Learning flows of control systems

Anonymous Author(s)
Affiliation
Address
email

Abstract

A recurrent neural network architecture is presented to learn the flow of a causal and time-invariant control system from data. For piecewise constant control inputs, we show that the proposed architecture is able to approximate the flow function by exploiting the system's causality and time-invariance. The output of the learned flow function can be queried at any time instant. We demonstrate the generalisation capabilities of the trained model with respect to the simulation time horizon and the class of input signals.

1 Introduction

Models play a vital role in designing control systems. For instance, in receding horizon control (Maciejowski [2002]), the model predicts the future evolution of the state variables and acts as a constraint in formulating the optimal control problem. With increasing complexity, the curse of dimensionality limits the usefulness of standard first-principle models. This limitation has motivated research on the data-driven approximation of such a physical model. Besides fast predictions for arbitrary initial conditions, in designing control systems, another advantage of many data-driven methods is the efficient computation of the model's gradient with respect to initial conditions, parameters or input signals. To leverage these advantages, in [Li et al., 2021, Geneva and Zabaras, 2022, Lu et al., 2021, Kissas et al., 2022], the focus is on learning the map from initial conditions, parameters and inputs to the solution of a differential equation.

The problem of approximating physical models from data is also tackled in system identification, see Schoukens and Ljung [2019] for an overview. Forgione and Piga [2021] have proposed methods for identifying dynamics of continuous-time control systems using neural ordinary differential equations (ODEs). However, as the dynamics correspond to the time derivative of the flow, the neural ODEs must be integrated through an ODE solver to obtain the system trajectories, representing an extra computational burden both for prediction and for computing gradients. Furthermore, errors in the learned dynamics will accumulate over time when the dynamics are integrated, and the error in the simulated trajectory can become unbounded.

To directly learn the flow of an autonomous dynamical system, Biloš et al. [2021] has proposed an alternative to neural ODEs that avoids the step of using an ODE solver and allows for faster prediction. This motivates the search for a corresponding learning scheme for controlled dynamical systems where inputs are present. However, this problem is more difficult since the domain of the flow of a continuous-time control system is infinite-dimensional.

Hanson and Raginsky [2020] have shown that continuous-time recurrent neural networks are universal approximators for flow functions of stable continuous-time dynamical systems, where the approximation quality is uniform over time. But the question of whether learning such a model from data is suitable in practice is to the best of our knowledge open.

Submitted to the DLDE Workshop in the 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

In this paper, we propose a neural network-based architecture for learning flows of controlled dynamical systems. We consider the class of piece-wise constant inputs which are practically relevant since digital controllers typically produce such control inputs. Taking advantage of the causality and time-invariance properties of the considered class of systems, we show that the flow function can be efficiently approximated by a recurrent neural network-based architecture. We illustrate using examples of nonlinear oscillators such as the Van der Pol oscillator, highlighting the generalisation capabilities of the proposed methodology.

2 Problem formulation

2.1 Considered class of control systems and flow functions

A control system Σ consists of the following quadruple (see Sontag [1998], pp. 26)

$$\Sigma = (\mathcal{T}, \mathcal{X}, \mathbb{U}, \varphi), \tag{1}$$

describing the evolution of state-variables of the dynamical system over a time interval $\mathcal{T}$ depending on its initial condition $x \in \mathcal{X}$ and input $u \in \mathbb{U}$, where $\mathbb{U}$ is a set of functions $u : \mathcal{T} \to \mathcal{U}$. The *flow*, dictating this evolution, is defined as a mapping $\varphi : \mathcal{T} \times \mathcal{X} \times \mathbb{U} \to \mathcal{X}$.

We assume that Σ time-invariant and finite dimensional. In particular, $\mathcal{T} \subseteq \mathbb{R}_{\geq 0}$, $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{U} \subset \mathbb{R}^m$. We also assume that $\mathbb{U}$ is the set of piecewise constant controls[1] of period Δ. In other words, given a sequence $\{u_k\}_{k=1}^{\infty}$ with $u_k \in \mathcal{U}$, the control input u is defined by

$$u(t) = u_k, \ (k-1)\Delta \leq t < k\Delta, \ k \in \mathbb{N}. \tag{2}$$

We will exploit two properties of the flow: *causality* which implies that the flow at time $T \geq 0$, $\varphi(T, x, u)$ depends only on the values of $u(t)$ for $0 \leq t < T$, and *continuity* in the sense that $t \mapsto \varphi(t, x, u)$ is continuous for each x, u.

As an example, we can consider the flow function generated by a system of ordinary differential equations $\dot{\xi}(t) = f(\xi(t), u(t)), \ t \geq 0$ with initial condition x where u is generated by a digital controller.

2.2 Mathematical formulation of the learning problem

We are interested in learning the flow from data on a time interval $[0, T]$ with $T > \Delta$. The input signals u and initial conditions x of interest are assumed to be drawn from probability distributions P_u on $\mathbb{U}$ and P_x on $\mathcal{X}$, respectively. Finding an approximated flow $\hat{\varphi}$ over a hypothesis class $\mathcal{H}$ amounts to minimising the following loss function

$$\ell_T(\hat{\varphi}) := \mathbf{E}\left[\frac{1}{T}\int_0^T \|\hat{\varphi}(t, X, U) - \varphi(t, X, U)\|^2 \, \mathrm{d}t\right], \tag{3}$$

where $X \sim P_x$ and $U \sim P_u$ are independent. In practice, the data from a control system is typically available in the following form

$$\left\{\left(\left\{t_k^i\right\}, x^i, u^i, \left\{\xi_k^i\right\}\right), \ k = 1, \ldots, K, \ i = 1, \ldots, N\right\},$$

for a given $K, N \in \mathbb{N}$ where $\xi_k^i = \varphi(t_k^i, x^i, u^i)$, $t_k^i \in [0, T]$ is an increasing sequence of time samples, and x^i, u^i are i.i.d samples from P_x and P_u. Thus we define the following empirical loss function $\hat{\ell}_T$ and search for a minimiser in $\mathcal{H}$:

$$\hat{\ell}_T(\hat{\varphi}) := \frac{1}{N}\sum_{i=1}^{N}\frac{1}{K}\sum_{k=1}^{K}\left\|\xi_k^i - \hat{\varphi}(t_k^i, x^i, u^i)\right\|^2. \tag{4}$$

The objective of this paper is to define an hypothesis space $\mathcal{H}$ which renders the above problem tractable and provides an approximation $\hat{\varphi}$ of the true flow function φ while preserving causality and continuity. In the next section, we propose a neural network-based architecture to solve this problem.

[1] The approach can be generalised for any class of input signals that admit a finite-dimensional causal parameterisation.

3 Proposed model architecture

Due to causality and the considered class of inputs (2), the flow $\varphi(s, x, u)$ at a time instant $s \geq 0$ depends only on a finite number of the input values $\{u_k\}$. Thus, at any time during the first control period $[0, \Delta]$, only the value of $u_1 \in \mathcal{U}$ and the initial condition $x \in \mathcal{X}$ are required to define φ. Therefore, we define $\Phi : [0, 1] \times \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ as

$$\Phi(\tau, x, u_1) := \varphi(\tau\Delta, x, u),$$

such that a finite-dimensional vector of parameters (as opposed to functions) directly maps to the flow. For an arbitrary time instant $s > 0$, the flow $\varphi(s, x, u)$ can be computed as follows:

1. Construct a map $\mathrm{d}_\Delta : (s, u) \mapsto \{\tau_k, u_k\}_{k=1}^{k_s+1}$ such that u_k is given according to (2) and

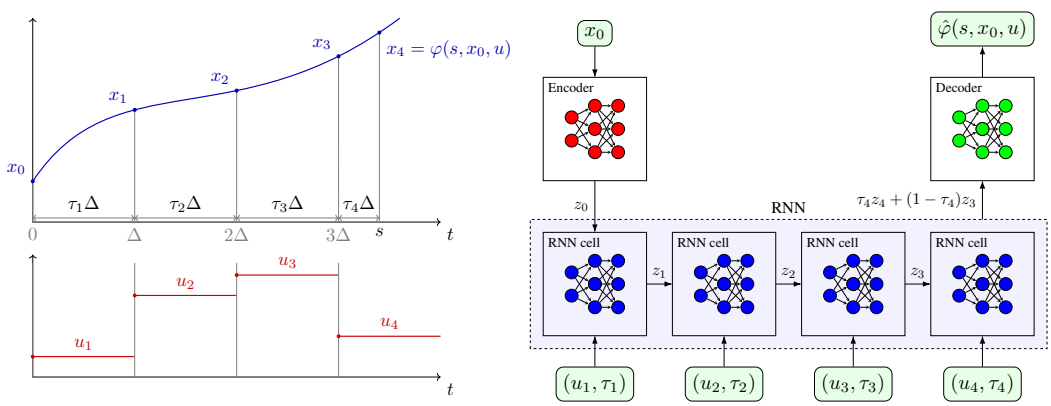
$$k_s := \lfloor s/\Delta \rfloor, \qquad \tau_k := \begin{cases} 1, & k \leq k_s \\ \dfrac{s - k_s\Delta}{\Delta}, & k = k_s + 1. \end{cases}$$

2. Define the sequence $x_k \in \mathcal{X}$ for all $k = 1, \ldots, k_s + 1$ as

$$\begin{aligned} x_0 &= x, \\ x_k &= \Phi(\tau_k, x_{k-1}, u_k). \end{aligned} \tag{5}$$

As a consequence of these two steps, we obtain $x_{k_s+1} = \varphi(s, x, u)$. Thus, trajectories of φ can be equivalently represented by the trajectories of a discrete dynamical system with inputs (τ_k, u_k).

To approximate (5), we define the hypothesis class $\mathcal{H}$ as a subset of the set of causal and continuous functions $\hat{\varphi} : \mathbb{R}_{\geq 0} \times \mathcal{X} \times \mathbb{U} \to \mathcal{X}$, defined by the composition of three neural networks as illustrated in Figure 1b.



(a) (b)

Figure 1: (a) Schematic illustration of true flow function φ for parameters $\{u_k, \tau_k\}_{k=1}^4$. (b) Corresponding model for the approximated flow $\hat{\varphi}$. In the approximated model, we first map the initial condition to a higher dimensional space through a feedforward *encoder* network. Then, the encoded state is propagated in time through a recurrent neural network (RNN). Each cell of the RNN sequentially takes (u_k, τ_k) as inputs. The two last hidden states are interpolated and mapped back to $\mathcal{X}$ through another feedforward *decoder* network.

More precisely, the hypothesis space is defined as

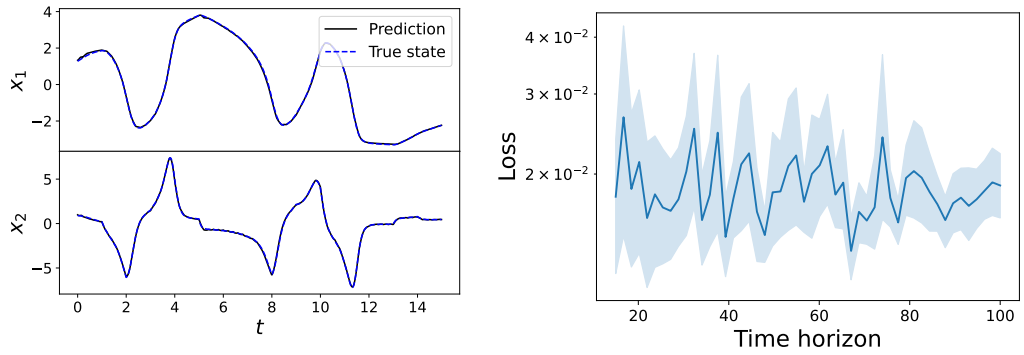
$$\mathcal{H} := \{\hat{\varphi} : \hat{\varphi}(s, x, u) = h_{\mathrm{dec}}(g_z(h_{\mathrm{RNN}}(h_{\mathrm{enc}}(x), \mathrm{d}_\Delta(s, u)), \mathrm{d}_\Delta(s, u)))\},$$

where $h_{\mathrm{enc}}, h_{\mathrm{dec}}$ and h_{RNN} are the maps corresponding to the encoder, decoder and the recurrent network, respectively. The map g_z ensures that the approximated mapping is continuous in time by interpolating between the two last RNN states.

4 Experimental evaluation

We illustrate the proposed method on a Van der Pol oscillator model, see Appendix A for details. The model is trained on $N = 300$ trajectories on $[0, 15]$ using the stochastic gradient descent algorithm `Adam` with early stopping and a gradient descent scheduler monitoring the validation loss.

The RNN is a single-layer LSTM network with 24 hidden states. The encoder network maps the initial state x to the initial LSTM hidden and cell states, and is a 3-layer feedforward net with 48 nodes in the hidden layers. The decoder network maps the hidden LSTM state to the flow value and is a 3-layer feedforward network with 24 nodes in the hidden layers. All networks use $\tanh$ activations. For each trajectory, $K = 200$ time points t_k^i are sampled using Latin hypercube sampling.



(a) (b)

Figure 2: (a) Actual (blue, dashed) and predicted (black) trajectories for the Van der Pol model for initial condition and input drawn from the corresponding distributions. (b) Estimate of $\ell_t(\hat{\varphi})$ as a function of t for the Van der Pol model trained with $T = 15$.

Figure 2a shows a predicted trajectory on $[0, 15]$ for two new pairs of initial conditions and inputs drawn from P_x and P_u, that are unseen during training. Further examples of trajectory prediction are shown in Appendix B.1. Note that the actual trajectory and the predicted trajectory are nearly indistinguishable.

The trajectories considered for training have a time horizon of 15 seconds. Due to the recurrent structure of the model and the stability of the system under consideration, we expect that the prediction performance is maintained for longer prediction horizons. To check this, we compute an estimate of the mean and variance of $\ell_t(\hat{\varphi})$ as a function of t, where $\hat{\varphi}$ is the model trained on $[0, 15]$. To this end, we compute the test loss on a different set of 300 trajectories on $[0, t]$ for a set of gridded values $15 \leq t \leq 100$. The result is shown in Figure 2b, where the coloured area represents the 95% confidence interval approximated using the empirical variance estimate. We observe that ℓ_t remains approximately constant as t increases, indicating that the model gives reliable predictions for t much larger than the value of T used for the training trajectories.

In Appendix B.2 we additionally investigate the generalisation with respect to the input distribution.

5 Concluding remarks

We presented a recurrent neural network architecture to learn the flow of a continuous causal and time-invariant control system in continuous time from trajectory data. Exploiting causality and time-invariance, we show that the problem of learning the flow function can be cast as the problem of learning a discrete dynamical system, motivating the use of an RNN-based architecture. Our experimental results on the Van der Pol oscillator show that the learned model has good prediction performance, and furthermore demonstrate that the model is able to generalise to longer prediction time horizons and new classes of input signals. We envision that our approach can provide an alternative to traditional models in control problems, bypassing the need of solving complex dynamics equations by directly predicting trajectories using our model.

References

Marin Biloš, Johanna Sommer, Syama Sundar Rangapuram, Tim Januschowski, and Stephan Günnemann. Neural flows: Efficient alternative to neural ODEs. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=XCs9rM255KZ.

Marco Forgione and Dario Piga. Continuous-time system identification with neural networks: Model structures and fitting criteria. *European Journal of Control*, 59:69–81, May 2021. doi: 10.1016/j.ejcon.2021.01.008. URL https://doi.org/10.1016/j.ejcon.2021.01.008.

Nicholas Geneva and Nicholas Zabaras. Transformers for modeling physical systems. *Neural Networks*, 146:272–289, 2022. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2021.11.022. URL https://www.sciencedirect.com/science/article/pii/S0893608021004500.

Joshua Hanson and Maxim Raginsky. Universal simulation of stable dynamical systems by recurrent neural nets. In Alexandre M. Bayen, Ali Jadbabaie, George Pappas, Pablo A. Parrilo, Benjamin Recht, Claire Tomlin, and Melanie Zeilinger, editors, *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pages 384–392. PMLR, 10–11 Jun 2020. URL https://proceedings.mlr.press/v120/hanson20a.html.

Georgios Kissas, Jacob H. Seidman, Leonardo Ferreira Guilhoto, Victor M. Preciado, George J. Pappas, and Paris Perdikaris. Learning operators with coupled attention. *CoRR*, abs/2201.01032, 2022. URL https://arxiv.org/abs/2201.01032.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, March 2021. doi: 10.1038/s42256-021-00302-5. URL https://doi.org/10.1038/s42256-021-00302-5.

Jan M. Maciejowski. *Predictive Control with Constraints.* Prentice Hall, England., 2002.

Johan Schoukens and Lennart Ljung. Nonlinear system identification: A user-oriented road map. *IEEE Control Systems*, 39(6):28–99, December 2019. doi: 10.1109/mcs.2019.2938121. URL https://doi.org/10.1109/mcs.2019.2938121.

Eduardo D. Sontag. *Mathematical Control Theory: Deterministic Finite Dimensional Systems*, pages 25–80. Springer New York, New York, NY, 1998. ISBN 978-1-4612-0577-7.

A Van der Pol oscillator

The Van der Pol oscillator is described by the system of ordinary differential equations

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= -x_1(t) + \left(1 - x_1(t)^2\right) \mu x_2(t) + u(t), \end{aligned} \tag{6}$$

with $\mu = 1$. We take $x(0) \sim N(0, I)$, i.e. a standard normal distribution. The control input sampling time is $\Delta = 0.1$ and the inputs considered are square wave inputs with period 5Δ and amplitudes sampled i.i.d. from $N(0, \sigma = 5)$, i.e.

$$\begin{aligned} u_{1+5k} &\sim N(0, 5), \\ u_{j+5k} &= u_{1+5k}, j = 2, 3, 4, 5 \end{aligned}$$

holds for all $k \in \mathbb{N}$. To generate the data used to train the model in Section 4, we integrate (6) with an RK45 solver.

B Additional experimental results

B.1 Van der Pol prediction results

Figure 3 shows four additional examples of trajectories predicted by the model described in Section 4, with $x \sim P_x, u \sim P_u$. As in Figure 2a, the predicted trajectory (in black) is nearly indistinguishable from the actual trajectory (in blue, dashed).

Figure 4 shows four additional trajectories on the larger prediction horizon $T = 100$. Recall that the model used to predict these was trained with trajectories on the time interval $[0, 15]$, confirming the observation in Figure 2b that the loss function ℓ_t remains approximately constant even as the prediction horizon increases well beyond the length of the trajectories considered during training.

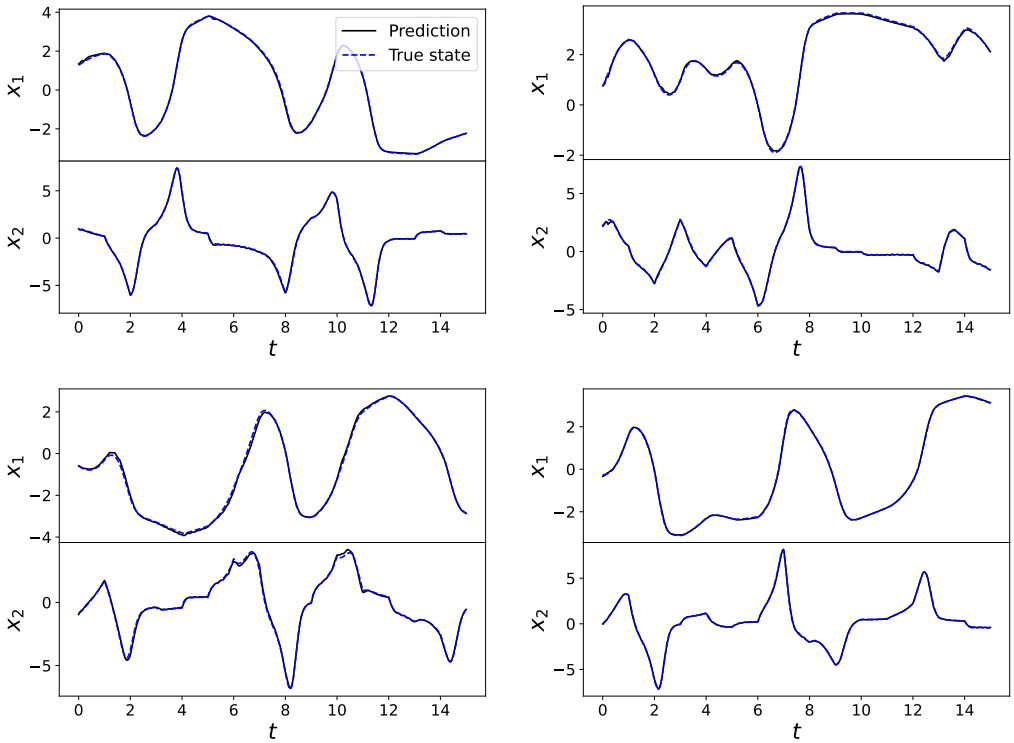


Figure 3: Actual and predicted trajectories for the Van der Pol oscillator model

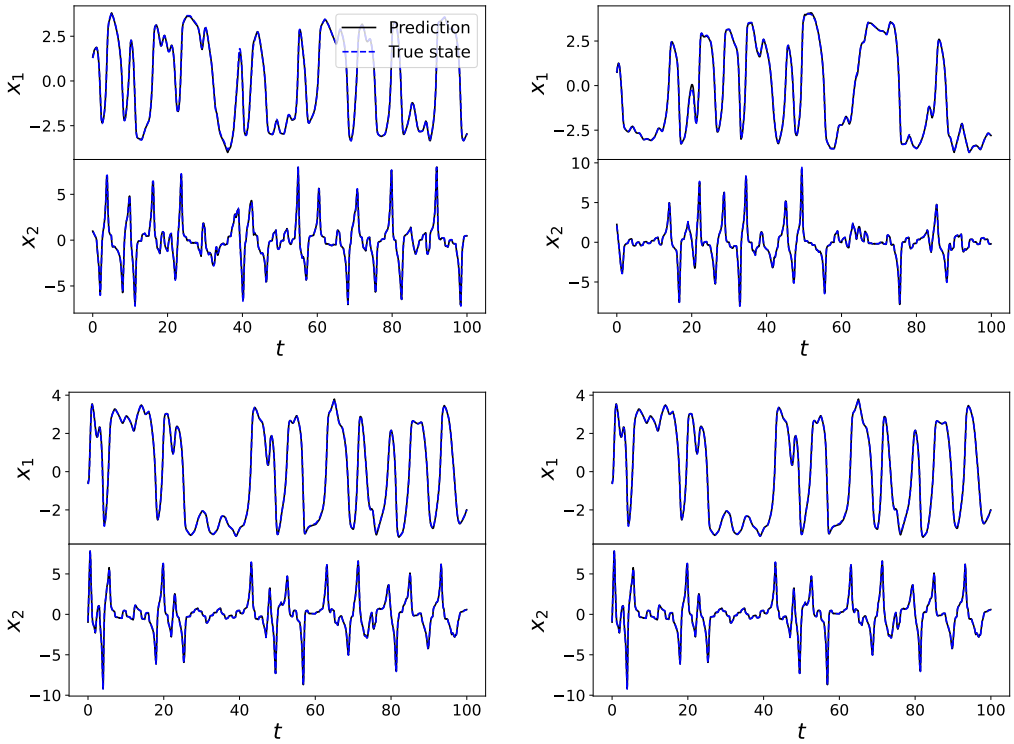


Figure 4: Actual and predicted trajectories for the Van der Pol oscillator model on the longer time horizon $T = 100$.

B.2 Generalisation to different input distributions

We additionally investigate the performance of the model trained in section 4 to an input distribution different from the training distribution P_u. In particular, we consider here a distribution Q_u on $\mathbb{U}$ consisting of sinusoidal sequences with random amplitude and frequency, that is,

$$u_k = A \sin\left(k \frac{\Omega}{2}\right),$$

where $A \sim \mathrm{LogNormal}(0, 1)$ and $\Omega \sim \mathrm{Uniform}(0, 2\pi)$. This corresponds to sinusoidal signals with frequency in the band $[0, 1/2\Delta]$, which is chosen to prevent aliasing.

To verify the performance on this class of inputs, we compute an estimate of $\ell_T(\hat{\varphi}; Q_u)$, defined as in equation (3) with the expectation taken with $U \sim Q_u$. Figure 5 shows four trajectories with initial conditions and input signals drawn from P_x and Q_u, respectively. Figure 6, shows a box plot of the distribution of the estimate of ℓ_t computed on 500 trajectories for each of the two input distributions P_u and Q_u. As expected, the mean and variance of the prediction loss for the distribution Q_u are slightly higher than for P_u, but remain reasonably closer to that of P_u. This is confirmed by Figure 5, where we observe that there is little degradation in prediction performance compared to the prediction on inputs drawn from P_u.

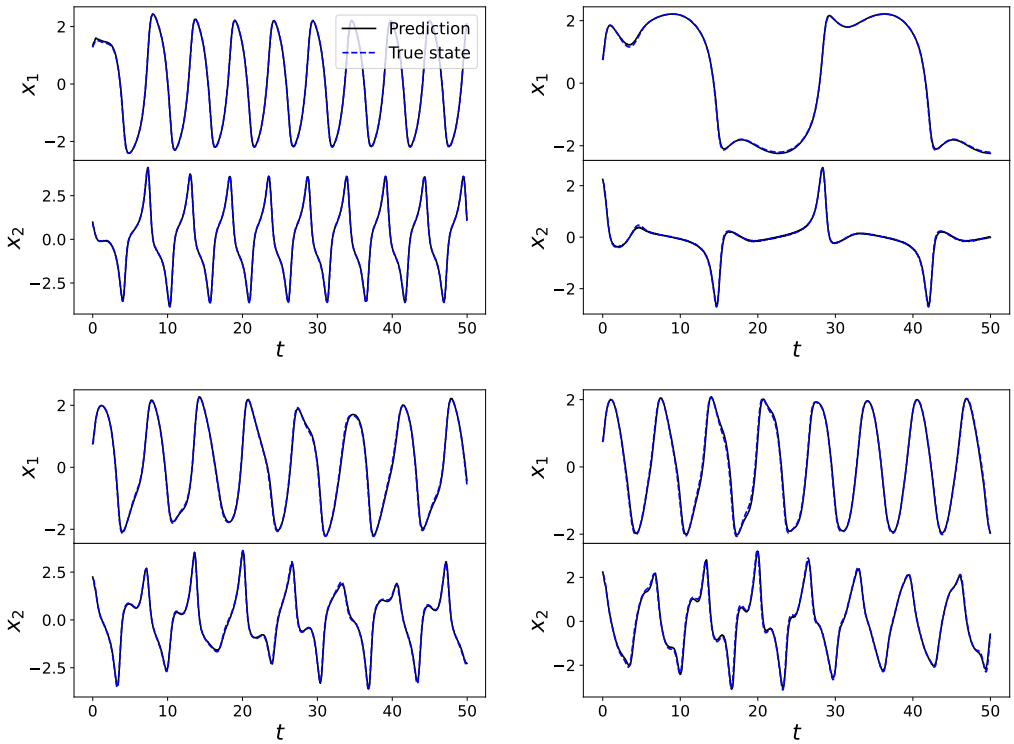


Figure 5: Actual and predicted trajectories for the Van der Pol oscillator model of section 4 with inputs drawn from the distribution Q_u.

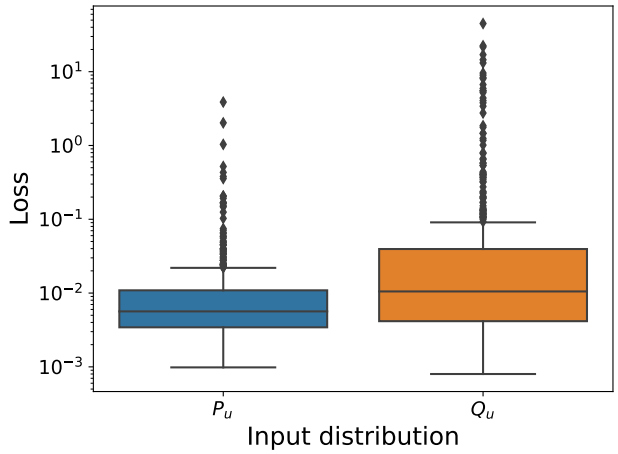


Figure 6: Distribution of the loss $\ell_T(\hat{\varphi})$ with input distributions P_u (blue) and Q_u (orange).