2022 American Control Conference (ACC)
Atlanta, USA, June 8-10, 2022

Learning-based Adaptive-Scenario-Tree Model Predictive Control with Probabilistic Safety Guarantees Using Bayesian Neural Networks

Yajie Bao, Kimberly J. Chan, Ali Mesbah, and Javad Mohammadpour Velni

Abstract— **This paper proposes a learning-based adaptive-scenario-tree model predictive control (MPC) approach with probabilistic safety guarantees using Bayesian neural networks (BNNs) for nonlinear systems. First, a data-driven description of the model uncertainty (i.e., plant-model mismatch) is learned using a BNN. Then, the learned description is employed to generate adaptive scenarios online for scenario-based MPC (sMPC). To accurately represent the evolution of uncertainties, we use a moment-matching method to compute the probabilities of the generated time-varying scenarios. Moreover, probabilistic safety guarantees are provided by ensuring that the trajectories of the scenarios contain the real trajectory of the system and all the generated scenarios satisfy the constraints with a high probability. By realizing a less conservative estimation of the model uncertainty, the proposed approach can improve robust control performance with respect to sMPC with a fixed scenario tree. Closed-loop simulations on a cold atmospheric plasma system with prototypical applications in (bio)materials processing demonstrate that the proposed approach results in an improved control performance compared to sMPC with a fixed scenario tree.**

I. Introduction

Model uncertainty is a common challenge in model-based control of safety-critical systems, where plant-model mismatch of the model to the system can result in unsafe operation [1], [2]. This mismatch can arise from a variety of factors (e.g., hard-to-model and time-varying system dynamics and disturbances) and, as a result, has a variety of assumed representations [3]–[5]. Learning-based strategies provide an interesting avenue of exploration because they can strategically incorporate "learned" knowledge about the system from data. As a result, learning-based MPC has gained increasing interest for the control of complex safety-critical systems operating in uncertain and hard-to-model environments [6]–[8]. While the system and environment dynamics can be learned from data to improve control performance and constraint satisfaction, the statistical nature of learning-based approaches introduces important challenges in guaranteeing robust constraint satisfaction [9].

Gaussian process (GP) regression is a popular choice for describing plant-model mismatch [7], [10], [11], because it provides a means to capture the time-varying and state-dependent nature of structural uncertainty (i.e., plant-model mismatch) [4], [12]. However, GPs have a few disadvantages with regard to providing a nonlinear description of the plant-model mismatch: i.) cubic complexity to data size while a variety of scalable GPs have been presented [13], and ii.) reliance on the assumption that the mismatch can be mapped with jointly Gaussian distributions. With these disadvantages in mind, we choose Bayesian neural networks (BNNs) to model the mismatch. First, BNNs provide a fast evaluation of the mismatch as well as a fast update of the statistical properties of the mismatch estimation. In particular, BNNs assume the parameters in the neural networks are random variables with given prior distributions and will approximate the posterior distributions conditioned on data using variational inference [14]. Furthermore, BNNs are general in that they can approximate arbitrary posteriors given a prior, which can enable us to model and predict a variety of mismatch representations.

In addition to the explicit quantification of the plant-model mismatch by the BNN, we must also utilize a control strategy that explicitly accounts for different realizations of the model uncertainty. A common strategy to account for uncertainties in MPC is to construct a tree of discrete uncertainty realizations, as done in scenario-based MPC (sMPC) [15]. By approximating the continuous mismatch description into discrete scenarios, sMPC solves an optimal control problem that incorporates an estimation of the deviation between the model and system. By creating a state- and input-dependent BNN-based mismatch model, as in [11], we address a limitation encountered in sMPC which involves the offline generation of time-invariant scenarios based on worst-case uncertainty descriptions. As such, the mismatch description makes the scenario tree adaptive by nature and, as such, can reduce the conservatism with respect to the worst-case, fixed scenario tree. Furthermore, we use a moment-matching technique estimate the probability of each scenario in order to: i.) explicitly account for the probability of each uncertainty scenario in the sMPC problem and ii.) establish probabilistic guarantees for constraint satisfaction. Towards safe control, different from [11], we enforce constraints of the system by ensuring that the trajectories of the scenarios contain the real trajectory of the system and requiring all such scenarios to satisfy the constraints.

The main contribution of this paper lies in the development of an adaptive-tree sMPC scheme with safety guarantees using the state- and input-dependent model uncertainty (i.e., plant-model mismatch) that is described by a BNN. The remainder of the paper is organized as follows: Section II introduces the problem formulation and related preliminaries.

Y. Bao and J. Mohammadpour Velni are with the School of Electrical and Computer Engineering at the University of Georgia, Athens, GA 30602, USA. {yajie.bao, javadm}@uga.edu

K. J. Chan and A. Mesbah are with the Department of Chemical and Biomolecular Engineering at the University of California, Berkeley, CA 94720, USA. {kchan45, mesbah}@berkeley.edu

This work was supported by the US National Science Foundation under grant 1912757 and 1912772.

Scenario generation and the probabilistic safety guarantee are discussed in Section III. Section IV presents closed-loop simulation results to demonstrate the proposed approach. Concluding remarks are finally provided in Section V.

II. Problem Statement and Related Preliminaries

Consider a constrained, discrete-time nonlinear system with state- and input-dependent uncertainty of the form

$$x(k+1) = f(x(k), u(k)) + B_d(g(x(k), u(k)) + d), \quad (1)$$

where x is the state, $k \in \mathbb{N}$ is the time instant, u is the control input, and $d \in \mathcal{D} \subseteq \mathbb{R}^{n_d}$ is normally-distributed process noise. $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ represents the known nominal part of the model (1) while $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathcal{D}$ describes unknown structural mismatch that lies in the subspace spanned by the known matrix $B_d \in \mathbb{R}^{n_x \times n_d}$. B_d can be used to embed the knowledge of how the noise affects the states and assumed to be the identity matrix when such knowledge does not exist. Additionally, the state and input constraints are described by

$$x \in \mathcal{X} \subseteq \mathbb{R}^{n_x}, \quad u \in \mathcal{U} \subseteq \mathbb{R}^{n_u}. \quad (2)$$

We aim to design an adaptive-scenario-tree MPC (asMPC) which incorporates the state- and input-dependent description of the mismatch denoted by $y \triangleq g+d$. Using $\kappa : \mathcal{X} \times \mathbb{N} \to \mathcal{U}$ to denote the asMPC law, the closed-loop system can be described by

$$\begin{aligned} &x(k+1) \\ =&f(x(k), \kappa(x(k), k)) + B_d(g(x(k), \kappa(x(k), k) + d) \\ \triangleq&\Phi_\kappa(x(k), d, k). \end{aligned} \quad (3)$$

Additionally, we use $\mathbf{x}(k|x_0)$ to denote the solution $x(k)$ to (3), given the initial state x_0.

Definition 1: Given an initial state $x_0 \in \mathcal{X}$, the system (1) is said to be **safe** under a control law κ if for $\forall k \in \mathbb{N}$,

$$\Phi_\kappa(x(k), d, k) \in \mathcal{X}, \quad \kappa(x(k), k) \in \mathcal{U}. \quad (4)$$

Moreover, the system (1) is said to be δ**-safe** under the control law κ if $\forall k \in \mathbb{N}$,

$$\Pr\left[\Phi_\kappa(x(k), d, k) \in \mathcal{X}, \kappa(x(k), k) \in \mathcal{U}\right] \geq 1 - \delta, \quad (5)$$

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

In general, the hard constraints (4) cannot be enforced without additional assumptions [7], especially when (1) is unknown. However, δ-safety relaxes the requirements of safety to *safety with a high probability*.

Here, we use Bayesian neural networks (BNNs) [14] to model the mismatch y, which is comprised of g and d together. BNNs provide an ensemble of models such that the unknown system behavior is contained within the model ensemble. Then, safety may be ensured by requiring all models to satisfy (2).

A. Plant-model Mismatch Quantification using BNNs

In this paper, we use a multi-layer, fully-connected BNN as depicted in Fig. 1 to model the structural mismatch y comprised of the noisy vector-valued function g and the process noise d. The N_s-sample dataset for training the BNN is defined as

$$\mathcal{T} = \{\mathrm{x}^{(j)} = (x^{(j)}, u^{(j)}), \mathrm{y}^{(j)}\}_{j=1}^{N_s} \quad (6)$$

where $\mathrm{x}^{(j)}$ is the j-th input datum to the BNN and $\mathrm{y}^{(j)} = B_d^\dagger(x(k+1) - f(x(k), u(k))) = g(x^{(j)}, u^{(j)}) + d$ is the j-th output datum, representing the plant-model mismatch.

A BNN treats the parameters of the models as random variables and are composed of DenseVariational layers. The DenseVariational layers approximate the posterior density of the parameters by variational inference (VI) given a prior density. Specifically, VI solves

$$\min_{\theta_j} \ \mathrm{KL}\Big(q(w_j; \theta_j) \| p(w_j | \mathcal{T})\Big) \quad (7)$$

$$\begin{aligned} \Leftrightarrow \quad \min_{\theta_j} \ &\Big(\mathbb{E}_{q(w_j;\theta_j)}\left[\log q(w_j; \theta_j)\right] - \mathbb{E}_{q(w_j;\theta_j)}\left[\log p(w_j)\right] \\ &- \mathbb{E}_{q(w_j;\theta_j)}\left[\log p(\mathcal{T} | w_j)\right]\Big), \end{aligned} \quad (8)$$

where w_j denotes the parameters in the j-th layer of the BNN, $q(w_j; \theta_j)$ denotes a family of densities with parameters θ_j, and $p(w_j|\mathcal{T})$ denotes the posterior. To solve (8) by Monte Carlo (MC) methods and backpropagation, a reparameterization trick is used to parameterize $q(w_j; \theta_j)$, i.e.,

$$w_j = \mu_{w_j} + \sigma_{w_j} \bigodot \epsilon_{w_j} \quad (9)$$

where $\bigodot$ denotes element-wise multiplication, $\epsilon_{w_j} \sim \mathcal{N}(0, I)$, and thus $\theta_j = (\mu_{w_j}, \sigma_{w_j})$.

The BNN is trained by minimizing

$$\frac{1}{N_{\text{BNN}}} \sum_{i=1}^{N_{\text{BNN}}} \left[\log q(w^{(i)}; \theta) - \log p(w^{(i)}) - \log p(\mathcal{T} | w^{(i)})\right] \quad (10)$$

over θ via stochastic gradient descent where $w^{(i)}$ are the i-th MC sample for approximating the (10), and N_{BNN} is the MC sample size determined such that (10) is convergent.

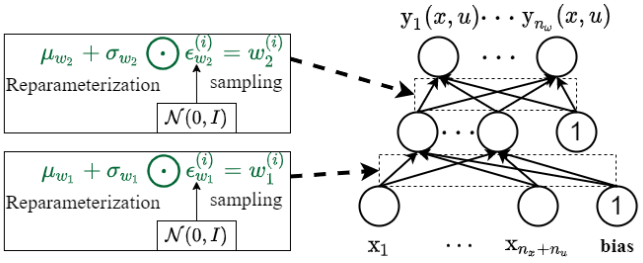


Fig. 1: A BNN composed of multiple DenseVariational layers with reparameterization trick. BNNs use data to estimate the parameters μ_w and σ_w of the posterior density function.

Using the trained BNN, the density of $\hat{\mathrm{y}} = \mathrm{BNN}(\mathrm{x})$ at given $(x(k), u(k))$ can be evaluated by drawing samples from the posteriors of weights and calculating the possible $\hat{\mathrm{y}}$'s with each set of sampled weights. Rather than directly estimating the density from samples, we calculate the moments,

such as the mean and standard deviation of each dimension of $\hat{\mathrm{y}}$, which is efficient and sufficient for constructing a confidence interval of $x(k+1)$ to check (5). To provide safety guarantees, we need reliable estimates of the mismatch y inside the operating region $\mathcal{X}\times\mathcal{U}$, which is similar to [16] and formally described in the following assumption:

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

$$\begin{aligned}&\forall k\in\mathbb{N}, |y_i(k+1)-\hat{\mu}_{\hat{\mathrm{y}}_i(k+1)}| \le \beta_i\hat{\sigma}_{\hat{\mathrm{y}}_i(k+1)} < |\mathcal{Y}_i|,\\ &i=1,2,\cdots,n_x,\end{aligned} \tag{11}$$

given $(x(k),u(k))\in\mathcal{X}\times\mathcal{U}$ for any $d\in\mathcal{D}$, where $\hat{\mu}_{\hat{\mathrm{y}}_i(k+1)}$ and $\hat{\sigma}_{\hat{\mathrm{y}}_i(k+1)}$ denote the estimated mean and standard deviation of the i-th entry of $\hat{\mathrm{y}}(k+1)$, respectively, using the learned BNN model with Monte Carlo methods, and $|\mathcal{Y}_i|$ is used to denote the range of valid y_i.

By Assumption 1, the learned model is sufficiently accurate such that the values of y are contained in the confidence intervals of our statistical model. It is noted that a larger $\beta_i\hat{\sigma}_{\hat{\mathrm{y}}_i(k+1)}$ means larger uncertainties of the model and gives a more conservative estimate of $y_i(k+1)$. This overestimates the probability of constraints violation and reduces the feasible region of control inputs, leading to degraded control performance. If $\beta_i\hat{\sigma}_{\hat{\mathrm{y}}_i(k+1)} \ge |\mathcal{Y}_i|$, the estimate is worse than a random guess of $y_i(k+1)$, and therefore, not useful for control. The above assumption can be enforced by a proper BNN and empirically verified on the testing set after model training. Moreover, δ is estimated as the relative frequency of the training data that violates (11) given β and validated using the testing data.

Furthermore, an input sequence is **valid** for a system with x_0 if applying the input sequence to the system is safe.

Lemma 1: Given x_0, a valid control input sequence $\mathbf{u}$, a BNN model that fulfills Assumption 1, and a confidence level δ, there exists a $\bar{N}_{\mathrm{MC}}$ such that $\forall k\in\mathbb{N}$,

$$\begin{aligned}&\Pr\left[\mathbf{x}_j(k|x_0)\in[\hat{\mathbf{x}}_{j,\min}(k|x_0),\hat{\mathbf{x}}_{j,\max}(k|x_0)]\right]\\ \ge& 1-\delta, j=1,\cdots,n_x\end{aligned} \tag{12}$$

where $\hat{\mathbf{x}}_{j,\min}(k|x_0)=\min_i \hat{\mathbf{x}}_j^{(i)}(k|x_0)$ and $\hat{\mathbf{x}}_{j,\max}(k|x_0)=\max_i \hat{\mathbf{x}}_j^{(i)}(k|x_0)$. In particular, $\hat{\mathbf{x}}_j^{(i)}(k|x_0) = f(\hat{\mathbf{x}}_j^{(l)}(k-1|x_0),\mathbf{u}(k-1))+B_d\hat{\mathrm{y}}^{(i)}(\hat{\mathbf{x}}_j^{(l)}(k-1|x_0),\mathbf{u}(k-1))$ where $\hat{\mathrm{y}}^{(i)}$ is the prediction of $y(\hat{\mathbf{x}}_j^{(l)}(k-1|x_0),\mathbf{u}(k-1))$ using the i-th sampled model from the BNN model, $i=1,\cdots,N_{\mathrm{MC}}(k)$, $l=1,\cdots,N_{\mathrm{MC}}(k-1)$. Specifically, $N_{\mathrm{MC}}(k)$ is used to denote the number of models drawn from the BNN model using MC methods at time instant k, and $\bar{N}_{\mathrm{MC}}=\max_k N_{\mathrm{MC}}(k)$.

Proof: When $k=0$, $\hat{x}_0=x_0$. Then, using Assumption 1, there exists a $N_{\mathrm{MC}}(0)$ at time instant 0 such that

$$\begin{aligned}\mathbf{x}_j(1|x_0) &= f_j(x_0,\mathbf{u}(0)) + B_{d,[j,:]}\mathrm{y}(x_0,\mathbf{u}(0))\\ &\in[\hat{\mathbf{x}}_{j,\min}(1|x_0),\hat{\mathbf{x}}_{j,\max}(1|x_0)], j=1,\cdots,n_x\end{aligned}$$

hold almost surely, i.e., $\delta\to 0$, as $|y_j(x_0,\mathbf{u}(0)) - \hat{\mu}_{\mathrm{y}_j(x_0,\mathbf{u}(0))}| \le \beta_j\hat{\sigma}_{\mathrm{y}_j(x_0,\mathbf{u}(0))}$ and the support of the weights in the BNN model are unbounded. Using induction, (12) is obtained using $\bar{N}_{\mathrm{MC}}=\max_k N_{\mathrm{MC}}(k)$. ∎

Lemma 1 guarantees that, with a high probability, the real system state trajectory is always contained in the multiple trajectories simulated by the BNN model.

III. sMPC Design Using BNNs

In this section, we detail the sMPC using the BNN model. In particular, the scenario generation method is presented, and the probabilistic safety guarantee is formulated.

A. Scenario-based Model Predictive Control

sMPC assumes that the uncertainty of a system may be represented by a tree of discrete scenarios. Any particular branch stemming from a node represents a particular scenario of an unknown, uncertain influence (e.g., from a disturbance or model error) [17]. To represent the trajectories generated by some number S scenarios, we adopt the notation $(x^s(i),u^s(i))$, where the addition of the superscript s without parentheses indicates the particular scenario $s\in\{1,\ldots,S\}$. The sMPC for an uncertain system at time step k can then be formulated as follows

$$\min_{x^s,u^s} \quad \sum_{s=1}^{S} p^s\left[\sum_{i=0}^{N-1} V\left(x^s(i),u^s(i)\right)\right] \tag{13a}$$

$$\text{s.t.} \quad x^s(i+1)=f\left(x^s(i),u^s(i)\right)+\hat{\mathrm{y}}^s(i), \tag{13b}$$

$$(x^s(i),u^s(i))\in\mathcal{X}\times\mathcal{U}, \tag{13c}$$

$$x^s(0)=x(k), \tag{13d}$$

$$u^s(i)=u^l(i) \text{ if } x^{b(s)}(i)=x^{b(l)}(i), \tag{13e}$$

where p^s is the probability of a particular scenario, $V\left(x^s(i),u^s(i)\right)$ is the standard MPC cost (which can be comprised of a stage cost and terminal cost) for the trajectory of a particular scenario, N is the prediction horizon, $\hat{y}^s$ is generated based on some uncertainty estimation strategy, and (13e) enforces a *non-anticipativity* constraint, which represents the fact that each control input that branches from the same parent node must be equal ($x^{b(s)}(i)$ is the parent state of $x^s(i+1)$). The non-anticipativity constraint is crucial so that the control inputs do not anticipate the future (i.e., decisions cannot realize the uncertainty). The solution to this optimization problem is used to generate the control law,

$$\kappa\left(x(k)\right)=u^\star(0). \tag{14}$$

A potential challenge in the sMPC is the exponential nature of the scenario tree formulation. To combat this, we utilize a method described in [17], in which a robust horizon $N_r<N$ is defined. This robust horizon stops the branching of the scenario tree up to a certain stage, and the uncertainty is assumed to be constant thereafter. Consequently, assuming the number of scenarios n_s at each node of a stage is constant, the total number of scenarios S can be represented by $S=n_s^{N_r}$.

Here, we propose the use of BNNs to estimate the plant-model mismatch. Using BNNs to represent the mismatch not only allows us to have a more expressive uncertainty estimation with time variance and state dependence, but also allows us to update and compute the mismatch estimations online. The details of this update is outlined in the following

section, and we denote this particular form of adaptable-tree sMPC as adaptive-scenario-tree MPC (asMPC).

B. Learning-based Scenario Generation

At each time step k, we draw $\bar{N}_{\text{MC}}$ samples from normal distributions and calculate weights $w^{(i)}$ by applying the transformation (9) of the reparameterization trick to the i-th sample. While Lemma 1 claims that the trajectories of the sampled $\bar{N}_{\text{MC}}$ models contain the system trajectory, $\bar{N}_{\text{MC}}$ can be too large for online optimization of the sMPC problems. To reduce the number of scenarios, instead, we estimate $\hat{\text{y}}^{(i)}$ using $w^{(i)}$, compute

$$\hat{\mu}_{\hat{\text{y}}(k)} = \frac{1}{\bar{N}_{\text{MC}}} \sum_{i=1}^{\bar{N}_{\text{MC}}} \hat{\text{y}}^{(i)}, \tag{15}$$

$$\hat{\sigma}_{\hat{\text{y}}(k)} = \sqrt{\frac{1}{\bar{N}_{\text{MC}}} \sum_{i=1}^{\bar{N}_{\text{MC}}} \|\hat{\text{y}}^{(i)} - \hat{\mu}_{\hat{\text{y}}(k)}\|_2^2}, \tag{16}$$

and use $\hat{\mu}_{\hat{\text{y}}(k)}$, $\hat{\mu}_{\hat{\text{y}}(k)} + a^j \hat{\sigma}_{\hat{\text{y}}(k)}$, $\hat{\mu}_{\hat{\text{y}}(k)} - a^j \hat{\sigma}_{\hat{\text{y}}(k)}, j = 1, \cdots, \frac{n_s-1}{2}$ where a^j are the tuning multipliers, as n_s scenarios at each node of a stage. To maintain the original statistical properties of the posteriors by the BNN, the probability of scenarios are calculated using moment-matching method [18]. Specifically, the first four central moments are matched by solving the following optimization problem

$$\begin{aligned}
\min_{\mathbf{p}} \sum_{i}^{m} \Big(& v_i^1 \left(M_i^- + M_i^+\right) + v_i^3 \left(T_i^- + T_i^+\right) \\
& + v_i^4 \left(Q_i^- + Q_i^+\right) \Big) + \sum_{i,j=1}^{m} v_{i,j}^1 \left(\Sigma_{i,j}^- + \Sigma_{i,j}^+\right), \\
\text{s.t.} \quad & \mathbf{X}\mathbf{p} + M^- - M^+ = M \\
& \sum_{i=1}^{n_s} (\mathbf{X}^i - \mathbf{X}\mathbf{p})^2 p^i + \Sigma^- - \Sigma^+ = \Sigma, \\
& \sum_{i=1}^{n_s} (\mathbf{X}^i - \mathbf{X}\mathbf{p})^3 p^i + T^- - T^+ = T, \\
& \sum_{i=1}^{n_s} (\mathbf{X}^i - \mathbf{X}\mathbf{p})^4 p^i + Q^- - Q^+ = Q, \\
& \sum_{i=1}^{n_s} p^i = 1, \quad p^i \geq 0, \quad \forall i \in [1, n_s], \\
& M_i^+, M_i^-, T_i^+, T_i^-, Q_i^+, Q_i^- \geq 0, \quad \forall i \in [1, m], \\
& \Sigma_{ij}^+, \Sigma_{ij}^- \geq 0, \quad \forall i, j \in [1, m],
\end{aligned}$$

where $(\mathbf{X}^i - \mathbf{X}\mathbf{p})^n$ denotes the n-th central moment, M, Σ, T, and Q are the first four central moments estimated from samples[1], and $v_i^0, v_{i,j}^1, v_i^3, v_i^4$ are weighting coefficients. Furthermore, $\mathbf{p} = (p^1, \cdots, p^{n_s})^{\text{T}}$, where p^i is the probability of the i-th scenario, and $\mathbf{X} = (\mathbf{X}^1, \cdots, \mathbf{X}^{n_s}) \in \mathbb{R}^{m \times n_s}$, where $\mathbf{X}^i = (X_1^i, \cdots, X_m^i)$ denotes the realization of the uncertainty in the i-th scenario, and m is the dimension of the realization. The number n_s are determined such that the cost value of the optimization problem is acceptable.

Remark 1: It is noted that the computational cost of the proposed scenario generation approach and moment-matching method is high when $\bar{N}_{\text{MC}}$ is large as well as with regards to the need to solve a second optimization problem. However, the computations can be done offline via a uniform-grid approach. Specifically, we discretize $\mathcal{X} \times \mathcal{U}$ using uniform grids, evaluate the BNN model at the grid points for $\bar{N}_{MC}$ times such that Lemma 1 is fulfilled, and solve the moment-matching optimization problem. The grid size is determined such that the estimation of $\hat{\mu}_{\hat{\text{y}}}$ and $\hat{\sigma}_{\hat{\text{y}}}$ is stable. Thus, the scenarios and the probability of scenarios at (x, u) can be retrieved from the offline computation results by finding the results at the grid point that is closest to (x, u).

To save computational cost, we only update the uncertainty estimation every time step and fix the scenarios over the prediction horizon. In particular, we use the solution $u^*(1|k-1)$ to (13) at $k-1$ and the state $x(k)$ to estimate the overall source of uncertainty $y(k)$ at k, and $\hat{y}(i|k) = \hat{y}(k), i = 0, \cdots, N-1$ for (13) at k. Generation of the scenarios in this data-driven framework is two-fold: i.) it allows us to update the uncertainty estimation at each time step k, which adapts the scenario tree within the sMPC; ii.) it allows us to compute the probabilities of each scenario, which allows us to assign a probabilistic safety certification, as discussed in the following subsection. Furthermore, we are able to update our beliefs about the mismatch between the true system and the model as we gain more experience through interactions with the system, by systematically updating the priors of the parameters of the BNN with new data. Then, the BNN may be updated under the framework in [19] such that it provides a more informative description of the mismatch.

C. Probabilistic Safety Guarantee

Using the scenario generation approach in III-B, the certificate of safety can be formalized into our main result.

Theorem 1: Let the hypotheses of Assumption 1 and Lemma 1 be satisfied. Then, the system under the scenario-based MPC law is δ-safe if (13) is recursively feasible.

Proof: By Lemma 1, (12) holds using $\bar{N}_{\text{MC}}$ samples. Consequently, at time step k, there exists a^j for the scenario generation using $\hat{\mu}_{\hat{\text{y}}(k)}$ and $\hat{\sigma}_{\hat{\text{y}}}(k)$ estimated from the $\bar{N}_{\text{MC}}$ samples such that the predictions $\hat{x}(k+1)$ by the generated scenarios contain the real $x(k+1)$ of the system under Assumption 1. Furthermore, (13) is feasible at every step, thus (5) holds for all k, which proves the system is δ-safe by Definition 1. ■

IV. Case Study

The asMPC approach is demonstrated via closed-loop simulations using a physics-based model of a RF-excited atmospheric pressure plasma jet (APPJ) in Argon. The plant model is described by a differential-algebraic system of equations (see [20] for details). Similar to [21], subspace identification was used to obtain the linear time-invariant

[1]The superscripts $+, -$ denote the positive and negative parts of the variable.

(LTI) model

$$x(k+1) = \begin{bmatrix} 0.68 & 0.05 \\ 0.69 & 0.20 \end{bmatrix} x(k) + \begin{bmatrix} 0.86 & -0.17 \\ 6.19 & -12.10 \end{bmatrix} u(k)$$

as the nominal prediction model $f(x,u)$ in (1). The states are $x = [T_s; T_g]$ where T_s is the surface temperature and T_g is the gas temperature, and the inputs are $u = [P; q]$ where P is the applied power and q is the flow rate of Argon. The thermal dose delivered to the target surface measured in terms of cumulative equivalent minutes (CEM) is defined as

$$\text{CEM}(k+1) = \text{CEM}(k) + K^{(43-T_s(k))}\delta t,$$

where $K = \begin{cases} 0.5, & \text{if } T_s \geq 35 \text{ °C} \\ 0, & \text{otherwise} \end{cases}$. The control objective is to achieve a specified thermal dose CEM_{sp} while satisfying the constraints

$$25 \text{ °C} \leq T_s \leq 42.5 \text{ °C}; 20 \text{ °C} \leq T_g \leq 80 \text{ °C};$$
$$1.5 \text{ W} \leq P \leq 8 \text{ W}; 1.0 \text{ slm} \leq q \leq 6 \text{ slm}.$$

A. BNN Representation of the Plant-Model Mismatch

1) Trajectory Data: Input-output data was gathered by applying an input sequence to the plant model. This data is transformed into the dataset $\mathcal{T}$ described in II-A for BNN training. The mismatch in predictions of T_s and T_g are denoted by y_1 and y_2, respectively. The dataset consisted of $22,366$ samples and was randomly split into training and testing sets by 67%/33%.

2) BNN Structure and Training: A DenseVariational layer with linear activation connected to a four-layer fully-connected neural network with the Exponential Linear Unit activation functions was used to represent the plant-model mismatch y. The prior $p(W^5) = \pi\mathcal{N}(W^5|0,(\sigma_1)^2) + (1-\pi)\mathcal{N}(W^5|0,(\sigma_2)^2)$ with $\pi = 0.5$, $\sigma_1 = 1.5$, and $\sigma_2 = 0.1$. Each of the three hidden layers has 32 units. As in [14], we first trained an ANN model that shares the same architecture with the BNN model and then transferred the weights of the ANN model to the BNN model to improve the training efficiency of the BNN model. The BNN was trained using the Adam optimizer in Keras [22]. The learning rate of Adam was set to 10^{-3} and decay to 10^{-6}. All other parameters of Adam were left as default. We trained the model for $10,000$ epochs with batch size 16.

3) BNN Prediction Evaluation: Validation results of the learned BNN model based on the testing dataset are shown in Fig. 2 for the surface temperature T_s. 97.82% of y_1 in the training dataset are within $[\hat{\mu}_{\hat{\text{y}}_1} - 3\hat{\sigma}_{\hat{\text{y}}_1}, \hat{\mu}_{\hat{\text{y}}_1} + 3\hat{\sigma}_{\hat{\text{y}}_1}]$ while 96.90% of y_1 in the testing dataset are within $[\hat{\mu}_{\hat{\text{y}}_1} - 3\hat{\sigma}_{\hat{\text{y}}_1}, \hat{\mu}_{\hat{\text{y}}_1} + 3\hat{\sigma}_{\hat{\text{y}}_1}]$. Moreover, the estimation of σ_y is very conservative at some testing points, as these points are not well represented in the training data.

B. sMPC Formulation

As stated above, the control objective is to deliver a specified thermal dose CEM_{sp} to a target surface. Thus, the objective function of the sMPC formulation is defined as

$$V(\text{CEM}(k)) = ||\text{CEM}_{\text{sp}} - \text{CEM}(N)||^2, \qquad (17)$$

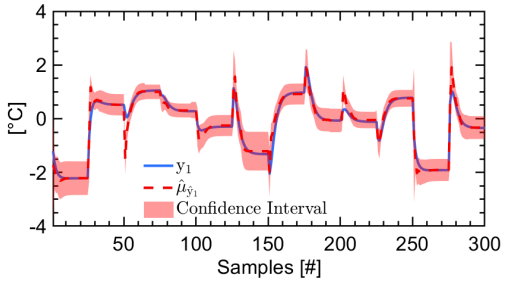


Fig. 2: Mean predictions of the BNN model of the mismatch along with a confidence interval of three estimated standard deviations $\hat{\sigma}_{\hat{\text{y}}_1}$ around the estimated mean $\hat{\mu}_{\hat{\text{y}}_1}$ are represented as a red, shaded area and a dashed red line. The true values of the mismatch y_1 are denoted by the solid blue line.

where $\text{CEM}(k)$ denotes the current CEM dose achieved at time step k and $\text{CEM}(N)$ is the predicted CEM by the end of the prediction horizon N. For comparison, we examine the performance of several variants of sMPC. In the first case, we examine the common approach of precomputing the worst-case uncertainty bounds, which is denoted as "worst-case" sMPC. The worst-case sMPC assumes a fixed scenario tree with scenarios generated based on the maximum error of estimation between the plant and the nominal model. Additionally, the scenarios were weighted uniformly, as we made no assumption of the prevalence of over-/under-estimation. In the second case, we examine the performance of sMPC using $\hat{\mu}_{\hat{y}}$ as the only scenario. This mismatch quantification is a function of the states and inputs, but does not consider the variance of uncertainty predictions. In the final case, we consider the proposed asMPC approach, wherein both the mean and variance of the BNN predictions of the plant-model mismatch are considered. In the asMPC case, we account for the probabilities of each scenario.

C. Closed-loop Simulations

At each time instant, we sampled $\bar{N}_{\text{MC}} = 100$ models to estimate the mean μ_y and standard deviation σ_y of the mismatch y. Subsequently, at each node of the scenario tree, we used $\hat{\mu}_{\hat{y}}$, $\hat{\mu}_{\hat{y}} + 3\hat{\sigma}_{\hat{y}}$, and $\hat{\mu}_{\hat{y}} - 3\hat{\sigma}_{\hat{y}}$ as three discrete scenarios of the plant-model mismatch. Furthermore, we set $|y_1| \leq 3$ and $|y_2| \leq 20$ based on $\max_j |\text{y}_i^{(j)}|, i = 1,2$ in the dataset $\mathcal{T}$. When the predictions of the scenarios are out of the bounds of y due to the limited generalization of the BNN model, we use the bounds instead of the predictions and uniform distribution as the probability of scenarios to avoid too conservative uncertainty estimation.

In our comparison of the aforementioned sMPC controllers, we used a prediction horizon $N = 5$ and a robust horizon $N_r = 2$. The sMPC problem (13) was solved in less than 0.1390 s (at all time instants) in our closed-loop runs while the sampling time was 0.5 s. As shown in Fig. 3, employing the proposed asMPC approach (green line) achieves better control performance than using sMPC with $\hat{\mu}_{\hat{y}}$ as the only scenario (blue line labeled by $\hat{\mu}_{\hat{y}}$) and the worst-case sMPC (black line). Furthermore, we performed 100 closed-loop simulations using asMPC under different measurement

noise realizations drawn from normal distribution $\mathcal{N}(0, 0.2)$. The states and control inputs that satisfy the constraints under the asMPC control law from three closed-loop simulations are shown in Fig. 4. Moreover, the closed-loop simulations revealed that all runs satisfied constraints by increasing a to 20.

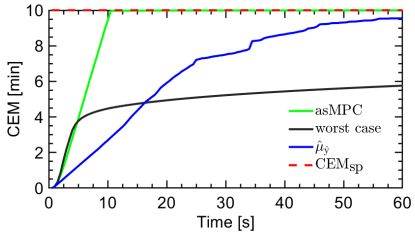
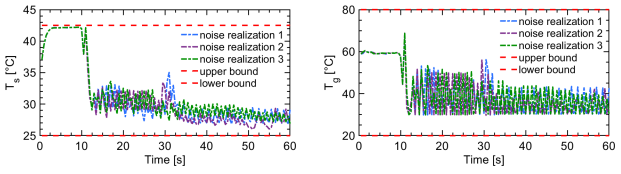
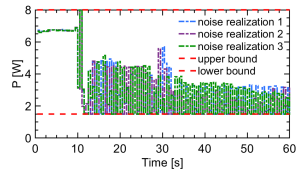


Fig. 3: Closed-loop simulation results of various scenario-based MPC strategies for CEM setpoint tracking.

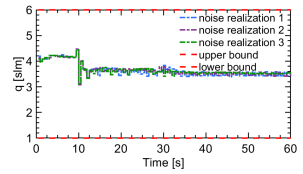


(a) T_s profile.

(b) T_g profile.



(c) Control input P.



(d) Control input q.

Fig. 4: State and input profiles for three closed-loop simulations of the proposed adaptive-scenario-tree MPC under different noise realizations.

V. Conclusions

In this paper, a learning- and scenario-based MPC approach was proposed to safely control nonlinear systems with state- and input-dependent uncertainties. In particular, a BNN was employed to model the unknown and/or time-varying system dynamics using trajectory data of the system, and adaptive scenario trees were constructed online based on the BNN model. Additionally, a moment-matching optimization method was used to compute the probabilities of the generated time-varying scenarios. Moreover, probabilistic safety was guaranteed by ensuring that the generated scenarios contain the real system and the constraints are satisfied by all the scenarios in the prediction horizon. Closed-loop simulations on a physics-based plasma jet model demonstrated that the proposed approach can improve control performance compared to scenario-based MPC with a fixed scenario tree.

References

[1] A. S. Badwe, R. S. Patwardhan, S. L. Shah, S. C. Patwardhan, and R. D. Gudi, "Quantifying the impact of model-plant mismatch on controller performance," *Journal of Process Control*, vol. 20, no. 4, pp. 408–425, 2010.

[2] M. Kano, Y. Shigi, S. Hasebe, and S. Ooyama, "Detection of significant model-plant mismatch from routine operation data of model predictive control system," *IFAC Proceedings Volumes*, vol. 43, no. 5, pp. 685–690, 2010.

[3] J. M. Simkoff, S. Wang, M. Baldea, L. H. Chiang, I. Castillo, R. Bindlish, and D. B. Stanley, "Plant-model mismatch estimation in unconstrained state-space mpc," in *2017 American Control Conference (ACC)*, pp. 3078–3083, IEEE, 2017.

[4] R. Soloperto, M. A. Müller, S. Trimpe, and F. Allgöwer, "Learning-based robust model predictive control with state-dependent uncertainty," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 442–447, 2018.

[5] X. Xu, J. M. Simkoff, M. Baldea, L. H. Chiang, I. Castillo, R. Bindlish, and B. Ashcraft, "Data-driven plant-model mismatch quantification for mimo mpc systems with feedforward control path," in *2020 American Control Conference (ACC)*, pp. 2760–2765, 2020.

[6] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, "Provably safe and robust learning-based model predictive control," *Automatica*, vol. 49, no. 5, pp. 1216–1226, 2013.

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

[8] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 269–296, 2020.

[9] A. Mesbah, "Stochastic model predictive control with active uncertainty learning: A survey on dual control," *Annual Reviews in Control*, vol. 45, pp. 107–117, 2018.

[10] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using gaussian process regression," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2736–2743, 2019.

[11] A. D. Bonzanini, J. A. Paulson, and A. Mesbah, "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, 2020.

[12] C. J. Ostafew, A. P. Schoellig, T. D. Barfoot, and J. Collier, "Learning-based nonlinear model predictive control to improve vision-based mobile robot path tracking," *Journal of Field Robotics*, vol. 33, no. 1, pp. 133–152, 2016.

[13] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When Gaussian process meets big data: A review of scalable GPs," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 11, pp. 4405–4423, 2020.

[14] Y. Bao, J. Mohammadpour Velni, and M. Shahbakhti, "Epistemic uncertainty quantification in state-space LPV model identification using Bayesian neural networks," *IEEE Control Systems Letters*, vol. 5, no. 2, pp. 719–724, 2021.

[15] D. Bernardini and A. Bemporad, "Scenario-based model predictive control of stochastic constrained linear systems," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pp. 6333–6338, IEEE, 2009.

[16] Y. Bao and J. M. Velni, "Safe control of nonlinear systems in LPV framework using model-based reinforcement learning," *International Journal of Control*, vol. 0, no. 0, pp. 1–12, 2022.

[17] S. Lucia, T. Finkler, and S. Engell, "Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty," *Journal of Process Control*, vol. 23, no. 9, pp. 1306–1319, 2013.

[18] K. Høyland and S. W. Wallace, "Generating scenario trees for multistage decision problems," *Management science*, vol. 47, no. 2, pp. 295–307, 2001.

[19] Y. Bao, J. Mohammadpour Velni, and M. Shahbakhti, "An online transfer learning approach for identification and predictive control design with application to RCCI engines," in *Dynamic Systems and Control Conference*, vol. 84270, p. V001T21A003, American Society of Mechanical Engineers, 2020.

[20] D. Gidon, D. B. Graves, and A. Mesbah, "Effective dose delivery in atmospheric pressure plasma jets for plasma medicine: a model predictive control approach," *Plasma Sources Science and Technology*, vol. 26, no. 8, p. 085005, 2017.

[21] A. D. Bonzanini, J. A. Paulson, G. Makrygiorgos, and A. Mesbah, "Fast approximate learning-based multistage nonlinear model predictive control using Gaussian processes and deep neural networks," *Computers & Chemical Engineering*, vol. 145, p. 107174, 2021.

[22] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.