Robust physical parameter identification through global linearisation of system dynamics

Citation for published version:

Link:
Link to publication record in Edinburgh Research Explorer

Published In:
Proc. NeurIPS Workshop on Safe and Robust Control of Uncertain Systems, 2021

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Robust Physical Parameter Identification through Global Linearisation of System Dynamics

Yordan Hristov
School of Informatics
University of Edinburgh
yordan.hristov@ed.ac.uk

Subramanian Ramamoorthy
School of Informatics
University of Edinburgh
s.ramamoorthy@ed.ac.uk

Abstract

Using neural networks to learn dynamical models from data is a powerful technique but can also suffer from problems like brittleness, overfitting and lack of safety guarantees. These problems are particularly acute when a distributional shift is observed in the dynamics of the same underlying system, caused by different values of its physical parameters. Casting the learned models in the framework of linear systems enhances our abilities to analyse and control them. However, it does not stop them from failing when having to extrapolate outside of their training distribution. By globally linearising the system’s dynamics, using ideas from Deep Koopman Theory, and combining them with off-the-shelf estimation techniques like Kalman filtering, we demonstrate a way of knowing when and what the model does not know and respectively how much we can trust its predictions. We showcase our ideas and results in the context of different rod lengths of the classical pendulum control environment.

1 Introduction

Analytical models of dynamical systems give us the capability to investigate them offline—e.g. the concept of Digital twins (Jones et al., 2020)—and optimally control them online (Garcia et al., 1989). However, formulating a sound model is often a nontrivial task, especially if the system of interest is complex and non-linear. Thus, in recent years, a more popular approach has been to employ data-driven, high-capacity neural models which, in a black-box fashion, can recover the underlying dynamics from solely observing the system. However, such approaches often suffer from a number of problems, typical for the deep learning field—hard to provide safety guarantees and lacking robustness and good generalisation capabilities. Therefore, in our work, we choose to employ deep neural models in assisting us to approximate nonlinear dynamics under the linear-system framework by learning to perform coordinate re-projection of the original state space to an alternative state space that can be linearly evolved through time. The linear nature of the approximated dynamics is particularly important because they allow us to draw on analysis approaches that are otherwise not available to systems with nonlinear dynamics (Dullerud & Paganini, 2013).

However, the fact that such approximated linear dynamics are defined on top of features that are learned from data means that the resultant models are prone to problems related to data shift. In the representation learning literature this is commonly known as the Domain Shift problem (Amodei et al., 2016) or Open World recognition for classification and regression (Lakkaraju et al., 2017; Bendale & Boul, 2016). In the context of modelling dynamics, such distributional shifts might be caused by observing different parameter values for the same system during and after training—e.g. different mass, different damping coefficients, etc. Being able to interpolate between already-observed data points and extrapolate beyond them is crucial since rarely every possible scenario can be encompassed in a training set of fixed size.

Figure 1: Network architecture for learning a feature space $\mathbf{y}$, from state-transition observations $\mathbf{x}$, on top of which the Koopman matrix $\mathbf{K}$ can be fit. The values of any physical parameters of interest $\mathbf{p}$ and actions $\mathbf{u}_t$ are attached to the state $\mathbf{y}_t$ before progressing it to $\mathbf{y}_{t+1}$ through $\mathbf{K}$. Therefore $\mathbf{y}$ is meant to be a parameter-independent (e.g. pendulum length) linearised state.

Performing online system identification is one way to cope with these issues. For example, taking gradient descent steps in the neural weights space of the learnt dynamical model at test time, to minimise the error between what is predicted and what is observed (Battaglia et al., 2016; Chang et al., 2016). However, such optimisation is known to be sample inefficient, especially for over-parametrised neural models which are prone to the problem of catastrophic forgetting (French, 1999; Kumaran et al., 2016). Building surrogate locally-linear models on the fly, only from observing state-action transition trajectories is another cheaper option (Williams et al., 2015; Arbabi et al., 2018; Kutz et al., 2016). Often, this paradigm is employed to identify the dynamics of the same underlying system but with varying physical parameter values (Li et al., 2019). However, any novel parameter values of interest are implicitly represented by the fit dynamics and there is no way of knowing how certain the model is in its outputs, especially when doing the estimations from a limited set of observations.

In our work, we argue that structured models, equipped with specific inductive biases (e.g. globally linear dynamics through Koopman theory) allow us to utilise sample-efficient uncertainty quantification techniques and be inherently more robust and reliable as a result. Globally-linear dynamics means that the state space observations of a non-linear system can be projected to an alternative state space where a single transition matrix can be found to characterise the whole system. Specifically, we choose to employ the ideas of Koopman Theory (Koopman, 1931) to learn globally linear dynamics that account for varying physical parameters—namely, once fit, the inferred dynamical model is kept fixed. What changes are part of the state, on top of which the dynamics evolve, that represents the different possible parameter values. Because we approximate the system as a linear one, at test time we use an off-the-shelf Kalman filter to estimate the underlying physical parameters from system observations. This allows us to effectively do parameter estimation in a single-shot fashion—from a single trajectory observation. Moreover, we have access to the corrections terms the filter does at each step. Therefore, instead of having a single point estimate for the underlying physical parameters, we also have a notion of how much we can trust these estimates and the respective system simulations based on them, hence being more robust to out-of-distribution samples.

2 Background

The main ideas behind Koopman theory date back to the 1930s (Koopman, 1931; Koopman & Neumann, 1932) and postulate that the state of any nonlinear dynamical system can be projected to an infinite-dimension Hilbert space, where there exists a Koopman Operator which linearly evolves the system through time. Namely, for a discrete-time dynamical system $\mathbf{x}_{t+1} = \mathbf{F}(\mathbf{x}_t)$, $\mathbf{x}_t \in \mathcal{X} \subset \mathbb{R}^n$, the corresponding Koopman operator $\mathcal{K} : \mathcal{F} \rightarrow \mathcal{F}$ is defined as $\mathcal{K} \phi \triangleq \phi \circ \mathbf{F}$, where $\mathcal{F}$ is the set of
all observable functions that form the infinite-dimensional Hilbert space. For any observable function \( \phi : \mathcal{X} \rightarrow \mathbb{R} \), \( \phi \in \mathcal{F} \), the following is true: \( (K\phi)(x_t) = \phi(F(x_t)) = \phi(x_{t+1}) \).

The infinite-dimensional nature of \( \mathcal{K} \) is often circumvented by approximating it using Dynamic Mode Decomposition (DMD) methods (Schmid, 2010; Kutz et al., 2016; Arbabi & Mezic, 2017; Arbabi et al., 2018) which aim to identify spatio-temporal coherent structures in the state-transition trajectories of the system of interest. DMD methods usually operate over linear observations of the system—\( \phi(x) = x \)—even though extensions to nonlinear observations also exist (Williams et al., 2016). Alternatively, one can choose to approximate \( \mathcal{K} \) with a finite-dimensional matrix \( \mathbf{K} \in \mathbb{R}^{m \times m} \), called the Koopman Matrix. It operates over \( \mathcal{M} \subset \mathbb{R}^{m} \)—a subspace of the original Hilbert space, spanned by a set of observable functions \( \{\phi_1, \ldots, \phi_m\} \). Given enough knowledge about the dynamical system of interest, the observable functions can be hand-crafted—e.g. low-dimensional systems (Brunton et al., 2016), soft robots (Bruder et al., 2019), etc. Recently, data-driven methods have proven useful in scenarios where good observations functions are intractable to design but can be instead learned from data—e.g. high-dimensional systems like fluids (Lusch et al., 2018), deformable objects (Li et al., 2019), etc. In our work we choose to represent the learnt observable functions as fully-connected neural networks, while \( \mathbf{K} \) is regressed to best fit transitions \( \phi(x_{t+1}) = \mathbf{K}\phi(x_t) \). In this way the observable functions are optimised to produce a manifold \( \mathcal{M} \) where the discrete-time dynamics of the system are linear. The final system that we work with can also accommodate actions \( u_t \), which are incorporated into the next time step state through a control matrix \( \mathbf{B} : \phi(x_{t+1}) = \mathbf{K}\phi(x_t) + \mathbf{B}u_t \).

### 3 Methodology

Let \( x = \{x_1, \ldots, x_T\} \) be a single roll out of length \( T \), for a system parametrised by physical parameters \( \mathbf{p} \), and \( y_t = \phi(x_t) \). Our main task is to jointly optimise the functions \( \phi \) and \( \phi^{-1} \) to map \( x \) to \( y \) and backwards such that \( \mathbf{K} \), recovered from one-step transitions in \( y \), can reliably model the system’s dynamics—see Figure[1]. There are two main components of the overall loss function we optimise - \( \mathcal{L}_{\text{rec}} \) and \( \mathcal{L}_{\text{pred}} \). The first one is an \( L1 \) loss between each observation \( x_t \) and its auto-encoding \( \phi^{-1}(\phi(x_t)) \). The second one guarantees that if we are to project a given state to the space of Koopman observables, roll out the system \( t \) time steps into the future and reconstruct back to \( x \) this will be equivalent to just rolling out the original nonlinear dynamics \( t \) time steps into the future.

\[
\mathcal{L}_{\text{rec}} = \sum_{t=1}^{T} |x_t - \phi^{-1}(\phi(x_t))| \\
\mathcal{L}_{\text{pred}} = \sum_{t=1}^{T-1} |\phi^{-1}(\mathbf{K}^t\phi(x_1)) - x_{t+1}| 
\]

Before calculating \( \mathcal{L}_{\text{pred}} \) we regress \( \mathbf{K} \) as the best-candidate matrix that would progress the state in \( y \) one time step in the future—\( \min_{\mathbf{K}, \mathbf{B}} = ||\mathbf{K}\mathbf{y}_{1:T-1} + \mathbf{B}u_{1:T-1} - \mathbf{y}_{2:T}|| \). \( \mathbf{K} \) and \( \mathbf{B} \) can be represented as a single matrix and the actions \( u \) can be added as another dimension in \( y \). In our work we are particularly interested in the physical parameters \( \mathbf{p} \) associated with a given roll out \( x \). Namely, we want \( \mathbf{K} \) to be simultaneously encompassing the dynamics of different parameter values for the same system. Therefore, for each roll out \( x \), we add its parameters \( \mathbf{p} \) as static (non-changing) dimensions to the Koopman feature space—\( y_{t+1} = \mathbf{K}g_t, g_t = \{y_t, p\} \). Estimating \( \mathbf{K} \) on a per-roll-out basis like this might result in the dynamics in \( y \) being piece-wise linear (Karl et al., 2016) depending on the lengths of the roll outs used for training. Therefore, similar to Li et al. (2019), we choose to fit \( \mathbf{K} \) on a batch level where a set of \( B \) rollouts are concatenated one after the other to form \( \mathbf{G} = \{g^{(1)}_1, g^{(1)}_2, \ldots, g^{(B)}_1, g^{(B)}_2, \ldots, g^{(B)}_{T-1}\} \) and \( \mathbf{Y} = \{y^{(1)}_1, y^{(1)}_2, \ldots, y^{(B)}_1, y^{(B)}_2, \ldots, y^{(B)}_{T}\} \). Then \( \mathbf{K} \) and \( \mathbf{B} \) are found by performing least squares over transitions in \( \mathbf{G} \) and \( \mathbf{Y} \)—\( \min_{\mathbf{K}, \mathbf{B}} = ||\mathbf{KG} + \mathbf{BU} - \mathbf{Y}|| \). Li et al. (2019) construct their batches in a deterministic fashion, such that each batch contains roll outs from a version of the system with the same underlying physical parameters. As a result, they manage to globally linearise individual instantiations of the system of interest. At test time, assuming they get multiple observations from a single parameter value of interest, they can still reliably identify \( \mathbf{K} \) on the fly and then use it for forward simulations. However, our goal is to globally linearise the system across different parameters such that at test time it is the parameter estimates that change and \( \mathbf{K} \) is the transition Koopman matrix to which we’ve converged during training. To achieve
that we essentially construct $G$ and $Y$ in a stochastic fashion, containing data from different system parameters, before fitting $K$ at each iteration of the training algorithm.

Kalman filtering over parameters vs point estimates with a trained module  Post-training we use a Kalman filter that tracks both the latent state $y_t$ and the physical parameters $p$ but only observes the next-step latent state $y_{t+1}$. The state-transition model is the Koopman matrix $K$, augmented with additional rows that do not alter the parameters. We allow for an initial burn-in period, similar to MCMC methods (Johansen, 2010), for the mean of the estimates of the parameter values to settle and then measure how much do the filter corrections vary for the rest of the rollout. If there are still variations (can be calibrated through the training data) in the Kalman corrections after the burn-in period, this effectively provides us with an uncertainty quantification for the quality of the estimated parameter values and the downstream simulations they are used for—see Figure 2. We argue the proposed filtering approach, possible due to the globally linearised dynamics, is better-informed than simply using a point estimate for the parameters from a separately-trained model—see Figure 2.

4 Experiments and Results

Models  In the performed experiments $\phi$ and $\phi^{-1}$ are implemented as fully-connected MLP networks, with 2 hidden layers per network, 128 neurons each. The latent space $y$ of the model, where the dynamics are linearised, is 16-dimensional. Each experiment has three main phases: (1) Train the model to learn a feature space which, when concatenated to the ground-truth parameter values, linearises the dynamics of the system; (2) For a new observation at test time, estimate the latent parameter value; (3) Using the estimates and an initial state, simulate the system for a fixed time horizon. Then, for 10 uniformly distributed initial conditions $(\theta, \dot{\theta})$, we measure Mean Squared Error (MSE) between the simulations and the actual trajectories.

Two baselines are trained alongside the model utilising a Kalman filter at test time (KLM)—one which has a separate MLP network that is trained to regress parameter values from fixed-length trajectory observations of the system (MLP) and one in which the individual batches are crafted to include rollouts from individual parametrisations of the system (CBM), similar to (Li et al., 2019). In the latter, there is no explicit parameter estimation (step (2) above)—the dynamics are identified online, for every system observed, and therefore the underlying physical parameters are implicitly contained in the matrix $K$ identified on the fly. We test this model both in a single-shot scenario (identifying $K$ from a single roll out at test time) and a few-shot scenario (identifying $K$ from multiple
Figure 3: Data from an undamped oscillating pendulum used to train (left) and test (right) the same model. Two pendulum lengths are seen during training, three more are added during testing. The 2-dimensional state observations are normalised—angle of deviation $\theta$ of the pendulum along the X axis and angular velocity $\dot{\theta}$ along the Y axis in both plots.

Data Trajectories from the classical pendulum environment are used for which the small-angle approximation is invalid—initial angles of deviation $\theta$ for each simulated rollout are in the range $[\pi/2, \pi \times 0.9]$ relative to the $\pi = 0$ stable equilibrium point. Initial angular velocity is always $\dot{\theta} = 0$. During training two different versions of the pendulum are seen with pendulum lengths $L \in \{2, 5\}$—while during testing there are five pendulum lengths $L \in \{0.5, 1, 2, 3.5, 5\}$. Two values are outside of the range seen during training (testing extrapolation) and one is inside it (testing interpolation). Example phase plane plots for the different pendulum lengths, for the same initial condition, can be seen in Figure 3. Each rollout is 128 steps long while the sequences used for training are 64 steps long ($T=64$ in Eq. 2).

Results From Table 1, we can see that both the MLP and the KLM models perform equally well when interpolating in parameter space (low MSE in the last three rows) but, as expected, struggle significantly with extrapolating to parameter values outside of previously-seen ranges (high MSE in the first two rows). The model, trained on curated batches (CBM), shows varying results, depending on the evaluation regime. For the few-shot evaluation, similar to the way it was trained, described in Section 3, it is better while extrapolating and worse while interpolating, compared to KLM and MLP. However, that is partly expected because the model is exposed to 10 trajectories at test time, when identifying $K$, compared to the single trajectories seen by MLP and KLM. In the single-shot regime, CBM reports the best results when extrapolating but at the same time performs worse on the previously seen parameter values (e.g. $L = 5$). This is partly because the CBM model effectively fits new Koopman matrices $K$ each time it needs to simulate an observed system. In that sense it is more flexible and has better extrapolation capabilities but can also be less stable—the transition matrix we fit for a particular trajectory might "explain" it very well but not others, corresponding to different initial conditions or even the same trajectory but further into the future.

Lastly, it is worth emphasizing that even though the KLM model shows sufficiently bad performance for unseen pendulum lengths it is also the only model whose estimations can signal for this before actually rolling out the simulation and comparing it to the true state trajectories—high MSE values correlate to high standard deviations $\sigma$ for the Kalman correction term in the Filter (last column).

Qualitative results corresponding to the KLM model estimating the latent pendulum lengths and the MLP model predicting them can be seen in Figures 4 and 5 respectively.
Table 1: MSE results from all three models for simulating forward dynamics after either identifying the Koopman matrix on the fly (CBM model) or after estimating the latent parameter variables and using them for forward simulation (MLP and KLM models). Last column contains the standard deviation for the Kalman corrections after the initial burn-in period (T=32) which can be used as a proxy for how certain the filter is in the estimates for the values of the latent pendulum length.

Figure 4: Kalman Filtering results for initial conditions $\theta = 0.9\pi$: the latent parameter values to which the mean estimate of the filter converges in 64 time steps (left), the Kalman corrections as a function of time (middle) and the distribution of the Kalman corrections after the initial burn-in phase at T=32 (right). The initial hypothesis for the pendulum length is always $L = 0$.

Figure 5: MLP predictions for individual roll outs (10 per parameter value). The predictions of the trained parameter MLP are plotted in blue vs the ground-truth parameter values in orange. The model’s predictions fail to extrapolate ($L=0.5, L=1.0$) but does well when interpolating ($L=3.5$)
5 Conclusion

Data-driven techniques for learning dynamical models from data suffer from distributional shift problems and fail to extrapolate especially when different parameter values of the same underlying system render different behaviours. In our work we showcase how by globally linearising the observed system’s dynamics, using ideas from Deep Koopman Theory, and combining them with off-the-shelf estimation techniques like Kalman filtering, we can account for simulation failures cause by out-of-distribution observations. We demonstrate our ideas and results in the context of different pendulum lengths for the classical pendulum control environment.

Acknowledgments

This work has benefitted from funding by the UK National Physical Laboratory through a Metrology Fellowship, on the topic of Trustworthy Control Synthesis. We thank Dr. Spencer Thomas and Prof. Alistair Forbes from NPL for discussions and insightful feedback.

References


