International Journal of Control

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tcon20

Safe control of nonlinear systems in LPV framework using model-based reinforcement learning

Yajie Bao & Javad Mohammadpour Velni

To cite this article: Yajie Bao & Javad Mohammadpour Velni (2022): Safe control of nonlinear systems in LPV framework using model-based reinforcement learning, International Journal of Control, DOI: 10.1080/00207179.2022.2029945

To link to this article: https://doi.org/10.1080/00207179.2022.2029945

INTERNATIONAL JOURNAL OF CONTROL
https://doi.org/10.1080/00207179.2022.2029945

Taylor & Francis
Taylor & Francis Group

Safe control of nonlinear systems in LPV framework using model-based reinforcement learning

Yajie Bao and Javad Mohammadpour Velni

School of Electrical & Computer Engineering, The University of Georgia, Athens, GA, USA

ABSTRACT

This paper presents a safe model-based reinforcement learning (MBRL) approach to control nonlinear systems described by linear parameter-varying (LPV) models. A variational Bayesian inference Neural Network (BNN) approach is first employed to learn a state-space model with uncertainty quantification from input-output data collected from the system; the model is then utilised for training MBRL to learn control actions for the system with safety guarantees. Specifically, MBRL employs the BNN model to generate simulation environments for training, which avoids safety violations in the exploration stage. To adapt to dynamically varying environments, knowledge on the evolution of LPV model scheduling variables is incorporated in simulation to reduce the discrepancy between the transition distributions of simulation and real environments. Experiments on a parameter-varying double integrator system and a control moment gyroscope (CMG) simulation model demonstrate that the proposed approach can safely achieve desired control performance.

ARTICLE HISTORY
Received 21 February 2021
Accepted 11 January 2022

KEYWORDS
Safe nonlinear control; model-based reinforcement learning; LPV framework

1. Introduction

Linear parameter-varying (LPV) models use a linear structure to capture time-varying and nonlinear dynamics of complex systems; these models then facilitate the formulation of computationally efficient control design algorithms for nonlinear systems (Hanema, 2018). Additionally, the linear relationship between control inputs and outputs depends on external *scheduling variables*, known as LPV parameters. The future evolution of the scheduling variables beyond the current time instant is generally unknown, which introduces uncertainties to the control design. Model predictive control (MPC) is widely used for constrained control design problems (including for LPV models). Minmax feedback control problem is formulated over all the possible scheduling trajectories and knowledge on possible future trajectories is exploited to reduce the uncertainty in scheduling evolution. Hanema et al. (2020) proposed a heterogeneously parameterised tube-based MPC approach with recursive feasibility and closed-loop stability guarantees for LPV systems without considering disturbance or uncertainty in system models.

Available theoretical works on LPV control design using MPC assume that representative LPV models are perfectly known, which is not a practical assumption. Global identification of LPV models using data-driven methods has been examined recently (see Bao, Velni, Basina et al., 2020).; Rizvi et al., 2018 and references therein). Moreover, uncertainty quantification in system identification has attracted increasing attention especially for safety-critical systems. Becker and Preciado (2019) assumed linear parameterisation of matrix functions and used the Gaussian process (GP) to estimate the posterior distributions of system parameters and latent variables. Moreover, Karl et al. (2016) proposed deep variational Bayesian filters to extract informative latent state-space embedding. Furthermore, Bao et al. (2021) developed a variational Bayesian inference Neural Network (BNN) approach to quantify uncertainty in state-space LPV model identification, which provides a posterior density estimation of the system model parameters given an input-output data set. Learning the system dynamics for MPC design, also known as learning-based MPC, largely focuses on closed-loop control performance and leaves potential for theoretical analysis due to the technical difficulties of deriving theoretical properties for stochastic models (Hewing et al., 2020).

Reinforcement Learning (RL) provides another paradigm for control design, which learns a policy that maximises the (so-called) cumulative reward by exploration and exploitation. RL can be categorised into model-free and model-based learning. Model-free RL can save efforts for a model identification but suffers from low sampling efficiency while model-based RL reduces sampling complexity by using a learned model of the system. Janner et al. (2019) provided conditions under which model usage can facilitate policy optimisation. Moreover, exploration may result in the violation of safety constraints. Therefore, methods have been developed for safe RL, such as constrained RL (Yu et al., 2019), control barrier functions based compensating control with model-free RL (Cheng et al., 2019), using reachability analysis in (Akametalu et al., 2014) or Lyapunov stability verification in (Berkenkamp et al., 2017) to construct a safe region, and combining RL and robust MPC in Gros et al. (2020) and Gros and Zanon (2020). Compared with (learning-based) MPC, MBRL is less restricted by the complexity of models, as models are only used to generate transition data for MBRL but

CONTACT Yajie Bao yb18054@uga.edu

cannot be too complex for MPC to solve the online optimisation problem in time. Additionally, there are works using ANNs to approximate the MPC control laws, e.g. see Chen et al. (2018), while RL policy gradient algorithms directly give the trained policy network.

This paper aims to develop a model-based reinforcement learning (MBRL) approach for control with safety guarantees for nonlinear systems described by LPV models by employing uncertainty quantification provided by the identified BNN model of the system. Time-varying property of systems of interest poses challenges for RL which assumes a stationary environment model. Padakandla (2020) gives a survey of RL algorithms for dynamically varying environments. In this paper, we assume no true system model and instead use input-output data to learn a crude model. We employ the BNN approach of Bao et al. (2021) to identify an LPV model with uncertainty quantification. Moreover, the uncertainties in the identified LPV model and the future scheduling trajectory will be considered simultaneously and approaches to incorporate the knowledge of the evolution of the scheduling variables will be proposed. To guarantee safety, inspired by the sim-to-real transfer of control with dynamics randomisation (Peng et al., 2018), we train the RL agents on the random dynamics from the identified BNN model without interacting with the real system. The safety of the learned policy is enforced by terminating an episode when system constraints are violated for any one of the generated dynamics. To learn an optimal control policy, we use Deep Deterministic Policy Gradient (DDPG) in Lillicrap et al. (2015), an actor-critic algorithm for environments with continuous action spaces. Furthermore, the BNN model is updated using new observations collected by applying learned policy to the real system. The updated model is expected to better characterise the uncertainty of the system, which in turn reduces the conservativeness required to ensure safety and improve the control performance.

The main contribution of this paper lies in developing a safe MBRL approach to control systems represented in LPV framework with safety guarantees. The remainder of the paper is organised as follows: Section 2 introduces the problem formulation and preliminaries. Our proposed safe MBRL using the identified BNN model is presented in Section 3. Section 4 shows validation results using numerical experiments. Concluding remarks are finally provided in Section 5.

2. Problem statement and preliminaries

We consider a constrained system represented by the following discrete-time, state-space LPV model with the initial condition $x_0 = x(0)$:

$$x(k+1) = A\left(\theta(k)\right)x(k) + B\left(\theta(k)\right)u(k), \tag{1}$$

$$x(k) \in \mathbb{X}, \quad u(k) \in \mathbb{U}, \quad k \in \mathbb{N}, \tag{2}$$

where $\theta(k) \in \Theta \subseteq \mathbb{R}^{n_\theta}$, $u(k)$, and $x(k)$, respectively, denote the scheduling variables, inputs, and states at time k; $\mathbb{X} \subseteq \mathbb{R}^{n_x}$ and $\mathbb{U} \subseteq \mathbb{R}^{n_u}$ are the state and input constraint sets; matrices A and B are smooth matrix functions of $\theta(k)$; $\mathbb{N}$ denotes the set of non-negative integers. Finally, $x(k)$ and $\theta(k)$ can be measured at every time k. In the LPV model above, the future behaviour of θ is not known at k. Using the set Θ to describe the future scheduling variables for control design is too restrictive and thus knowledge on the evolution of θ has been explored. In practice, θ generally varies within a bounded rate-of-variation (ROV), i.e.

$$\forall k \in \mathbb{N}, \quad |\theta(k+1) - \theta(k)| \leq \delta_{ROV}, \tag{3}$$

which gives a 'cone' expanding outwards from the current $\theta(k)$ to describe the possible future trajectories of θ. Other instances of the knowledge on the future θ that have been explored in the literature can be found in Hanema et al. (2020).

Given a control action $\pi : \mathbb{X} \times \Theta \times \mathbb{N} \to \mathbb{U}$, the closed-loop system can be described by

$$\begin{aligned} x(k+1) &= A(\theta(k))x(k) + B(\theta(k))\pi(x(k), \theta(k), k) \\ &\triangleq \Phi_\pi(x(k), \theta(k), k). \end{aligned} \tag{4}$$

Additionally, we use $\mathbf{x}(k|\theta, x_0)$ (resp. $\hat{\mathbf{x}}(k|\theta, x_0)$) to denote the solution $x(k)$ (resp. $\hat{x}(k)$) to (4) (resp. a data-driven model) given a scheduling signal $\theta : \mathbb{N} \to \Theta$ and the initial state x_0.

Definition 2.1: System (1) is *safe* under a control policy π if

$$\forall k \in \mathbb{N} : \Phi_\pi(x(k), \theta(k), k) \in \mathbb{X}, \quad \pi(x(k), \theta(k), k) \in \mathbb{U}. \tag{5}$$

Moreover, system (1) is said to be $\boldsymbol{\delta}$-*safe* under the control policy π if

$$\begin{aligned} \Pr\,[\forall k \in \mathbb{N} : \Phi_\pi(x(k), \theta(k), k) \in \mathbb{X}, \\ \pi(x(k), \theta(k), k) \in \mathbb{U}] \geq 1 - \delta, \end{aligned} \tag{6}$$

where $\Pr[\cdot]$ denotes the probability of an event.

In general, (5) cannot be enforced without additional assumptions (Koller et al., 2018) especially when (1) is unknown. Furthermore, δ-safety relaxes the requirements of safety to *safety with a high probability*. It is assumed that a data set $\mathcal{D} = \{(\theta(i), x(i), u(i)), x(i+1)\}_{i=0}^{N-1}$ has been collected, but matrix functions $A(\cdot)$ and $B(\cdot)$ are unknown. The problem addressed in this paper is to learn a BNN model (denoted as f^w) from $\mathcal{D}$ and a policy $\pi : \mathbb{X} \times \Theta \times \mathbb{N} \to \mathbb{U}$ in the simulation environments generated by f^w, such that the system is steered from some initial state x_0 to the origin by π while satisfying constraints in (2) with a high probability at each time instant. First, we briefly introduce the BNN approach to identify the state-space LPV (LPV-SS) model of the system.

2.1 LPV-SS model identification using BNN

A BNN is composed of DenseVariational layers[1] which approximate the posterior density of the parameters by variational inference given a prior density. In particular, a scaled mixture of two Gaussian densities (Blundell et al., 2015)

$$p(w_j) = \rho_{\text{mix},j}\mathcal{N}(w_j|0, \sigma_{j,1}^2) + (1 - \rho_{\text{mix},j})\mathcal{N}(w_j|0, \sigma_{j,2}^2), \tag{7}$$

with the tuning parameter $\rho_{\text{mix},j}$ is used as the prior density of the parameters w_j (including the weights and bias if exists) in the j-th layer. Equation (7) can represent both a heavy

tail by a large $\sigma_{j,1}$ and concentration by a small $\sigma_{j,2}$. Furthermore, $\rho_{\text{mix},j}$, $\sigma_{j,1}$, and $\sigma_{j,2}$ are determined using cross-validation (Hastie et al., 2009). Variational inference (VI) approximates difficult-to-compute probability density functions by finding a member from a family of densities that is closest to the target in the sense of Kullback–Leibler (KL) divergence (Blei et al., 2017). To approximate the posterior $p(w_j|\mathcal{D})$, VI solves

$$\min_{\vartheta_j} \mathrm{KL}\Big(q(w_j;\vartheta_j)\|p(w_j|\mathcal{D})\Big) \tag{8}$$

$$\Leftrightarrow \min_{\vartheta_j} \mathrm{KL}\Big(q(w_j;\vartheta_j)\|p(w_j)\Big) - \mathbb{E}_{q(w_j;\vartheta_j)}\left[\log p(\mathcal{D}|w_j)\right]$$

$$\Leftrightarrow \min_{\vartheta_j} \Big(\mathbb{E}_{q(w_j;\vartheta_j)}\left[\log q(w_j;\vartheta_j)\right] - \mathbb{E}_{q(w_j;\vartheta_j)}\left[\log p(w_j)\right] - \mathbb{E}_{q(w_j;\vartheta_j)}\left[\log p(\mathcal{D}|w_j)\right]\Big), \tag{9}$$

where $q(w_j;\vartheta_j)$ denotes a family of densities with parameters ϑ_j. The function in (9) is known as the evidence lower bound (ELBO) (Blei et al., 2017). To solve (9) by Monte Carlo (MC) methods and backpropagation, a reparameterisation trick is used to parameterise $q(w_j;\vartheta_j)$, i.e. $w_j = \mu_{w_j} + \sigma_{w_j} \bigodot^2 \epsilon_{w_j}$, where $\epsilon_{w_j} \sim \mathcal{N}(0, I)$ and thus $\vartheta_j = (\mu_{w_j}, \sigma_{w_j})$ here. Compared with a Dense layer (i.e. a fully connected layer with parameters w_j), a DenseVariational layer (with parameters μ_{w_j} and σ_{w_j}) doubles the number of parameters and requires minimising ELBO in (9) for uncertainty quantification of w_j. Similar to ANNs, a BNN can be composed of multiple fully connected DenseVariational layers.

Based on the LPV-SS model identification using artificial neural networks in Bao, Velni, Basina et al. (2020).), Bao et al. (2021) uses BNNs to represent matrix functions $A(\theta)$ and $B(\theta)$ in (1), which provides quantification of the uncertainties of the learned model. Figure 1 shows how a BNN is used to represent $A(\theta)$, and $B(\theta)$ is represented similarly by another BNN. Using f_A^w and f_B^w to denote the BNNs representing A and B, respectively, the BNN model of the system is described by

$$\begin{aligned} x(k+1) &= f^w(\theta(k), x(k), u(k)) \\ &= f_A^w(\theta(k))x(k) + f_B^w(\theta(k))u(k), \end{aligned} \tag{10}$$

where f^w can be learned by minimising

$$\frac{1}{N_{\text{MC}}} \sum_{i=1}^{N_{\text{MC}}} \left[\log q(w^{(i)};\vartheta) - \log p(w^{(i)}) - \log p(\mathcal{D}|w^{(i)})\right] \tag{11}$$

over ϑ using the data set $\mathcal{D}$, where $w^{(i)}$ is the ith sample generated by Monte Carlo (MC) for approximating the ELBO, and N_{MC} is the MC sample size.

Using the trained BNNs, the density of the matrix functions at a given scheduling variable can be evaluated by drawing samples from the posteriors of weights and calculating the possible matrices with each set of sampled weights. Rather than directly estimating the density from samples, we calculate the statistics such as the mean and standard deviation of each element of the matrices, which is efficient and sufficient for constructing a confidence interval of $x(k+1)$ to check (6). The number of samples is determined to guarantee a stable estimation. In order

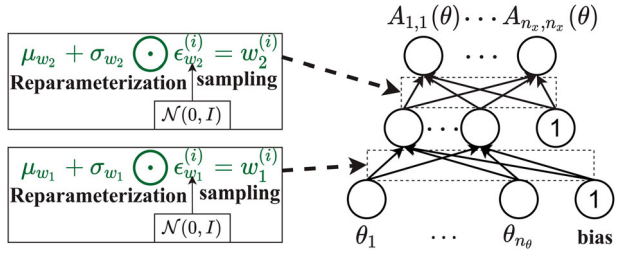

Figure 1. Using a BNN composed of multiple (two) DenseVariational layers to represent $A(\cdot)$ with reparameterisation trick. The input to the BNN is θ and the output is the vectorised $A(\theta)$. We reshape the output to restore matrix A. BNNs use data to learn the parameters μ_w and σ_w of the posterior density function.

to ensure safety, we need a reliable estimation of the system model, which is formally described in the following assumption:

Assumption 2.2: *For a confidence level $\delta \in (0, 1]$, there exists a scaling factor β such that with probability greater than $1-\delta$,*

$$\forall k \in \mathbb{N} : |x_i(k+1) - \mu_{x_i(k+1)}| \le \beta_i \sigma_{x_i(k+1)} < |\mathbb{X}_i|, \quad i = 1, 2, \ldots, n_x, \tag{12}$$

given $(x(k), \theta(k), u(k))$, where $\mu_{x_i(k+1)}$ and $\sigma_{x_i(k+1)}$, respectively, denote the estimated mean and standard deviation of the ith entry of $x(k+1)$ using the learned BNN model with Monte Carlo methods, and $|\mathbb{X}_i|$ is used to denote the range of valid x_i.

It is noted that a larger $\beta_i \sigma_{x_i(k+1)}$ means larger uncertainties of the model and gives a more conservative estimate of $x_i(k+1)$ which overestimates the probability of constraints violation, reduces the feasible region of control inputs, and thus degrades control performance. If $\beta_i \sigma_{x_i(k+1)} \ge |\mathbb{X}_i|$, the estimate is worse than the random guess of $x_i(k+1)$, which is not useful for control. The above assumption can be enforced by a proper BNN and empirically verified on the testing set[3] after model training. Moreover, δ can be estimated as the relative frequency of the testing data that violates (12) given β.

Lemma 2.3: *Given x_0, a scheduling signal θ, and a BNN model that fulfils Assumption 2.2, there exists an N_{MC} and a confidence level δ such that*

$$\Pr\Big[\forall k \in \mathbb{N} : \mathbf{x}_j(k|\theta, x_0) \in \left[\min_i \hat{\mathbf{x}}_j^{(i)}(k|\theta, x_0), \max_i \hat{\mathbf{x}}_j^{(i)}(k|\theta, x_0)\right]\Big] \ge 1 - \delta, \quad i = 1, \ldots, N_{\text{MC}}, j = 1, 2, \ldots, n_x, \tag{13}$$

where N_{MC} is the number of models drawn from the BNN model using MC methods.

Proof: When $k = 0$, $\hat{x}_0 = x_0$, as x_0 and $\theta(0)$ are known. Then, using Assumption 2.2, there exists an $N_{\text{MC}}(0)$ such that $\mathbf{x}_j(1|x_0, \theta(0)) \in [\min_{1 \le i \le N_{\text{MC}}(0)} \hat{\mathbf{x}}_j^{(i)}(1|x_0, \theta(0)), \max_{1 \le i \le N_{\text{MC}}(0)} \hat{\mathbf{x}}_j^{(i)}(1|x_0, \theta(0))]$ almost surely ,[4] $j = 1, \ldots, n_x$. Using induction, (13) is obtained using $N_{\text{MC}} = \max_k N_{\text{MC}}(k)$. ■

Lemma 2.3 guarantees that, with a high probability, the real system state trajectory is always contained in the multiple trajectories simulated by the BNN model. The uncertainties in the evolution of scheduling variables will be addressed in Section 3.

2.2 RL and policy gradient

RL in Markov Decision Processes (MDP) is generally described by a set of environment and agent states s, a set of actions a of the agent, a state transition model $p(s_{t+1}|s_t, a_t)$ (at time t) from s_t to s_{t+1} under action a_t, and a reward function $r(s, a)$, and aims to learn a policy function $a = \pi^*(s)$[5] such that the expected sum of discounted rewards $\eta_0(\pi)$ within a horizon T is maximised, i.e.

$$\begin{aligned}\pi^* &= \arg\max_{\pi} \eta_0(\pi) \\ &= \arg\max_{\pi} \mathbb{E}_{\tau\sim p(\tau|s_0,\pi)}\left[\sum_{t=0}^{T-1} \gamma^t r(s_t, \pi(s_t))\right], \end{aligned} \tag{14}$$

where $\mathbb{E}$ denotes the expectation, $\gamma \in (0, 1]$ is the discount factor and $p(\tau|s_0, \pi)$ represents the probability of a trajectory $\tau = (s_0, a_0, s_1, \ldots, a_{T-1}, s_T)$ that starts from s_0 under policy π and

$$p(\tau|s_0, \pi) = p(s_0) \prod_{t=0}^{T-1} p(s_{t+1}|s_t, \pi(s_t)).$$

In the case of control design for LPV systems, the state is $s_t = [x(t), \theta(t)]$, the immediate reward can be $r_t = -(x^{\mathrm{T}}(t+1)Qx(t+1) + u^{\mathrm{T}}(t)Ru(t))$, where $Q, R \succ 0$ are tuning parameters, and action is $a_t = u(t)$. For model-free RL, $p(s_{t+1}|s_t, a_t)$ is unknown and τ's are generated for learning by interacting with the real system. Instead, model-based RL first learns a model from data and then uses the model to generate trajectories $\tilde{\tau}$'s (Deisenroth et al., 2013). Therefore, the performance of the model-based RL depends on the discrepancy between the τ's and $\tilde{\tau}$'s.

Deterministic policy gradient algorithms (Silver et al., 2014) for RL use a parameterised function with respect to ψ (e.g. a neural network), that we denote by $\pi_\psi(s)$, as action. Moreover, action-value function $Q^\pi(s, a) = \mathbb{E}[G_0|s_0 = s, a_0 = a; \pi]$ is used to assess the expected return of a pair of state and action (s, a) following π where $G_t = \sum_{k=t}^{\infty} \gamma^{k-t} r(s_k, a_k)$ is the total discounted reward from time step t onwards. The update law of the model is based on the deterministic policy gradient theorem

$$\nabla_\psi J(\psi) = \int_{\mathcal{S}} \rho^\pi(s) \nabla_a Q^\pi(s, a) \nabla_\psi \pi_\psi(s)|_{a=\pi_\psi(s)} \mathrm{d}s, \tag{15}$$

where $J(\psi) = \int_{\mathcal{S}} \rho^\pi(s) Q^\pi(s, \pi_\psi(s))\mathrm{d}s$ denotes the performance objective and $\rho^\pi(s)$ represents the stationary distribution of s under π. Furthermore, actor-critic methods use a parameterised function $Q_\phi(s, a)$ (a.k.a. critic) with respect to ϕ to learn the value function where the actor refers to $\pi_\psi(s)$.

Off-policy policy gradient uses a known behaviour policy π_D to collect samples and estimate $Q(s, a)$ with regard to the target policy π_ψ. In this way, policy improvement is achieved. Moreover, off-policy approaches do not require full trajectories, and experience replay (Mnih et al., 2013) can be used to improve sample efficiency by randomly drawing samples from the replay memory which consists of all the episode steps during Q-learning updates. Deterministic policy gradient makes deterministic decision but is hard to guarantee enough exploration unless there is sufficient noise in the environment (Silver et al., 2014). Either adding noise to the deterministic policy or using stochastic behaviour policies can help the exploration for training.

3. Model-based RL using the BNN model

Model-based RL employs a model learned from the interactions with the real environment to simulate further episodes and uses the simulated episodes to train the agent, rather than directly learning from the experiences as model-free RL does. Consequently, the interactions with the real system are reduced, which speeds up the learning process by saving the waiting time for the system to respond or reset. Moreover, by learning in the simulated environments instead of interacting with the real system, model-based RL can avoid harm to the system, as the control inputs given by the policy network of deep RL might result in the violation of the system constraints in the exploration phase.

The performance of model-based RL depends on the accuracy of the model it employs. When there is a large model-plant mismatch, the agent trained in the simulation environments would fail to work in the real environment. Stochastic models can better capture the true system dynamics than deterministic models but increase the difficulty in optimising the RL algorithms. Deisenroth and Rasmussen (2011) proposed PILCO which uses Gaussian process (GP) models and analytical policy gradients for policy improvement. Gal et al. (2016) further improve the computational efficiency and uncertainty estimation of PILCO using BNN dynamic models to sample dynamic function realisations via particle methods. Furthermore, Higuera et al. (2018) proposed to use fixed random numbers, clipping gradients, and BNNs with multiplicative parameter noise to extend the approach of Gal et al. (2016). In this work, we also use BNNs to model the system dynamics. Given a bounded generalisation error of the model, the following theorem gives a sufficient condition for improving the true returns $\eta(\pi)$, i.e. the improvement of $\eta(\pi)$ is guaranteed when the return $\hat{\eta}(\pi)$ under the BNN model is improved by more than $C(\epsilon_m, \epsilon_\pi)$.

Lemma 3.1 (Janner et al., 2019)**:** *Let the expected total variation distance (TV-distance)*[6] $\mathbb{E}_{s\sim\pi_{D,k}}[D_{TV}(p(s', r|s, a)\|p_w(s', r|s, a))]$ *between two transition distributions be bounded at each time step by* ϵ_m *and the policy divergence* $D_{TV}[\pi\|\pi_D]$ *be bounded by* ϵ_π*. Then the true returns* $\eta(\pi)$ *and model returns* $\hat{\eta}(\pi)$ *of the policy are bounded as*

$$\eta(\pi) \geq \hat{\eta}(\pi) - \underbrace{\left[\frac{2\gamma r_{\max}(\epsilon_m + 2\epsilon_\pi)}{(1-\gamma)^2} + \frac{4 r_{\max}\epsilon_\pi}{1-\gamma}\right]}_{C(\epsilon_m,\epsilon_\pi)},$$

$D_{TV}(\cdot\|\cdot)$ denotes the total variation distance, $p(s', r|s, a)$ denotes the unknown transition distribution of the system while $p_w(s', r|s, a)$ denotes the learned transition distribution by the BNN model with parameters w, π is the target policy while $\pi_{D,k}$ denotes the data-collecting policy at time step k, γ is the discount factor, and $r_{\max}$ represents an upper bound on rewards.

In practice, ϵ_m can be estimated using the loss of the model on the time-dependent state distribution of the validation data set and ϵ_π can be evaluated using KL divergence and Pinsker's inequality (Reid & Williamson, 2009). For LPV systems of interest in this work, both the imprecise knowledge of the future scheduling signals and the epistemic uncertainties of the learned

system model can result in a discrepancy between the transition distributions of the simulation environments and real environments, which in turn degrades the performance of the learned policy when applied to the real system. Therefore, either the uncertainties in the evolution of the scheduling variables should decrease or the accuracy of the BNN model should increase to minimise $C(\epsilon_m, \epsilon_\pi)$ and achieve guaranteed improvement of the true returns. This paper investigates employing both domain knowledge of scheduling variables and fine-tuning of the BNN model using closed-loop data to reduce the discrepancy ϵ_m and thus $C(\epsilon_m, \epsilon_\pi)$.

To ensure safety, different from prior work (Gal et al., 2016; Higuera et al., 2018), we explicitly consider the system constraints in the learning process. Specifically, we check the safety constraints when a valid control input given by the policy network is applied to the dynamic functions sampled from the BNN model and terminate an episode when any predicted states violate the constraints.

Theorem 3.2: *Let π be the policy that is learned via DDPG using the data generated by Algorithm 1 with the conditions of Lemma 2.3 satisfied and $\hat{\eta}(\pi)$ is improved by more than $C(\epsilon_m, \epsilon_\pi)$. If*

$$\exists K < K_{\text{term}} : \forall k \geq K, \theta \in \Theta_{\text{sele}}, i = 1, \ldots, N_{\text{MC}}, \\ \epsilon > 0, |\hat{\mathbf{x}}^{(i)}(k|\theta, x_0)| \leq \epsilon, \tag{16}$$

where $K_{\text{term}} = \arg\min_k \hat{x}^{(i)}(k) \notin \mathbb{X}$, Θ_{sele} represents the set of selected scheduling signals which contains the scheduling trajectory of the system, and ϵ denotes the error tolerance under π, then the system is δ-safe and stabilised by the policy π.

Proof: Since an episode is terminated at K_{term}, then for $k = 0, \ldots, K_{\text{term}} - 1$, $\hat{\mathbf{x}}(k|\theta, x_0) \in \mathbb{X}$ and thus $\mathbf{x}(k|\theta, x_0) \in \mathbb{X}$ with a high probability by (13) from Lemma 2.3. Furthermore, if (16) holds (i.e. π can stabilise all the models sampled from the BNN model without violating state constraints), then for a given scheduling signal of the system $\theta \in \Theta_{\text{sele}}$, $\exists K < K_{\text{term}}, \forall k \geq K$ and $\epsilon > 0$, $|\mathbf{x}(k|\theta, x_0)| \leq \epsilon$ and (6) holds by (13). Therefore, the system is δ-safe and stabilised by the policy π. ■

Additionally, we incorporate the knowledge of scheduling variables into the generation of simulation environments to reduce the discrepancy between simulations and applications. Instances of knowledge in Hanema et al. (2020) can be easily integrated into simulations as Step 10 in the Algorithm 1 which presents the procedure of generating an episode using the BNN model. When the transition distribution $p(\theta(k+1)|\theta(k))$ of θ is known, we can generate $\theta(k+1)$ from the transition distribution; otherwise, if only the ROV is known, we can obtain the range of $\theta(k+1)$ given $\theta(k)$ and generate $\theta(k+1)$ from a uniform distribution over the range; else if only Θ is known, we can generate $\theta(k+1)$ from a uniform distribution over Θ.

For policy optimisation in the simulated environments, we use an off-policy actor-critic algorithm called deep deterministic policy gradient (DDPG). DDPG learns both a Q-function and a deterministic policy. For Q-learning of the critic that aims to minimise the difference between the Q-function represented by a network Q_ϕ and the target Q value, DDPG uses another

Algorithm 1 An episode using the BNN model

1: Require the current policy network π
2: Define the maximal length of an episode K.
3: Reset the simulation environment.
4: **while** $k < K$ **do**
5: **for** $i = 1$ to N_{MC} **do**
6: Evaluate $u(k)$ using π with $x(k)$ and $\theta(k)$
7: Sample $w^{(i)}$ from the BNN model
8: Evaluate $A^{(i)}(\theta(k))$ and $B^{(i)}(\theta(k))$ with $f_A^{w^{(i)}}$ and $f_B^{w^{(i)}}$, respectively
9: Calculate $x^{(i)}(k+1)$ by (10) with $A^{(i)}(\theta(k))$, $x(k)$, $B^{(i)}(\theta(k))$, and $u(k)$
10: Generate $\theta(k+1)$ using the knowledge ($p(\theta(k+1)|\theta(k))$, ROV, or Θ)
11: **if** $x^{(i)}(k+1) \notin \mathbb{X}$ **then**
12: Break and jump to 4.
13: **end if**
14: **end for**
15: Calculate $x(k+1) = \frac{1}{N_{\text{MC}}} \sum_{i=1}^{N_{\text{MC}}} x^{(i)}(k+1)$ and $k = k+1$
16: **end while**

network $Q_{\phi_{\text{targ}}}$ to evaluate the target Q value $r(s,a) + \gamma(1-d)\max_{a'} Q(s', a')$, where s' is the next state and a' is the next action. Furthermore, to simplify the evaluation of the target Q value, DDPG uses a target policy network $\pi_{\psi_{\text{targ}}}$ to compute an action which approximately maximises $Q_{\phi_{\text{targ}}}$, rather than directly calculating the action that maximises the target. In addition, the target policy network (resp. target Q network) shares the same architecture and parameter initialisation with the policy network (resp. Q network). Specifically, the critic is trained by minimising the following mean-squared Bellman error (MSBE) loss with stochastic gradient descent:

$$L(\phi, \mathcal{D}_{\text{repl}}) = \mathbb{E}_{(s,a,r,s',d)\sim\mathcal{D}_{\text{repl}}}\Big[(Q_\phi(s,a) - (r + \gamma(1-d) \\ \times Q_{\phi_{\text{targ}}}(s', \pi_{\psi_{\text{targ}}}(s'))))^2\Big], \tag{17}$$

where ϕ is the Q network parameters, ψ is the policy parameters, and ϕ_{targ} and ψ_{targ} are target parameters. The target networks are updated once per main network update by Polyak averaging, i.e.

$$\phi_{\text{targ}} \leftarrow \rho\phi_{\text{targ}} + (1-\rho)\phi, \\ \psi_{\text{targ}} \leftarrow \rho\psi_{\text{targ}} + (1-\rho)\psi, \tag{18}$$

where ρ is a hyperparameter. In this way, the target network values are constrained to change slowly, which greatly improves stability in learning. Experience replay is used to improve data efficiency, remove correlations in the observation sequences, and smooth over changes in the data distribution for Q-learning (Mnih et al., 2013). $\mathcal{D}_{\text{repl}}$ in (17) is the reply buffers of transitions (s, a, r, s', d), where d indicates whether state s' is terminal. It is noted that $\mathcal{D}_{\text{repl}}$ can contain previous experiences that are obtained using an outdated policy, as Q-learning is not related to the distribution of transitions and the optimal Q-function that minimises MSBE should satisfy the Bellman equation for all

possible transitions. Importance sampling is used to correct the mismatch between behaviour and target policies for off-policy policy gradient algorithms with stochastic policies but avoided for DDPG, as the deterministic policy gradient removes the integral over actions by deterministic policy gradient theorem (Silver et al., 2014). For policy learning of the actor, a deterministic policy $\pi_\psi(s)$ is learned to provide the action that maximises $Q_\phi(s,a)$. Specifically,

$$\max_{\psi} \mathbb{E}_{s\sim\mathcal{D}_{\text{repl}}}[Q_\phi(s,\pi_\psi(s))],$$

in which Q-function parameters ϕ are fixed, is solved using gradient ascent with respect to policy parameters only. Since the policy function is deterministic, the agent may not be able to explore a sufficient variety of policies at the training time to find the optimal policy. To help with the exploration, noise is added to actions at training time, i.e. the exploration policy $\pi_{\text{expl}}(s) = \pi_\psi(s) + \mathcal{N}$, where $\mathcal{N}$ denotes the selected noise. $\mathcal{N}$ can be an Ornstein–Uhlenbeck process[7] (Uhlenbeck & Ornstein, 1930) to generate temporally correlated exploration (Lillicrap et al., 2015) or uncorrelated mean-zero Gaussian noise. When exploiting the learned policy, $\pi_\psi(s)$ is evaluated without adding noise. To enforce the control input constraints, the outputs of the policy network are projected onto $\mathbb{U}$. In particular, given $\mathbb{U} = [a_{\min}, a_{\max}]$, $a = \text{clip}(\pi_\psi(s) + \mathcal{N}, a_{\min}, a_{\max})$ for exploration and $a = \text{clip}(\pi^*_\psi(s), a_{\min}, a_{\max})$ for exploitation where $\pi^*(s) = \pi^*(x,\theta)$ denotes the learned policy (i.e. the control law).

Only when the agent learns a policy that can ensure safety for all the simulation environments generated by the BNN model, interaction with the real system will be allowed. Moreover, using the new observations of the system from the interactions can update the BNN model, which can in turn reduce the conservativeness introduced for safety and improve the policy search. Also, instead of retraining the BNN model from scratch, we can use fine-tuning, a transfer learning technique discussed in Bao et al. (2020) for neural networks to update the BNN. Fine-tuning decreases the number of trainable parameters of the network by fixing parameters in the first few layers and retraining the rest of the parameters to increase computational efficiency and generalisation ability. The number of parameters to fix decreases when the change in the data distribution is more significant.

Furthermore, the generalisation of the RL depends on the discrepancy between the simulation environments and the real environments while the environments (i.e. the true LPV models of nonlinear systems under study) depend on the scheduling variables. When the real evolution pattern of the LPV model changes, the performance of the RL agent trained for the previous scheduling trajectories can degrade even to failure. In this case, we can still use fine-tuning to update the actor and critic network to improve the performance.

4. Numerical experiments and results

In this section, the proposed methods are validated using a numerical example and a complex physical system model.

4.1 Parameter-varying double integrator

In this subsection, the proposed methods are validated using a parameter-varying double integrator model. The LPV-SS representation of the system is as follows:

$$x(k+1) = \left(\begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix} + \begin{bmatrix}0.1 & 0\\ 0 & 0.1\end{bmatrix}\theta_1(k) + \begin{bmatrix}0.5 & 0.5\\ 0 & 0\end{bmatrix}\theta_2(k) + \begin{bmatrix}0 & 0\\ 0 & 0.2\end{bmatrix}\theta_3(k)\right)x(k) + \begin{bmatrix}0.5\\ 1\end{bmatrix}u(k) \tag{19}$$

with constraints and scheduling sets as

$$\mathbb{X} = \{x \in \mathbb{R}^2 | \|x\|_\infty \le 6\}, \quad \mathbb{U} = \{u \in \mathbb{R} | \, |u| \le 1\},$$
$$\Theta = \{\theta \in \mathbb{R}^3 | \|\theta\|_\infty \le 1\}.$$

As observed, matrix $A(\cdot)$ is an affine function of the scheduling variables and matrix B is constant.

4.1.1 System identification and results

We use slowly varying trajectories in Figure 2(a) for the scheduling variables to collect observations $\mathcal{D} = \{(\theta(i), x(i), u(i)), x(i+1)\}_{i=0}^{499}$ for system identification. Pseudo-random binary sequences (PRBS) of the inputs with an amplitude of 0.01 in Figure 2(b) are used to excite the system and the generated state sequence with initial state $x_0 = [2.7; 0]$ is shown in Figure 2(c,d). A total of 500 samples are collected and split into training and testing sets with a ratio of 80%/20%.

We use a DenseVariational layer with 4 hidden units to represent $A(\cdot)$ and a Dense layer with 2 hidden units to represent B. Neither of the layers use activation functions and the Dense layer further does not use bias, which aims to exactly represent the class of models to which (19) belongs. However, the input to the DenseVariational layer is θ while the input to the Dense layer is u to compute Bu, as B is constant. Adam optimiser is used with a learning rate set to 0.01 and the other hyper-parameters as default. We trained the BNN model for 1000 epochs and the validation results are shown in Figure 3. Only 2% of samples are out of $2\sigma(x_1)$ and 6% of samples are out of $2\sigma(x_2)$. By increasing β, the true states are guaranteed to lie in the interval $[\mu - \beta\sigma, \mu + \beta\sigma]$ almost surely.

4.1.2 Validation of safe MBRL using theBNN model

First, we tested the model-fee RL for 1000 steps. As shown in Figure 4, the system resets 62 times due to the violation of the constraints and most of the episodes last less than 40 steps, which shows the necessity of using model-based RL.

Technical details of DDPG: For the actor-network, a 5-layer fully-connected neural network is used. Each of the 3 hidden layers has 16 units and uses ReLU as the activation function while the output layer uses linear activation function. The critic network shares the same structure as the actor-network but each hidden layer has 32 units. The agent takes actions sampled from a uniform distribution over the action space for the first 100 steps to warm up the networks. Moreover, the Ornstein–Uhlenbeck process (see Uhlenbeck & Ornstein, 1930) is added to the action during training for exploration. The discount factor is set to $\gamma = 0.99$ and $\rho = 0.001$ for target network update. The maximal length of a roll-out is set to 200 and the

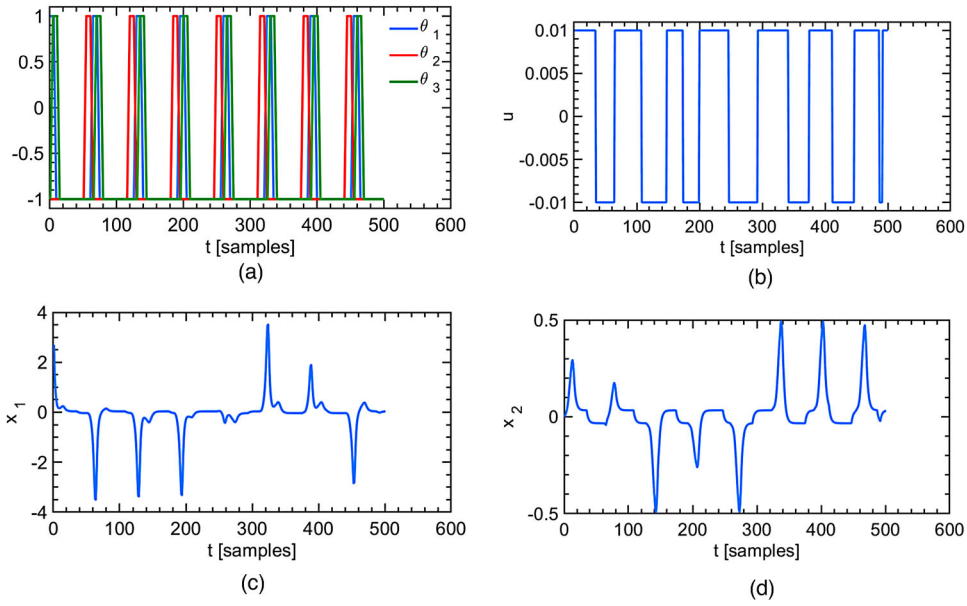

Figure 2. Data generated for model identification. (a) Scheduling trajectories θ^a; (b) inputs to the system; (c) sequence of x_1 and (d) sequence of x_2.

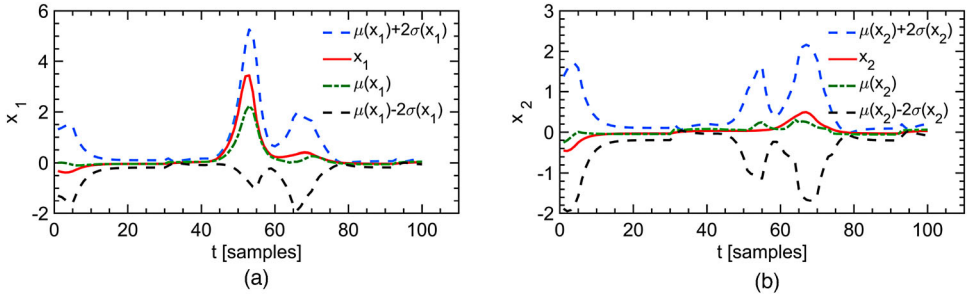

Figure 3. Validation results of system identification. The area between the two dashed lines is within two estimated standard deviations of the estimated mean, which is about 95% confidence interval. (a) Validation results of x_1 and (b) validation results of x_2.

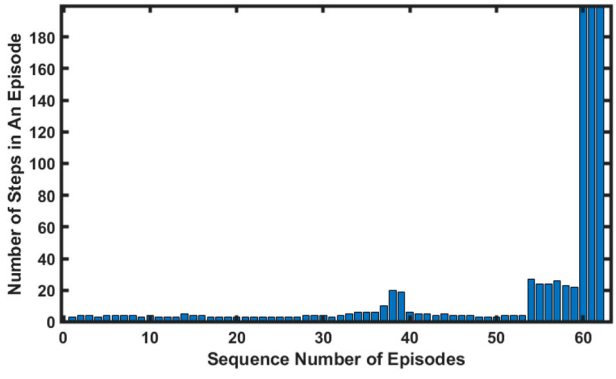

Figure 4. Number of time steps in episodes for the first 1000 interactions with the real system.

number of steps per epoch is 10, 000 . We trained the agent for 2 epochs with a limit of the sequential memory as 100,000 and a batch size as 32 using Adam optimiser with a learning rate set as 0.001. Moreover, 10 dynamic functions are sampled from the BNN model at each time instant to evaluate the next state.

Results and discussion: First, we assume the scheduling trajectories are known, which indicates that the true scheduling signals of the system can be used for simulation. We tested the control performance for both the scheduling signal θ^a in Figure 2(a) and θ^b in Figure 5(b) using the same BNN model identified in Section 4.1.1. From (19), θ^b in Figure 5(b) is easier to cause constraint violation than θ^a in Figure 2(a) at training. As shown in Figure 5(a,c), the constraints are satisfied but the control performance of the RL agent degrades as the scheduling signals change. This performance degradation results from the accuracy drop of the BNN model which is identified for the θ^a in Figure 2(a) and the difficulty for the agent to adapt to environments with θ^b in Figure 5(b).

BNN model update: Since the trained RL agent can interact with the system safely, we collect new observations to update the BNN model and improve the control performance. Using the learned policy, we collected 5000 observations and fine-tuned the BNN model for 1000 epochs using the same hyper-parameters as in Section 4.1.1. Then, the agent updated the policy by learning in the simulation environments that are created by the updated BNN. The control result is shown in Figure 5(d). As observed, the overshooting is moderated and the

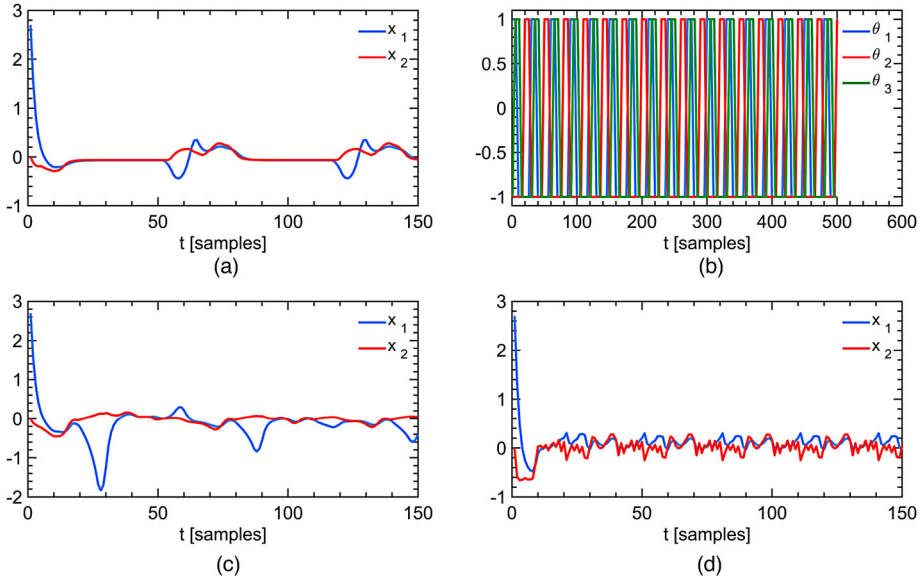

Figure 5. Control results using model-based RL with known scheduling trajectories. (a) When the scheduling signals of the system are the same as the trajectory for model identification; (b) scheduling signals θ^b that are different from the trajectory for model identification; (c) when the scheduling signals of the system are θ^b in (b); (d) when the scheduling signals of the system are θ^b in (b) and the updated BNN model is used.

oscillation around the origin is in a smaller range, compared to Figure 5(c).

4.1.3 Uncertainties in the evolution of scheduling variables

When the scheduling signals are unknown, we randomly generate scheduling variables and sample dynamic functions from the updated BNN to estimate the next states at each time instant. The scheduling signal for training is shown in Figure 6(a). We trained the agent for 4 epochs. Figure 6(b,c) shows that the control performance for Figure 2 degrades and the agent cannot regulate (stabilise) the system with θ^b in Figure 5(b), as the joint uncertainty of scheduling variables and system models caused significant discrepancy between the transition distributions of the training environments and the real environment. To reduce the joint uncertainty, we use the knowledge of the ROV of the scheduling variables and $\delta_{ROV} = 0.5$ in (3) for both θ^a in Figure 2(a) and θ^b in Figure 5(b). Specifically, for every 20 time steps, we randomly generate a θ, and then, we use the ROV to calculate the range of the next θ and randomly select one from the linear space of the range with 20 elements. The agent was trained for 2 epochs. Figure 7 shows that the usage of the ROV knowledge significantly improves the control performance.

4.1.4 Adaptation to new scheduling signals

Since LPV models depend on the scheduling variables, the environments change as the scheduling trajectory changes and the RL agent trained in the previous environments can perform worse in the new environment and even fail to work. As shown in Figure 8(a,b), the control performance degrades when the agent transfers from slowly varying environments to fast-varying environments while the performance improves when the agent transfers from fast-varying environments to slowly varying environments, which is even better than the performance of the agent specially trained in the slowly varying environments as shown in Figure 5(a). This motivated us to use a scheduling trajectory that varies faster than the true scheduling trajectory in practice to guarantee performance.

4.2 Control moment gyroscope

In this section, we validate the proposed methods using experiments on a complex 4 degree-of-freedom control moment gyroscope (CMG) simulation model from Parks (1999). The detailed plant description can be found in Abbas et al. (2014) and the coordinate frames are shown in Figure 9. The states of the simulation model consist of angles q_i and angular speeds $\omega_i, i = 1, 2, 3, 4$ of the 4 gimbals, and it is generally required to control q_3 and q_4 using torques τ_1 and τ_2 provided by two dc motors. Moreover, the authors in Abbas et al. (2014) show that the nonlinear model of CMG can be linearised around a moving operating point ω_1 and converted into an LPV state-space representation, which proves to be accurate and suitable for control design. The states, inputs, outputs and scheduling variables of the obtained LPV model are

$$\begin{aligned} x &= [q_3\ q_4\omega_2\ \omega_3\ \omega_4]^{\mathrm{T}}, \\ u &= [\tau_1\ \tau_2]^{\mathrm{T}}, \\ \theta &= [q_2\ q_3]^{\mathrm{T}}. \end{aligned} \tag{20}$$

Additionally, the controller is designed to be in the static state-feedback form with parameter-varying gain, i.e. $u = K(\theta)x$, where K is a smooth nonlinear matrix function represented by NN as Bao and Velni (2021).

4.2.1 System identification and results

Band-Limited White Noise (BLWN) with noise power [0.09; 0.7] is used to excite the system. A total of 95,000 samples are collected with a sampling frequency of 0.1 kHz. The first 65,000 samples are used as the training data and the rest for testing.

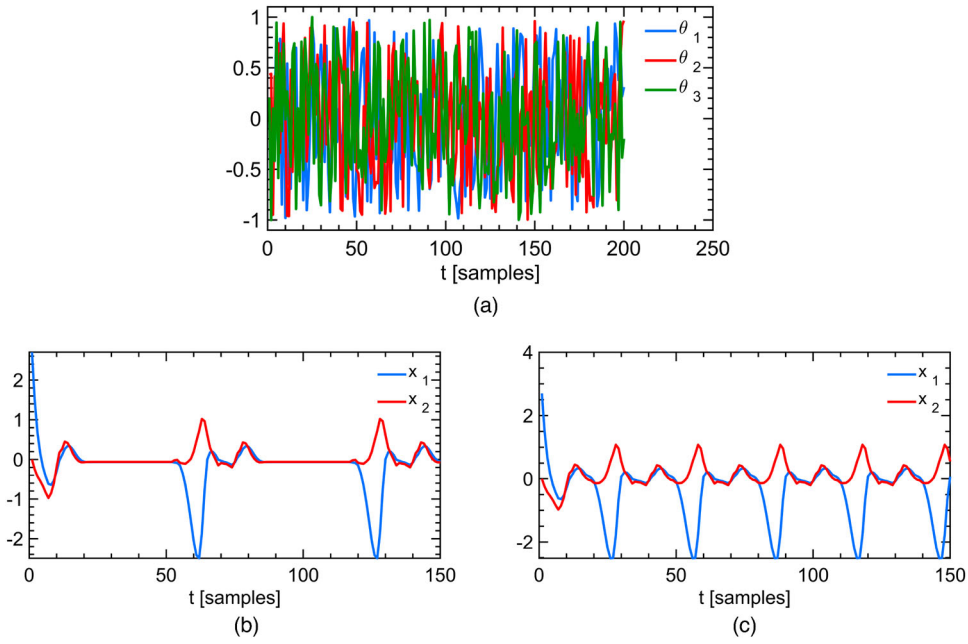

Figure 6. Control results using model-based RL with unknown scheduling signals. (a) Random scheduling signals for simulation when the evolution of the scheduling variables of the system is unknown; (b) when the scheduling signals are θ^a in Figure 2(a) and (c) when the scheduling signals are θ^b in Figure 5(b).

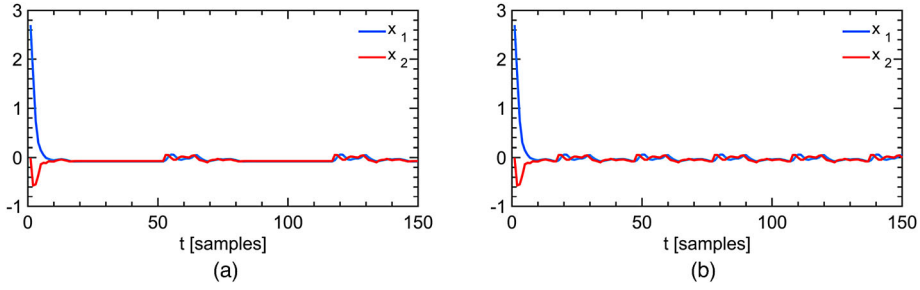

Figure 7. Control results using model-based RL and prior knowledge on the ROV of the scheduling signals. (a) Control results of using the knowledge of ROV when the scheduling signals are θ^a in Figure 2(a) and (b) control results of using the knowledge of ROV when the scheduling signals are θ^b in Figure 5(b).

We use a BNN composed of 3 DenseVariational layers with respective hidden units $\{4, 16, 25\}$ to represent $A(\cdot)$ and an ANN composed of 3 Dense layers with respective hidden units $\{4, 16, 10\}$ to represent $B(\cdot)$. It is noted that both A and B can be represented by BNNs. However, as discussed in Bao et al. (2021), using BNNs to represent both A and B increases not only the expressiveness of the LPV model but also the computational cost and the convergence efficiency of BNN training. It is more reasonable to only represent A with BNNs, as A has a larger impact on the system description than other matrix functions. Therefore, in this example, we consider using BNNs only to represent A. The first two hidden layers of both the BNN and ANN

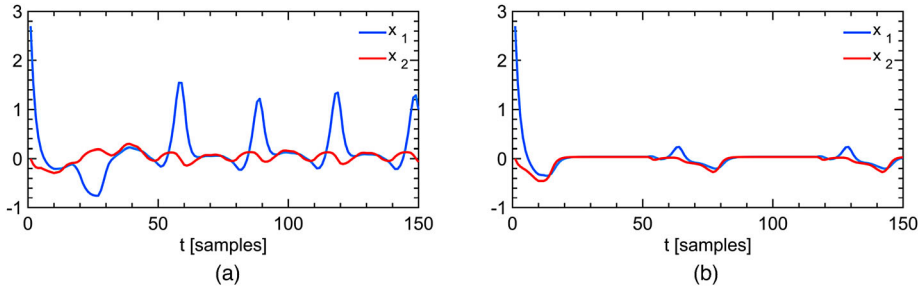

Figure 8. Control results of adaptation to new scheduling signals using model-based RL. (a) Applying the policy learned for scheduling signals in 2(a) to the environments with the signals in Figure 5(b) and (b) applying the policy learned for scheduling signals in 5(b) to the environments with the signals in Figure 2(a).

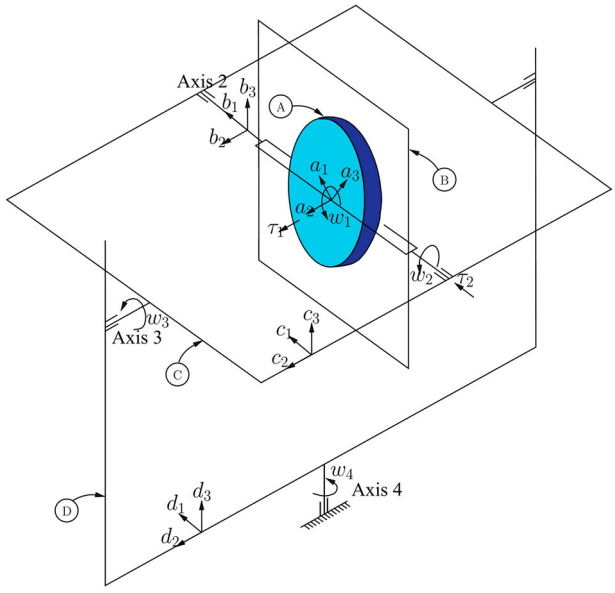

Figure 9. Coordinate frames of CMG (Parks, 1999).

use Exponential Linear Units[8] (ELU) first introduced in Clevert et al. (2015) as activation functions and the output layers do not use activation function. Adam optimiser is used with a learning rate set to 0.001 and the other hyper-parameters as default. We trained the BNN model for 5000 epochs and the validation results are shown in Figure 10. No samples are out of 2σ of q_3 and q_4, which shows that the models sampled from the learned BNN model contain the real model.

4.2.2 Network architecture and hyperparameters

An MLP with two hidden layers is used to model $K(\theta)$. Each of the hidden layers contains 512 units, uses ReLU as the activation function and is followed by a batch normalisation layer[9] to normalise different physical units of features. The output layer uses a linear activation function and the output of this MLP is multiplied by x to constitute the policy network. For Q network, the state s is transformed by an MLP with one 16-unit layer followed by one 32-unit layer and action a is transformed by an MLP with one 32-unit hidden layer. The transformed s and a are concatenated and then transformed by an MLP with

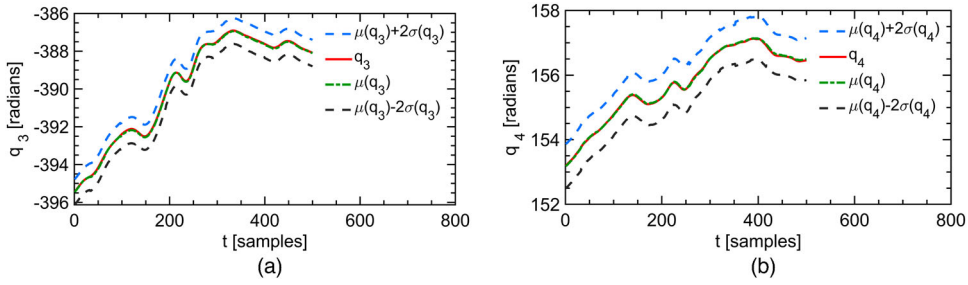

Figure 10. Validation results of system identification. For the sake of clarity, only 500 of the testing data points are shown here. The area between the two dashed lines is within 2 estimated standard deviations of the estimated mean, which is about 95% confidence interval. (a) Validation results of q_3 and (b) validation results of q_4.

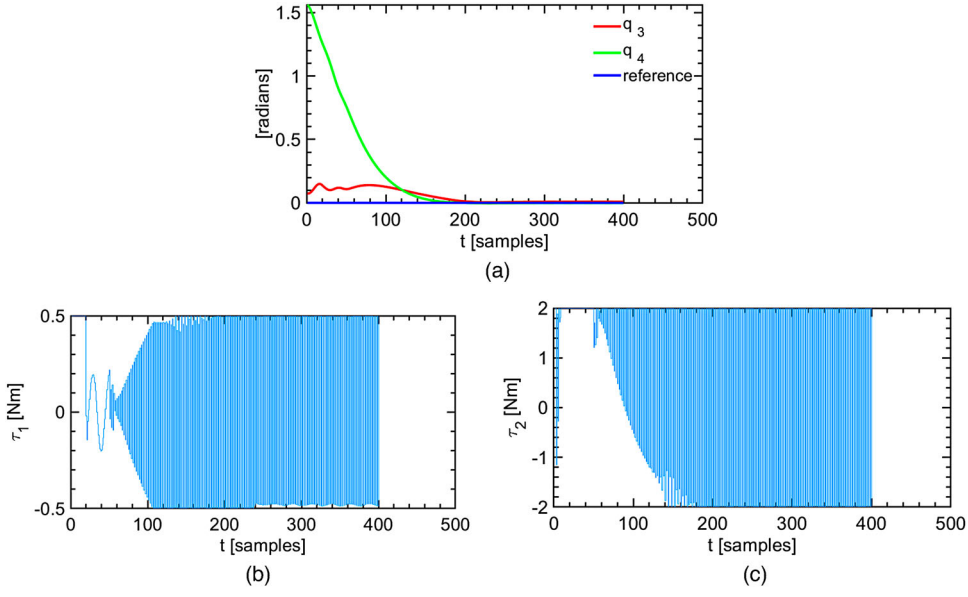

Figure 11. Control results for the CMG platform. (a) State trajectory of the controlled CMG platform; (b) control input τ_1 to the CMG platform and (c) control input τ_2 to the CMG platform.

two 512-unit hidden layers to estimate Q value. All the hidden layers are followed by a batch normalisation layer. Gaussian noise $\mathcal{N}(0, 0.03u(k))$ is added to an initial control sequence $\{u(k)\}_{k=1}^{T}$ from an MPC at time k for the first 40,000 steps to help with the exploration which is used in Bao and Velni (2021) to improve learning efficiency. Moreover, an Ornstein–Uhlenbeck process with initial standard deviation $\sigma = [0.1 \;\; 0.4]^{\mathrm{T}}$ is added to the action at training time for exploration. Also considered are the discount factor $\gamma = 0.99$ and $\rho = 0.001$ for target network update. The maximum length of an episode is 400. We trained the agent for 2800 episodes with a limit of the sequential memory as 50,000 and batch size as 64 using Adam optimiser with a learning rate as 10^{-4} for the Q network and 10^{-5} for the policy network. Additionally, the policy was applied to the simulation model every 100 training episodes to test the performance of the controller and determine whether to stop training.

4.2.3 Experimental settings, results and discussion

The state of environment s_t consists of $\{u(t+k-l), \theta(t+k-l), x(t+k-l), e(t+k-l)\}_{k=1}^{l}$, where $l = 4$ denotes the memory length and is used for the agent to implicitly build the process model (Spielberg et al., 2019). The immediate reward is chosen as $r_t = -\|e(t+1)\|_1$ and the control objective is to track reference $q_3 = q_4 = 0$.

Figure 11 shows the control results using the proposed MBRL design approach in the LPV framework. Despite the CMG platform being far more complex than the numerical example discussed earlier, very precise tracking performance was achieved in moderate training episodes. Additionally, the control inputs are shown in Figure 11(b,c).

5. Concluding remarks

In this paper, a safe model-based reinforcement learning (MBRL) approach was proposed to control nonlinear systems described using LPV models. BNNs were used to learn, from input–output data, an LPV-SS model with epistemic uncertainty quantification. Then, the epistemic uncertainty from the system identification and imprecise knowledge of the future scheduling variables were jointly considered for control design with safety guarantees. Model-based RL was proposed to learn policy in the simulation environments that are created by the BNN model rather than directly interacting with the real system. Knowledge on the evolution of the scheduling variables was incorporated into the simulation environments to reduce the joint uncertainty and adapt to varying environments. Numerical experiments show that the proposed approach can ensure safety and achieve desired control performance.

Notes

1. Refer to Tran et al., 2018 for the implementation of the DenseVariational layer.
2. $\odot$ denotes element-wise multiplication.
3. The data set $\mathcal{D}$ is randomly split into a training set for training a model and a testing set for testing the generalisation of the trained model.
4. *Almost surely* is used to avoid the analysis of Pr in (13) which involves the analysis of the closed-loop system and BNN models and is unnecessary for the proposed approach, although using the confidence level can decrease N_{MC}.
5. It is noted that, here, we only consider deterministic policies for control.
6. The total variation distance between two probability measures P and Q on a σ-algebra $\mathcal{F}$ of subsets of the sample space Ω is defined via $D_{TV}(P\|Q) = \sup_{A\in\mathcal{F}} |P(A) - Q(A)|$.
7. The Ornstein–Uhlenbeck process is a stationary Gauss–Markov process which tends to drift towards its mean function over time.
8. $$f(x) = \begin{cases} x & \text{if } x > 0, \\ \alpha(e^x - 1) & \text{if } x \le 0. \end{cases}$$
9. We refer to DDPG in Keras (Chollet, 2015) for implementing the proposed method.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was financially supported by The United States National Science Foundation under award #1762595.

References

Abbas, H. S., Ali, A., Hashemi, S. M., & Werner, H. (2014). LPV state-feedback control of a control moment gyroscope. *Control Engineering Practice*, *24*(1), 129–137. https://doi.org/10.1016/j.conengprac.2013.05.008

Akametalu, A. K., Fisac, J. F., Gillula, J. H., Kaynama, S., Zeilinger, M. N., & Tomlin C. J. (2014). *Reachability-based safe learning with Gaussian processes*. 53rd IEEE Conference on Decision and Control (pp. 1424–1431). IEEE. https://doi.org/10.1109/CDC.2014.7039601

Bao, Y., & Velni, J. M. (2021). *Model-free control design using policy gradient reinforcement learning in LPV framework*. 2021 European Control Conference (ECC) (pp. 150–155). IEEE. https://doi.org/10.23919/ECC54610.2021.9655004

Bao, Y., Velni, J. M., Basina, A., & Shahbakhti, M. (2020). *IFAC-PapersOnLine* (Vol. 53.2, pp. 5286–5291). IFAC.

Bao, Y., Velni, J. M., & Shahbakhti, M. (2020). *An online transfer learning approach for identification and predictive control design with application to RCCI engines*. Dynamic Systems and Control Conference (Vol. 84270, p. V001T21A003). American Society of Mechanical Engineers.

Bao, Y., Velni, J. M., & Shahbakhti, M. (2021). Epistemic uncertainty quantification in state-space LPV model identification using Bayesian neural networks. *IEEE Control Systems Letters*, *5*(2), 719–724. https://doi.org/10.1109/LCSYS.7782633

Becker, C. O., & Preciado, V. M. (2019). *Variational inference for linear systems with latent parameter space*. In 2019 American Control Conference (ACC) (pp. 5662–5667). IEEE.

Berkenkamp, F., Turchetta, M., Schoellig, A., & Krause, A. (2017). *Safe model-based reinforcement learning with stability guarantees*. Advances in Neural Information Processing Systems (pp. 908–918). NeurIPS.

Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, *112*(518), 859–877. https://doi.org/10.1080/01621459.2017.1285773 http://dx.doi.org/10.1080/01621459.2017.1285773 .

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). *Weight uncertainty in neural network*. International Conference on Machine Learning (pp. 1613–1622). ICML.

Chen, S., Saulnier, K., Atanasov, N., Lee, D. D., Kumar, V., Pappas, G. J., & Morari, M. (2018). *Approximating explicit model predictive control using constrained neural networks*. 2018 Annual American Control Conference (ACC) (p. 1520-1527). IEEE.

Cheng, R., Orosz, G., Murray, R. M., & Burdick, J. W. (2019). *End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks*. Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, pp. 3387–3395). AAAI Press.

Chollet, F. (2015). Keras. https://keras.io.

Clevert, D. A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (ELUS). arXiv preprint arXiv:1511.07289.

Deisenroth, M., & Rasmussen, C. E. (2011). *PILCO: A model-based and data-efficient approach to policy search*. Proceedings of the 28th International Conference on Machine Learning (ICML-11) (pp. 465–472). ICML.

Deisenroth, M. P., Neumann, G., & Peters, J. (2013). *A survey on policy search for robotics*. Now Publishers.

Gal, Y., McAllister, R., & Rasmussen, C. E. (2016). *Improving PILCO with Bayesian neural network dynamics models*. Data-Efficient Machine Learning Workshop, ICML (Vol. 4, p. 34). ICML.

Gros, S., & Zanon, M. (2020). Safe reinforcement learning with stability & safety guarantees using robust MPC. arXiv preprint arXiv:2012.07369.

Gros, S., Zanon, M., & Bemporad, A. (2020). Safe reinforcement learning via projection on a safe set: How to achieve optimality? *IFAC-PapersOnLine*, *53*(2), 8076–8081. https://doi.org/10.1016/j.ifacol.2020.12.2276

Hanema, J. (2018). *Anticipative model predictive control for linear parameter-varying systems* [Unpublished doctoral dissertation]. Technische Universiteit Eindhoven.

Hanema, J., Lazar, M., & Tóth, R. (2020). Heterogeneously parameterized tube model predictive control for LPV systems. *Automatica*, *111*(2020) 108622. https://doi.org/10.1016/j.automatica.2019.108622

Hastie, T., Tibshirani, R., & Friedman, J. (2009). Model Assessment and Selection. In: The Elements of Statistical Learning. Springer Series in Statistics. Springer, New York, NY. https://doi.org/10.1007/978-0-387-84858-7_7

Hewing, L., Wabersich, K. P., Menner, M., & Zeilinger, M. N. (2020). Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, *3*(1), 269–296. https://doi.org/10.1146/control.2020.3.issue-1

Higuera, J. C. G., Meger, D., & Dudek, G. (2018). *Synthesizing neural network controllers with probabilistic model-based reinforcement learning*. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 2538–2544). IEEE.

Janner, M., Fu, J., Zhang, M., & Levine, S. (2019). *When to trust your model: Model-based policy optimization*. Advances in Neural Information Processing Systems (pp. 12519–12530). NeurIPS.

Karl, M., Soelch, M., Bayer, J., & van der Smagt, P. (2016). Deep variational Bayes filters: Unsupervised learning of state space models from raw data. *arXiv preprint arXiv:1605.06432*.

Koller, T., Berkenkamp, F., Turchetta, M., & Krause, A. (2018). *Learning-based model predictive control for safe exploration*. 2018 IEEE Conference on Decision and Control (CDC) (pp. 6059–6066). IEEE.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

Padakandla, S. (2020). A survey of reinforcement learning algorithms for dynamically varying environments. arXiv preprint arXiv:2005.10619.

Parks, T. R. (1999). *Manual for model 750: Control moment gyroscope*. Educational Control Products.

Peng, X. B., Andrychowicz, M., Zaremba, W., & Abbeel, P. (2018). *Sim-to-real transfer of robotic control with dynamics randomization*. 2018 IEEE International Conference on Robotics and Automation (ICRA) (pp. 1–8). IEEE.

Reid, M. D., & Williamson, R. C. (2009). Generalised Pinsker inequalities. *arXiv preprint arXiv:0906.1244*.

Rizvi, S. Z., Mohammadpour Velni, J., Abbasi, F., Tóth, R., & Meskin, N. (2018). State-space LPV model identification using kernelized machine learning. *Automatica*, *88*(9), 38–47. https://doi.org/10.1016/j.automatica.2017.11.004

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). *Deterministic policy gradient algorithms*. International Conference on Machine Learning (pp. 387–395). ICML.

Spielberg, N. A., Brown, M., Kapania, N. R., Kegelman, J. C., & Gerdes, J. C. (2019). Neural network vehicle models for high-performance automated driving. *Science Robotics*, *4*(28), 425. https://doi.org/10.1126/scirobotics.aaw1975

Tran, D., Dusenberry, M. W., van der Wilk, M., & Hafner, D. (2018). Bayesian layers: A module for neural network uncertainty. CoRR abs/1812.03973. http://arxiv.org/abs/1812.03973

Uhlenbeck, G. E., & Ornstein, L. S. (1930). On the theory of the Brownian motion. *Physical Review*, *36*(5), 823–841. https://doi.org/10.1103/PhysRev.36.823

Yu, M., Yang, Z., Kolar, M., & Wang, Z. (2019). *Convergent policy optimization for safe reinforcement learning*. Advances in Neural Information Processing Systems (pp. 3127–3139). NeurIPS.