MODEL-BASED UNKNOWN INPUT ESTIMATION VIA PARTIALLY OBSERVABLE MARKOV DECISION PROCESSES

Anonymous authors
Paper under double-blind review

ABSTRACT

In the context of condition monitoring for structures and industrial assets, the estimation of unknown inputs, usually referring to acting loads, is of salient importance for guaranteeing safe and performant engineered systems. In this work, we propose a novel method for estimating unknown inputs from measured outputs, particularly for the case of dynamical systems with known or learned dynamics. The objective is to search for those system inputs that will reproduce the actual measured outputs, which can be reformulated as a Partially Observable Markov Decision Process (POMDP) problem and solved with well-established planning algorithms for POMDPs. The cross-entropy method is adopted in this paper for solving the POMDP due to its efficiency and robustness. The proposed method is demonstrated using simulated dynamical systems for structures with known dynamics, as well as a real wind turbine with learned dynamics, which is inferred via use of a Replay Overshooting (RO) scheme, a deep learning-based dynamics method for learning stochastic dynamics.

1 INTRODUCTION

Across a range of scientific fields, including structural engineering, robotics, economics, and biology, it is essential to obtain information on the external inputs that are applied onto dynamical systems (Sanchez & Benaroya, 2014; Rajamani et al., 2017). To what concerns the domains of Structural Health Monitoring (SHM) and Prognostics and Health Management (PHM), in particular, the assessment of performance or condition, e.g in terms of fatigue accumulation and reliability, can be evaluated more efficiently under adequate estimation of the acting loads. One typical such application is input estimation for vehicles (e.g. via estimation of the road roughness profile); a use case which has found increasing use in recent years, as part of the so called on board monitoring or mobile sensing platforms (Jin et al., 2022; Kang et al., 2019; Xue et al., 2020). However, for real-world dynamical systems, it is difficult, if not impossible, to acquire direct measurements of acting inputs (excitation) due to the fact that these inputs are of distributed and temporally continuous nature, as well as due to the typically sparse and noisy available observations from limited sensors. This is the case, for instance, with wind loads acting on wind turbines or towering structures. As a consequence, various approaches have been proposed for estimating unknown inputs from the measurable outputs, in what is typically referred to as an inverse problem, which requires the evaluation of the invertibility of dynamical systems, as studied in (Sain & Massey, 1969; Silverman, 1969; Hou & Patton, 1998; Maes et al., 2015).

The problem of unknown input estimation has long been investigated under the class of input or input-state estimation methods, such as unknown input observers (Valcher, 1999; Sundaram & Hadjicostis, 2007), optimal filters (Darouach & Zasadzinski, 1997; Gillijns & De Moor, 2007), generalized inverse approaches (Ansari & Bernstein, 2019), and Kalman-based input-state estimation methods(Maes et al., 2016; Azam et al., 2017), including augmented Kalman filters (AKF) (Lourens et al., 2012) and dual Kalman filters (DKF) (Azam et al., 2015). These well-established methods require explicit definition of system dynamics (usually state-space) models, which hardens the integration with deep learning frameworks.

In this paper, we investigate the input estimation problem from a new perspective. The input estimation problem for a dynamical system bears similarities to sequential decision-making problems, with the main distinction residing in the objective. Typically, for decision-making problems, the ultimate goal is to find the system inputs that can maximize the rewards or minimize the costs associated to some policy, which includes actions and observations. On the other hand, the input estimation problem seeks the inputs that reproduce the measured system responses, which are regarded as the actual reference outputs. With such a consideration, the input estimation problem can be formulated as a Partially Observable Markov Decision Process (POMDP), with the primary difference to a typical decision-making problem pertaining to the definition of reward/cost functions. Under this premise, a number of state-of-the-art model predictive control and model-based reinforcement learning algorithms can be repurposed to solve the POMDP problem for input estimation. Since the reformulated MDP has relatively simple reward functions and the proposed method is model-based, which assumes the underlying dynamics model is either known or learned, we choose the cross-entropy method for policy search due to its efficiency and robustness (Mannor et al., 2003).

2 Literature Review

Model-based Reinforcement Learning and Dynamics Modeling Model-based reinforcement learning (MBRL) refers to a vast family of approaches that rely on explicit estimates of the transition or dynamics function Moerland et al. (2020). Model-based reinforcement learning comprises two phases: 1) learning a forward dynamics model from observed data, which is more commonly referred to as system identification in the control literature; and 2) utilizing the inferred dynamics model to predict the distribution over state trajectories resulting from applying a sequence of actions, often by means of model-predictive control (MPC). By calculating the expected reward over state trajectories, it is possible to assess multiple candidate action sequences, and identify the optimal one. Learning reliable predictive models of high-dimensional dynamics is often the bottleneck in scaling model-based approaches to complex tasks (Atkeson & Schaal, 1997), and various approaches utilize different dynamics learning models and planning methods for the two phases. Nagabandi et al. (2018) use deterministic NN models and MPC with random shooting to achieve data-efficient control in higher dimensional tasks than can be modeled by Gaussian Processes. Chua et al. (2018) proposed PETS by fusing uncertainty-aware deep network dynamics models with sampling-based uncertainty propagation. On the basis of probabilistic neural networks serving as the dynamics model, the cross-entropy method is applied to search for the optimal policy. Igl et al. (2018) proposed the Deep Variational RL, which directly uses a network to output distribution of the latent state from the observation, with a particle filter approximating the intractable computation of the belief update. Many other approaches solve POMDPs by Recurrent Neural Networks (RNNs). One typical instance is the Deep Recurrent Q-network (DRQN), introduced by Hausknecht & Stone (2015), which employs RNNs to integrate historical trajectory and is robust to partial observability on Atari games.

Model Predictive Control Model predictive control (MPC) (Camacho & Alba, 2013)) is a class of control methods that plan an optimized sequence of actions based on a model. Learning-based MPC (Hewing et al., 2020) has a tight connection with MBRL, particularly for nonlinear and black-box models. In general, at each time step, MPC obtains an optimal action sequence by sampling multiple sequences and applying the first action of the sequence to the environment. Formally, at time step t_0, a MPC agent will search for an action sequence $\mathbf{u}_{t_0:t_0+H}$ by optimizing:

$$\max_{\mathbf{u}_{t_0:t_0+H}} \mathbb{E}_{\mathbf{z}_{t+1}\sim p(\mathbf{z}_{t+1}|\mathbf{z}_t,\mathbf{u}_t)}\big[\sum_{t=t_0}^{t_0+H} r(\mathbf{z}_t,\mathbf{u}_t)\big], \tag{1}$$

where H denotes the planning horizon. Then, only the first action $\mathbf{u}_t$ from the action sequence will be selected and applied to the environment, and the same process is repeated. As a black-box optimization problem, black-box MPC employs zero-order, or gradient-free, optimization methods to solve Eq. 1. The random shooting, a simple Monte Carlo method, is used in the Mb-Mf (Nagabandi et al., 2018). The random shooting method uniformly and randomly samples a number of action sequences $\mathbf{u}_{t:t+H}$ from the space of action sequences. Following the transition function, the current state $\mathbf{z}_t$ is transited to $\mathbf{z}_{t+H}$ by applying the action sequences as defined in the model. Evaluation of action sequences is based on the returns gathered throughout the transition process. The action sequence with the highest reward will be preserved as the solution of Eq. 1. The random shooting method is straightforward to implement and does not demand a great deal of computational resources. However, because it samples from the same distribution at every iteration, it suffers from low efficiency of the random sampling process and high variance. Thus, when dealing with higher dimensional action spaces and longer decision horizons, it could fail to sample a high reward action sequence. Recent advances in MPC methods have centered on altering the sampling strategies (Chua et al., 2018; Hafner et al., 2019) and the sample space (Wang & Ba, 2019). Replacing the random shooting method with the cross-entropy method (CEM) (Botev et al., 2013), PETS (Chua et al., 2018), and PlaNet (Hafner et al., 2019) improve the optimization efficiency. Instead of sampling randomly and uniformly at every iteration, CEM samples the action sequences from an initial multivariate normal distribution, and then update the sampling distribution according to the evaluation of the sampled sequences so that the mean value gradually moves toward the the high-reward sequences, which will then be sampled with a higher probability. This principle resembles many other gradient-free optimization methods (Hansen, 2016; Yu et al., 2016; Hu et al., 2017). As a result, other gradient-free optimization methods can also be used to solve Eq. 1 and integrated into the proposed framework. We choose to use CEM throughout this work because of its effectiveness and because it solved all the considered tasks. The algorithm and its properties are detailed in Section 3.2.

3 Background

3.1 Partially Observable Markov Decision Process

A partially observable Markov decision process (POMDP) is defined by a tuple $(\mathcal{Z}, \mathcal{X}, \mathcal{U}, f, g, r)$, where $\mathcal{Z}$ is the state space, $\mathcal{X}$ is the observation space, and $\mathcal{U}$ is the action (input) space, while f, g and r are the respective transition, observation and reward functions (Sutton et al., 1998). The system dynamics is described by the following stochastic equations, which reflect the Markovian assumption:

$$\begin{aligned} \mathbf{z}_t &= f(\mathbf{z}_{t-1}, \mathbf{u}_{t-1}) + w_t, \\ \mathbf{x}_t &= g(\mathbf{z}_t) + v_t, \\ \mathbf{r}_t &= r(\mathbf{x}_t, \mathbf{u}_{t-1}). \end{aligned} \tag{2}$$

The latent states $\mathbf{z}_t$ evolve according to the transition function f, for an imposed instantaneous input and a random noise disturbance w_t. Subsequently, we observe a noisy or partially observed observation according to the observation function g, contaminated with another random noise v_t, reflecting measurement and modeling imprecision; and finally we receive a reward $\mathbf{r}_t$ based on the reward function r, the generated observation, and the imposed input. The ultimate goal is to search for a policy $\mathbf{u}_{1:T}$ that maximizes the sum of rewards (or minimizes costs).

At a high level, all standard reinforcement learning algorithms follow the same loop: the agent interacts with the POMDP by using a trail policy, which may or may not match the true policy, by observing the current observation $\mathbf{x}_t$, selecting an action $\mathbf{u}_t$, then observing the resulting next observation $\mathbf{x}_{t+1}$ and a reward value $\mathbf{r}_{t+1} = r(\mathbf{x}_{t+1}, \mathbf{u}_t)$. This procedure is repeated for multiple iterations, and the agent uses the observed tuple $(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1}, \mathbf{r}_{t+1})$ to update its policy. In this paper, we propose to apply a similar framework for the input estimation problem in dynamical systems. In this case, the policies to be optimized are the system inputs.

3.2 Cross Entropy Method

Among numerous RL algorithms, the cross-entropy method (CEM) is selected in this paper for input estimation on availability of a dynamics model. The selection is motivated by the efficiency and robustness of this approach given the dynamics model and the reward function (Mannor et al., 2003; De Boer et al., 2005), which is the case for the scenarios we considered. CEM is a sampling-based optimization algorithm that infers an input sequence distribution that optimizes the objective reward function. As detailed in Alg. 1, it begins by initializing a standard normal Gaussian belief over optimal input sequences, where t represents the current step and H is the length of the input estimation horizon.

As there is no backward value iteration from the last time step to the first, the complexity of the CEM algorithm increases only marginally in terms of the decision horizon, which is one of the advantages of CEM. The main difference brought by longer decision horizon is sampling a vector of higher dimension, and this does not require significantly higher computational resources. Additionally, it is a sampling-based approach and hence derivative-free, which also makes the algorithm fast.

Algorithm 1 Cross-Entropy Method (CEM)

```
Initialize:  $\theta_0$  Parameter of the belief distribution
             $H$  Input estimation horizon
             $I$  Optimization iterations
             $K$  Candidates per iteration
             $\rho$  Quantile of candidates to be selected
for iteration $i = 1, 2, ..., I$ do
    Draw a set of candidate solutions $\mathbf{u}^{(k)} = (\mathbf{u}_t^{(k)}, ..., \mathbf{u}_{t+H}^{(k)})$, $k = 1, ..., K$, from the current belief distribution $f(\cdot; \theta_i)$
    Calculate the performances $S(\mathbf{u}^{(k)})$ for $k = 1, ..., K$
    Rank the $K$ performances $r^{(k)}$ from smallest to largest: $r^{(1)} \leq r^{(2)} \leq ... \leq r^{(K)}$ (ties are broken arbitrarily).
    Compute the sample $(1-\rho)$-quantile of the performances, given by $\hat{\mathbf{r}} = r(\lceil (1-\rho)K \rceil)$, and let the elite set $\mathcal{N}$ denote the indices of $k$ for which $r(\mathbf{u}^{(k)}) \geq \hat{\mathbf{r}}$
    Calculate estimated parameter $\hat{\theta}_i$ of the belief distribution based on the samples of the elite set $\{k \in \mathcal{N} : \mathbf{u}^{(k)}\}$
    Update the parameter vector according to $\theta_{i+1} = (1-\alpha_i)\theta_i + \alpha_i \hat{\theta}_{i+1}$
end for
return the first input $\mathbf{u}_t$ in the input sequence
```

The CEM algorithm is most-commonly implemented using a constant smoothing parameter (Rubinstein & Kroese, 2004), that is, $\alpha_t = \alpha$ for all t, where $\alpha \in (0, 1]$. In general, this yields a significantly faster rate of convergence of the sampling distribution $f(\cdot; \theta_i)$ compared with decreasing smoothing schemes, which is a main reason for its popularity. For this special yet important case, the theorem in Appendix A shows that the sampling distribution always converges to a unit mass located at some candidate and that the limiting probability of generating the optimal solution can be made arbitrarily close to 1 by selecting a sufficiently small value of α. It is noted that using a smaller value of α effectively reduces the rate of convergence of $f(\cdot; \theta_i)$ from the initial uniform distribution to a unit mass. Therefore, when adopting a constant smoothing parameter, competing objectives are formed, namely achieving the optimal solution with high probability, and achieving a fast rate of convergence of the sampling distribution. For the majority of applications, including what we consider in this work, choosing $\alpha = 1$ delivers accurate results while maintaining a high solution speed.

3.3 The Unknown Input Estimation Problem recast as a POMDP

To formulate the unknown input estimation problem as a POMDP, the reward function is defined as

$$r(\hat{\mathbf{u}}_t) = \|\hat{\mathbf{x}}_t - \mathbf{x}_t\|. \tag{3}$$

The objective is to find a candidate input $\hat{\mathbf{u}}_t$ that can minimize the difference between the observation that is generated from this candidate input and the true (measured) observation. The key idea is to define the reward function as the difference between the generated observation from the candidate input and the true observation, and then solve the POMDP based on a priori known or an inferred dynamics model.

The advantage of reformulating the input estimation as a POMDP is twofold: 1) When the dynamics model is available, it is straightforward to solve the reformulated POMDP with different well-established MPC algorithms, which provides ample solution schemes. 2) When the dynamics model is not available, it is possible to apply MBRL methods for coupling this scheme with deep learning-based dynamics models in order to accomplish simultaneous input estimation and dynamics model learning. In this work, we mainly investigate the use of the proposed framework for cases

Algorithm 2 Unknown Input Estimation with CEM

```
Input:  $H$  Input estimation horizon
        $I$  Optimization iterations
        $K$  Candidates per iteration
        $n$  Number of top candidates to be selected
Initialize the belief distribution over inputs $\mathbf{u}_{t:t+H} \sim N(\mathbf{0}, \mathbf{I})$
for iteration $i = 1, 2, ..., I$ do
    Draw a set of candidate solutions $\mathbf{u}^{(k)}_{t:t+H}$, $k = 1, ..., K$, from the current belief $N(\mu_i, \text{diag}(\sigma_i))$
    Evaluate the rewards $r_k$ for $k = 1, ..., K$
    Rank the $K$ rewards $r_k$ and note their indices $k$ an elite set $\mathcal{N} = \{k \in \{1, ..., K\} : r_k \text{ is one of the minimal } n \text{ rewards}\}$
    Update $\mu_{i+1} = \frac{1}{n}\sum_{k\in\mathcal{N}} \mathbf{u}^{(k)}_{t:t+H}$ and $\sigma_{i+1} = \frac{1}{n-1}\sum_{k\in\mathcal{N}} |\mathbf{u}^{(k)}_{t:t+H} - \mu_{i+1}|$
end for
return the first input $\mathbf{u}_t$
```

where the dynamics is known a priori or inferred (learned) by means of deep learning methods. The task of simultaneous estimation and learning is left for future work.

4 Simulation and Experimental Results

We first demonstrate the proposed method on a simulated structural system. The performance and applicability are investigated based on different types of input-measurement sets. The problem of mobile sensing is examined, where the inputs correspond to road and rail profiles, for the case of road versus railway infrastructure, respectively. These example demonstrate potential on real-world problems of practical value. Finally, the proposed method is applied on the further problem of input estimation using a real-world dataset obtained from a wind turbine. Here, a deep learning-based dynamics model is first inferred from the available data. The results validate the effectiveness and robustness of the proposed framework and indicate how deep learning-based dynamics modeling methods can be integrated into the framework.

4.1 Structural System

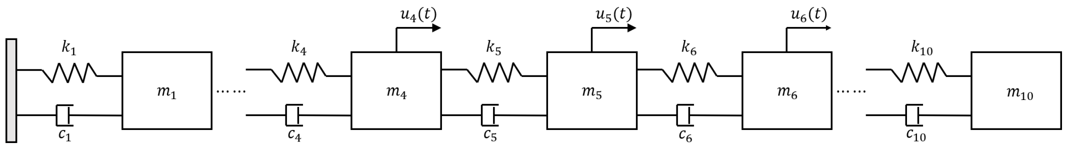


Figure 1: Illustration of the simulated 10-DOF structural system.

In this section, we implement the proposed framework on a simulated 10 Degree of Freedom (10-DOF) structural system, which is shown in Fig. 1. The structural system is governed by the following differential equations:

$$\mathbf{M}\ddot{\mathbf{q}}(t) + \mathbf{C}\dot{\mathbf{q}}(t) + \mathbf{K}\mathbf{q}(t) = \mathbf{S_u}\mathbf{u}(t), \tag{4}$$

where the displacement vector $\mathbf{q} = [q_1, ..., q_{10}]^T$; the mass matrix $\mathbf{M} = \text{diag}(m_1, ..., m_{10})$, and $m_1 = ... = m_{10} = 1$; the damping matrix $\mathbf{C} = \text{diag}(c_1, ..., c_{10})$, and $c_1 = ... = c_{10} = 1$; and the stiffness matrix

$$\mathbf{K} = \begin{bmatrix} k_1+k_2 & -k_2 & 0 & \dots & 0 & 0 \\ -k_2 & k_2+k_3 & -k_3 & \dots & 0 & 0 \\ 0 & -k_3 & k_3+k_4 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & k_9+k_{10} & -k_{10} \\ 0 & 0 & 0 & \dots & -k_{10} & k_{10} \end{bmatrix},$$

where $k_1 = ... = k_{10} = 1$. In order to embed this within the POMDP framework, the aforementioned differential equation can be brought in state-space form, by introducing the state vector $\mathbf{z} = [\mathbf{q}^T, \dot{\mathbf{q}}^T]^T$. Consequently, Eq. 4 can be rewritten in a first order ODE form:

$$\dot{\mathbf{z}}(t) = \mathbf{A}_c\mathbf{z}(t) + \mathbf{B}_c\mathbf{u}(t), \tag{5}$$

where the system matrices are

$$\mathbf{A}_c = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{C} \end{bmatrix}, \quad \mathbf{B}_c = \begin{bmatrix} \mathbf{0} \\ \mathbf{M}^{-1}\mathbf{S_u} \end{bmatrix}. \tag{6}$$

Eq. 5 can be further discretized in time, using for example a zero order hold (zoh) scheme, which leads into the following formulation:

$$\mathbf{z}_{t+1} = \mathbf{A}\mathbf{z}_t + \mathbf{B}\mathbf{u}_t \tag{7}$$

As for the observation equation, we consider the most general case, where a combination of the displacement, velocities and accelerations of this system can be measured, and thus the measurement vector can be formulated as the following form by using Eq. 4:

$$\mathbf{x}_t = \mathbf{C}\mathbf{z}_t + \mathbf{D}\mathbf{u}_t, \tag{8}$$

where

$$\mathbf{C} = \begin{bmatrix} \mathbf{S_d} & \mathbf{0} \\ \mathbf{0} & \mathbf{S_v} \\ \mathbf{S_a}\mathbf{M}^{-1}\mathbf{K} & \mathbf{S_a}\mathbf{M}^{-1}\mathbf{C} \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{S_a}\mathbf{M}^{-1}\mathbf{S_u} \end{bmatrix}. \tag{9}$$

Here, $\mathbf{S}_d$, $\mathbf{S}_v$ and $\mathbf{S}_a$ are the selection matrices that determine which DOFs in terms of available displacements, velocities and accelerations are measured.

We conduct a comprehensive investigation on the proposed method in terms of various traits: 1) input type (random, sinusoidal, and sine-swept inputs) and input localization; and 2) measurement availability.

To illustrate the workings of the proposed method, we evaluate the method for different input types, including random noise (sampled from $\mathcal{N}(0, 1)$), harmonic inputs (generated as $\sin(t)$) and swept sine inputs (generated by $\sin(t^2)$). We first set the fourth, fifth and sixth DOFs to be loaded with the same type of inputs, for the three different types respectively, while all other DOFs remain unloaded, as shown in Fig. 1, so as to investigate its capability of input localization. Then we also test scenarios where all three different types of inputs are applied to the fourth, fifth and sixth DOFs respectively and simultaneously, while other DOFs of the inputs are set to be zero. For a comparison across various input types, accelerations of all DOFs are assumed as available measurements. We evaluate the root mean square error (RMSE) for the simulation cases. As an instance, the input estimation

Table 1: Performance for different input types

Input type	**RMSE**	**Measurement availability**	**RMSE**
Random noise	.0219	Displacement	.0105
Sine	.0068	Velocity	.0317
Sinesweep	.0064	Acceleration	.0553
Mixed	.0116	Mixed	.0341

The results reveal that the proposed method can effectively tackle diverse input types, and can further estimate whether external inputs are applied on remaining DOFs, thus delivering input localization information.

Additionally, we wish to investigate the efficacy of the proposed approach under availability of diversified measurements, i.e., different quantities from the possible set of displacement, velocity and acceleration outputs. In accounting for such a mixed measurement case, it is assumed that the displacements of the first 5 DOFs and the accelerations of the last 5 DOFs are measured. The results are presented in the right hand side of Table 1. It is observed that the proposed method achieved high accuracy for all the considered scenarios, which demonstrates the effectiveness of the proposed method under different availability of measurements.

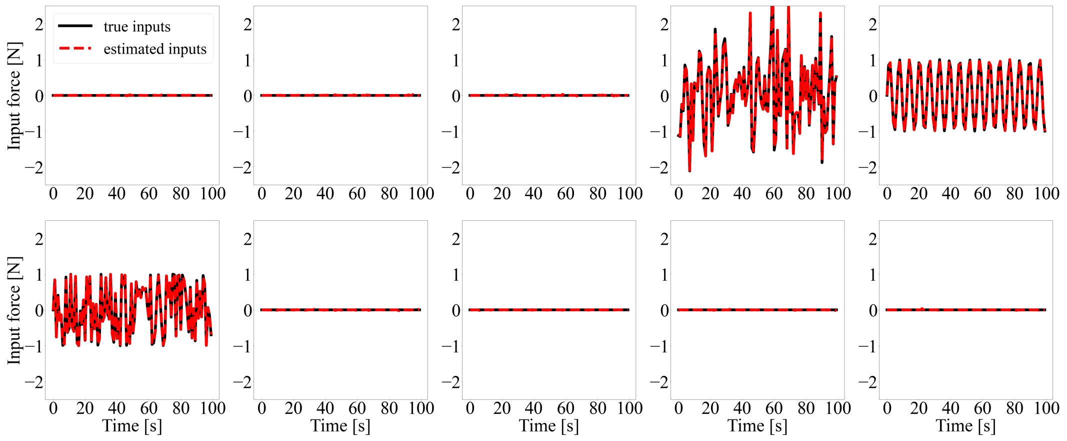


Figure 2: Input estimation results for scenario with mixed input types.

4.2 Road and Railway Profile

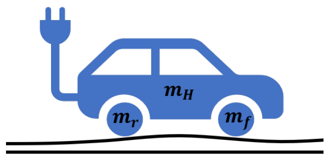


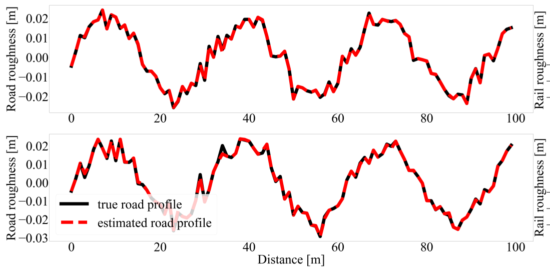
Figure 3: Road condition evaluation via input estimation algorithms. m_f, m_r, and m_H represent the masses of the front tire, rear tire, and the car body, respectively.

The degraded road surface reduces the driving comfort and poses traffic safety concerns; if not maintained promptly, the repair costs might increase. Although high-accuracy profilers equipped with lasers, inertia sensors, and cameras have been developed, their use is not practical for frequently evaluating the road network, owing to their high initial and operation costs. Lower cost alternatives often suffer from poorer precision, unless boosted with an appropriate processing or data analysis technique. Here, we suggest a scheme that can be appropriate for such a use case.

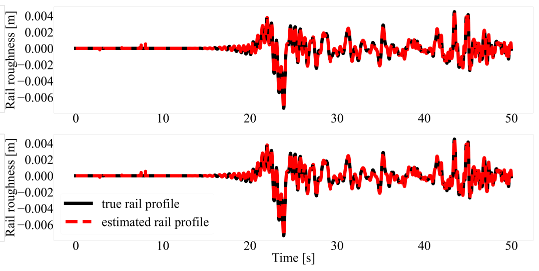
In accounting for the interacting vehicle/road-surface system, a half-car model is used. Its formulation is similar to what was presented in Section 4.1, albeit with different system matrices $\mathbf{M}$, $\mathbf{C}$ and $\mathbf{K}$. The state vector is defined as $\mathbf{z} = [\mathbf{z}_H, \theta, \mathbf{z}_f, \mathbf{z}_r]$, where $\mathbf{z}_H$, $\mathbf{z}_f$ and $\mathbf{z}_r$ are the displacements of the car body, front tire and rear tire, respectively, corresponding to the half car structure shown in Fig. 3; and θ is the pitching angle of the car body. The system matrices are detailed in Appendix E. We assume an accelerometer is mounted on the car body, allowing to track acceleration. By checking the invertibility condition stated in Appendix B, it is verified that a mere measurement of the acceleration of the car body is enough for identifying the input (i.e. the road profile). The road is simulated as a sinusoidal function, which represents large wavelength variation in terms of an uphill and downhill profile, with addition of random noise, for representing shorter-wavelength variation, corresponding to the local roughness of the road. The estimation results are shown in the Fig. 4a, with the average RMSE of 4.14×10^{-4} and the average R^2 of 0.9991. It can be observed that both large-scale patterns and the local roughness can be accurately identified, using only the acceleration of the car body.

Further to road surface roughness evaluation, in an affiliated domain of transport infrastructure, namely railways, rail roughness also needs to be evaluated for ensuring safety and comfort. Traditionally, rail roughness monitoring relies on visual methods or on highly specialized diagnostic vehicles equipped with optical and inertial sensors that gather geometric data (Hoelzl et al., 2022). Alternative approaches deploy in-service diagnostic vehicles, or even revenue trains, which more frequently run the network. These are equipped with acceleration sensors that collect vibration data, which are then exploited to extract rail roughness profiles (Dertimanis et al., 2020).

For this application, a simple system consisting of a wheelset running on a track is adopted. The wheelset is modelled as a rigid body with 6 degrees of freedom (three translations and three rotations). The track is assumed to be rigid with known roughness, which constitutes the baseline



(a) Road profile estimation.



(b) Rail profile estimation.

Figure 4: Estimation results of road and rail profiles.

for assessing the roughness profile obtained via acceleration data from the wheelset. To extract rail roughness (in the vertical direction), vertical acceleration measurements, corrupted with noises, from the wheelset are deployed. This system satisfies the invertibility condition of Section B, which implies that vertical acceleration measurements from the wheelset are sufficient to obtain the roughness profile of the rails. Fig. 4b shows the measured (true) rail profile and the one obtained via acceleration measurements (estimated rail profile), with the average RMSE of 4.39×10^{-6} and the average R^2 of 0.9999, demonstrating the applicability and performance of the proposed method.

4.3 Wind Turbine

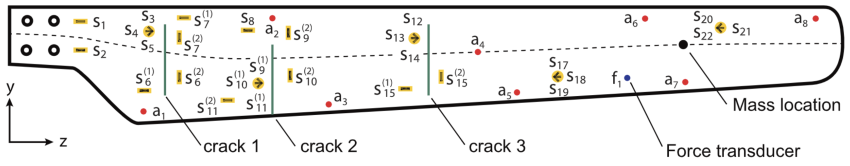


Figure 5: Position of sensors on the experimentally tested wind turbine blade. A total of eight accelerometers, a_i, are mounted on the blade, marked with a red color. Some strain information is also collected, s_{ij}, but remains unused here. The data from the accelerometers serve as measured outputs, while the data from the force transducer f_1 reflect the input excitation. The figure is reused from (Ou et al., 2021)

Among various application fields of Structural Health Monitoring (SHM), the assessment of wind turbine structures is gaining increased attention due to their critical significance and competitiveness as a renewable energy resource. To further demonstrate the value of the proposed framework for SHM of wind energy infrastructure, we validate use of the proposed scheme for vibration monitoring of operational wind turbines. The data used in this paper were obtained and illustrated in (Ou et al., 2021) by experimentally testing a small-scale wind turbine blade.

Replay Overshooting Replay Overshooting (RO), originally proposed in Li et al. (2021), is a deep learning framework to capture the dynamics of complex systems. The framework can be described in the form of a nonlinear state space model:

$$\mathbf{z}_t = f_{\theta_\mathbf{t}}(\mathbf{z}_{t-1}, \mathbf{u}_{t-1}) + w_t, \ (transition) \tag{10}$$

$$\mathbf{x}_t = g_{\theta_\mathbf{o}}(\mathbf{z}_t) + v_t, \qquad (observation) \tag{11}$$

where $f_{\theta_\mathbf{t}}$ and $g_{\theta_\mathbf{o}}$ are learnable functions (i.e., not defined a priori) governing the transition and observation models, both parameterized by neural networks with parameters $\theta = \theta_\mathbf{t} \bigcup \theta_\mathbf{o}$. The process, w_t, and observation, v_t, noise sources are assumed to follow Gaussian distributions, with respective covariances set as learnable parameters during the training process.

Within this framework, the model of Eq. 11 can be learned by maximizing an evidence lower bound of the data log-likelihood

$$\log p(\mathbf{x}) \geq \mathcal{L}(\theta, \phi; \mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{u})}[\log p_{\theta_\mathbf{o}}(\mathbf{x}|\mathbf{z})] - \mathrm{KL}\big(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{u})||p_{\theta_\mathbf{t}}(\mathbf{z},\mathbf{u})\big), \tag{12}$$

It is worth noting that, while most of the dynamical VAE setups (Girin et al., 2020; Krishnan et al., 2017; Rangapuram et al., 2018; Fraccaro et al., 2017; Chung et al., 2015; Karl et al., 2017; Higgins et al., 2017), which extend VAEs to a dynamical version by considering the temporal evolution of the latent variables, or deep state space models, use an additional neural network as the inference model. However, here the inference model of the RO follows the format of an Extended Kalman Filter, which is a well-established Bayesian filter approximation for nonlinear systems of known nonlinear functions. The RO in essence extends the EKF framework to a learnable observer representation This alleviates the need for deriving a separate inference network, q_ϕ, parameterized by a parameter vector ϕ that is independent of parameters within f_{θ_t} and g_{θ_o}. Since the objective ELBO largely depends on the goodness of reconstruction and inference, a separate inference network can weaken the training of the transition and observation models.

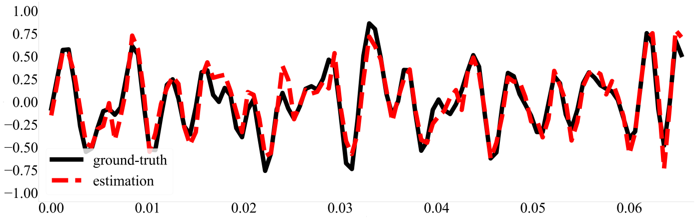


Figure 6: Input estimation of the wind turbine blade.

Here, we use RO as a means to demonstrating applicability of the herein suggested POMDP approach to input estimation, even for cases where a model is not known a priori (as was the case in the previous examples). Thus, the RO serves for recovering an underlying (latent) dynamics model under availability of data. With this learned model, we can use the presented POMDP framework to conduct model-based input estimation. In this example, we infer a dynamics model using RO based on the acceleration measurements collected from accelerometers a_1 to a_8, which serve as system outputs, while the data collected from the force transducer f_1 serve as information on the system input. For the training purpose, the system inputs are also required to be measured for learning an input-output dynamics model, and simultaneous model learning and input estimation will be considered for future work. Then, based on the learned dynamics model, we utilize the proposed POMDP approach to conduct input estimation on further test datasets, this time assuming that the input is unmeasured (unknown). The results are shown in Fig. 6, with a RMSE of 0.129, and the R^2 for the linear regression between the ground-truth and estimation is 0.87, indicating a strong consistency between the both. It is observed that the input estimation is sufficiently accurate, even for use of a learned dynamics model, which intrinsically contains modeling errors and approximations. Since in a real-world scenario, such modeling errors are ubiquitous, it is important to establish flexible inference schemes, which account for uncertainties. This is a trait of the proposed POMDP approach, where, further, the accumulated errors during the input estimation process do not grow to be unbounded.

5 Conclusion

In this work, we investigate the input estimation problem from a new perspective by reformulating it as a Partially Observable Markov Decision Process (POMDP). The ground-truth system inputs are shown to be well-approximated by iteratively selecting those candidate inputs, whose corresponding outputs can best approximate the actual measurements, and then updating the belief distribution of the inputs. We show the applicability of the proposed methodology in theory and real-world applications, and adopt a straight forward algorithm, the cross-entropy method, to solve the reformulated POMDP. Different model-based reinforcement learning frameworks and dynamics modeling methods can be integrated into the proposed methodology. This work aims to set the idea of such a use case for POMDPs in place. The influence of various reinforcement learning methods and a thorough comparison against further input estimation frameworks are left for further work.

References

Ahmad Ansari and Dennis S Bernstein. Deadbeat unknown-input state estimation and input reconstruction for linear discrete-time systems. *Automatica*, 103:11–19, 2019.

Christopher G Atkeson and Stefan Schaal. Learning tasks from a single demonstration. In *Proceedings of International Conference on Robotics and Automation*, volume 2, pp. 1706–1712. IEEE, 1997.

Saeed Eftekhar Azam, Eleni Chatzi, and Costas Papadimitriou. A dual kalman filter approach for state estimation via output-only acceleration measurements. *Mechanical systems and signal processing*, 60:866–886, 2015.

Saeed Eftekhar Azam, Eleni Chatzi, Costas Papadimitriou, and Andrew Smyth. Experimental validation of the kalman-type filters for online and real-time state and input estimation. *Journal of vibration and control*, 23(15):2494–2519, 2017.

Zdravko I Botev, Dirk P Kroese, Reuven Y Rubinstein, and Pierre L'Ecuyer. The cross-entropy method for optimization. In *Handbook of statistics*, volume 31, pp. 35–59. Elsevier, 2013.

Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer science & business media, 2013.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. *arXiv preprint arXiv:1506.02216*, 2015.

Andre Costa, Owen Dafydd Jones, and Dirk Kroese. Convergence properties of the cross-entropy method for discrete optimization. *Operations Research Letters*, 35(5):573–580, 2007.

Mohamed Darouach and Michel Zasadzinski. Unbiased minimum variance estimation for systems with unknown exogenous inputs. *Automatica*, 33(4):717–719, 1997.

Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

Vasilis Dertimanis, Manuel Zimmermann, Francesco Corman, and Eleni Chatzi. On-board monitoring of raill roughness via axle box accelerations of revenue traubs with uncertain dynamics. In *Model Validation and Uncertainty Quantification*, volume 3, pp. 16–171, 2020.

Marco Fraccaro, Simon Kamronn, Ulrich Paquet, and Ole Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. *arXiv preprint arXiv:1710.05741*, 2017.

Steven Gillijns and Bart De Moor. Unbiased minimum-variance input and state estimation for linear discrete-time systems with direct feedthrough. *Automatica*, 43(5):934–937, 2007.

Laurent Girin, Simon Leglaive, Xiaoyu Bie, Julien Diard, Thomas Hueber, and Xavier Alameda-Pineda. Dynamical variational autoencoders: A comprehensive review. *arXiv preprint arXiv:2008.12595*, 2020.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019.

Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 aaai fall symposium series*, 2015.

Lukas Hewing, Kim P Wabersich, Marcel Menner, and Melanie N Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Sy2fzU9gl.

Cyprien Hoelzl, Vasilis Dertimanis, Matthias Landgraf, Lucian Ancu, Marcel Zurkirchen, and Eleni Chatzi. On-board monitoring for smart assessment of railway infrastructure: A systematic review. In *The Rise of Smart Cities*, pp. 223–259. Advanced Structural Sensing and Monitoring Systems, 2022.

Ming Hou and Ron J Patton. Input observability and input reconstruction. *Automatica*, 34(6): 789–794, 1998.

Yi-Qi Hu, Hong Qian, and Yang Yu. Sequential classification-based optimization for direct policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for pomdps. In *International Conference on Machine Learning*, pp. 2117–2126. PMLR, 2018.

Nan Jin, Vasilis K Dertimanis, Eleni N Chatzi, Elias G Dimitrakopoulos, and Lambros S Katafygiotis. Subspace identification of bridge dynamics via traversing vehicle measurements. *Journal of Sound and Vibration*, 523:116690, 2022.

R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960. ISSN 0021-9223. doi: 10.1115/1.3662552.

Sun-Woo Kang, Jung-Sik Kim, and Gi-Woo Kim. Road roughness estimation based on discrete kalman filter with unknown input. *Vehicle System Dynamics*, 57(10):1530–1544, 2019.

Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=HyTqHL5xg.

Rahul Krishnan, Uri Shalit, and David Sontag. Structured inference networks for nonlinear state space models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Albert H Li, Philipp Wu, and Monroe Kennedy. Replay overshooting: Learning stochastic latent dynamics with the extended kalman filter. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 852–858. IEEE, 2021.

E Lourens, Edwin Reynders, Guido De Roeck, Geert Degrande, and Geert Lombaert. An augmented kalman filter for force identification in structural dynamics. *Mechanical systems and signal processing*, 27:446–460, 2012.

Kristof Maes, E Lourens, Katrien Van Nimmen, Edwin Reynders, Guido De Roeck, and Geert Lombaert. Design of sensor networks for instantaneous inversion of modally reduced order models in structural dynamics. *Mechanical Systems and Signal Processing*, 52:628–644, 2015.

Kristof Maes, AW Smyth, Guido De Roeck, and Geert Lombaert. Joint input-state estimation in structural dynamics. *Mechanical Systems and Signal Processing*, 70:445–466, 2016.

Shie Mannor, Reuven Y Rubinstein, and Yohai Gat. The cross entropy method for fast policy search. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 512–519, 2003.

Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. Model-based reinforcement learning: A survey. *arXiv preprint arXiv:2006.16712*, 2020.

Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7559–7566. IEEE, 2018.

Yaowen Ou, Konstantinos E Tatsis, Vasilis K Dertimanis, Minas D Spiridonakos, and Eleni N Chatzi. Vibration-based monitoring of a small-scale wind turbine blade under varying climate conditions. part i: An experimental benchmark. *Structural Control and Health Monitoring*, 28 (6):e2660, 2021.

Rajesh Rajamani, Yan Wang, Garrett D Nelson, Ryan Madson, and Ali Zemouche. Observers with dual spatially separated sensors for enhanced estimation: Industrial, automotive, and biomedical applications. *IEEE Control Systems Magazine*, 37(3):42–58, 2017.

Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. *Advances in neural information processing systems*, 31:7785–7794, 2018.

Herbert E Rauch, F Tung, and Charlotte T Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8):1445–1450, 1965.

Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*, volume 133. Springer, 2004.

Michael Sain and James Massey. Invertibility of linear time-invariant dynamical systems. *IEEE Transactions on automatic control*, 14(2):141–149, 1969.

J Sanchez and Haym Benaroya. Review of force reconstruction techniques. *Journal of Sound and Vibration*, 333(14):2999–3018, 2014.

LMJP Silverman. Inversion of multivariable linear systems. *IEEE Transactions on automatic control*, 14(3):270–276, 1969.

Shreyas Sundaram and Christoforos N Hadjicostis. Delayed observers for linear systems with unknown inputs. *IEEE Transactions on Automatic Control*, 52(2):334–339, 2007.

Richard S Sutton, Andrew G Barto, et al. Introduction to reinforcement learning. 1998.

Maria Elena Valcher. State observers for discrete-time linear systems with unknown inputs. *IEEE Transactions on Automatic Control*, 44(2):397–401, 1999.

Tingwu Wang and Jimmy Ba. Exploring model-based planning with policy networks. *arXiv preprint arXiv:1906.08649*, 2019.

Kai Xue, Tomonori Nagayama, and Boyu Zhao. Road profile estimation and half-car model identification through the automated processing of smartphone data. *Mechanical Systems and Signal Processing*, 142:106722, 2020.

Yang Yu, Hong Qian, and Yi-Qi Hu. Derivative-free optimization via classification. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

A Convergence of Cross Entropy Method

We have the following three theorems for the convergence properties of CEM. The detailed proof can be found in Costa et al. (2007).

Theorem 1. *(Necessary condition) The optimal solution is generated eventually by the CE algorithm with probability 1 only if the smoothing condition* $\sum_{i=1}^{\infty}\prod_{m=1}^{t}(1-a_m)=\infty$.

Theorem 2. *(Sufficient condition) The optimal solution is generated eventually by the CE algorithm with probability 1 if the smoothing condition* $\sum_{i=1}^{\infty}\prod_{m=1}^{t}(1-a_m)^n=\infty$.

Theorem 3. *The sequence of probability mass function* $f(\cdot;\theta_i), i \geq 1$, *converges with probability 1 to a unit mass located at some candidate* θ *only if* $\sum_{t=1}^{\infty}\alpha_t=\infty$.

B Invertibility Conditions

If the unknown input $\mathbf{u}$ can be uniquely identified from the output $\mathbf{x}$, the system is called invertible. The conditions for invertibility have already been well studied in the literature (Sain & Massey, 1969). Here, we state a sufficient and necessary condition for invertibility of linear dynamical systems with different kinds of measurements.

Theorem 4. *The system is invertible if and only if the matrix* $\mathbf{N}$ *has full column rank, where*

$$\mathbf{N} = \begin{bmatrix} \mathbf{D} & 0 & \dots & 0 \\ \mathbf{CB} & \mathbf{D} & \dots & 0 \\ \mathbf{CAB} & \mathbf{CB} & \dots & 0 \\ \vdots & \vdots & & \vdots \\ \mathbf{CA}^{n-1}\mathbf{B} & \mathbf{CA}^{n-2}\mathbf{B} & \dots & \mathbf{D} \\ \mathbf{CA}^{n}\mathbf{B} & \mathbf{CA}^{n-1}\mathbf{B} & \dots & \mathbf{CB} \\ \vdots & \vdots & & \vdots \\ \mathbf{CA}^{2n-1}\mathbf{B} & \mathbf{CA}^{2n-2}\mathbf{B} & \dots & \mathbf{CA}^{n-1}\mathbf{B} \end{bmatrix} \tag{13}$$

and n is the dimension of the state space.

If the system is not invertible, there would be infinitely many candidate inputs that can generate the same output, which makes the search for the true system inputs impossible, since the problem is intrinsically ill-conditioned.

C Replay Overshooting: Extended Kalman Filters with Learnable Dynamics Models

C.1 Extended Kalman Filtering and Smoothing

The Extended Kalman Filter (EKF) has been a popular choice for nonlinear state estimation and parameter identification in engineering problems, mainly due to its ease of implementation, robustness and suitability for real-time applications. It assumes a sequence of measurements $\mathbf{x}_{1:T}$ from a monitored dynamical system, which are generated by some latent states $\mathbf{z}_{1:T}$ that are not necessarily directly observed. The transition from a state $\mathbf{z}_{t-1}$ to the next state $\mathbf{z}_t$ is termed as *transition*, and the process from a state z_{t1} to its corresponding observation $\mathbf{x}_t$ is termed as *observation*. The EKF assumes that the transition and observation models are given, as described by the following two equations:

$$\mathbf{z}_t = f(\mathbf{z}_{t-1}, \mathbf{u}_{t-1}) + w_t, \ (transition) \tag{14}$$

$$\mathbf{x}_t = g(\mathbf{z}_t) + v_t, \qquad (observation) \tag{15}$$

where f and g are two known linear/nonlinear functions governing the transition and observation models, and $w_t \sim \mathcal{N}(0, \mathbf{Q}_t)$, $v_t \sim \mathcal{N}(0, \mathbf{R}_t)$ are Gaussian noise sources with mean zero and covariances $\mathbf{Q}_t$ and $\mathbf{R}_t$, respectively. EKF assumes that the inferred posterior distributions of latent states are based on past and current observations $\mathbf{x}_{1:t}$ and driving force $\mathbf{u}_{1:t}$, and follow a Gaussian distribution:

$$q(\mathbf{z}_t|\mathbf{x}_{1:t}, \mathbf{u}_{1:t}) = \mathcal{N}(\mu_{t|t}, \Sigma_{t|t}). \tag{16}$$

Here $\boldsymbol{\mu}_{t|t}$ and $\boldsymbol{\Sigma}_{t|t}$ represent the mean and covariance of the posterior distribution. The Kalman Filter (KF) (Kalman, 1960) offers a closed-form of the posterior distributions $q(\mathbf{z}_t|\mathbf{x}_{1:t}, \mathbf{u}_{1:t})$ of latent states and provides exact inference for linear systems. The EKF conducts approximate inference, employing a linearization of the nonlinear equations. For EKF, the inference of the posterior distribution is obtained by iteratively executing the following two steps. The first step *predict* is to compute the posterior distribution $q(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{u}_{1:t-1}) = \mathcal{N}(\mu_{t|t-1}, \boldsymbol{\Sigma}_{t|t-1})$, which is simply based on previous observations $\mathbf{x}_{1:t-1}$. The second step *update* pertains in updating the posterior distribution from the *predict* step using the current observation $\mathbf{x}_t$, which gives $q(\mathbf{z}_t|\mathbf{x}_{1:t}, \mathbf{u}_{1:t}) = \mathcal{N}(\mu_{t|t}, \boldsymbol{\Sigma}_{t|t})$

The EKF is a recursive filtering method for conducting inference based on $\mathbf{x}_{1:t}$, i.e., the observations until the present time t. Since we assume that a sequence of observations $\mathbf{x}_{1:T}$ is available, it is possible and reasonable to further update the inference of posterior distributions with the whole sequence of observations, with respect to a time step within the sequence. This process is termed as *smoothing*. The Rauch-Tung-Striebel smoothing (Rauch et al., 1965) is a Kalman smoothing algorithm to infer such posterior distributions $q(\mathbf{z}_t|\mathbf{x}_{1:T}, \mathbf{u}_{1:T}) = \mathcal{N}(\boldsymbol{\mu}_{t|T}, \boldsymbol{\Sigma}_{t|T})$ of latent states based on the whole sequence of available observations.

A salient limitation of the EKF when applied to learning dynamical systems lies in the requirement for the transition model f and the observation model g to be explicitly defined or at least of known functional format. However, this is not practically feasible when applied to real-world complex systems. In tackling this limitation, the key idea of the proposed RO is to replace f and g by learnable functions, typically by neural networks. By doing this, the transition and observation models turn to be trainable and efficiently learned by minimizing the defined loss function, making the RO a flexible tool for learning the dynamics of complex systems. Also, another benefit compared to VAEs, where the inference model is parameterized by neural networks, is that the inference model of the RO follows a closed-form model. In this section, we will detail the modeling and training of the RO framework.

In Eqs. 14 and 15, if we replace f and g by learnable transition and observation functions, the framework can be described as:

$$\mathbf{z}_t = f_{\theta_\mathbf{t}}(\mathbf{z}_{t-1}, \mathbf{u}_{t-1}) + w_t, \ (transition) \tag{17}$$

$$\mathbf{x}_t = g_{\theta_\mathbf{o}}(\mathbf{z}_t) + v_t, \qquad (observation) \tag{18}$$

where $f_{\theta_\mathbf{t}}$ and $g_{\theta_\mathbf{o}}$ are the learnable functions governing the transition and observation models, both parameterized by neural networks with parameters $\theta = \theta_\mathbf{t} \bigcup \theta_\mathbf{o}$. The process noise sources w_t and observation noise v_t are assumed to follow Gaussian distributions, with respective time-invariant covariances, i.e., $w_t \sim \mathcal{N}(0, \mathbf{Q})$ and $v_t \sim \mathcal{N}(0, \mathbf{R})$ for all time steps t. The time-invariant covariances $\mathbf{Q}$ and $\mathbf{R}$ are also set as learnable parameters during the training process.

The inference model of the RO follows the format of the EKF. This is different from the inference model in VAEs, where q_ϕ is a separate inference network parameterized by ϕ independent of parameters within f_{θ_t} and g_{θ_o}. Since the objective ELBO largely depends on the goodness of reconstruction and inference, a separate inference network is thus weakening the training of the transition and observation models.

C.2 Evidence Lower Bound and Training

With Kalman filters conducting inference, the parameters to be learned are summarized in the vector θ, which includes neural network parameters of both the transition model $f_{\theta_\mathbf{t}}$ and the observation model $g_{\theta_\mathbf{x}}$. In addition, the initial values $\boldsymbol{\mu}_{0|0}$ and $\boldsymbol{\Sigma}_{0|0}$ and covariances $\mathbf{Q}$, $\mathbf{R}$ for respective noises are also parameters to be learned. Similar to the VAE, the training of ROs is embedded in the variational inference methodology, with the EKF algorithm charged with conducting inference. Given any inference model $q_\phi(\mathbf{z}_t|\mathbf{x})$ and Markovian property implied by the dynamics, the ELBO in Eq. 12 can be expressed in a factorized form ($\mathbf{x}_{1:T}$ and $\mathbf{u}_{1:T}$ are abbreviated as $\mathbf{x}$ and $\mathbf{u}$ in the following formulations for simplicity):

$$\begin{aligned}\mathcal{L}(\theta, \phi; \mathbf{x}) = \sum_{t=1}^{T} \Big(& \mathbb{E}_{q_\phi(\mathbf{z}_t|\mathbf{x},\mathbf{u})}[\log p_{\theta_\mathbf{o}}(\mathbf{x}_t|\mathbf{z}_t)] \\ & - \mathbb{E}_{q_\phi(\mathbf{z}_{t-1}|\mathbf{x},\mathbf{u})}\big[\mathrm{KL}\big(q_\phi(\mathbf{z}_t|\mathbf{x},\mathbf{u})||p_{\theta_\mathbf{t}}(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{u}_{t-1})\big)\big]\Big),\end{aligned} \tag{19}$$

where, in the RO, $q_\phi(\mathbf{z}_t|\mathbf{x}, \mathbf{u})$ is actually $q_\theta(\mathbf{z}_t|\mathbf{x}, \mathbf{u})$, which alleviates the requirement of additional parameters, further to θ, within the transition and observation models, and thus $\mathcal{L}(\theta, \phi; \mathbf{x})$ reduces to $\mathcal{L}(\theta; \mathbf{x})$. Since the posterior distributions $q_\theta(\mathbf{z}_t|\mathbf{x}, \mathbf{u})$ can be computed in closed form by EKF, the distributions in Eq. 19 can thus be computed explicitly given $\mathbf{z}_t \sim \mathcal{N}(\boldsymbol{\mu}_{t|T}, \boldsymbol{\Sigma}_{t|T})$. Thus, the

ELBO Eq. 19 can be computed in a surrogate way as:

$$\begin{aligned}\mathcal{L}(\theta;\mathbf{x}) &= -\frac{1}{2}\sum_{t=1}^{T}\Big[\log|\mathbf{C}_{t|T}\mathbf{\Sigma}_{t|T}\mathbf{C}_{t|T}^T+\mathbf{R}| \\ &+(\mathbf{x}_t-g_{\theta_o}(\boldsymbol{\mu}_{t|T}))^T(\mathbf{C}_{t|T}\mathbf{\Sigma}_{t|T}\mathbf{C}_{t|T}^T)^{-1}(\mathbf{x}_t-g_{\theta_o}(\boldsymbol{\mu}_{t|T}))+d_x\log(2\pi) \\ &+\log\frac{|\mathbf{A}_{t-1|T}\mathbf{\Sigma}_{t-1|T}\mathbf{A}_{t-1|T}^T+\mathbf{Q}|}{|\mathbf{\Sigma}_{t|T}|}-d_z+\mathrm{Tr}((\mathbf{A}_{t-1|T}\mathbf{\Sigma}_{t-1|T}\mathbf{A}_{t-1|T}^T+\mathbf{Q})^{-1}\mathbf{\Sigma}_{t|T}) \\ &+(f_{\theta_t}(\boldsymbol{\mu}_{t-1|T})-\boldsymbol{\mu}_{t|T})^T(\mathbf{A}_{t-1|T}\Sigma_{t-1|T}\mathbf{A}_{t-1|T}^T+\mathbf{Q})^{-1}(f_{\theta_t}(\boldsymbol{\mu}_{t-1|T})-\boldsymbol{\mu}_{t|T})\Big],\end{aligned} \tag{20}$$

where $\mathbf{A}_{\cdot|\cdot}=\frac{\partial f(\boldsymbol{\mu}_{\cdot|\cdot},\mathbf{u}_{t-1})}{\partial\boldsymbol{\mu}_{\cdot|\cdot}}$ is the Jacobian of f at $\boldsymbol{\mu}_{\cdot|\cdot}$, and $\mathbf{C}_{\cdot|\cdot}=\frac{\partial g(\boldsymbol{\mu}_{\cdot|\cdot})}{\partial\boldsymbol{\mu}_{\cdot|\cdot}}$ is the Jacobian of g at $\boldsymbol{\mu}_{\cdot|\cdot}$.

Typically, the variational objective function for the dynamical VAE framework focuses on the reconstruction loss, which is heavily dependent on the inference model (encoder) and observation model (decoder), whereas the transition model plays a minor role in the variational objective and works as an intermediate process for training. This often results in an accurate inference model and a less meaningful transition model, which is not applicable for prediction because the transition model cannot reflect the true underlying dynamics. Therefore, it is necessary that the objective function also takes the accuracy of the transition model into consideration to ensure its closeness to the true latent dynamic process. To address this issue, we adopt the overshooting method proposed in Li et al. (2021), which is termed as *replay overshooting*. The key point lies in simply introducing the prediction loss of the generative model into the objective function. After obtaining a sequence of means and covariances $\{(\mu_{t|T},\Sigma_{t|T})\}$, the initial value $(\bar{\mu}_0,\bar{\Sigma}_0)=(\mu_{0|T},\Sigma_{0|T})$ will be further used for prediction process. Then similarly as the prediction step in EKF, the predicted values are obtained via the generative model as:

$$\bar{\boldsymbol{\mu}}_t = f(\bar{\boldsymbol{\mu}}_{t-1},\mathbf{u}_t), \tag{21}$$

$$\bar{\mathbf{\Sigma}}_t = \mathbf{A}_{t-1}\bar{\mathbf{\Sigma}}_{t-1}\mathbf{A}_{t-1}^T+\mathbf{Q}, \tag{22}$$

and we receive another set of distributions $\bar{q}(\mathbf{z}_t)=\mathcal{N}(\bar{\mu}_t,\bar{\Sigma}_t)$ from the generative model. The final reconstruction loss is composed of the both sets of posterior distributions, i.e., the posterior distributions obtained from the Kalman inference model as well as those obtained from the transition model, weighted by α (in this paper, we set $\alpha=0.5$). These form the objective function combined with the KL loss, expressed as:

$$\begin{aligned}\mathcal{L}(\theta;\mathbf{x}) = \sum_{t=1}^{T}\Big(&\alpha\mathbb{E}_{q_\theta(\mathbf{z}_t)}[\log p_\theta(\mathbf{x}_t|\mathbf{z}_t)]+(1-\alpha)\mathbb{E}_{\bar{q}_\theta(\mathbf{z}_t)}[\log p_\theta(\mathbf{x}_t|\mathbf{z}_t)] \\ &-\mathbb{E}_{q_\theta(\mathbf{z}_{t-1})}[\mathrm{KL}(q(\mathbf{z}_t)||p(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{u}_t))]\Big),\end{aligned} \tag{23}$$

where the first and third term are the same as in Eq. 20, while the second term is for overshooting.

The pipeline of the RO is summarized in Algorithm 3.

Algorithm 3 Replay Overshooting

```
Initialize parameters $\theta, \boldsymbol{\mu}_{0|0}, \mathbf{\Sigma}_{0|0}, \mathbf{Q}, \mathbf{R}$
while $\theta$ not converged do
    for batch $b = 1, ..., B$ do
        $(\boldsymbol{\mu}_{t|t}, \mathbf{\Sigma}_{t|t})$ = EKF($\theta, \mathbf{x}_{1:T}, \mathbf{Q}, \mathbf{R}$)
        $(\boldsymbol{\mu}_{t|T}, \mathbf{\Sigma}_{t|T})$ = EKS($\theta, \mathbf{x}_{1:T}, \mathbf{Q}, \mathbf{R}$)
        $(\bar{\boldsymbol{\mu}}_t, \bar{\mathbf{\Sigma}}_t)$ = EKP($\theta, \boldsymbol{\mu}_{0|0}, \mathbf{\Sigma}_{0|0}, \mathbf{Q}, \mathbf{R}$)
        Compute the objective value $\mathcal{L}(\theta; \mathbf{x})$
        Update $\theta, \boldsymbol{\mu}_{0|0}, \mathbf{\Sigma}_{0|0}, \mathbf{Q}, \mathbf{R}$ with stochastic
        gradient ascent on $\mathcal{L}$
    end for
end while
```

D Implementation Details

For transparency and reproducibility purposes, we here further provide the details of the employed architectures of the adopted Replay Overshooting method for learning dynamics, as follows. The architecture of each network is presented in the following format: hidden units of the first hidden layer + hidden units of the second hidden layer + number of outputs for mean value and number of outputs for covariance. Each number of units is followed by the activation function employed for that layer.

- Input: 300 timesteps of 1 dimension
- Latent Space: 16 dimensions
- Transition Network (f_{θ_t}): 128 Softplus + 128 Softplus + 128 Softplus + 16 Linear
- Emission Network (g_{θ_o}): 128 Softplus + 128 Softplus + 128 Softplus + 16 Linear

Since at the very first stage of the training process, the accumulated error over a long sequence due to an inaccurate dynamics model can lead to exploding gradients, it is not feasible to directly train over the whole range of sequences. Here a ramped training strategy is adopted, where the training data length increases gradually from 2 to T as the training proceeds. It is easier to train on shorter sequences at the beginning, and the model gradually adapts to increasing length and incoming new data, thus ensuring a stable training process.

E Half-car Model

A half-car model is here used to simulate the vehicle dynamics. The corresponding system matrices serve as the dynamics model describing the vehicle, which allows to conduct model-based input estimation for the road profile. The system matrices for the half-car model are as follows:

$$\begin{aligned}
\mathbf{M} &= \begin{bmatrix} m_H & 0 & 0 & 0 \\ 0 & I_y & 0 & 0 \\ 0 & 0 & m_f & 0 \\ 0 & 0 & 0 & m_r \end{bmatrix}, \\
\mathbf{C} &= \begin{bmatrix} c_f + c_r & L_r c_r - L_f c_f & -c_f & -c_r \\ L_r c_r - L_f c_f & L_f^2 c_f + L_r^2 c_r & L_f c_f & -L_r c_r \\ -c_f & L_f c_f & c_f & 0 \\ -c_r & -L_r c_r & 0 & c_r \end{bmatrix}, \\
\mathbf{K} &= \begin{bmatrix} k_f + k_r & L_r k_r - L_f k_f & -k_f & -k_r \\ L_r k_r - L_f k_f & L_f^2 k_f + L_r^2 k_r & L_f k_f & -L_r k_r \\ -k_f & L_f k_f & k_f + k_{tf} & 0 \\ -k_r & -L_r k_r & 0 & k_r + k_{tr} \end{bmatrix},
\end{aligned} \tag{24}$$

where $m_H = 2200, I_y = 1100, m_f = 106, m_r = 152, c_f = c_r = 2500, k_f = 2 \times 10^4, k_r = 2.6 \times 10^4, k_{tf} = k_{tr} = 4 \times 10^5$.