International Journal of Control

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

A learning- and scenario-based MPC design for nonlinear systems in LPV framework with safety and stability guarantees

Yajie Bao, Hossam S. Abbas & Javad Mohammadpour Velni

To cite this article: Yajie Bao, Hossam S. Abbas & Javad Mohammadpour Velni (2023): A learning- and scenario-based MPC design for nonlinear systems in LPV framework with safety and stability guarantees, International Journal of Control, DOI: 10.1080/00207179.2023.2212814

To link to this article: https://doi.org/10.1080/00207179.2023.2212814

INTERNATIONAL JOURNAL OF CONTROL
https://doi.org/10.1080/00207179.2023.2212814

Taylor & Francis
Taylor & Francis Group

A learning- and scenario-based MPC design for nonlinear systems in LPV framework with safety and stability guarantees

Yajie Bao[a], Hossam S. Abbas[b] and Javad Mohammadpour Velni[c]

[a]School of Electrical & Computer Engineering, The University of Georgia, Athens, GA, USA; [b]Institute for Electrical Engineering in Medicine, University of Lübeck, Lübeck, Germany; [c]Department of Mechanical Engineering, Clemson University, Clemson, SC, USA

ABSTRACT

This paper presents a learning- and scenario-based model predictive control (MPC) design approach for systems modelled in the linear parameter-varying (LPV) framework. Using input-output data collected from the system, a state-space LPV model with uncertainty quantification is first learned through the variational Bayesian inference Neural Network (BNN) approach. The learned probabilistic model is assumed to contain the true dynamics of the system with a high probability and is used to generate scenarios that ensure safety for a scenario-based MPC. Moreover, to guarantee stability and enhance the performance of the closed-loop system, a parameter-dependent terminal cost and controller, as well as a terminal robust positive invariant set are designed. Numerical examples will be used to demonstrate that the proposed control design approach can ensure safety and achieve desired control performance.

ARTICLE HISTORY
Received 7 July 2022
Accepted 4 May 2023

KEYWORDS
Safe scenario-based model predictive control; learning-based control design; linear parameter-varying framework; Bayesian neural networks

1. Introduction

Model predictive control has been widely used to control a given process while satisfying a set of constraints and found applications in various domains including vehicular technology (Luo et al., 2021) and chemical processes (Ellis et al., 2017). Furthermore, learning-based model predictive control (L-MPC) has increasingly received interest for control of complex safety-critical systems operating in uncertain and hard-to-model environments (Aswani et al., 2013; Hewing et al., 2020; Koller et al., 2018). Uncertainties and hard-to-model dynamics of the environments are learned from data for L-MPC to improve the control performance and guarantee constraints satisfaction. However, the statistical nature of learning-based methods brings new challenges including the generalisability of the learned models and the computational complexity involved in the MPC design (Bonzanini et al., 2020; Mesbah, 2018).

Suitable and sufficiently accurate model representations of the system dynamics are crucial to MPC performance, and distributional information on the uncertainties reduces the conservativeness of compact uncertainty sets. Gaussian process (GP) regression is a commonly used non-parametric learning method to identify residual model uncertainty, which provides a point-wise approximation of unknown errors with mean and covariance characterisation. However, GP suffers from high computational complexity which grows with the number of recorded data points. Additionally, GP-based MPC design faces the challenge of uncertainty propagation (i.e. the propagation of the resulting stochastic state distributions) over the MPC prediction horizon. This problem becomes even more exacerbated when the known nominal part of the system model is nonlinear. To address this, Koller et al. (2019) linearised the nominal part for uncertainty propagation.

Linear parameter-varying (LPV) models use a linear structure to capture time-varying and nonlinear dynamics of a system with system matrices dependent on so-called *scheduling variable(s)*, and this allows developing computationally efficient design methods (Hanema, 2018). Nonlinear systems can be embedded in LPV representations (Abbas et al., 2021); as an example, linear switching systems and Markov jump linear systems can be viewed as particular cases of LPV systems with the scheduling variables being a switching sequence and a Markov chain, respectively. Moreover, learning-based methods for the global identification of state-space LPV (LPV-SS) models with arbitrary scheduling dependency using input-output data have been developed (Bao, Velni et al., 2020), and a variational Bayesian inference Neural Network (BNN) approach (Bao et al., 2021) has been proposed to quantify uncertainties in state-space LPV model identification of nonlinear systems, which provides a posterior density estimation of the system model parameters given an input-output dataset.

Different LPV-MPC design schemes given system models have been recently surveyed in Morato et al. (2020). One challenge with the MPC design in the LPV framework lies in the unknown future evolution of the LPV scheduling variables over the prediction horizon. Two main approaches have been considered in the literature to handle this difficulty: min-max MPC formulation over all possible scheduling trajectories (Lee & Yu, 1997), and tube-based design, where possible future trajectories are exploited to reduce the uncertainty in the scheduling variables evolution. Hanema et al. (2020) proposed a

CONTACT Yajie Bao yb18054@uga.edu

heterogeneously parameterised tube-based MPC approach with recursive feasibility and stability guarantees for LPV systems without considering uncertainty in system models and disturbance. Moreover, Calafiore and Fagiano (2013) proposed a scenario-based MPC for constrained discrete-time LPV models with bounded scheduling dependency and stochastic scheduling variables. However, Calafiore and Fagiano (2013) assumed that the terminal control law associated with the terminal set of the system model is given, which is not practical. Therefore, the existing LPV-MPC approaches are not directly applicable to learning-based LPV models, due to the high complexity of the learned models with arbitrary scheduling dependency and the complex joint uncertainty in the learned models and the scheduling variables.

In this paper, we assume no true system model is available, but input-output data are given. For data-driven LPV-SS model identification using only input-output data, the majority of the current LPV identification methods, including direct prediction-error minimisation (PEM) methods, as well as global subspace and realisation-based techniques (SID), assume an affine scheduling dependency with known basis functions, which restricts the complexity of a representation (Cox, 2018). Rizvi et al. (2018) used kernelized canonical correlation analysis (KCCA) to estimate the state sequence and then a least-squares support vector machine (LS-SVM) to capture the dependency structure, which suffers from the kernel function selection and computational complexity. The expectation-maximisation algorithms estimate states and matrices alternatively (Wills & Ninness, 2012). To simultaneously estimate states and explore LPV model structural dependency, Bao, Velni et al. (2020) presented an integrated architecture of artificial neural networks (ANNs). However, the aforementioned methods focus on estimating a set of deterministic parameters rather than characterising the statistical properties of the estimation, which typically produce good models in the sense of minimising the expected loss. However, the accuracy under a few operating points can be poor, which can later result in a low-performing controller and safety violation. Furthermore, robust control techniques cannot be employed without quantifying the uncertainty of the estimated model. Gaussian process (GP) has been used to quantify model uncertainty but suffers from cubic complexity to data size and assumes joint Gaussian distributions to describe uncertainties (Liu et al., 2020). Instead, BNNs can provide a fast evaluation of uncertainties after training and approximate arbitrary posterior distributions. Bao et al. (2021) proposed a BNN training approach based on Bao, Velni et al. (2020) to quantify the epistemic uncertainty in the ANN model. In this paper, we employ the proposed BNN architecture to identify an LPV model with uncertainty quantification for robust estimation and control purposes.

Moreover, the quantified uncertainties in the learned LPV model and the future scheduling trajectory will be considered simultaneously for L-MPC design. In particular, based on the characterisation of the joint uncertainties, we construct tubes of linear models that contain the true system dynamics almost surely. To ensure safety, the system constraints are enforced by considering the worst case in the model tube. Additionally, chance constraints can be handled by adjusting the tube based on the BNN model. Furthermore, we use the expectation of the cost over the possible system trajectories as the cost function. *Scenario-based MPC* (SMPC) is adopted here to approximate analytically intractable evolution of uncertainties and improve online computational efficiency. Several methods of scenario generation for SMPC have been proposed in the literature, including Monte Carlo (MC) sampling methods (Shapiro, 2003), moment matching methods (Høyland et al., 2003), and even machine learning techniques (Defourny, 2010). Despite these efforts, the existing methods are typically only applied to convex problems and assume full recourse. Since BNNs are evaluated using MC methods, a straightforward approach for scenario generation is to use models drawn from the posterior distributions as scenarios. However, the number of models required for safety guarantees can be too large for online optimisation of the SMPC design. To reduce the number of scenarios, Bao et al. (2022) used $\hat{\mu}_{\hat{\mathrm{y}}(k)}$, $\hat{\mu}_{\hat{\mathrm{y}}(k)} + a^j\hat{\sigma}_{\hat{\mathrm{y}}(k)}, \hat{\mu}_{\hat{\mathrm{y}}(k)} - a^j\hat{\sigma}_{\hat{\mathrm{y}}(k)}, j = 1, \ldots, \frac{n_s-1}{2}$ where $\hat{\mu}_{\hat{\mathrm{y}}(k)}$ and $\hat{\sigma}_{\hat{\mathrm{y}}(k)}$ are the sample mean and standard deviation of the predictions $\hat{\mathrm{y}}(k)$ of uncertainties at time step k by BNNs, and a^j's are the tuning multipliers and n_s is the number of scenarios at each node of a stage. However, Bao et al. (2022) used fixed a^j's for each time step, which may not well represent the distributions of uncertainties. In this work, we use K-means (Lloyd, 1982), a popular clustering method in machine learning with convergence guarantees, to quantize the sample models. In particular, we apply K-means clustering to possible system matrices at each time step to construct scenarios. Additionally, the distributions of the system matrices are estimated using the identified LPV model and the knowledge of the scheduling variables by Monte Carlo methods. With the scenarios generated using K-means, to maintain the original statistical properties of the system matrices distributions, the probability of the scenarios is estimated by a moment-matching optimisation method for matching the first four central moments.

Furthermore, to guarantee the stability of the closed-loop system and the recursive feasibility of the associated MPC optimisation problem, we present a learning-based approach for terminal ingredients design. In particular, we transform the BNN model with nonlinear scheduling dependency into an LPV form with affine scheduling dependency and compute a terminal constraint as a robust positive invariant set and a terminal cost as a parameter-dependent poly-quadratic Lyapunov function using related LPV tools (Pandey & de Oliveira, 2017) based on the transformed model. The latter is computed by solving a linear matrix inequality (LMI) problem corresponding to the extreme realisations of the scenarios, which provide a parameter-dependent terminal controller based on the generated scenarios that can improve the control performance. To the best of the authors' knowledge, this is the first work on learning-based terminal control design in the LPV framework that is applicable to general nonlinear systems using only the input-output data. Additionally, the BNN model can be updated online using the framework developed in Bao, Mohammadpour Velni et al. (2020) with new observations collected by applying the MPC law to the real system. The updated model is anticipated to better characterise the uncertainty of the system, which in turn reduces the conservativeness required to ensure safety and hence improve the control performance. Figure 1

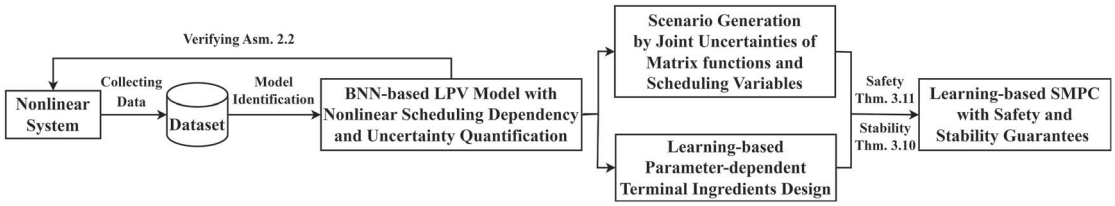

Figure 1. The flow chart of the overall learning-based MPC design procedure.

shows the flow chart of the overall learning-based SMPC design procedure.

The contributions of this paper are three-fold:

(1) We present a learning- and scenario-based robust MPC design approach. The proposed approach learns an LPV model with generally nonlinear scheduling dependency from data and thus is applicable to a broader class of nonlinear systems than the existing works that assume a given LPV model with affine scheduling dependency. The proposed approach tackles the challenges of model bias for learning-based control by uncertainty quantification using BNNs and robust control using scenario optimisation. The proposed approach can also handle the joint uncertainties of the matrix functions and scheduling variables in the LPV model while most existing LPV-MPC works only consider the uncertainty of scheduling variables.
(2) We present a learning-based terminal ingredient design for scenario-based MPC using BNN models in the LPV framework. The proposed design reduces conservativeness by considering parameter-dependent terminal ingredients while most existing works consider static terminal ingredients. The proposed design is applicable to LPV models with nonlinear scheduling dependency while most existing works only assume affine scheduling dependency.
(3) We provide safety and stability guarantees for the proposed MPC scheme.

The challenges of the proposed approach lie in learning a sufficiently accurate model from data and reducing the conservativeness of uncertainty quantification and scenario generation for control design purposes. The remainder of the paper is organised as follows: Section 2 describes the problem formulation and preliminaries. Scenario-based MPC design approach using identified BNN models is presented in Section 3. Section 4 then presents numerical results to validate the proposed learning-based control design method. Concluding remarks are finally made in Section 5.

2. Preliminaries

2.1 Basic definitions

A set with a non-empty interior that contains the origin is called a proper set, and a proper set that is also compact and convex is called a PC-set. In data-driven methods, a dataset is randomly split into a *training set* for training a model and a *testing set* for testing the generalisation of the trained model. In probability theory, an event is said to happen *almost surely* if it happens with probability 1 (or Lebesgue measure 1). A function $f : \mathbb{R}_+ \to \mathbb{R}_+$ is of class $\mathcal{K}_\infty$ if it is continuous, strictly increasing, $f(0) = 0$, and $\lim_{\xi\to\infty} f(\xi) = \infty$. A variable θ is said to evolve according to a bounded rate-of-variation (ROV) if for all time samples $k \in \mathbb{N}$, there exists a δ such that $|\theta(k+1) - \theta(k)| \le \delta$.

2.2 Problem formulation

We consider a constrained discrete-time nonlinear system represented by

$$x(k+1) = f(x(k), u(k)) \tag{1a}$$
$$x(k) \in \mathbb{X}, \quad u(k) \in \mathbb{U}, \quad k \in \mathbb{N}, \tag{1b}$$

where $f(\cdot)$ is an unknown nonlinear function, $x(k)$ and $u(k)$ denote the states and control inputs at time sample k, respectively. $\mathbb{X} \subseteq \mathbb{R}^{n_x}$ and $\mathbb{U} \subseteq \mathbb{R}^{n_u}$ are the input and state constraint sets. We can embed the nonlinear representation (1) into the following discrete-time state-space LPV representation

$$x(k+1) = A(\theta(k))\,x(k) + B(\theta(k))\,u(k), \tag{2}$$
$$x(k) \in \mathbb{X}, \quad u(k) \in \mathbb{U}, \quad k \in \mathbb{N}, \tag{3}$$

where $\theta(k) \in \Theta \subseteq \mathbb{R}^{n_\theta}$ denotes the scheduling variables at time sample k. The scheduling variables are (nonlinear) functions of inputs/states, but are converted into an exogenous signal by confining the values of θ to some suitable set Θ such that the associated set of admissible trajectories (i.e. the set of input and output signals that are compatible with the dynamics) of (2) is a superset of the set of trajectories of the original nonlinear system (1a) (Hanema, 2018). Furthermore, A and B are smooth nonlinear matrix functions of $\theta(k)$. $x(k)$ and $\theta(k)$ can be measured at every time instant k. $\mathbb{X}$ and $\mathbb{U}$ are assumed to be PC-sets. Additionally, it is assumed that the future behaviour of θ is not known exactly at time instant k and that the matrix functions $A(\cdot)$ and $B(\cdot)$ are unknown.

Given an initial state x_0, a scheduling signal $\theta : \mathbb{N} \to \Theta$, and a control law $\kappa : \mathbb{X} \times \Theta \times \mathbb{N} \to \mathbb{U}$, the closed-loop system can be described by

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

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) with the representation (2) (resp. a data-driven model).

Definition 2.1: Given an initial state $x_0 \in \mathbb{X}$, the system (2) is said to be **safe** under a control law κ if

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

Moreover, the system (2) is said to be $\boldsymbol{\delta}$**-safe** under the control law κ if

$$\Pr\left[\forall k \in \mathbb{N} : \Phi_\kappa(x(k), \theta(k), k) \in \mathbb{X}, \kappa(x(k), \theta(k), k) \in \mathbb{U}\right] \geq \delta \quad (6)$$

where $0 \leq \delta \leq 1$, and $\Pr[\cdot]$ denotes the probability of an event.

In general, (5) cannot be enforced without additional assumptions (Koller et al., 2018) especially when (2) is unknown. Furthermore, δ-safety relaxes the requirements of safety to *safety with a high probability*. The problem addressed in this paper is to design a learning-based model predictive controller $\kappa : \mathbb{X} \times \Theta \times \mathbb{N} \to \mathbb{U}$ using a dataset $\mathcal{D} = \{(\theta(k), x(k), u(k)), x(k+1)\}_{k=1}^{N_\mathcal{D}}$ collected from the system, which yields $x(k) \to 0$ as $k \to \infty$ with the constraints (6) to be satisfied. First, we briefly describe the proposed probabilistic approach to identify the state-space LPV (LPV-SS) model of the system using the available dataset $\mathcal{D}$.

2.3 LPV-SS model identification using BNN

The data-driven LPV model identification problem is to learn matrix functions $A(\cdot)$ and $B(\cdot)$ from the dataset $\mathcal{D}$. To model arbitrary scheduling dependency and have parametric representations of the system, Bao, Velni et al. (2020) used fully-connected ANNs to represent $A(\cdot)$ and $B(\cdot)$, and learned the parameters of the ANNs by minimising the mean squared error (MSE) of the predictions of the ANN model. To quantify the epistemic uncertainty in the ANN model for robust estimation and control, Bao et al. (2021) treated the parameters of the matrix functions represented by ANNs as random variables and learned the posterior distributions of the parameters by BNNs (Blundell et al., 2015) composed of DenseVariational layers to represent the matrix functions.

A BNN approximates 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)

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

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 jth 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., 2017Feb). To approximate the posterior $p(w_j|\mathcal{D})$, VI solves

$$\min_{\vartheta_j} \text{KL}\left(q(w_j; \vartheta_j) \| p(w_j|\mathcal{D})\right) \quad (8)$$

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

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

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., 2017Feb). To solve (9) by Monte Carlo (MC) methods and backpropagation, a reparameterisation trick is used to parameterise $q(w_j; \vartheta_j)$, i.e. $w_j = \mu_{w_j} + \sigma_{w_j} \bigodot \epsilon_{w_j}$ where $\bigodot$ denotes the element-wise multiplication, $\epsilon_{w_j} \sim \mathcal{N}(0, I)$, 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.

Figure 2 shows how a BNN is used to represent $A(\theta)$; $B(\theta)$ 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

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

where f^w can be learned by minimising

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

over ϑ using the dataset $\mathcal{D}$ where $w^{(i)}$ is the ith sample generated by MC for approximating the ELBO, and N_{BNN} is the MC sample size determined such that (10) is convergent. Furthermore, as the discussion of the trade-off between bias and variance (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 compromises the convergence efficiency of the BNN training. It is hence reasonable to only represent A with BNNs and still represent B with ANNs, as A has a larger impact on the system description than other matrix functions. Therefore, in this paper, we consider using BNNs only to represent the matrix A, but the proposed approaches can be easily extended to the case where both A and B are represented by BNNs.

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. To provide safety guarantees, we need reliable estimates of the state x inside the operating region $\mathbb{X} \times \mathbb{U}$, which is similar to Bao and Mohammadpour Velni (2022) and formally described in the following assumption:

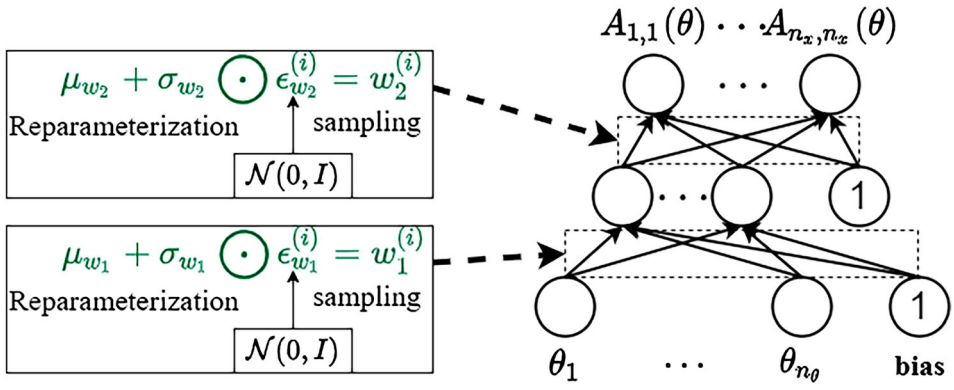

Figure 2. Using a BNN composed of multiple (here, two) DenseVariational layers to represent $A(\cdot)$ with reparameterisation trick. Here, the input to the BNN is θ and the output is the vectorised $A(\theta)$, which once reshaped, provides the full matrix A. BNNs use data to learn the parameters μ_w and σ_w of the posterior density function.

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

$$\forall k \in \mathbb{N} : |x_j(k+1) - \hat{\mu}_{x_j(k+1)}| \leq \beta_j \hat{\sigma}_{x_j(k+1)} < |\mathbb{X}_j|, \quad j = 1, 2, \ldots, n_x, \tag{12}$$

given $(x(k), \theta(k), u(k))$ where $\hat{\mu}_{x_j(k+1)}$ and $\hat{\sigma}_{x_j(k+1)}$ respectively denote the estimated mean and standard deviation of the jth entry of $x(k+1)$ using the learned BNN model with Monte Carlo methods, and $|\mathbb{X}_j|$ is used to denote the range of valid x_j.

By Assumption 2.2, the learned model is sufficiently accurate such that the values of $x(k+1)$ of the system are contained in the confidence intervals of our statistical model. It is noted that a larger $\beta_j \hat{\sigma}_{x_j(k+1)}$ means larger uncertainties of the model and gives a more conservative estimate of $x_j(k+1)$ which overestimates the probability of constraints violation, reduces the feasible region of control inputs, and thus degrades control performance. If $\beta_j \hat{\sigma}_{x_j(k+1)} \geq |\mathbb{X}_j|$, the estimate is worse than random guess of $x_j(k+1)$, which is not useful for control. The above assumption can be enforced by a well-designed BNN trained on a sufficient dataset and empirically verified on the testing set after model training. Moreover, δ_p can be estimated as the relative frequency of the testing data that violates (12) given β. If Assumption 2.2 does not hold, the architecture of the BNN should be adjusted or more data should be collected for training to improve the accuracy of the BNN until the hypotheses of Assumption 2.2 are satisfied.

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

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

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

Proof: Let $k = 0$, $\hat{x}_0 = x_0$, as x_0 and $\theta(0)$ are known. Then, using Assumption 2.2, there exists an $N_{\mathrm{MC}}(0)$ at time step 0 such that

$$\mathbf{x}_j(1|x_0, \theta(0)) \in \left[\min_{1 \leq i \leq N_{\mathrm{MC}}(0)} \hat{\mathbf{x}}_j^{(i)}(1|x_0, \theta(0)), \max_{1 \leq i \leq N_{\mathrm{MC}}(0)} \hat{\mathbf{x}}_j^{(i)}(1|x_0, \theta(0))\right]$$

almost surely, $j = 1, \ldots, n_x$. Using induction, (13) is obtained as $N_{\mathrm{MC}} = \max_k N_{\mathrm{MC}}(k)$. ■

It is noted that *almost surely* is used in the proof to avoid the analysis of Pr in (13) which involves the analysis of the closed-loop system and the BNN models and is unnecessary for the proposed approach, although using confidence level can decrease N_{MC}. Lemma 2.3 guarantees that, with a high probability, the 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.

2.4 Scenario-based MPC design approach

Given the distribution of the uncertainties described by the BNN model, stochastic MPC can be used to stabilise the system at the origin. In particular, the objective of the stochastic MPC problem is to minimise

$$\mathbb{E}\left\{\sum_{i=0}^{N-1} \ell(x(i|k), u(i|k)) + V_f(x(N|k))\right\}, \tag{14}$$

where $\mathbb{E}$ denotes the expected value operator over the random matrix functions and scheduling variables, $\ell(\cdot)$ is the stage cost function, and $V_f(\cdot)$ is the terminal cost function. It is noted that the joint uncertainties of matrix functions and scheduling signals are propagated forward through the prediction model (10) over the prediction horizon and thus the closed-form probability density function of x is hard to derive. Therefore, the problem of optimising (14) with the BNN model is not directly solvable.

Scenario-based MPC (SMPC) assumes that the uncertainty of a system may be represented by a tree of discrete scenarios which facilitates multi-step ahead predictions and feasibility guarantees. As a sufficiently large number of independent uncertainty realisation paths by sampling and simulation can

represent system uncertainty, applying reduction techniques to the paths can obtain representative scenarios while preserving statistical properties (Xu et al., 2012) and reduce the computational complexity of SMPC. Any particular branch stemming from a node represents a particular scenario of an unknown, uncertain influence (e.g. from a disturbance or model error) (Lucia et al., 2013). To represent the trajectories generated by some number C scenarios, we adopt the notation $(x^j(i), u^j(i))$, where the addition of the superscript j indicates the particular scenario $j \in \{1, \ldots, C\}$.

The scenario-based optimal control problem for an uncertain system at time step k can then be formulated as follows

$$\min_{x^j, u^j} \quad \sum_{j=1}^{C} p^j \left[\sum_{i=0}^{N-1} \ell\left(x^j(i|k), u^j(i|k)\right) + V_f\left(x^j(N|k)\right) \right] \tag{15a}$$

$$\text{s.t.} \quad x^j(i+1|k) = f_A^w(\theta^j(i|k))x^j(i|k) + f_B^w(\theta^j(i|k))u^j(i|k), \tag{15b}$$

$$\left(x^j(i|k), u^j(i|k)\right) \in \mathbb{X} \times \mathbb{U}, \tag{15c}$$

$$x^j(0|k) = x(k), \tag{15d}$$

$$u^j(i|k) = u^l(i|k) \quad \text{if} x^{p(j)}(i|k) = x^{p(l)}(i|k), \tag{15e}$$

where p^j is the probability of the jth scenario, $\ell(x^j(i|k), u^j(i|k))$ is the stage cost, and $V_f(x^j(N|k))$ is the terminal cost for the trajectory of the jth scenario, N is the prediction horizon, and (15e) enforces a *non-anticipativity* constraint, which represents the fact that each control input that branches from the same parent node must be equal ($x^{p(j)}(i|k)$ is the parent state of $x^j(i+1|k)$). The non-anticipativity constraint is crucial to accurately model the real-time decision-making problem such that the control inputs do not anticipate the future (i.e. decisions cannot realise the uncertainty). The solution to this optimisation problem is used to generate the control law

$$\kappa\left(x(k)\right) = u^{0*}(0|k). \tag{16}$$

Given the structure of the scenario tree, it is crucial to generate appropriate scenarios at each stage of the optimisation to accurately represent the uncertainty of the system under consideration. Additionally, the computational cost of SMPC is proportional to the number of scenarios which is positively correlated with the coverage of the uncertainty space. Hence, the objective of constructing a scenario tree is to accurately approximate (14) with a relatively small number of scenarios.

To express the joint uncertainties of matrix functions and scheduling signals by scenario trees, we generate model paths by sampling the scheduling signals and simulating the BNN model and apply reduction techniques to the model paths for generating representative scenarios while preserving the statistical properties of uncertainty quantified by the BNN model. In particular, we use MC sampling methods and K-means, a clustering method in machine learning, to generate scenarios, as the uncertainties are described by a BNN model such that the propagation of uncertainties is intractable to analyse. In particular, MC methods are employed to sample models from the BNN model for selected scheduling trajectories. While Lemma 2.3 claims there exists a scalar N_{MC} such that the trajectories of the sampled N_{MC} models contain the system trajectory, N_{MC} can be too large for online optimisation of the SMPC problems. Instead, we apply K-means clustering to the N_{MC} models to reduce the number of scenarios. K-means clustering is a vector quantisation method which partitions N_{s} observations/samples $\{\mathrm{x}^{(i)}\}_{i=1}^{N_{\mathrm{s}}}$ into C disjoint clusters $\{S_c\}_{c=1}^{C}$ by minimising the within-cluster sum-of-squares variances (squared Euclidean distances) $\sum_{c=1}^{C} \sum_{\mathrm{x} \in S_c} \|\mathrm{x} - \mu_c\|^2$, and each cluster is described by the mean (a.k.a. *centroid*) of the samples in the cluster. We use the cluster centroids of the models sampled from the BNN model as the models of scenarios. However, the C scenarios may lose the property of the N_{MC} models in Lemma 2.3.

To incorporate a probabilistic safety certificate into the scenario generation, we add extra scenarios corresponding to the worst cases based on the N_{MC} models. Then, the system is safe under (16) if (15), where all the scenarios are subject to the constraints, is feasible. Details of the scenario generation with safety guarantees will be provided in the next section.

3. Scenario-based MPC design using the learned BNN models

In this section, we present the techniques employed to design learning-based SMPC for nonlinear systems in the LPV framework with safety and stability guarantees. First, K-means clustering for scenario generation based on the BNN model is presented. Then, the use of the moment-matching method to compute the probability of scenarios is described. Next, the SMPC problem and terminal ingredients are presented, and finally, conditions for the stability and safety guarantees are provided.

3.1 Proposed method for scenario generation

In this work, we consider both the uncertainty in the evolution of θ and the epistemic uncertainty from the learning-based modelling. The scenario tree is designed to cover the joint uncertainty space while considering the computational cost. Considering that the matrix functions given a θ are evaluated using MC methods, we generate multi-stage scenario trees by applying K-means to the models drawn from the BNN model, which is summarised by the following procedure.

1: **procedure** Scenarios Generation Using K-means
2: Generate L scheduling trajectories $\{\theta^{(l)}(k), k = 1, \ldots, K\}_{l=1}^{L}$ with K time steps using the knowledge of θ.
3: Evaluate $A(\theta^{(l)}(k)), l = 1, \ldots, L, k = 1, \ldots, K$ for each time instant and each scheduling trajectory.
4: For each time instant k, apply K-means to $\{\mathrm{vec}(A(\theta^{(l)}(k)))\}_{l=1}^{L}$ to cluster the L evaluations of the matrix function A at time instant k into C clusters.
5: Use the cluster centres as scenarios at time instant k.
6: **end procedure**

It is noted that any knowledge of the scheduling variable can be easily incorporated into the scheduling trajectory generation (line 2 of the above procedure) to reduce the conservativeness of the generated scenarios. When no knowledge except Θ exists,

the future θ (within a prediction horizon) is assumed to be uniformly distributed over Θ for scheduling trajectory generation. Additionally, the number of clusters is related to the number of scenarios. Using a larger number C of clusters can better describe the distribution of the matrices and thus improve the control performance but also increases the computational cost of multi-stage MPC (Lucia et al., 2013).

Moreover, to ensure safety with a given confidence level δ, we add 2 extra scenarios which correspond to the worst cases and thus have $C+2$ scenarios at each branching node. Specifically, we estimate the mean μ_{M} and standard deviation σ_{M} for each element of the identified system matrices $A(\theta)$ and $B(\theta)$ over the range of the scheduling variables and determine $\beta_{\mathrm{M}}, \mathrm{M} = A, B$ such that $P(x(k+1) \in \mathbb{X}) \geq 1-\delta$ when using $\hat{\mu}_{\mathrm{M}} \pm \beta_{\mathrm{M}}\hat{\sigma}_{\mathrm{M}}$ as the worst-case scenarios. Since the elements in the matrices are bounded and the trained BNN is assumed to contain the true dynamics of the system by Assumption 2.2, there must exist a β_{M} such that the behaviours of the scenarios contain those of the system. A larger β_{M} indicates a more conservative estimation of the uncertainty and can degrade control performance, which is verified by our experiments. Specifically, β_{M} can be determined using probabilistic safety methods for BNNs (Bao et al., 2023; Wicker et al., 2020). In particular, using f^{ω} to denote the BNN, probabilistic safety calculates the lower bound of the probability $P_{\mathrm{safe}}(T,S) = P_{\omega\sim q(\omega;\theta)}(\forall \mathrm{x} \in T, f^{\omega}(\mathrm{x}) \in S)$ guaranteeing that for all inputs in T, the output of the BNN is in the safety set S by estimating the maximal safe sets of weights $H = \{\omega | \forall \mathrm{x} \in T, f^{\omega}(\mathrm{x}) \in S\}$. Additionally, H is approximated by continuously combining safe sets of weights for a given number of iterations using Interval Bound Propagation (IBP) (Gowal et al., 2018) or Linear Bound Propagation (LBP) (Zhang et al., 2018). In particular, IBP or LBP propagates the input interval, i.e. $T = [x^L, x^U]$, through the first layer, to find values $z^{(1),L}$ and $z^{(1),U}$ such that $z^{(1)} \in [z^{(1),L}, z^{(1),U}]$, and then iteratively propagate the bound through each consecutive layer to obtain an interval in the output, which is guaranteed to contain the network output. In our case, we find $H = \{\beta_{\mathrm{M}} | \forall \theta \in \Theta, x \in \mathbb{X}, u \in \mathbb{U}, f^{\beta_{\mathrm{M}}}(\theta, x, u) \in \mathbb{X}\}$ such that $P_{\mathrm{safe}}([\Theta; \mathbb{X}; \mathbb{U}]; \mathbb{X}) \geq 1-\delta$.

Additionally, given the number of clusters at each time instant, the number of scenarios grows exponentially with respect to the horizon. To maintain the computational tractability, branching is only applied for the first $N_b < N$ steps (aka the robust horizon (Lucia et al., 2013)), and the realisation of the uncertainty at step N_b is used for the remaining $N - N_b$ steps, which results in C^{N_b} scenarios and C is the number of clusters. Figure 3 shows an illustrative example of the scenarios in the robust horizon and prediction horizon. It is noted that the number of scenarios $|r(j)|$ and the matrices A_k^i at time k reflect the joint uncertainty of epistemic uncertainty in the matrix functions from the system identification and the unknown evolution of the scheduling variables.

3.2 Probability of scenarios

After generating the scenario tree, the probability of each scenario is calculated using the moment-matching method (Høyland & Wallace, 2001) to maintain the original statistical properties. Generally, it is sufficient to use the first four moments as the statistical features to be matched in scenario generation (Ji et al., 2005). Specifically, the first four central moments are matched by solving the following optimisation problem

$$\min_{\mathbf{p}} \sum_{i}^{m} \left(w_i^1 \left(M_i^- + M_i^+\right) + w_i^3 \left(S_i^- + S_i^+\right) + w_i^4 \left(Q_i^- + Q_i^+\right)\right) + \sum_{i,j=1}^{m} w_{i,j}^1 \left(\Sigma_{i,j}^- + \Sigma_{i,j}^+\right), \tag{17a}$$

$$\text{s.t.} \quad \mathbf{X}\mathbf{p} + M^- - M^+ = M \tag{17b}$$

$$\sum_{s=1}^{C+2} (\mathbf{X}^s - \mathbf{X}\mathbf{p})(\mathbf{X}^s - \mathbf{X}\mathbf{p})^{\mathrm{T}} p^s + \Sigma^- - \Sigma^+ = \Sigma, \tag{17c}$$

$$\sum_{s=1}^{C+2} (\mathbf{X}^s - \mathbf{X}\mathbf{p})^3 p^s + S^- - S^+ = S, \tag{17d}$$

$$\sum_{s=1}^{C+2} (\mathbf{X}^s - \mathbf{X}\mathbf{p})^4 p^s + Q^- - Q^+ = Q, \tag{17e}$$

$$\sum_{s=1}^{C+2} p^s = 1, \quad p^s \geq 0, \quad s = 1, \ldots, C, C+1, C+2, \tag{17f}$$

$$M_i^+, M_i^-, S_i^+, S_i^-, Q_i^+, Q_i^- \geq 0, \quad i = 1, \ldots, m, \tag{17g}$$

$$\Sigma_{ij}^+, \Sigma_{ij}^- \geq 0, \quad i,j = 1, \ldots, m. \tag{17h}$$

where the third- and fourth-power operations in (17d) and (17e) are defined on the elements of the vector $(\mathbf{X}^s - \mathbf{X}\mathbf{p})$, M, Σ, S, and Q are the first four central moments estimated from samples with superscripts $+$, $-$ denoting the positive and negative parts of the associated variables, and $w_i^0, w_{ij}^1, w_i^3, w_i^4$ in the objective function are weighting coefficients. Furthermore, $\mathbf{p} = (p^1, \ldots, p^C, p^{C+1}, p^{C+2})^{\mathrm{T}}$ and p^s is the probability of the sth scenario, $\mathbf{X} = (\mathbf{X}^1, \ldots, \mathbf{X}^C, \mathbf{X}^{C+1}, \mathbf{X}^{C+2}) \in \mathbb{R}^{m\times(C+2)}$ where $\mathbf{X}^s = (X_1^s, \ldots, X_m^s)$ denotes the realisation of the uncertainty in the sth scenario and m is the dimension of the realisation. For example, $m = n_x^2$ when clustering vectorised matrix function value A. The optimal value of the cost function is greater than 0 and indicates the degree to which the generated scenarios preserve the statistical properties of uncertainty quantified by BNNs. Therefore, we choose C such that the optimal value is close to 0 while satisfying the computational resource limitations of multi-stage MPC.

3.3 Scenario-based MPC problem

Given the constructed tree, the MPC problem can be formulated at every time instant as

$$\min_{x^j(i|k), u^j(i|k)} \sum_{j=1}^{C^{N_b}} p^j J_j \left(\mathbf{x}^j(0:N|k), \mathbf{u}^j(0:N-1|k)\right) \tag{18a}$$

$$\text{s.t.} \quad x^j(i+1|k) = A_i^{r(j)} x^{p(j)}(i|k) + B_i^{r(j)} u^j(i|k), \tag{18b}$$

$$x^j(i|k) \in \mathbb{X}, u^j(i|k) \in \mathbb{U}, \forall (j,i) \in I, \tag{18c}$$

$$u^j(i|k) = u^l(i|k) \text{ if } x^{p(j)}(i|k) = x^{p(l)}(i|k), \quad \forall (j,i), (l,i) \in I, \tag{18d}$$

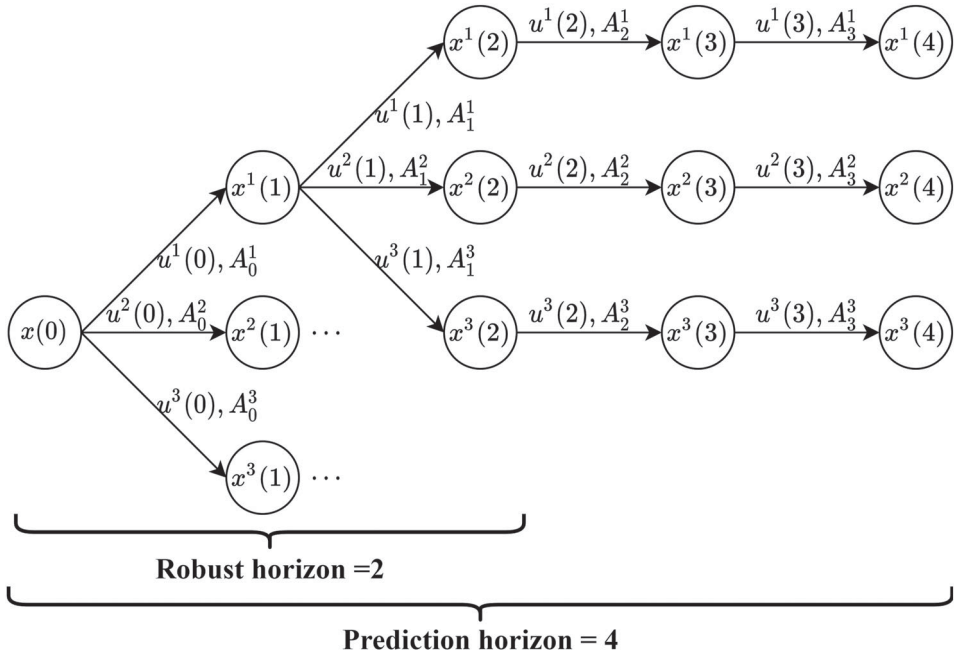

Figure 3. Scenario tree representation of the joint uncertainty evolution for MPC. In the figure, $A_k^{r(j)}$ refers to the matrix at time k in the $r(j)$th scenario.

$$x^j(N|k) \in \mathbb{X}_f, \forall (j, N) \in I, \tag{18e}$$

where the weight p^j is the probability of the jth scenario computed using the method described in Section 3.2, the function $p(j)$ refers to the index of the parent node (the parent of the node indexed by j), and $r(j)$ gives the considered realisation of the joint uncertainty via $A_i^{r(j)}$ and $B_i^{r(j)}$. Furthermore, I denotes the set of all occurring index pairs (j, i). The constraints in (18d) are non-anticipativity constraints to guarantee that control inputs from the same parent node are identical, and $\mathbb{X}_f$ is the terminal set. Since each realisation of the joint uncertainty gives a linear system model at time instant i in the prediction horizon, the constraint satisfaction can be guaranteed by only considering the extreme scenarios, which is employed to establish safety guarantees. The objective function in (18a) is the weighted sum of the cost for each scenario J_j which is defined as

$$J_j = V_f(x^j(N|k)) + \sum_{i=0}^{N-1} \ell(x^{p(j)}(i|k), u^j(i|k)), \tag{19}$$

in which $V_f(\cdot)$ is the terminal cost and $\ell(\cdot)$ is the stage cost. The terminal cost $V_f(\cdot)$ will be discussed in the next subsection. In this paper, we consider

$$\ell(x, u) = x^{\mathrm{T}} Q x + u^{\mathrm{T}} R u \tag{20}$$

where $Q, R \succ 0$ are tuning parameters.

3.3.1 Terminal ingredients

In this section, we show how to compute the three terminal ingredients (Mayne et al., 2000), i.e. a terminal cost, a terminal controller, and a terminal invariant set, that are required to achieve stability of the closed-loop system with the proposed MPC scheme.

First, we transform the BNN model into an LPV form with affine scheduling dependency described as

$$\hat{A}(\hat{\theta}(k)) = \sum_{i=1}^{q} \hat{\theta}_i(k)\hat{A}_i, \quad \hat{B}(\hat{\theta}(k)) = \sum_{i=1}^{q} \hat{\theta}_i(k)\hat{B}_i,$$
$$\sum_{i=1}^{q} \hat{\theta}_i(k) = 1, \quad \hat{\theta}_i(k) \geq 0, \tag{21}$$

where $\hat{A}_i$ and $\hat{B}_i$ are extreme realisations of $A(\theta)$ and $B(\theta)$ in (2), respectively, and $\hat{\theta}$ is the new scheduling variable such that (2) and (21) are equivalent. Additionally, Theorem 2.1 in Nguyen (2014) shows that the LPV models with different numbers of extreme realisations of A_i and B_i can be transformed into the form of (21). In particular, we use the scenarios including the worst-case scenarios in Section 3.1 to obtain the extreme realisations of matrix functions $\hat{A}_i$ and $\hat{B}_i$. It is noted that the conservativeness of the extreme realisations is related to the accuracy of the learned BNN model. Additionally, the number of extreme realisations $\hat{A}_i$ is $2^{|A|}$ where $|A|$ denotes the number of elements in matrix A, and that number for $\hat{B}_i$ is $2^{|B|}$. However, we can only measure θ of the system but not $\hat{\theta}$. Moreover, we assume only input-output data exist without a true system model. Therefore, we learn a coordinate transformation $\mathcal{T}$ from θ in (2) to $\hat{\theta}$ in (21) from data, which can be formulated as a regression problem. While lots of regression algorithms can be used to learn the transformation, ANN can approximate arbitrary nonlinear functions and learn features automatically from data, and thus we use a fully-connected ANN to parameterise $\hat{\theta}(k) = \mathcal{T}(\theta)$ and build the ANN model of $\hat{x}(k+1) = (\sum_{i=1}^{q} \hat{\theta}_i(k)\hat{A}_i)x(k) + (\sum_{i=1}^{q} \hat{\theta}_i(k)\hat{B}_i)u(k)$ where $\hat{A}_i, \hat{B}_i$ are estimated extreme realisations. Then, the optimal transformation $\mathcal{T}^\star$ is obtained by minimising the Mean Squared Error (MSE)

loss function $\frac{1}{N_{\mathcal{D}}-1}\sum_{k=1}^{N_{\mathcal{D}}-1}(x(k+1)-\hat{x}(k+1))^2$ via stochastic gradient descent (SGD) with respect to the parameters of $\mathcal{T}$ on the dataset $\mathcal{D}$. Additionally, the softmax activation function is used in the last layer to satisfy the constraints of $\hat{\theta}$. The advantage of this approach for coordinate transformation is to further moderate the negative effect of the scenario generation by constraining the scenarios to be compatible with the existing dataset. It is noted that the performance of the proposed approach depends on the sufficiency of the dataset, as well as the architecture design and training of ANNs.

Based on the above formulation, we show how to compute the terminal cost and the related terminal controller. We consider parameter-dependent poly-quadratic terminal cost functions in the form of

$$V(x(k),\hat{\theta}(k)) = x(k)^{\mathrm{T}}P(\hat{\theta}(k))x(k),$$
$$P(\hat{\theta}(k)) = \sum_{i=1}^{q}\hat{\theta}_i(k)P_i \succ 0. \tag{22}$$

Note that using such a parameter-dependent formulation can reduce conservativeness significantly in comparison with the parameter-independent counterpart. In general, the closed-loop system can be asymptotically stabilised by the MPC law if there exists a terminal feedback controller $u_k = K_f(x(k))$ such that the following sufficient conditions are satisfied (Mayne et al., 2000):

(1) $V_f(\cdot)$ is a Lyapunov function on a terminal set $\mathbb{X}_f$ under the terminal controller $K_f(\cdot)$ and
$$V_f(x(k+1)) - V_f(x(k)) \le -\ell(x(k),K_f(x_k)) < 0,$$
$$\forall x(k) \in \mathbb{X}_f. \tag{23}$$
(2) If $x(k) \in \mathbb{X}_f$, then $x(k+1) = \hat{A}(\hat{\theta}(k))x(k) + \hat{B}(\hat{\theta}(k))K_f(x(k)) \in \mathbb{X}_f, \forall \hat{\theta}(k) \in \hat{\Theta}$, i.e. $\mathbb{X}_f$ is positively invariant under K_f.
(3) $K_f(x) \in \mathbb{U}, \forall x \in \mathbb{X}_f \subseteq \mathbb{X}$, i.e. the input and state constraints are satisfied under the control law.

To enlarge the terminal region, we consider the following parameter-dependent state-feedback terminal controller

$$K_f(x(k);\hat{\theta}(k)) = \left(\sum_{i=1}^{q}\hat{\theta}_i(k)K_i\right)x(k). \tag{24}$$

Based on the condition for the stability of discrete-time LPV systems (Pandey & de Oliveira, 2017), we compute the terminal cost function and controller by the following proposition:

Proposition 3.1: *For the discrete-time LPV systems described by* (21)*, condition* (23) *is satisfied if there exist matrices* $Q_i \succ 0, X_i \in \mathbb{R}^{n_x\times n_x}, L_i \in \mathbb{R}^{n_u\times n_x}, Y_i \in \mathbb{R}^{n_u\times n_x}, Z_i \in \mathbb{R}^{n_u\times n_x}, i = 1, \ldots, q$ *such that*

$$\begin{bmatrix} X_i + X_i^{\mathrm{T}} - Q_i & X_i^{\mathrm{T}}\hat{A}_i^{\mathrm{T}} & -L_i^{\mathrm{T}} \\ \star & Q_j - R_{i,j} & \hat{B}_iZ_j - Y_j^{\mathrm{T}} \\ \star & \star & Z_j + Z_j^{\mathrm{T}} \\ \star & \star & \star \\ \star & \star & \star \end{bmatrix}$$
$$\begin{bmatrix} (Q^{1/2}X_i)^{\mathrm{T}} & (R^{1/2}L_i)^{\mathrm{T}} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ I & \mathbf{0} \\ \star & I \end{bmatrix} \succ 0$$
$$\text{for} \quad \forall i,j = 1,\ldots q, \tag{25}$$

where ★ *represents the symmetric blocks omitted for brevity, and* $R_{i,j} = \hat{B}_iY_j + (\hat{B}_iY_j)^{\mathrm{T}}$, $P_i = Q_i^{-1}$, *using the terminal controller with gain in the form of* (24) *and* $K_i = L_iX_i^{-1}$.

Proof: The proof is based on the proof of Theorem 2 in Pandey and de Oliveira (2017). Since $X_i + X_i^{\mathrm{T}} \succ Q_i \succ 0$, $X_i^{\mathrm{T}}Q_i^{-1}X_i \succeq X_i + X_i^{\mathrm{T}} - Q_i$. Additionally, substituting $K_i = L_iX_i^{-1}$ into (25), we have

$$\begin{bmatrix} X_i^{\mathrm{T}}Q_i^{-1}X_i & X_i^{\mathrm{T}}\hat{A}_i^{\mathrm{T}} & -(K_iX_i)^{\mathrm{T}} \\ \star & Q_j - R_{i,j} & \hat{B}_iZ_j - Y_j^{\mathrm{T}} \\ \star & \star & Z_j + Z_j^{\mathrm{T}} \\ \star & \star & \star \\ \star & \star & \star \end{bmatrix}$$
$$\begin{bmatrix} (Q^{1/2}X_i)^{\mathrm{T}} & (R^{1/2}K_iX_i)^{\mathrm{T}} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ I & \mathbf{0} \\ \star & I \end{bmatrix} \succ 0. \tag{26}$$

Applying the following congruent transformation

$$S_i^{\mathrm{T}} = \begin{bmatrix} X_i^{-\mathrm{T}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & I \end{bmatrix}$$

to (26) gives

$$\begin{bmatrix} Q_i^{-1} & \hat{A}_i^{\mathrm{T}} & -K_i^{\mathrm{T}} & (Q^{1/2})^{\mathrm{T}} & (R^{1/2}K_i)^{\mathrm{T}} \\ \star & Q_j - R_{i,j} & \hat{B}_iZ_j - Y_j^{\mathrm{T}} & \mathbf{0} & \mathbf{0} \\ \star & \star & Z_j + Z_j^{\mathrm{T}} & \mathbf{0} & \mathbf{0} \\ \star & \star & \star & I & \mathbf{0} \\ \star & \star & \star & \star & I \end{bmatrix} \succ 0 \tag{27}$$

which can be rewritten as

$$\begin{bmatrix} P_i & \hat{A}_i^{\mathrm{T}} & -K_i^{\mathrm{T}} \\ \star & P_j^{-1} + M_{i,j} & \hat{B}_iH_j^{-\mathrm{T}} - P_j^{-\mathrm{T}}F_jH_j^{-1} \\ \star & \star & H_j^{-\mathrm{T}} + H_j^{-1} \\ \star & \star & \star \\ \star & \star & \star \end{bmatrix}$$
$$\begin{bmatrix} (Q^{1/2})^{\mathrm{T}} & (R^{1/2}K_i)^{\mathrm{T}} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ I & \mathbf{0} \\ \star & I \end{bmatrix} \succ 0 \tag{28}$$

by defining $P_i = Q_i^{-1}, H_i = Z_i^{-\mathrm{T}}, F_i = P_i Y_i^{\mathrm{T}} H_i$ and $M_{i,j} = -\hat{B}_i H_j^{-\mathrm{T}} F_j^{\mathrm{T}} P_j^{-1} - (\hat{B}_i H_j^{-\mathrm{T}} F_j^{\mathrm{T}} P_j^{-1})^{\mathrm{T}}$. Then, applying another congruent transformation

$$S_j^{\mathrm{T}} = \begin{bmatrix} I & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & H_j & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & P_j & F_j & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & I \end{bmatrix}$$

to (28) produces

$$\begin{bmatrix} P_i & -(H_j K_i)^{\mathrm{T}} & (P_j \hat{A}_i - F_j K_i)^{\mathrm{T}} \\ \star & H_j + H_j^{\mathrm{T}} & (P_j \hat{B}_i + F_j)^{\mathrm{T}} \\ \star & \star & P_j \\ \star & \star & \star \\ \star & \star & \star \end{bmatrix} \begin{bmatrix} (Q^{1/2})^{\mathrm{T}} & (R^{1/2} K_i)^{\mathrm{T}} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ I & \mathbf{0} \\ \star & I \end{bmatrix} \succ 0. \tag{29}$$

Taking convex combinations of (29) over i and j gives

$$\begin{bmatrix} P(\hat{\theta}(k)) & -(H(\hat{\theta}(k+1))K(\hat{\theta}(k)))^{\mathrm{T}} \\ \star & H(\hat{\theta}(k+1)) + H(\hat{\theta}(k+1))^{\mathrm{T}} \\ \star & \star \\ \star & \star \\ \star & \star \end{bmatrix} \begin{bmatrix} (P(\hat{\theta}(k+1))\hat{A}(\hat{\theta}(k))) & & \\ -F(\hat{\theta}(k+1)) & (Q^{1/2})^{\mathrm{T}} & (R^{1/2}K(\hat{\theta}(k)))^{\mathrm{T}} \\ K(\hat{\theta}(k)))^{\mathrm{T}} & & \\ (P(\hat{\theta}(k+1))\hat{B}(\hat{\theta}(k)) & & \\ +F(\hat{\theta}(k+1)))^{\mathrm{T}} & \mathbf{0} & \mathbf{0} \\ P(\hat{\theta}(k+1)) & \mathbf{0} & \mathbf{0} \\ \star & I & \mathbf{0} \\ \star & \star & I \end{bmatrix} \succ 0. \tag{30}$$

Finally, multiplying (30) by

$$S(\hat{\theta}(k)) = \begin{bmatrix} I & K(\hat{\theta}(k))^{\mathrm{T}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & I \end{bmatrix}$$

from the left and by its transpose from the right yields

$$\begin{bmatrix} P(\hat{\theta}(k)) & P(\hat{\theta}(k+1))\hat{A}_c(\hat{\theta}(k))^{\mathrm{T}} \\ \star & P(\hat{\theta}(k+1)) \\ \star & \star \\ \star & \star \end{bmatrix} \begin{bmatrix} (Q^{1/2})^{\mathrm{T}} & (R^{1/2}K(\hat{\theta}(k)))^{\mathrm{T}} \\ \mathbf{0} & \mathbf{0} \\ I & \mathbf{0} \\ \mathbf{0} & I \end{bmatrix} \succ 0,$$

$$\text{for } \forall \hat{\theta}(k), \hat{\theta}(k+1) \in \hat{\Theta} \tag{31}$$

where $\hat{A}_c(\hat{\theta}(k)) = \hat{A}(\hat{\theta}(k)) + \hat{B}(\hat{\theta}(k))K(\hat{\theta}(k))$. Finally, it can be shown that (31) is equivalent to (23) by applying the Schur complement, and this concludes the proof. ■

Next, using the controller determined by solving (25), we can compute a terminal set as a maximal polyhedral robust positively invariant (RPI) set (Nguyen, 2014). The considered input and state constraints are in the form of

$$\mathbb{X} = \{x \in \mathbb{R}^{n_x} | F_x x \le g_x\}, \mathbb{U} = \{u \in \mathbb{R}^{n_u} | F_u x \le g_u\}. \tag{32}$$

Different from the Procedure 2.1 in Nguyen (2014), the state constraints of the system (21) are

$$x_c \in \mathbb{X}_c, \mathbb{X}_c = \{x \in \mathbb{R}^{n_x} | F_c x \le g_c\} \tag{33}$$

where $F_c = [F_x^{\mathrm{T}} \ (F_u K_1)^{\mathrm{T}} \ \cdots \ (F_u K_q)^{\mathrm{T}}]^{\mathrm{T}}$ and $g_c = [g_x^{\mathrm{T}} \ g_u^{\mathrm{T}} \ \cdots \ g_u^{\mathrm{T}}]^{\mathrm{T}}$, as a parameter-dependent controller is used. Therefore, the number of constraints is increased by $(q-1)qn_u$, compared against a parameter-independent controller. Then, using Algorithm 1, we can compute a maximal polyhedral RPI set $\Omega_{\max}$ as the terminal set Ω_f.

Algorithm 1 (Gilbert and Tan (1991), **Procedure 2.1**) Robustly controlled positively invariant set computation

Input: $\{\hat{A}_{ci}\}_{i=1}^{q}$, $\mathbb{X}_c$ defined in (33).
Output: The maximal RPI set $\Omega_{\max}$.
1: Set $i = 0, F_0 = F_c, g_0 = g_c$ and $\mathbb{X}_0 = \{x \in \mathbb{R}^{n_x} : F_0 x \le g_0\}$.
2: Set $\mathbb{X}_1 = \mathbb{X}_0$.
3: Eliminate redundant inequalities of the following polytope,

$$P = \left\{ x \in \mathbb{R}^{n_x} : \begin{bmatrix} F_0 \\ F_0 \hat{A}_{c1} \\ \vdots \\ F_0 \hat{A}_{cq} \end{bmatrix} x \le \begin{bmatrix} g_0 \\ g_0 \\ \vdots \\ g_0 \end{bmatrix} \right\}$$

4: Set $\mathbb{X}_0 = P$ and update consequently the matrices F_0 and g_0.
5: If $\mathbb{X}_0 = \mathbb{X}_1$ then stop and set $\Omega = \mathbb{X}_0$. Else continue.
6: Set $i = i + 1$ and go to step 2.

Furthermore, we can compute the robustly N-step controlled positively invariant sets based on the maximal RPI set as the domain of attraction (DOA) using Algorithm 2. Different from Procedure 2.3 in Nguyen (2014), we allow control inputs to be different in Step 2 of Algorithm 2 for different (A_i, B_i) when computing the expanded set, to enlarge the DOA in Step 2 of Algorithm 2. Therefore, the number of decision variables is increased by $q-1$ compared to the parameter-independent case.

Algorithm 2 (Gilbert and Tan (1991), **Procedure 2.3**) Robustly N-step controlled invariant set computation

Input: $\{\hat{A}\}_{i=1}^{q}$, $\{\hat{B}\}_{i=1}^{q}$ and the sets $\mathbb{X}$, $\mathbb{U}$ and $\Omega_{\max}$.
Output: The N-step robustly controlled invariant set C_N.

1: Set $i = 0$ and $C_0 = \Omega_{\max}$ and let the matrices F_0, g_0 be the half-space representation of C_0, i.e. $C_0 = \{x \in \mathbb{R}^n : F_0 x \leq g_0\}$.
2: Compute the expanded set $P_i \subset \mathbb{R}^{n_x+n_u}$

$$P_i = \left\{ (x,u) \in \mathbb{R}^{n_x+n_u} : \begin{bmatrix} F_i(\hat{A}_1 x + \hat{B}_1 u_1) \\ F_i(\hat{A}_2 x + \hat{B}_2 u_2) \\ \vdots \\ F_i(\hat{A}_q x + \hat{B}_q u_q) \end{bmatrix} x \leq \begin{bmatrix} g_i \\ g_i \\ \vdots \\ g_i \end{bmatrix} \right\}$$

3: Compute the projection $P_i^{(n)}$ of P_i on $\mathbb{R}^{n_x}$

$$P_i^{(n)} = \{x \in \mathbb{R}^{n_x} : \exists u \in \mathbb{U} \text{ s.t. } (x,u) \in P_i\}.$$

4: Set $C_{i+1} = P_i^{(n)} \cap \mathbb{X}$ and let F_{i+1}, g_{i+1} be the half-space representation of C_{i+1}, i.e.

$$C_{i+1} = \{x \in \mathbb{R}^{n_x} : F_{i+1} x \leq g_{i+1}\}.$$

5: If $C_{i+1} = C_i$, then stop and set $C_N = C_i$. Else continue.
6: If $i = N$, then stop else continue.
7: Set $i = i + 1$ and go to step 2.

3.3.2 Recursive feasibility, stability and safety

In this section, we establish the recursive feasibility, stability, and safety of the proposed learning-based SMPC scheme.

In particular, the recursive feasibility and stability are established by adopting the work (Maiworm et al., 2015) which considers a nonlinear discrete-time system represented by

$$x(k+1) = f(x(k), u(k), p(k)), \quad \text{s.t. } x(k) \in \mathbb{X}, \quad u(k) \in \mathbb{U}, \quad p(k) \in \mathbb{P} \tag{34}$$

where $p \in \mathbb{R}^{n_p}$ denotes the uncertain parameters and $\mathbb{P}$ is a discrete set of s parameter values, under the following assumptions:

Assumption 3.2: *(**Continuity**) The functions $f(x,u,p)$, $\ell(x,u)$ and V_f are continuous, with $f(0,0,p) = 0$ $\forall p \in \mathbb{P}$, $\ell(0,0) = 0$ and $V_f(0) = 0$.*

Assumption 3.3: *(**Constraints**) The sets $\mathbb{X}$ and $\mathbb{X}_f \subseteq \mathbb{X}$ are closed, and $\mathbb{U}$ is compact. Each set contains the origin.*

Establishing recursive feasibility for SMPC is equivalent to requiring that the terminal state $x^j(N)$ of each scenario ends in a common control invariant terminal region Ω_f which ensures that the state stays in Ω_f for all system instances when $x(N) \in \Omega_f$.

Assumption 3.4: *(**Common terminal region**) There exists a common terminal region Ω_f that is control invariant for $x(k+1) = f^j(x(k), u(k)), \forall j \in \{1, \ldots, s\}$ with $u \in \mathbb{U}$.*

Proposition 3.5: *Suppose that Assumptions* 3.2, 3.3 *and* 3.4 *hold. Then, the SMPC is recursively feasible.*

Proof: The proof is similar to that of Proposition 4 in Maiworm et al. (2015) and hence omitted here. ■

To establish stability, the following assumptions on the stage and terminal costs are made.

Assumption 3.6: *(**Basic stability assumption**) For $\forall x \in \Omega_f$ and $\forall j \in \{1, \ldots, N_s\}$,*

$$\min_{\tilde{u}(k) \in \mathbb{U}} \{V_f^j\left(f(x,u,p)\right) + \ell(x,u) | f(x,u,p) \in \Omega_f\} \leq V_f^j(x), \tag{35}$$

where $N_s = s^{N_b}$ denotes the number of scenarios and V_f^j denotes an individual terminal cost function to the jth scenario, holds for all $p \in \mathcal{P}$.

Assumption 3.6 ensures the descent property of V_f^j and implies Assumption 3.4 if $V_f^j(x)$ is a control Lyapunov function.

Assumption 3.7: *(**Bounds on stage and terminal costs**) The stage cost $\ell(x,u)$ and the terminal costs $V_f^j(x)$ satisfy*

$$\ell(x,u) \geq \alpha_1(|x|) \quad \forall x \in \Omega_N, \forall u \in \mathbb{U}$$
$$V_f^j(x) \leq \alpha_2^j(|x|) \quad \forall x \in \Omega_f \quad \text{and} \quad \forall j \in \{1, \ldots, N_s\},$$

in which $\alpha_1(\cdot)$ and $\alpha_2^j(\cdot)$ are $\mathcal{K}_\infty$ functions.

Assumptions 3.6 and 3.7 ensure that the value function is a Lyapunov function for $x(k+1) = f^j(x(k), \kappa_N(x(k)))$, $\forall j \in \{1, \ldots, s\}$ on the domain C_N. The following lemma and theorem are then given.

Lemma 3.8: *(SMPC stability (Maiworm et al., 2015)) Suppose that Assumptions* 3.2–3.7 *hold and that Ω_f contains the origin in its interior. Then, the origin is asymptotically stable with a region of attraction C_N for the system $x(k+1) = f^j(x(k), \kappa_N(x(k)))$ for all $j \in \{1, \ldots, s\}$.*

Furthermore, the above Lemma 3.8, which holds for general nonlinear systems with discrete sets of uncertain parameter values, can be adopted for systems in the LPV form (21) with affine scheduling dependency and continuous set of scheduling variables, resulting from the following lemma.

Lemma 3.9: *$\forall k \in \mathbb{N}$, if a control input $u(k)$ is feasible for all the extreme realisations of* (21) *given $x(k)$, then $u(k)$ is feasible $\forall \hat{\theta}(k) \in \{\hat{\theta} | \sum_{i=1}^{q} \hat{\theta}_i = 1$ and $\hat{\theta}_i \geq 0\}$ in* (21).

Proof: Since $\hat{A}_i x(k) + \hat{B}_i u(k) \triangleq x^i(k+1) \in \mathbb{X}, i = 1, \ldots, q$, and $\mathbb{X}$ is assumed to be a PC-set, then $\forall \hat{\theta}(k) \in \{\hat{\theta} | \sum_{i=1}^{q} \hat{\theta}_i = 1$ and $\hat{\theta}_i \geq 0\}, x(k+1) = \hat{A}(\hat{\theta}(k))x(k) + \hat{B}(\hat{\theta}(k))u(k) = (\sum_{i=1}^{q} \hat{\theta}_i(k)\hat{A}_i)x(k) + (\sum_{i=1}^{q} \hat{\theta}_i(k)\hat{B}_i)u(k) = \sum_{i=1}^{q} \hat{\theta}_i(k)(\hat{A}_i x(k) + \hat{B}_i u(k)) = \sum_{i=1}^{q} \hat{\theta}_i(k) x^i(k+1) \in \mathbb{X}$. ■

Lemma 3.9 shows that the feasibility of a control input for all the possible values of scheduling variables can be established by only considering the finite extreme realisations of (21).

Based on the stability theorem of SMPC and Lemma 3.9, we present the following theorem on the learning-based SMPC.

Theorem 3.10: *(**Learning-based SMPC stability and feasibility**) Suppose that Lemma* 1 *is fulfilled, and the terminal set* Ω_f *computed by Algorithm 1 contains the origin in its interior. Then, the SMPC with the terminal cost* (22) *and the terminal controller* (24) *is recursively feasible and the origin is asymptotically stable for* (21) *with a region of attraction* C_N*. Moreover, the original system* (2) *is stable with a high probability that is at least* $1 - \delta_c$*.*

Proof: Obviously, the LPV model with (21), the considered stage cost (20), and the terminal cost (22) fulfill Assumption 3.2. Assumption 3.3 also holds, as the terminal set Ω_f computed in Section 3.3.1 is polyhedral and thus closed while $\mathbb{X}$ and $\mathbb{U}$ are assumed to be PC-sets. Moreover, Ω_f by Algorithm 1 is control invariant for arbitrary scheduling variables under the terminal controller (24) and thus Assumption 3.4 holds. Furthermore, Assumption 3.6 holds, as the designed terminal controller satisfies the sufficient conditions in Section 3.3.1. Additionally, the quadratic stage cost (20) and the poly-quadratic terminal cost function (22) satisfy Assumption 3.7 with $\alpha_1(|x|) = \lambda_{\min}(Q)\|x\|^2$ and $\alpha_2^j(|x|) = \lambda_{\max}(P_j)\|x\|^2$ where λ denotes the eigenvalue of a matrix. Hence, the origin is asymptotically stable with a region of attraction C_N for the LPV model with (21) by Lemma 3.8. Moreover, the LPV model with (21) is transformed from the BNN model whose behaviours contain the behaviours of the system by Lemma 2.3. Therefore, the system is stabilised with a high probability that Lemma 2.3 is fulfilled. ■

Furthermore, using the scenario generation approach described in Sections 3.1 and 3.2, the certificate of safety under the SMPC law can be formalised as follows.

Theorem 3.11: *(**Learning-based SMPC safety**) Let the hypotheses of Assumption* 2.2 *and Lemma* 2.3 *be satisfied. Then, the system under the SMPC law* (16) *is* δ*-safe.*

Proof: By Assumption 2.2 and Lemma 2.3, the behaviours of the generated scenarios based on the N_{MC} sampled models from the BNN model contain the behaviours of the system. Furthermore, by Proposition 3.5, the SMPC is recursively feasible, which proves the system is δ-safe by Definition 2.1. ■

Additionally, Figure 4 shows the block diagram of the closed-loop learning-based SMPC scheme.

4. Numerical results

In this section, the proposed methods of this work are validated on a parameter-varying double integrator system model (Hanema et al., 2020), as well as a parameter-varying multiple-input multiple-output (MIMO) system with complex nonlinear scheduling dependency.

4.1 Parameter-varying double integrator

The LPV-SS representation of the system is assumed to be

$$x(k+1) = \left(\begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix} + \begin{bmatrix}0.1 & 0\\ 0 & 0.1\end{bmatrix}\theta_1(k) + \begin{bmatrix}0.5 & 0.5\\ 0 & 0\end{bmatrix}\theta_2(k) + \begin{bmatrix}0 & 0\\ 0 & 0.2\end{bmatrix}\theta_3(k)\right)x(k) + \begin{bmatrix}0.5\\ 1\end{bmatrix}u(k), \tag{36}$$

with constraints and scheduling sets as

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

In (36), $A(\cdot)$ is an affine function of the scheduling variables and B is constant.

4.1.1 System identification

We use slowly-varying trajectories for the scheduling variables in Figure 5(a) to collect observations $\mathcal{D} = \{(\theta(t), x(t), u(t)), x(t+1)\}$ for model identification. Pseudo random binary sequences (PRBS) input signal with a scale of 0.01 as shown in Figure 5(b) is used to excite the system, and the generated state sequence with initial state $x(0) = [2.7; 0]$ is shown in Figure 5(c, d). Furthermore, 500 samples are collected and split into training and testing sets with a ratio of 80%/20%.

We use one DenseVariational layer with 4 hidden units to represent $A(\cdot)$ and one Dense layer with 2 hidden units to represent B. Neither of the layers uses activation functions and the Dense layer further does not use bias, which aims to exactly represent the class of models to which (36) belongs. The tuning parameters in (7) are determined as $\sigma_1 = 0.3, \sigma_2 = 0.1$. Adam optimiser is used with a learning rate set to 0.01 and other hyper-parameters as default. Moreover, using the transfer learning approach (Bao et al., 2021), we first trained an ANN model with the same architecture as the BNN model, used the trained ANN weights to initialise the BNN model, and then trained the BNN model for 1, 000 epochs. The validation results are shown in Figure 6.

It is noted that the best fit ratio $\text{BFR} = 100\% \cdot \max(1 - \frac{\|x-\hat{x}\|_2}{\|x-\bar{x}\|_2}, 0) = [96.70\%; 87.04\%]$ using the estimated mean as predictions for outputs. None of the samples are out of $2\sigma_x$. By increasing $\beta\sigma_x$, the true states are guaranteed to lie in the interval $[\mu_x - \beta\sigma_x, \mu_x + \beta\sigma_x]$ almost surely.

4.1.2 Validation of the proposed approach

Without extra knowledge on the evolution of the scheduling variables beyond the scheduling sets, we randomly sample 500 θ's from the uniform distribution over Θ and evaluate $A(\cdot)$ for $N_{\text{MC}} = 500$ times using the dynamic functions sampled from the BNN model for each θ. Then, we apply K-means to the evaluated A's to generate the scenarios. The number of clusters is assumed to be 3. Also, $\beta_{\text{M}} = 1, \text{M} = A, B$ is considered here for the estimation of extreme realisations. Therefore, 5 scenarios were used including $\mu_{\text{M}} \pm \beta_{\text{M}}\sigma_{\text{M}}$. It is worth noting that the scenarios are fixed within the robust horizon of the tree generation in this case due to the limited time-invariant knowledge of θ. When further information (e.g. a bounded ROV (Casavola et al., 2008)) is known, we can generate time-varying scenarios for each step within the robust horizon. The probability

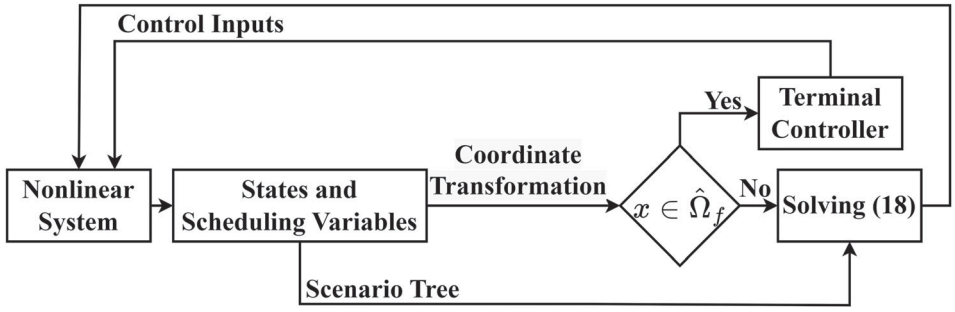

Figure 4. The block diagram of the closed-loop learning-based SMPC scheme.

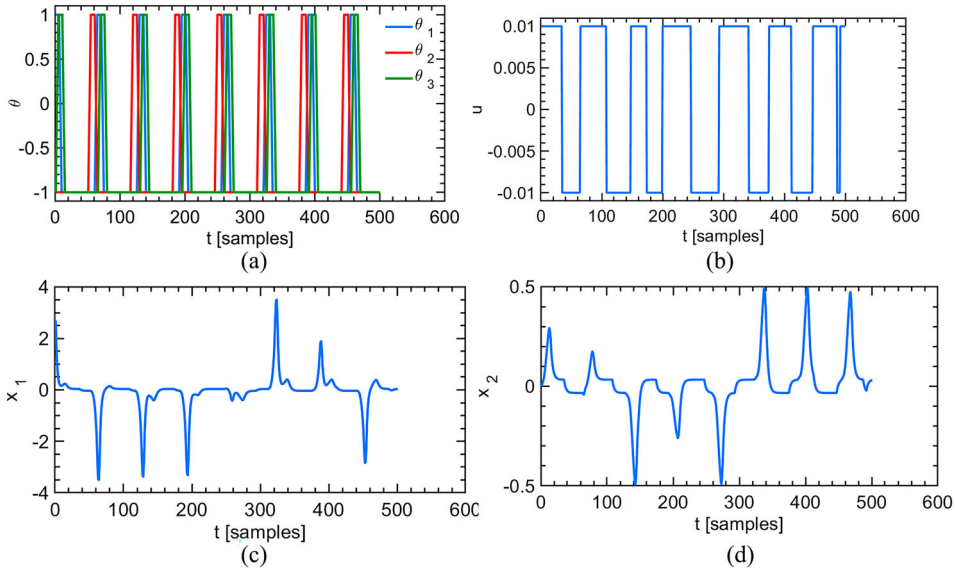

Figure 5. Data generated for model identification purposes. (a) Scheduling trajectories. (b) Inputs to the system. (c) Sequence of x_1. (d) Sequence of x_2.

of the 5 scenarios is $\mathbf{p} = [0.3560; 0.3383; 0.3037; 0.0010; 0.0010]$ using the moment matching method. Additionally, in our experiments, $Q = I_{2\times 2}, R = 1$ for the stage cost ℓ in (20). The prediction horizon is set to 10 and the robust horizon to 1. We computed RPI sets and 10-step robustly controlled positively invariant (RCPI) sets based on the system model (2) and the BNN model (21), respectively.

Results and discussion: As shown in Figure 7, the estimated sets are smaller than the system sets due to the conservativeness introduced to guarantee safety. The sets can be enlarged by numerical methods, which will be investigated in the future work.

The scheduling signals for control are shown in Figure 8(a), which vary faster than the signals used for model identification in Figure 5(a). The control results are shown in Figure 8, where Figure 8(b–c) demonstrate that using the terminal cost and terminal set can increase the convergence rate.

Figure 9 shows that the designed MPC can achieve high control performance even when the initial states are at the vertices of the state constraint set; this is something that was not demonstrated using the approach developed in Hanema et al. (2020).

Additionally, Figure 10 shows that the designed MPC is robust against the evolution of the scheduling variables in

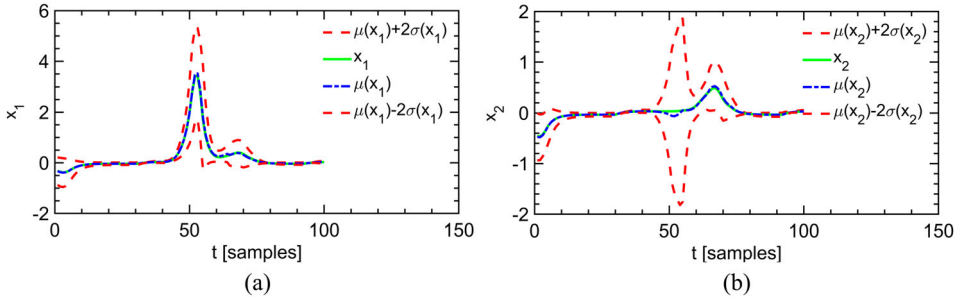

Figure 6. Validation results for the identified BNN model. The area between the two dashed lines is within 2 estimated standard deviations of the estimated mean, which is about 95% confidence interval.

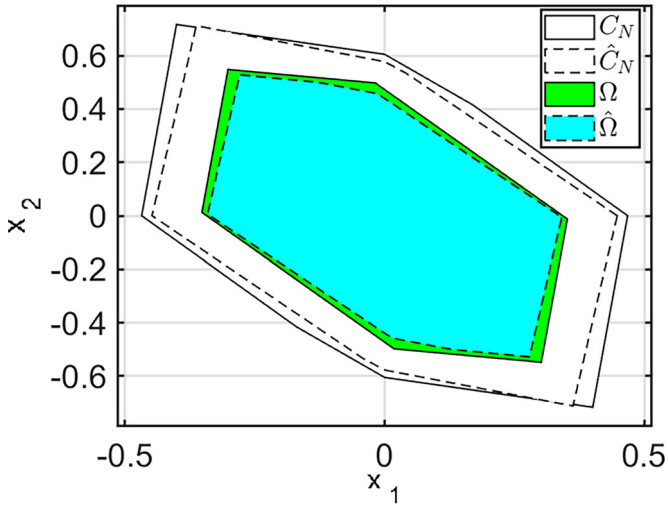

Figure 7. RPI sets: Ω_f for system and $\hat{\Omega}_f$ for BNN model; 10-step RCPI sets: C_N for system and $\hat{C}_N$ for BNN model.

Figure 10(a) and the real state trajectory is contained among the trajectories of the scenarios (See Figure 10(b, c)).

4.2 Parameter-varying MIMO system

The LPV-SS representation of the system is assumed to be

$$x(k+1) = \begin{bmatrix} \sin(\theta_1) & \theta_1^2 + \theta_1\theta_2 \\ \theta_2^3 & \cos(\theta_1 + \theta_2) \end{bmatrix} x(k) + \begin{bmatrix} \theta_2^4 & \cos(\theta_2) \\ \sin(\theta_1 + \theta_2) & \theta_1^3 \end{bmatrix} u(k), \quad (37)$$

with constraints and scheduling sets as

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

Here, both $A(\cdot)$ and $B(\cdot)$ are nonlinear functions of the scheduling variables.

4.2.1 Model identification

We use $\theta_1(k) = \sin(0.3k)$ and $\theta_2(k) = \sin(0.7k)$ in Figure 11(a) to collect observations $\mathcal{D} = \{(\theta(k), x(k), u(k)), x(k+1)\}$ for model identification purposes. Input signals in Figure 11(b) drawn from the uniform distribution $\mathcal{U}(-0.45, 0.45)$ are used to excite the system, and the generated state sequence with initial state $x(0) = [0; 0]$ is shown in Figure 11(c). Additionally, 1100 samples are collected and split into 800 and 300 samples as training and testing sets, respectively.

We use a DenseVariational layer connected to a three-layer fully-connected ANN to represent $A(\cdot)$ and another DenseVariational layer connected to another three-layer fully-connected ANN to represent $B(\cdot)$. All the hidden layers have 32 hidden units with the Exponential Linear Unit (ELU) activation functions (Clevert et al., 2015) while the output layers have 4 hidden units without activation functions. The tuning parameters in (7) are determined as $\sigma_1 = 0.3, \sigma_2 = 0.1$. Adam optimiser is used with a learning rate set to 0.001 and other hyper-parameters as default. Moreover, we first trained an ANN model with the same architecture as the BNN model, used the trained ANN weights to initialise the BNN model, and then trained the BNN model for 10,000 epochs. The validation results are shown in Figure 12.

4.2.2 Validation of the proposed approach

Without assuming extra knowledge on the evolution of the scheduling variables beyond the scheduling sets, we randomly sample 100 θ's from the uniform distribution over Θ and then evaluate both $A(\cdot)$ and $B(\cdot)$ for $N_{\text{MC}} = 500$ times using the dynamic functions sampled from the BNN model for each θ. Then, we apply K-means to the concatenations of the vectorised A's and B's to generate the scenarios. The number of clusters is assumed to be 3. Also, $\beta_{\text{M}} = 2, \text{M} = A, B$ is considered here for the worst-case scenarios. Therefore, 5 scenarios were used

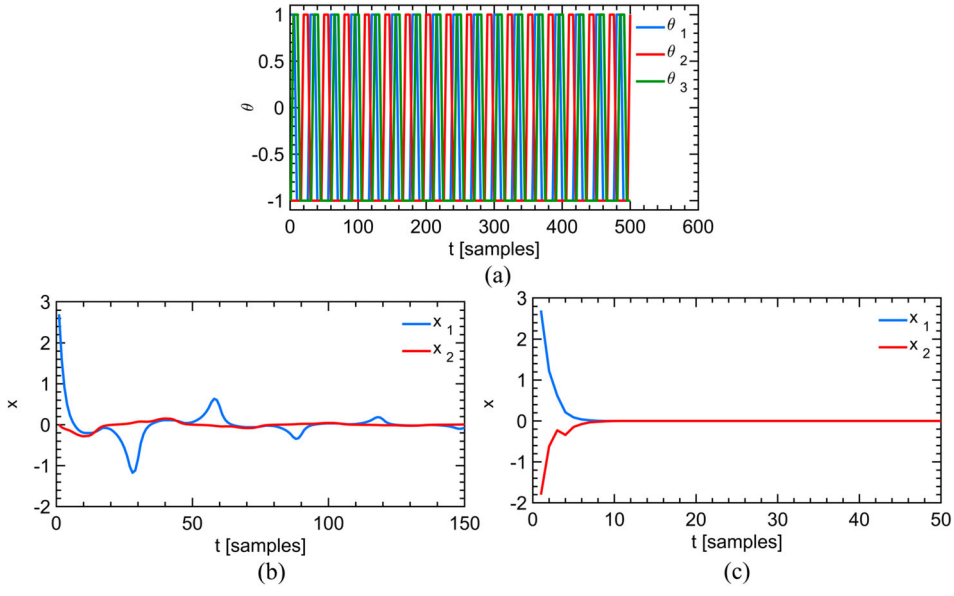

Figure 8. Control results using K-means to generate scenarios. (a) Scheduling signals used for control. (b) Control results without using terminal cost and terminal set. (c) Control results using terminal cost and terminal set.

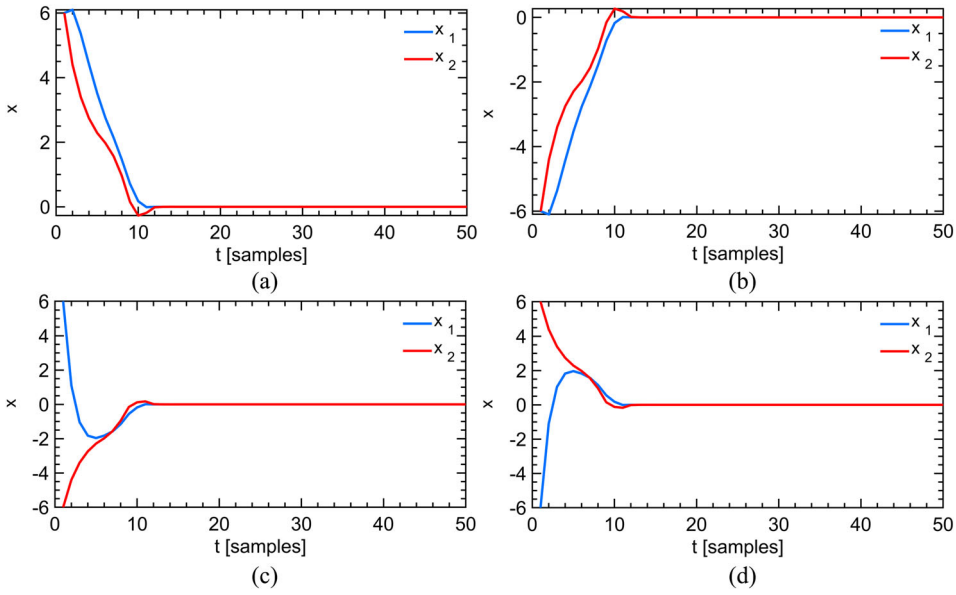

Figure 9. Control results using K-means to generate scenarios and terminal cost control when the initial states x_0 are at the vertices of the constraint set $\mathbb{X}$.

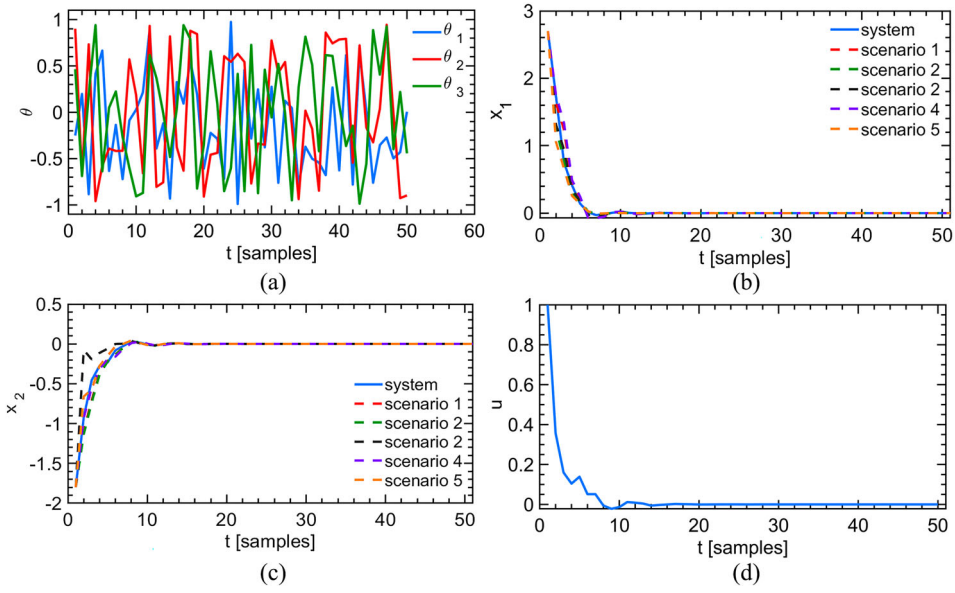

Figure 10. Control results using terminal cost control when $x_0 = [2.7; -1.8]$ and a random scheduling trajectory. (a) Random scheduling trajectories. (b) State x_1 trajectory of systems and scenarios. (c) State x_2 trajectory of systems and scenarios. (d) Control inputs.

including $\mu_M \pm \beta_M \sigma_M$. Additionally, the scenarios are fixed within the robust horizon of the tree generation in this case due to the limited time-invariant knowledge of θ. The probability of the 5 scenarios is $\mathbf{p} = [0.26; 0.30; 0.26; 0.09; 0.09]$ using the moment matching method. Moreover, in our experiments, $Q = I_{2\times 2}, R = I_{2\times 2}$ for the stage cost ℓ in (20). The prediction horizon is set to 10 and the robust horizon to 1. The RPI set was computed based on the BNN model.

Results and discussion: The computed RPI set based on the BNN model is shown in Figure 13(a). The scheduling signal for control is random, as shown in Figure 13(b), which varies faster than the signal for identification in Figure 11(a), to demonstrate that the designed MPC is robust against the evolution of the scheduling variable. The control results in Figure 13(c–f) show that the designed MPC can achieve good control performance even when the initial states are at the vertices of the state constraint set.

4.3 Two-tank system

The cascaded two-tank system (Hanema et al., 2021) can be described by

$$\rho S_1 \dot{h}_1 = -\rho A_1 \sqrt{2gh_1} + u, \tag{38a}$$

$$\rho S_2 \dot{h}_2 = \rho A_1 \sqrt{2gh_1} - \rho A_2 \sqrt{2gh_2}, \tag{38b}$$

where u is the flow of liquid with density $\rho = 0.001\,\mathrm{kgcm}^{-3}$ pumped into the upper tank. $S_1 = 2500\,\mathrm{cm}^2$, $S_2 = 1600\,\mathrm{cm}^2$,

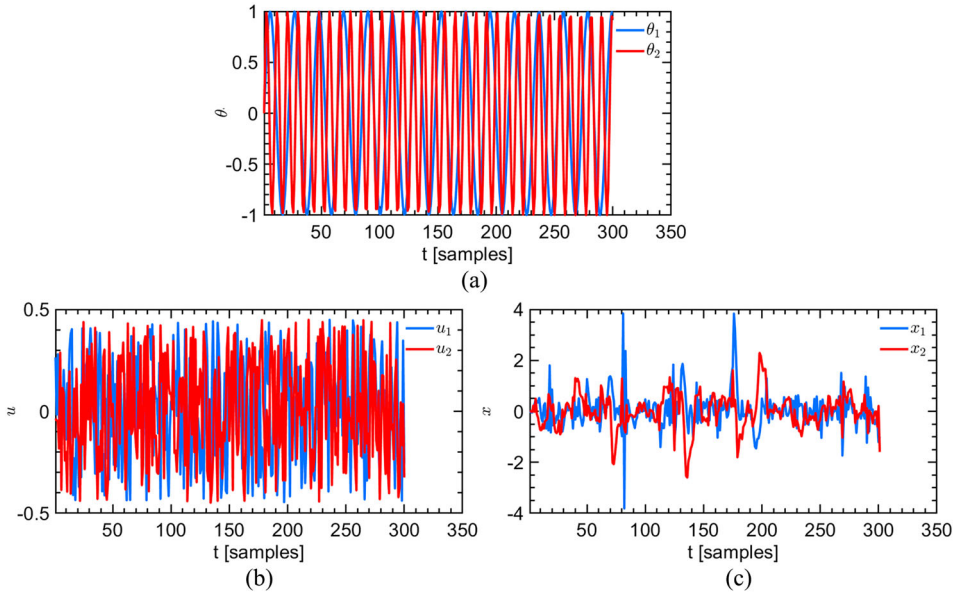

Figure 11. Data generated for model identification purposes. For the sake of clarity, only the first 300 training data points are shown here. (a) Scheduling trajectories. (b) Inputs to the system. (c) Sequence of x.

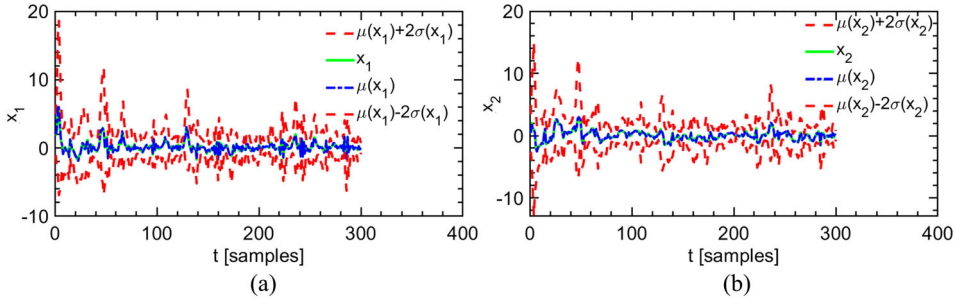

Figure 12. Validation results for the identified BNN model. The BFR = [93.14%; 92.47%] using the estimated mean as predictions for outputs. None of the samples are out of $2\sigma_x$.

$A_1 = 9\,\text{cm}^2$, and $A_2 = 4\,\text{cm}^2$ denote the cross-sectional areas of the upper tank, the lower tank, the pipe through which the liquid flows into the lower tank, and the pipe through which the liquid flows out, respectively. The control objective is to regulate the levels h_1 and h_2 at a given set point. u is available as a control input and subject to the constraint $\mathbb{U} = \{u | 0\,\text{kgs}^{-1} \leq u \leq 4\,\text{kgs}^{-1}\}$. Additionally, the liquid levels satisfy the bounds $\mathbb{X} = \{x = [h_1, h_2]^\text{T} | 1\,\text{cm} \leq h_1 \leq 35\,\text{cm}, 10\,\text{cm} \leq h_2 \leq 200\,\text{cm}\}$. The system model (38) is assumed to be unknown for control design and only used for simulation. In the simulation, the goal is to reach a reference value $h_2^* = 115\,\text{cm}$ of the lower tank. Moreover, the translated state and input variables $\tilde{x} = x - [22.72, 115]^\text{T}$ and $\tilde{u} = u - 1.90$ are introduced to convert the problem into a stabilisation problem.

4.3.1 System identification

We apply a random input signal drawn from uniform distribution $U[0, 4]$ to collect observations $\mathcal{D} = \{(x(t), u(t)), x(t+1)\}$ for model identification. The sampling time is 0.9 seconds. The input and the collected state sequences are shown in Figure 14. Furthermore, 1000 samples are collected and split into training and testing sets with a ratio of 65%/35%.

Since we assume (38) is unknown, we cannot choose the scheduling variables and transform (38) into an exact LPV embedding as Hanema et al. (2021), and thus we cannot use the approach in Hanema et al. (2021) for control design. Instead, we simply use the states as the scheduling variables to learn a model in the form of (10) but treat the scheduling variables as free variables in the prediction horizon of SMPC. In particular, we use a DenseVariational layer connected to a three-layer fully-connected ANN to represent $A(\cdot)$. All the hidden layers have 32 hidden units with ELU activation functions while the output layers have 4 hidden units without activation functions. Moreover, we use one Dense layer with 2 hidden units to represent $B(\cdot)$ and the dense layer does not use bias. The tuning parameters in (7) are determined as $\sigma_1 = 1.5, \sigma_2 = 0.1$. Adam optimiser is used with a learning rate set to 0.001 and other hyper-parameters as default. Moreover, we first trained an ANN model with the same architecture as the BNN model, used the trained ANN weights to initialise the BNN model, and then trained the BNN model for 50, 000 epochs. The validation results are shown in Figure 15.

4.3.2 Validation of the proposed approach

Without assuming extra knowledge on the evolution of the scheduling variables beyond the scheduling sets, we randomly

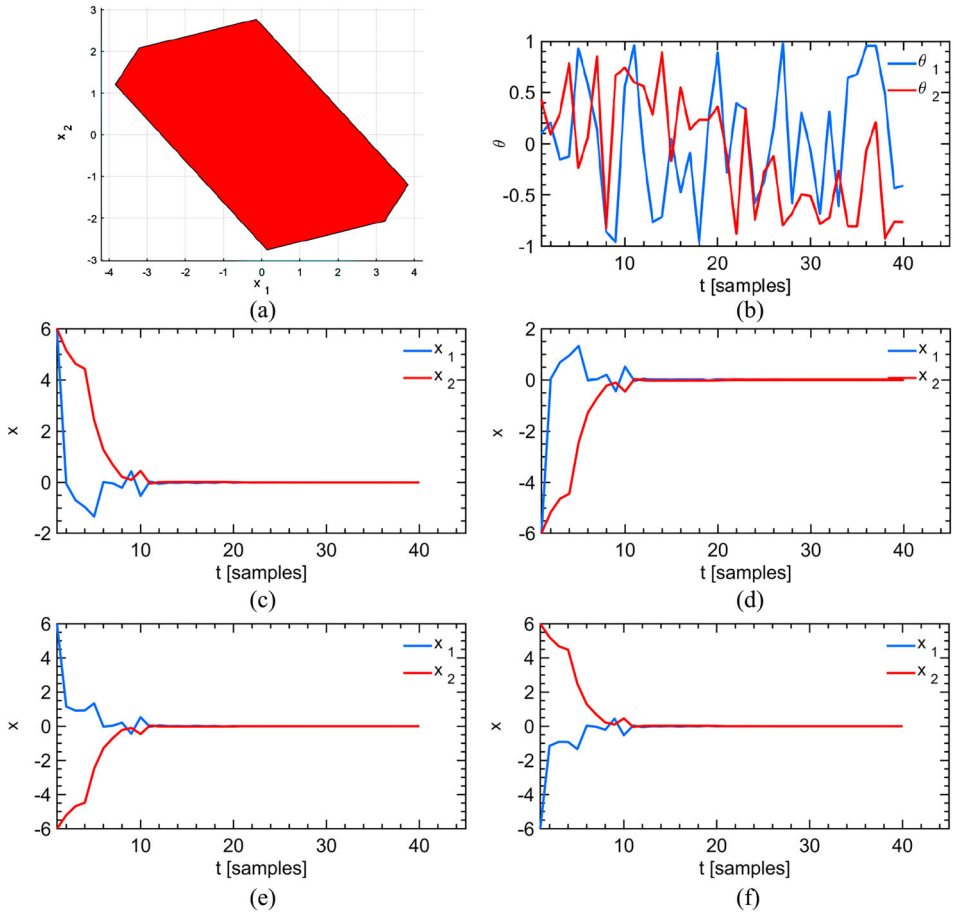

Figure 13. Control results using the proposed approach. (a) RPI set based on the BNN model. (b) Random scheduling signals for control.

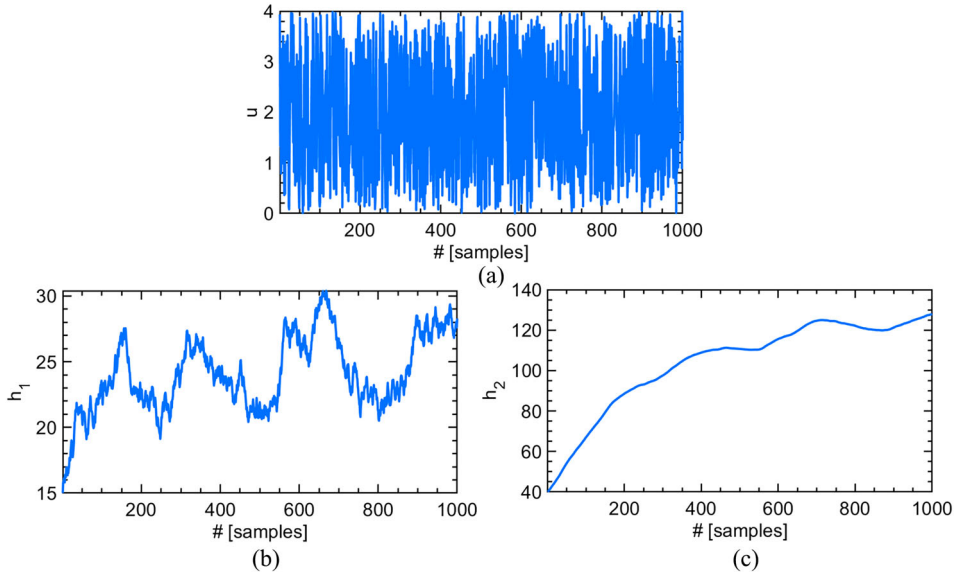

Figure 14. Data generated for system identification purposes. (a) Inputs to the system. (b) Sequences of $x_1 = h_1$. (c) Sequences of $x_2 = h_2$.

sample 1000 θ's from the uniform distribution over $\Theta = \mathbb{X}$ and then evaluate $A(\cdot)$ for $N_{\text{MC}} = 5000$ times using the dynamic functions sampled from the BNN model for each θ. Then, we apply K-means to the vectorised A's to generate the scenarios. The number of clusters is assumed to be 3. Also, $\beta_A = 0.1$ is considered here for the worst-case scenarios. Therefore, 5 scenarios were used including $\mu_A \pm \beta_A \sigma_A$. Additionally, the scenarios are fixed within the robust horizon of the tree generation in this case due to the limited time-invariant knowledge of θ. The probability of the 5 scenarios is $\mathbf{p} = [0.12; 0.81; 0.07; 0.00; 0.00]$ using the moment matching method. Moreover, in our experiments, $Q = I_{2\times 2}, R = 10$ for the stage cost ℓ in (20). The prediction horizon is set to 4 and the robust horizon to 1. Additionally, We use a 4-layer fully-connected NN with 4 and 8 units in the 2

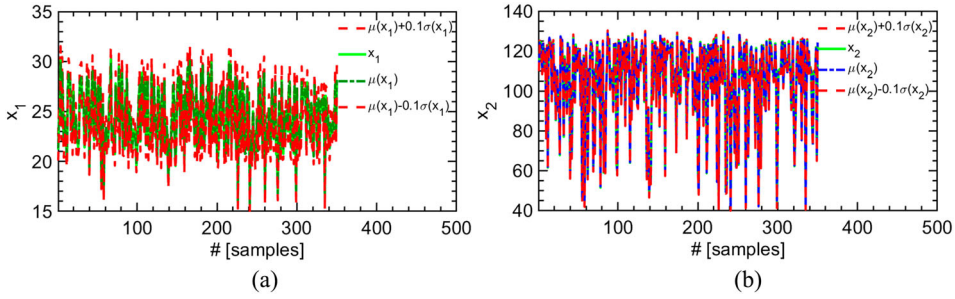

Figure 15. Validation results for the identified BNN model. The BFR = [92.04%; 97.52%] using the estimated mean as predictions for outputs. 96% of the states are within 0.1 standard deviations of the average predictions.

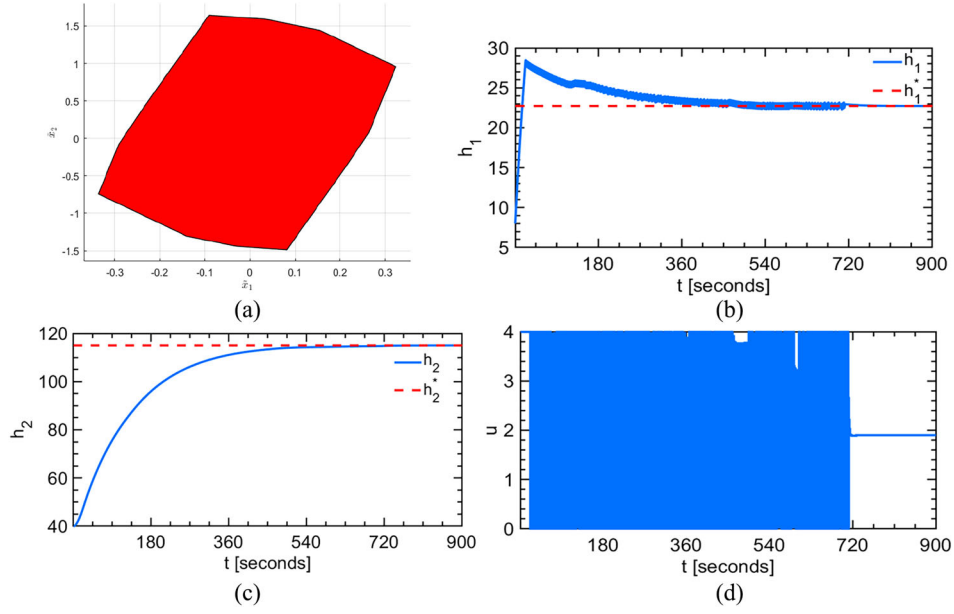

Figure 16. Control results using the proposed approach. (a) RPI set based on the BNN model. (b) x_1 profile. (c) x_2 profile. (d) Control inputs.

hidden layers to model the coordinate transformation from the scheduling variable in (10) to the scheduling variable in (21). The RPI set shown in Figure 16 (a) was computed based on the BNN model.

The control results in Figure 16(c) show that the designed SMPC can bring the liquid level h_2 of the lower tank to the reference value while satisfying the system constraints. The control inputs in Figure 16(d) fluctuate between the limits in the early stages of the control process, which may result from the conservatives of the BNN model and the generated scenarios. Moreover, it is noted that the proposed approach reached the set point slower than the approach that assumes a known system model and uses the exact LPV embedding in Hanema et al. (2021), as the data-driven model can be conservative, compared with the exact LPV model of the system. However, the data-driven model can be refined using the closed-loop data to improve the control performance, which will be investigated in the future work.

5. Concluding remarks

In this paper, a learning-based MPC design approach was proposed for systems described in the LPV framework. 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. SMPC was proposed to consider safety when generating scenarios. K-means clustering and moment matching were used to generate scenarios with probabilities that can retain the stochastic properties of the joint uncertainty of the model and the scheduling variables. To guarantee closed-loop stability, parameter-dependent terminal cost, and controller were designed, which can improve the control performance, together with a terminal RPI set. Numerical experiments and simulations were used to show that the proposed approach can ensure safety and achieve the desired control performance.

In our future work, we plan to consider the effects of measurement noise of scheduling variables on the proposed approach, as exact measurements of these parameters can be impractical in real applications. Moreover, we will improve the proposed approaches to evaluate the probabilistic safety of BNN models and develop online adaptation approaches to reduce the conservativeness of BNN models using closed-loop data.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was financially supported by National Science Foundation under award #1912757. The second author's work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under project #419290163.

References

Abbas, H. S., Tóth, R., Petreczky, M., Meskin, N., Mohammadpour Velni, J., & Koelewijn, P. J. (2021). LPV modeling of nonlinear systems: A multi-path feedback linearization approach. *International Journal of Robust and Nonlinear Control*, *31*(18), 9436–9465. https://doi.org/10.1002/rnc.v31.18

Aswani, A., Gonzalez, H., Sastry, S. S., & Tomlin, C. (2013). Provably safe and robust learning-based model predictive control. *Automatica*, *49*(5), 1216–1226. https://doi.org/10.1016/j.automatica.2013.02.003

Bao, Y., Chan, K. J., Mesbah, A., & Velni, J. M. (2022). Learning-based adaptive-scenario-tree model predictive control with probabilistic safety guarantees using Bayesian neural networks. In *2022 American control conference (ACC)* (pp. 3260–3265). IEEE. https://doi.org/10.23919/ACC53348.2022.9867798

Bao, Y., Chan, K. J., Mesbah, A., & Velni, J. M. (2023). Learning-based adaptive-scenario-tree model predictive control with improved probabilistic safety using robust Bayesian neural networks. *International Journal of Robust and Nonlinear Control*, *33*(5), 3312–3333. https://doi.org/10.1002/rnc.v33.5

Bao, Y., & Mohammadpour Velni, J. (2022). Safe control of nonlinear systems in LPV framework using model-based reinforcement learning. *International Journal of Control*, *96*(4), 1079–1090. https://doi.org/10.1080/00207179.2022.2117083.

Bao, Y., Mohammadpour Velni, J., & Shahbakhti, M. (2020). An online transfer learning approach for identification and predictive control design with application to RCCI engines. In *Dynamic systems and control conference* (Vol. 84270, pp. V001T21A003). ASME.

Bao, Y., Mohammadpour Velni, J., & 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

Bao, Y., Velni, J. M., Basina, A., & Shahbakhti, M. (2020). Identification of state-space linear parameter-varying models using artificial neural networks. *IFAC-PapersOnLine*, *53*(2), 5286–5291. https://doi.org/10.1016/j.ifacol.2020.12.1209

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

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015, June). Weight uncertainty in neural network. In *International conference on machine learning* (pp. 1613–1622). PMLR.

Bonzanini, A. D., Paulson, J. A., & Mesbah, A. (2020). Safe learning-based model predictive control under state- and input-dependent uncertainty using scenario trees. In *2020 59th IEEE conference on decision and control (CDC)* (pp. 2448–2454). IEEE. https://doi.org/10.1109/CDC42340.2020.9304310

Calafiore, G. C., & Fagiano, L. (2013). Stochastic model predictive control of LPV systems via scenario optimization. *Automatica*, *49*(6), 1861–1866. https://doi.org/10.1016/j.automatica.2013.02.060

Casavola, A., Famularo, D., & Franze, G. (2008). A predictive control strategy for norm-bounded LPV discrete-time systems with bounded rates of parameter change. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, *18*(7), 714–740. https://doi.org/10.1002/(ISSN)1099-1239

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

Cox, P. B. (2018). *Towards efficient identification of linear parameter-varying state-space models* [Unpublished doctoral dissertation]. Eindhoven University of Technology.

Defourny, B. (2010). *Machine learning solution methods for multistage stochastic programming* [PhD diss.]. University of Liege. https://www.lehigh.edu/defourny/PhDthesis_B_Defourny.pdf.

Ellis, M., Liu, J., & Christofides, P. D. (2017). Economic model predictive control. *Springer*, *5*(7), 65. https://doi.org/10.1007/978-3-319-41108-8

Gilbert, E. G., & Tan, K. T. (1991). Linear systems with state and control constraints: The theory and application of maximal output admissible sets. *IEEE Transactions on Automatic Control*, *36*(9), 1008–1020. https://doi.org/10.1109/9.83532

Gowal, S., Dvijotham, K., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T., & Kohli, P. (2018). *On the effectiveness of interval bound propagation for training verifiably robust models*. arXiv preprint arXiv:1810.12715.

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

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

Hanema, J., Tóth, R., & Lazar, M. (2021). Stabilizing non-linear model predictive control using linear parameter-varying embeddings and tubes. *IET Control Theory & Applications*, *15*(10), 1404–1421. https://doi.org/10.1049/cth2.v15.10

Hastie, T., Tibshirani, R., & Friedman, J. (2009). Model assessment and selection. In *The elements of statistical learning* (pp. 219–259). Springer. https://doi.org/10.1007/978-0-387-21606-5_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

Høyland, K., Kaut, M., & Wallace, S. W. (2003). A heuristic for moment-matching scenario generation. *Computational Optimization and Applications*, *24*(2/3), 169–185. https://doi.org/10.1023/A:1021853807313

Høyland, K., & Wallace, S. W. (2001). Generating scenario trees for multistage decision problems. *Management Science*, *47*(2), 295–307. https://doi.org/10.1287/mnsc.47.2.295.9834

Ji, X., Zhu, S., Wang, S., & Zhang, S. (2005). A stochastic linear goal programming approach to multistage portfolio management based on scenario generation via linear programming. *Iie Transactions*, *37*(10), 957–969. https://doi.org/10.1080/07408170591008082

Koller, T., Berkenkamp, F., Turchetta, M., Boedecker, J., & Krause, A. (2019). *Learning-based model predictive control for safe exploration and reinforcement learning*. arXiv preprint arXiv: 1906.12189.

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

Lee, J., & Yu, Z. (1997). Worst-case formulations of model predictive control for systems with bounded parameters. *Automatica*, *33*(5), 763–781. https://doi.org/10.1016/S0005-1098(96)00255-5

Liu, H., Ong, Y. S., Shen, X., & Cai, J. (2020). When Gaussian process meets big data: A review of scalable GPs. *IEEE Transactions on Neural Networks and Learning Systems*, *31*(11), 4405–4423. https://doi.org/10.1109/TNNLS.5962385

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, *28*(2), 129–137. https://doi.org/10.1109/TIT.1982.1056489

Lucia, S., Finkler, T., & Engell, S. (2013). Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty. *Journal of Process Control*, *23*(9), 1306–1319. https://doi.org/10.1016/j.jprocont.2013.08.008

Luo, Q., Nguyen, A. T., Fleming, J., & Zhang, H. (2021). Unknown input observer based approach for distributed tube-based model predictive control of heterogeneous vehicle platoons. *IEEE Transactions on Vehicular Technology*, *70*(4), 2930–2944. https://doi.org/10.1109/TVT.2021.3064680

Maiworm, M., Bäthge, T., & Findeisen, R. (2015). Scenario-based model predictive control: Recursive feasibility and stability. *IFAC-PapersOnLine*, *48*(8), 50–56. https://doi.org/10.1016/j.ifacol.2015.08.156

Mayne, D. Q., Rawlings, J. B., Rao, C. V., & Scokaert, P. O. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, *36*(6), 789–814. https://doi.org/10.1016/S0005-1098(99)00214-9

Mesbah, A. (2018). Stochastic model predictive control with active uncertainty learning: A survey on dual control. *Annual Reviews in Control*, *45*, 107–117. https://doi.org/10.1016/j.arcontrol.2017.11.001

Morato, M. M., Normey-Rico, J. E., & Sename, O. (2020). Model predictive control design for linear parameter varying systems: A survey. *Annual Reviews in Control*, *49*, 64–80. https://doi.org/10.1016/j.arcontrol.2020.04.016

Nguyen, H.-N. (2014). Set theoretic methods in control. In Constrained control of uncertain, time-varying, discrete-time systems (pp. 7–42). Springer International Publishing. https://doi.org/10.1007/978-3-319-02827-9_2

Pandey, A., & de Oliveira, M. C. (2017). Quadratic and poly-quadratic discrete-time stabilizability of linear parameter-varying systems. *IFAC-PapersOnLine*, *50*(1), 8624–8629. https://doi.org/10.1016/j.ifacol.2017.08.1512

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

Shapiro, A. (2003). Monte Carlo sampling methods. *Handbooks in Operations Research and Management Science*, *10*, 353–425. https://doi.org/10.1016/S0927-0507(03)10006-0

Wicker, M., Laurenti, L., Patane, A., & Kwiatkowska, M. (2020). *Probabilistic safety for Bayesian neural networks*. arXiv preprint arXiv:2004.10281.

Wills, A., & Ninness, B. (2012). System identification of linear parameter varying state-space models. In P. Lopes dos Santos, T. P. Azevedo Perdicoulis, & C. Novara (Eds.), *Linear parameter-varying system identification: New developments and trends* (pp. 295–315). World Scientific.

Xu, D., Chen, Z., & Yang, L. (2012). Scenario tree generation approaches using K-means and LP moment matching methods. *Journal of Computational and Applied Mathematics*, *236*(17), 4561–4579. https://doi.org/10.1016/j.cam.2012.05.020

Zhang, H., Weng, T. W., Chen, P. Y., Hsieh, C. J., & Daniel, L. (2018). Efficient neural network robustness certification with general activation functions. *Advances in Neural Information Processing Systems*, *31*.