuitive understanding of the systems under investigation. In combination with MPC, models are applied to control applications, resulting in models that outperform equivalent scale deep networks, due to the inherent parsimonious latent representation of dynamics. As future work, the approach will be applied to more complex soft robotics tasks, including incorporating vision for object handling.

REFERENCES

- [1] W. Michiels and S. Niculescu, *Stability, control, and computation for time-delay systems: an eigenvalue-based approach*. SIAM, 2014.
- [2] F. M. Ham and R. G. Brown, “Observability, eigenvalues, and kalman filtering,” *IEEE Transactions on Aerospace and Electronic Systems*, no. 2, pp. 269–273, 1983.
- [3] S. Grazioso, G. Di Gironimo, and B. Siciliano, “A geometrically exact model for soft continuum robots: The finite element deformation space formulation,” *Soft robotics*, vol. 6, no. 6, pp. 790–811, 2019.
- [4] S. H. Sadati, S. E. Naghibi, A. Shiva, B. Michael, L. Renson, M. Howard, C. D. Rucker, K. Althoefer, T. Nanayakkara, S. Zschaler *et al.*, “TMTDyn: A matlab package for modeling and control of hybrid rigid–continuum robots based on discretized lumped systems and reduced-order models,” *The International Journal of Robotics Research*, vol. 40, no. 1, pp. 296–347, 2021.
- [5] P. Yu and Q. Bi, “Analysis of non-linear dynamics and bifurcations of a double pendulum,” *Journal of Sound and Vibration*, vol. 217, no. 4, pp. 691–736, 1998.
- [6] D. Kim, S. Kim, T. Kim, B. B. Kang, M. Lee, W. Park, S. Ku, D. Kim, J. Kwon, H. Lee *et al.*, “Review of machine learning methods in soft robotics,” *PLoS One*, vol. 16, no. 2, p. e0246102, 2021.
- [7] A. Heuillet, F. Couthouis, and N. Díaz-Rodríguez, “Explainability in deep reinforcement learning,” *Knowledge-Based Systems*, vol. 214, p. 106685, 2021.
- [8] C. Rudin, “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead,” *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.
- [9] J. Huang and F. L. Lewis, “Neural-network predictive control for nonlinear dynamic systems with time-delay,” *IEEE Transactions on Neural Networks*, vol. 14, no. 2, pp. 377–389, 2003.
- [10] B. O. Koopman, “Hamiltonian systems and transformation in hilbert space,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 17, no. 5, p. 315, 1931.
- [11] 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.
- [12] D. A. Haggerty, M. J. Banks, P. C. Curtis, I. Mezić, and E. W. Hawkes, “Modeling, reduction, and control of a helically actuated inertial soft robotic arm via the Koopman operator,” *arXiv preprint arXiv:2011.07939*, 2020.
- [13] I. Abraham and T. D. Murphey, “Active learning of dynamics for data-driven control using Koopman operators,” *IEEE Transactions on Robotics*, vol. 35, no. 5, pp. 1071–1083, 2019.
- [14] Y. Han, W. Hao, and U. Vaidya, “Deep learning of koopman representation for control,” Oct. 2020.
- [15] H. Shi and M. Q. Meng, “Deep Koopman operator with control for nonlinear systems,” *arXiv preprint arXiv:2202.08004*, 2022.
- [16] A. Mauroy, I. Mezić, and Y. Suzuki, *Koopman Operator in Systems and Control*. Springer, 2020.
- [17] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic mode decomposition: data-driven modeling of complex systems*. SIAM, 2016.
- [18] M. Korda and I. Mezić, “Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control,” *Automatica*, vol. 93, pp. 149–160, 2018.
- [19] S. L. Brunton, M. Budišić, E. Kaiser, and J. N. Kutz, “Modern Koopman theory for dynamical systems,” *arXiv preprint arXiv:2102.12086*, 2021.
- [20] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz, “Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control,” *PLOS ONE*, vol. 11, no. 2, p. e0150171, 2016.
- [21] E. Kaiser, J. N. Kutz, and S. L. Brunton, “Data-driven discovery of Koopman eigenfunctions for control,” *arXiv preprint arXiv:1707.01146*, 2017.
- [22] P. Bevanda, S. Sosnowski, and S. Hirche, “Koopman operator dynamical models: Learning, analysis and control,” *arXiv preprint arXiv:2102.02522*, 2021.
- [23] L. Shi and K. Karydis, “Acd-edmd: Analytical construction for dictionaries of lifting functions in Koopman operator-based nonlinear robotic systems,” in *IEEE Robotics and Automation Letters*. IEEE, 2021.
- [24] G. Mamakoukas, M. Castano, X. Tan, and T. Murphey, “Local Koopman operators for data-driven control of robotic systems,” in *Robotics: science and systems*, 2019.
- [25] C. Folkestad, D. Pastor, and J. W. Burdick, “Episodic Koopman learning of nonlinear robot dynamics with application to fast multirotor landing,” in *2020 IEEE International Conference on Robotics and Automation*. IEEE, 2020, pp. 9216–9222.
- [26] 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.
- [27] F. E. Sotiropoulos and H. H. Asada, “Dynamic modeling of bucket-soil interactions using Koopman-dfl lifting linearization for model predictive contouring control of autonomous excavators,” *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 151–158, 2021.
- [28] A. Broad, I. Abraham, T. Murphey, and B. Argall, “Data-driven Koopman operators for model-based shared control of human–machine systems,” *The International Journal of Robotics Research*, vol. 39, no. 9, pp. 1178–1195, 2020.

- [29] I. Mezić, “Spectral properties of dynamical systems, model reduction and decompositions,” *Nonlinear Dynamics*, vol. 41, no. 1, pp. 309–325, 2005.
- [30] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. Henningson, “Spectral analysis of nonlinear flows,” *Journal of fluid mechanics*, vol. 641, no. 1, pp. 115–127, 2009.
- [31] 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.
- [32] N. Takeishi, Y. Kawahara, and T. Yairi, “Learning Koopman invariant subspaces for dynamic mode decomposition,” in *Advances in Neural Information Processing Systems*, 2017, pp. 1130–1140.
- [33] M. Kamb, E. Kaiser, S. L. Brunton, and J. N. Kutz, “Time-delay observables for Koopman: Theory and applications,” *SIAM Journal on Applied Dynamical Systems*, vol. 19, no. 2, pp. 886–917, 2020.
- [34] T. Peter and G. Plonka, “A generalized prony method for reconstruction of sparse sums of eigenfunctions of linear operators,” *Inverse Problems*, vol. 29, no. 2, p. 025001, 2013.
- [35] P. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, “A tutorial on the cross-entropy method,” *Annals of Operations Research*, vol. 134, no. 1, pp. 19–67, Feb. 2005.
- [36] K. Doya, “Reinforcement learning in continuous time and space,” *Neural computation*, vol. 12, no. 1, pp. 219–245, 2000.
- [37] S. Le Clainche and J. M. Vega, “Higher order dynamic mode decomposition,” *SIAM Journal on Applied Dynamical Systems*, vol. 16, no. 2, pp. 882–925, 2017.
- [38] G. Litak, M. Cocco, M. I. Friswell, S. F. Ali, S. Adhikari, A. W. Lees, and O. Bilgen, “Nonlinear oscillations of an elastic inverted pendulum,” in *2012 IEEE 4th International Conference on Nonlinear Science and Complexity*, Aug. 2012, pp. 113–116.
- [39] C. D. Santina, “The soft inverted pendulum with affine curvature,” in *2020 59th IEEE Conference on Decision and Control*, Dec. 2020, pp. 4135–4142.

APPENDIX

Data collection

•	Pendulum (no control)	Pendulum (PD control)	Soft pendulum
T	50	50	50
dt	0.02	0.01	0.05
q_0 range	[-3.1,3.1]	[-3.1,3.1]	[-1.5,1.5]
\dot{q}_0 range	[-2,2]	[-2,2]	[-2,2]
# Training	15000	15950	100763
# Validation	1000	1450	28791
# Evaluation	3000	2900	14394