Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

Fast approximate learning-based multistage nonlinear model predictive control using Gaussian processes and deep neural networks

Angelo D. Bonzanini[a,1], Joel A. Paulson[b,1], Georgios Makrygiorgos[a], Ali Mesbah[a,*]

[a] Department of Chemical and Biomolecular Engineering, University of California-Berkeley, Berkeley, CA 94720, USA
[b] Department of Chemical and Biomolecular Engineering, The Ohio State University, Columbus, OH 43210, USA

ARTICLE INFO

Article history:
Received 7 July 2020
Revised 12 October 2020
Accepted 14 November 2020
Available online xxx

Keywords:
Learning-based predictive control
Scenario-based MPC
Gaussian processes
Deep neural networks
Bayesian hyperparameter optimization

ABSTRACT

Scenario-based model predictive control (MPC) methods introduce recourse into optimal control and can thus reduce the conservativeness inherent to open-loop robust MPC. However, the uncertainty scenarios are often generated offline using worst-case uncertainty bounds quantified *a priori*, limiting the potential gains in control performance. This paper presents a learning-based multistage MPC (msMPC) for systems with hard-to-model dynamics and time-varying plant-model mismatch. Gaussian Processes (GP) are used to learn state- and input-dependent plant-model mismatch in real-time and accordingly adapt the scenario tree online. Due to the increased computational complexity associated with incorporating the GP predictions into the optimal control problem, the learning-based msMPC (LB-msMPC) law is approximated by a deep neural network (DNN) that is cheap-to-evaluate online and has a small memory footprint, which makes it suitable for embedded applications. In addition, we present a novel algorithm for training the DNN-based controller that uses a GP description of the plant-model mismatch to generate closed-loop simulation data, which ensures the LB-msMPC law is evaluated in regions of the state space most relevant to closed-loop operation. The proposed LB-msMPC strategy is demonstrated on a cold atmospheric plasma jet with applications in (bio)materials processing. The simulation results indicate the promise of the approximate LB-msMPC strategy for control of hard-to-model systems with fast dynamics on millisecond timescales.

1. Introduction

Model predictive control (MPC) is widely used for the control of multivariable and constrained systems (Mayne, 2014). The performance of MPC hinges on the availability of accurate and inexpensive system models. Learning-based MPC (LB-MPC) aims to leverage machine learning and statistical learning tools to enable control of systems with complex and hard-to-model dynamics, where purely model-based control approaches exhibit limited effectiveness (Aswani et al., 2013). LB-MPC has shown promise in various control applications, such as pH neutralization processes (Kocijan et al., 2004), gas-liquid separation processes (Likar and Kocijan, 2007), non-thermal plasma processing (Bonzanini and Mesbah, 2020), robot path tracking (Ostafew et al., 2014), and unmanned vehicles (Hewing et al., 2019).

However, using machine learning algorithms to augment – or even completely substitute – the model does not inherently equip the MPC controller with the ability to systematically account for uncertainty Bemporad and Morari (1999); Mesbah (2016, 2018). In particular, the presence of uncertainties may lead to undesired constraint violation and/or MPC performance deterioration. A popular approach to robustify MPC involves representing system uncertainties using a *scenario tree* in the optimal control problem (Bernardini and Bemporad, 2009). Scenario-based MPC methods, such as multi-stage MPC (msMPC) (Lucia et al., 2013), can mitigate the inherent conservativeness associated with robust MPC by introducing recourse into the optimal control problem. However, an important drawback of msMPC is the fact that scenarios are generated based on offline-characterized worst-case uncertainty descriptions that must hold throughout the entire state space. This neglects the potential time-varying and/or state-dependent nature of the uncertainty that, when unaccounted for, can lead to significant losses in control performance.

Adaptive approaches to msMPC have recently been proposed to improve knowledge about an uncertain system via online estimation of uncertainties and accordingly adapting the scenario tree (Subramanian et al., 2015; Holtorf et al., 2019; Thangavel et al., 2020). Although various disturbance estimators have been used

[*] Corresponding author.
E-mail address: mesbah@berkeley.edu (A. Mesbah).
[1] A. Bonzanini and J. Paulson contributed equally to this work.

https://doi.org/10.1016/j.compchemeng.2020.107174
0098-1354/© 2020 Elsevier Ltd. All rights reserved.

for scenario tree adaptation, they typically assume either a specific structure of the model, or a generic constant disturbance prediction, which may not capture structural plant-model mismatch adequately. Alternatively, state- and input-dependent descriptions of system uncertainty have shown to be useful for capturing the time-varying and state-dependent nature of structural model uncertainty (i.e., plant-model mismatch) (Bonzanini et al., 2020; Bonzanini and Mesbah, 2020; Hewing et al., 2019; Ostafew et al., 2014; Soloperto et al., 2018).

In this paper, we present a learning-based msMPC (LB-msMPC) strategy for hard-to-model dynamic systems in the presence of plant-model mismatch. The first contribution of this work is a learning-based paradigm for msMPC with adaptive scenario tree whereby Gaussian Processes (GP) (Rasmussen and Williams, 2006) are used to learn a data-driven description of the plant-model mismatch. The main advantages of using a GP uncertainty model are its ability to: (i) learn a description of any arbitrary functions as a result of its non-parametric form and (ii) quantify the uncertainty in the prediction of the plant-model mismatch that can be incorporated into robust MPC schemes. For online adaptation of the scenario tree, the GP is embedded into the predictions of the msMPC such that the scenarios (derived from either random sampling or some quadrature rule) depend on the current and predicted state and input values. Although online adaptation of the scenario tree in msMPC has been implemented using recursive Bayesian weighting (Krishnamoorthy et al., 2019b) and sigma point-based uncertainty propagation (Thangavel et al., 2020), this is the first work that uses GP to perform scenario tree adaptation.

However, GP-based adaption of the scenario tree further increases the computational complexity of msMPC that can make it impractical to implement in real-time, especially for systems with fast dynamics. This is mainly because the number of decision variables grows exponentially with the prediction horizon due to the branching of the scenario tree in addition to the calculation of each scenario requiring a Cholesky decomposition of the covariance matrix predicted by the GP uncertainty model. Since the LB-msMPC problem cannot be solved exactly offline using explicit/multiparametric MPC methods (Bemporad et al., 2002) due to the nonlinear nature of the GP, we propose to approximate the implicit LB-msMPC law in terms of a cheap-to-evaluate control law. There has recently been increased interest in approximate MPC strategies to yield an explicit expression for the controller using data generated from offline solution of an MPC problem. Previous works have used various function approximators, such as polynomials (Chakrabarty et al., 2016), radial basis functions (Csekő et al., 2015), and deep neural networks (DNNs) (Parisini and Zoppoli, 1995; Karg and Lucia, 2018; Chen et al., 2018) to derive approximate control laws. In particular, DNNs have been shown to be useful for approximating MPC laws due to their ability to capture the piecewise nature of the control law (Karg and Lucia, 2018; Bonzanini et al., 2020a; Paulson and Mesbah, 2020). The second contribution of this work is a novel framework for training fast approximate LB-msMPC using DNNs. In contrast to existing approximate MPC approaches that base their training on open-loop data, we demonstrate the advantages of training the DNN with closed-loop data, which yields an effective approximation of the control law in state-space regions relevant to closed-loop control. In addition, we show how Bayesian optimization methods (Shahriari et al., 2016) can be used to systematically and efficiently select hyperparameters of the DNN.

The proposed LB-msMPC strategy is demonstrated on a simulation case study of an atmospheric pressure plasma jet (APPJ) with prototypical applications in (bio)materials processing and plasma medicine. We explore the tradeoff between computational cost and performance for different update rules of the GP uncertainty model. Specifically, we compare LB-msMPC, whereby the GP is propagated in prediction, and a so-called adaptive msMPC, whereby the GP is updated at each sampling time and thus is external to the controller. In addition, we demonstrate the effects of training data and hyperparameter optimization on the performance of the DNN-based approximate control laws.

The structure of this paper is as follows. The problem formulation and standard worst-case msMPC are summarized in Section 2. Section 3 presents the proposed LB-msMPC strategy and discusses various schemes for scenario selection. This is followed by the approximation of LB-msMPC using DNNs in Section 4, with focus on generating closed-loop training data and hyperparameter selection using Bayesian optimization. In Section 5, the proposed methods are demonstrated on a simulation case study of a cold atmospheric plasma jet with prototypical applications in plasma medicine and biomaterials processing (Gidon et al., 2018). We benchmark the proposed LB-msMPC strategy against worst-case msMPC for control of nonlinear plasma dose delivery, and demonstrate that the approximate LB-msMPC law can be evaluated in the millisecond range.

2. Preliminaries

2.1. Problem formulation

Consider an uncertain discrete-time nonlinear system of the form

$$x^+ = f(x,u) + w, \tag{1}$$

where $x \in \mathbb{R}^{n_x}$ is the current state, x^+ is the successor state at the next time instant, $u \in \mathbb{R}^{n_u}$ is the control input, and $w \in \mathbb{R}^{n_w}$ is a vector of unknown disturbances. The disturbance sequence $\{w_k\}_{k\geq 0}$ is assumed to be a realization of a stochastic process that satisfies the following assumption.

Assumption 1. w_k for $k = 0, 1, \dots$ are independent and identically distributed (i.i.d.) random variables with known probability density function $p_w(w)$ and bounded support $\mathcal{W} \subset \mathbb{R}^{n_w}$.

Assumption 1 is made initially to derive a theoretically consistent scenario-based approximation to an underlying closed-loop stochastic MPC problem, which will be relaxed in the subsequent sections. The system (1) is subject to state and input constraints given by

$$(x,u) \in \mathcal{Z}, \tag{2}$$

where $\mathcal{Z} = \{(x,u) : h(x,u) \leq 0\}$ is a known set specified by the constraint function $h : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_h}$. Note that (2) implicitly defines a set of state constraints

$$\mathcal{X} = \text{Proj}_{\mathbb{R}^{n_x}}(\mathcal{Z}), \tag{3}$$

where $\text{Proj}_X(S) = \{x \in X : \exists y \in Y \text{ s.t. } (x,y) \in S\}$ is the orthogonal projection of the set $S \subset X \times Y$ onto the space X. We aim to solve the following finite-horizon closed-loop MPC problem under uncertainty

$$\min_{\Pi} J(x,\Pi) := \mathbb{E}_{\mathbf{w}}\left\{\sum_{k=0}^{N-1} \ell(x_k, \pi_k(x_k)) + \ell_f(x_N)\right\}, \tag{4a}$$

$$\text{s.t.} x_{k+1} = f(x_k, \pi_k(x_k)) + w_k, \tag{4b}$$

$$(x_k, \pi_k(x_k)) \in \mathcal{Z}, \tag{4c}$$

$$x_0 = x, \forall \mathbf{w} \in \mathcal{W}^N, \quad \forall k \in \{0,\dots,N-1\}, \tag{4d}$$

where the decision variables are defined by the *control policy* $\Pi = \{\pi_0(\cdot), \dots, \pi_{N-1}(\cdot)\}$ that is composed of a sequence of control laws

$\pi_k : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$ over the prediction horizon $N > 0$. The objective function is defined in terms of the expected value $\mathbb{E}_{\mathbf{w}}\{\cdot\}$ with respect to the disturbance sequence $\mathbf{w} = (w_0, \ldots, w_{N-1})$. $\ell(x, u)$ and $\ell_f(x_N)$ denote the stage cost and terminal cost, respectively. The objective function directly takes advantage of distributional information of the disturbances. The constraints, however, are enforced with respect to the worst-case realization of the disturbances. In principle, the hard constraints (2) could be replaced with *chance constraints*; however, this would complicate the analysis when Assumption 1 is relaxed due to the fact that the p_w could vary with time or the state. As such, we choose to enforce constraints in a worst-case sense, which is consistent with the standard formulation of multistage MPC (Lucia et al., 2013).

2.2. Multistage MPC

The closed-loop MPC problem (4) is not solvable directly due to the infinite dimensional nature of the control policy Π. However, even a finite parametrization of the control laws, e.g., $\pi_k(x_k) = K_k x_k + v_k$, results in (4) taking the form of a semi-infinite optimization problem since the uncertainty set $\mathcal{W} \subset \mathbb{R}^{n_w}$ is generally uncountable. A tractable approximation to (4) can be derived in the general nonlinear setting by considering a sample-based approximation of p_w, i.e.,

$$p_w(w) \approx p_{\tilde{W}}(w) := \textstyle\sum_{i=1}^{s} p^i \delta(w - w^i), \tag{5}$$

where w^i and p^i denote the location and probability of the ith sample, respectively, and s is the total number of samples in the approximation. This implies that the support of the disturbance distribution is approximated by $\mathcal{W} \approx \tilde{\mathcal{W}} = \{w^1, \ldots, w^s\}$. In this case, the future evolution of the uncertainty over the prediction horizon can be represented by a *scenario tree* (Fig. 1), where a scenario represents a path from the root node to a given leaf node. A key challenge in scenario tree approximations is that the size of the scenario tree grows exponentially with the prediction horizon.

The so-called multistage MPC (msMPC) formulation (Lucia et al., 2013) avoids branching after a certain number of time steps, called the *robust horizon* N_r, after which the disturbances are assumed constant. Consequently, the total number of scenarios is given by $S = s^{N_r}$. The resulting optimal control problem can be formulated as

$$\min_{u_{k,j}, x_{k,j}} \textstyle\sum_{j=1}^{S} \omega_j \Big[\sum_{k=0}^{N-1} \ell(x_{k,j}, u_{k,j}) + \ell_f(x_{N,j}) \Big], \tag{6a}$$

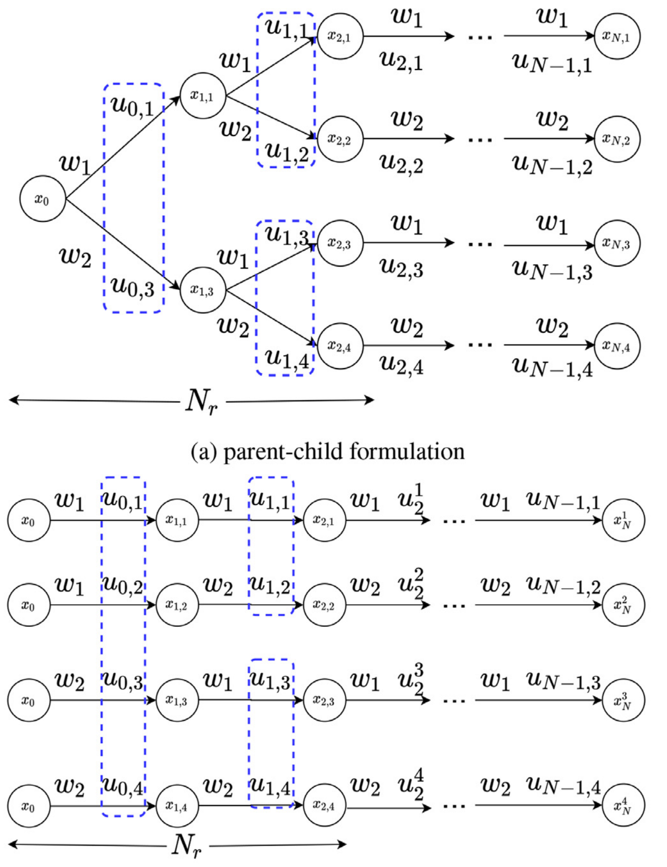
$$\text{s.t.} x_{k+1,j} = f(x_{k,j}, u_{k,j}) + w_{k,j}, \tag{6b}$$

$$(x_{k,j}, u_{k,j}) \in \mathcal{Z}, \tag{6c}$$

$$\textstyle\sum_{j=1}^{S} \bar{E}_j u_j = 0, \tag{6d}$$

$$x_{0,j} = x, \forall j \in \{1, \ldots, S\}, \quad \forall k \in \{0, \ldots, N-1\}, \tag{6e}$$

where the subscript $(\cdot)_{k,j}$ denotes the jth scenario at sample time k, ω_j is the probability of occurrence of the disturbance sequence $w_j = (w_{0,j}, \ldots, w_{N-1,j})$ that is equal to the products of conditional probabilities (an element of $\{p^1, \ldots, p^s\}$) along the path of scenario j, and (6d) enforces the *non-anticipativity constraints* that ensure the states that branch from the same parent node have the same control input (i.e., current decisions cannot anticipate the future). Here, the vector $u_j = (u_{0,j}, \ldots, u_{N-1,j}) \in \mathbb{R}^{n_u N}$ denotes the sequence of optimal control inputs for the jth scenario and the matrix $\bar{E}_j$ is given by Klintberg et al. (2016)



(a) parent-child formulation

(b) distributed formulation where every scenario $S = s^{N_r} = 2^2 = 4$ retains a local copy of the system variables

Fig. 1. Two equivalent scenario tree representations for $s = N_r = 2$.

$$\bar{E} = \begin{bmatrix} E_{1,2} & -E_{1,2} & & & \\ & E_{2,3} & -E_{2,3} & & \\ & & \ddots & \ddots & \\ & & & E_{S-1,S} & -E_{S-1,S} \end{bmatrix}, \\ = \begin{bmatrix} \bar{E}_1 & \bar{E}_2 & \ldots & \bar{E}_{S-1} & \bar{E}_S \end{bmatrix}, \tag{7}$$

where

$$E_{j,j+1} = \begin{bmatrix} I_{n_u} & & & 0 & \ldots & 0 \\ & \ddots & & \vdots & \ddots & \vdots \\ & & I_{n_u} & 0 & \ldots & 0 \end{bmatrix}. \tag{8}$$

The dimensions of these matrices are defined in terms of $n_{o,(j,j+1)}$, which denotes the number of common nodes between consecutive scenarios j and $j+1$. That is, $E_{j,j+1} \in \mathbb{R}^{r \times n_u N}$ and $\bar{E}_j \in \mathbb{R}^{q \times n_u N}$, where $r = n_u n_{o,(j,j+1)}$ and $q = n_u \sum_{j=1}^{S-1} n_{o,(j,j+1)}$.

Remark 1. Formulating the non-anticipativity constraints using the chain-like structure exhibited by the matrices $\{\bar{E}_j\}_{j=1}^{S}$ results in highly sparse structures that can be exploited by many solvers, as discussed in Klintberg et al. (2016). This structure can also be straightforwardly handled by both primal and dual distributed optimization algorithms, which has recently been explored in Krishnamoorthy et al. (2019a).

Remark 2. When $N_r = N$, constraints (6c) will be a direct enforcement of worst-case constraints under the approximated distribution (5) since the state and input evolution is required to stay within the constraints for all possible realizations of disturbances. For linear systems $f(x,u)=Ax+Bu$ with a polytopic disturbance support, these constraints exactly enforce worst-case constraint sanctification as long as the extreme values of $\mathcal{W}$ are included in $\tilde{\mathcal{W}}$Karg et al. (2019).

The design of a suitable scenario tree is a tradeoff between coverage of the uncertainty space and computational cost. One of the most commonly used methods for scenario tree construction is to select the combination of the minimum, nominal, and maximum values in each disturbance dimension, leading to $s=3^{n_w}$. Thus, the tree grows exponentially with respect to the number of considered uncertainties, indicating that the choice of scenarios is of paramount importance for tractability. Alternative methods for the construction of sparse scenario trees is discussed in more detail in Section 3.3.

3. Learning-based multistage MPC

3.1. Learning plant-model mismatch with Gaussian processes

The msMPC problem (6) treats the uncertainty scenarios $\tilde{\mathcal{W}}$ as perfectly known *a priori*, which can be a valid choice under Assumption 1. However, there are a variety of systems that do not satisfy this assumption, including those that exhibit a significant degree of time-varying uncertainty. One such example is plant-model mismatch, which corresponds to the case that the true underlying dynamics are unknown. We can represent this situation by

$$x^+=f(x,u)+B_d(g(x,u)+v), \tag{9}$$

where $v\in\mathcal{R}^{n_d}$ is a process noise that is assumed to be normally distributed, i.e., $v\sim\mathcal{N}(0,\Sigma^v)$ with a diagonal covariance matrix $\Sigma^v=\text{diag}([\sigma_1^2,\ldots,\sigma_{n_d}^2])$. The model (9) is composed of a known nominal function $f:\mathbb{R}^{n_x}\times\mathbb{R}^{n_u}\to\mathbb{R}^{n_x}$ and an initially unknown function $g:\mathbb{R}^{n_x}\times\mathbb{R}^{n_u}\to\mathbb{R}^{n_d}$ to be learned from data, which is assumed to lie in the subspace spanned by the matrix $B_d\in\mathbb{R}^{n_x\times n_d}$. For example, in the case that all states have unmodeled dynamics, then $n_d=n_x$ and $B_d=I_{n_x}$. We can interpret this representation of plant-model mismatch as a special case of (1), where the disturbance $w=B_d(g(x,u)+v)$ is now *state- and-input dependent*.

In this work, we choose to model the noisy vector-valued function g using Gaussian process (GP) regression (Rasmussen and Williams, 2006). Assuming that we have $M>0$ training points in the form of previously collected measurements of the states and inputs, we can evaluate the mismatch term as

$$y_j=B_d^\dagger(x_{j+1}-f(x_j,u_j))=g(x_j,u_j)+v_j, \tag{10}$$

for all $j\in\{1,\ldots,M\}$, where $B_d^\dagger$ denotes the Moore–Penrose pseudoinverse of B_d. Letting $z_j=(x_j,u_j)$, we can define the training dataset as

$$\mathcal{D}=\{\mathbf{y}=[y_1,\ldots,y_M]^\top,\ \mathbf{z}=[z_1,\ldots,z_M]^\top\}. \tag{11}$$

An important note about GP regression is that the evaluation at a new point, which involves the computation of the covariance matrix, becomes more expensive as more training data are used. In applications where there is a need for fast GP evaluations, one can conveniently populate the training set using optimal sampling techniques (Burbidge et al., 2007; Willett et al., 2006). Although this is beyond the scope of this work, it is an interesting future research direction. For simplicity, it is assumed that each dimension of g can be learned separately. First, we must specify a GP prior on each element $a\in\{1,\ldots,n_d\}$ of g, with known mean function $m^a(\cdot)$ and covariance kernel $k^a(\cdot,\cdot)$. Then, by conditioning on the training data $\mathcal{D}$, a Gaussian posterior distribution can be derived at any selected test point $z=(x,u)$ with the mean and covariance

$$\mu_a^d(z)=m^a(z)+K_{z\mathbf{z}}^a(K_{\mathbf{zz}}^a+\sigma_a^2 I)^{-1}([\mathbf{y}]_{\cdot,a}-m_{\mathbf{z}}^a), \tag{12a}$$

$$\Sigma_a^d(z)=K_{zz}^a-K_{z\mathbf{z}}^a(K_{\mathbf{zz}}^a+\sigma_a^2 I)^{-1}K_{\mathbf{z}z}^a, \tag{12b}$$

where K is the Gram matrix that is composed of the terms $[K_{\mathbf{zz}}^a]_{i,j}=k^a(z_i,z_j)$, $[K_{\mathbf{z}z}^a]_j=k^a(z_j,z)$, $K_{z\mathbf{z}}^a=(K_{\mathbf{z}z}^a)^\top$, $K_{zz}^a=k^a(z,z)$, and $[m_{\mathbf{z}}^a]_j=m^a(z_j)$. Many choices exist for the kernel function, one of the most popular being the squared exponential kernel

$$k^a(z_i,z_j)=\sigma_{f,a}^2\exp\left(-\frac{1}{2}(z_i-z_j)^\top L_a^{-1}(z_i-z_j)\right), \tag{13}$$

where $\sigma_{f,a}^2$ is the signal variance and L_a is a positive diagonal length-scale matrix (Rasmussen and Williams, 2006). The resulting state- and input-dependent GP approximation of the unknown function $g(x,u)$ is then given by

$$d(x,u)\sim\mathcal{N}(\mu^d(x,u),\Sigma^d(x,u)), \tag{14}$$

where the mean $\mu^d(\cdot)=[\mu_1^d(\cdot),\ldots,\mu_{n_d}^d(\cdot)]^\top$ and covariance $\Sigma^d(\cdot)=\text{diag}([\Sigma_1^d(\cdot),\ldots,\Sigma_{n_d}^d(\cdot)]^\top)$ are concatenated from the individual output predictions in (12).

Remark 3. The assumption that each dimension of g is learned separately is common in literature, e.g., see Hewing et al. (2019). Firstly, assuming independence of the outputs reduces computational complexity and makes the derivation of confidence intervals more straightforward. Secondly, although vector forms of GP do exist, it is not clear if they provide a noticeable improvement in predictions. The proposed LB-msMPC strategy can accommodate any number of outputs learned with a single GP regression.

Remark 4. We choose to model the unknown dynamics using the additive term $w=B_d(g(x,u)+v)$ in (9) because it provides an intuitive structure that separates the known part from the unmodeled dynamics. Since w is itself a nonlinear function, the dynamics in (9) can be expressed as a new nonlinear function, say, $f_{\text{tot}}(x,u,d)$, where d is defined in (14). When the uncertainty is not additive, the derivation of the scenarios requires an additional optimization step, i.e., maximizing $f_{\text{tot}}(x,u,d)$ over d.

3.2. GP-based formulation of msMPC

Under the GP approximation of $g(x,u)$ in (14), the effective disturbance in (1) becomes $w=B_d(d(x,u)+v)$. Due to the dependence on the state and input, the disturbance does not satisfy Assumption 1 and the msMPC problem (6) must be modified to explicitly account for plant-model mismatch. A key observation is that $w_k|w_{k-1},\ldots,w_0=w_k|x_k,u_k$ is Gaussian such that

$$w_k|x_k,u_k\sim\mathcal{N}(\mu^w(x_k,u_k),\Sigma^w(x_k,u_k)), \tag{15}$$

where $\mu^w(x,u)=B_d\mu^d(x,u)$ is the mean function and $\Sigma^w(x,u)=B_d\Sigma^d(x,u)B_d^\top+B_d\Sigma^v B_d^\top$ is the covariance matrix function. We can equivalently represent (15) via an affine transformation of standard Gaussian random variables $\xi_k\sim\mathcal{N}(0,I_{n_w})$, namely

$$w_k|x_k,u_k=\mu^w(x_k,u_k)+\Sigma^w(x_k,u_k)^{1/2}\xi_k, \tag{16}$$

where $(\cdot)^{1/2}$ denotes any matrix square root such as the Cholesky decomposition. We can model the random process $\{\xi_k\}_{k\geq 0}$ as a collection of i.i.d. random variables – a main requirement of Assumption 1 – such that the same basic msMPC formulation in Section 2 can be applied in this transformed space, i.e.,

$$\min_{u_{k,j},x_{k,j}}\ \textstyle\sum_{j=1}^{S}\gamma_j\Big[\sum_{k=0}^{N-1}\ell(x_{k,j},u_{k,j})+\ell_f(x_{N,j})\Big], \tag{17a}$$

$$\text{s.t.}x_{k+1,j}=f(x_{k,j},u_{k,j})+w_{k,j}, \tag{17b}$$

$$w_{k,j}=\mu^w(x_{k,j},u_{k,j})+\Sigma^w(x_{k,j},u_{k,j})^{1/2}\xi_{k,j}, \tag{17c}$$

$$(x_{k,j}, u_{k,j}) \in \mathcal{Z}, \tag{17d}$$

$$\textstyle\sum_{j=1}^{S} \bar{E}_j u_j = 0, \tag{17e}$$

$$x_{0,j} = x, \forall j \in \{1, \dots, S\}, \quad \forall k \in \{0, \dots, N-1\}, \tag{17f}$$

where $\xi_{k,j} \in \{\xi^1, \dots, \xi^s\}$ denote the scenarios at each stage in the transformed space and $\{\gamma_j\}_{j=1}^{S}$ denote the effective probability of each scenario over time. We refer to formulation (17) as learning-based msMPC (LB-msMPC) due to the fact that the learned GP model is embedded in the predictions (17c), which is the main difference between problems (6) and (17).

Let $\mathbf{U}^\star(x) = \{u_0^\star(x), \dots, u_S^\star(x)\}$ denote the optimal solution to (17) that is a function of the measured state value x. The LB-msMPC control law is implicitly defined in terms of this solution by

$$\kappa_{\text{lb}}(x) = u_{0,1}^\star(x), \tag{18}$$

where the first input in the sequence is equal for all scenarios according to the non-anticipativity constraints, i.e., $u_{0,1}^\star(x) = u_{0,2}^\star(x) = \ldots = u_{0,S}^\star(x)$. Let $\mathscr{X}_{\text{lb}} \subseteq \mathbb{R}^{n_x}$ be the domain of attraction of the LB-msMPC controller, which denotes the set of initial states such that (17) has a feasible solution. To ensure a feasible solution always exists in practice, i.e., $\mathscr{X}_{\text{lb}} = \mathbb{R}^{n_x}$, the constraints (17d) can be softened using an exact penalty function method as discussed in Kerrigan and Maciejowski (2000).

Remark 5. There may be additional parameters θ that appear in the cost or constraint functions in the MPC problem (e.g., setpoint values). In such cases, the control law (18) will depend on these parameters $\kappa_{\text{lb}}(x, \theta)$.

Due to the GP model used to approximate plant-model mismatch, the bounded disturbance assumption is no longer satisfied by the support $\mathcal{W} = \mathbb{R}^{n_w}$. Instead of trying to enforce constraints for all possible values of uncertainty in this case, we look to use confidence intervals to characterize the expected range of uncertainty. These confidence intervals can be straightforwardly computed from the GP model as

$$\mathcal{W}(x, u) = \left\{ w : \|w - \mu^w(x, u)\|^2_{\Sigma^w(x,u)} \le r(\alpha) \right\}, \tag{19}$$

where $r(\cdot)$ is the quantile function for the chi-squared distribution with n degrees of freedom ($\chi^2_{n_d}(\alpha)$) and $\alpha \in (0, 1)$ is the desired probability level. The choice of α is a tuning parameter, with larger values representing less confidence in our knowledge of the function $g(x, u)$. The ellipsoidal confidence region (19) can be used to select the minimum, nominal, and maximum values in each of the principal axes by selecting the samples of ξ in each dimension to be $\{-r^{-1/2}(\alpha), 0, r^{-1/2}(\alpha)\}$.

Remark 6. Even though the affine transformation (16) is valid at each individual test point, this does not directly imply that the sequence $\{\xi_k\}_{k\ge 0}$ satisfies the i.i.d. assumption. This is a valid representation under a stochastic interpretation of the GP in which different outputs can be predicted for evaluations at the exact same state value. The so-called scenario viewpoint results in every random GP sample being a deterministic function, which is a more realistic representation of plant-model mismatch. Interested readers are referred to Umlauft et al. (2018) for a more detailed discussion on these two methods that will be the subject of our future work.

Remark 7. The LB-msMPC problem does not guarantee closed-loop constraint satisfaction by design. One way to recover this property for systems with state- and input-dependent uncertainty was recently proposed in Bonzanini et al. (2020b), wherein the optimal input $\kappa_{\text{lb}}(x)$ is projected onto a safety set of states $\mathcal{S} \subseteq \text{Proj}_{\mathbb{R}^{n_x}}(\mathcal{Z})$, i.e.,

$$u_{\mathcal{S}}(x) = \underset{u \in \mathcal{S}_u(x)}{\text{argmin}} \; \|u - \kappa_{\text{lb}}(x)\|^2,$$

where

$$\mathcal{S}_u(x) = \{u : \{f(x, u)\} \oplus \mathcal{W}(x, u) \subseteq \mathcal{S}, \; (x, u) \in \mathcal{Z}\}$$

is the set of inputs that guarantee the system stays within the safe set $\mathcal{S}$ for all possible disturbances, with $A \oplus B$ denoting the Minkowski set addition of two sets $A, B \subset \mathbb{R}^n$. By selecting $\mathcal{S}$ to be a robust control invariant (RCI) set for the system $x^+ = f(x, u) + w$ and constraints (2) and (19), a feasible backup controller is guaranteed to exist due to the fact that $\mathcal{S}_u(x) \neq \emptyset$ for all $x \in \mathcal{S}$. A discussion on RCI set computation for a variety of systems (ranging from linear to nonlinear) is provided in Rakovic et al. (2006). It is important to note that RCI set construction is generally a challenging task, especially as n_x increases, and remains an active area of research.

3.3. Sparse scenario tree construction methods

There are two main alternatives for deriving sample-based approximations to the distribution p_ξ, which can both be interpreted as approximations to the expectation operator in the closed-loop MPC problem (4). The first approach is often referred to as sample average approximation (SAA) (Kleywegt et al., 2002), which applies Monte Carlo (MC) sampling to obtain realizations of the random disturbance ξ. The convergence rate of MC is $O(s^{-1/2})$ Caflisch (1998), which is independent of the number of uncertainties n_w; however, it is often slow in practice, implying a large number of samples may be needed to achieve an acceptable level of error. Numerical quadrature, on the other hand, looks to approximate the expected value operator as a sum over a finite set of samples as follows

$$\mathbb{E}_\xi\{F(\xi)\}\} = \textstyle\int F(\xi) p_\xi(\xi) d\xi \approx \sum_{i=1}^{s} p^i F(\xi^i), \tag{20}$$

where $F(\xi)$ is some arbitrary function. The quadrature nodes $\{\xi^i\}_{i=1}^{s}$ then correspond to the uncertainty realizations, whereas the quadrature weights $\{p^i\}_{i=1}^{s}$ represent their associated probabilities.

Gaussian quadrature rules are known to be highly efficient since they can exactly integrate polynomial functions up to order $2s$ using only s nodes (for s even) for univariate problems $n_w = 1$. The extension to $n_w > 1$, in the form of a tensor product of all combination of univariate quadrature points, still suffers from the *curse-of-dimensionality*. Sparse grid quadrature is a popular alternative to Gaussian quadrature for high-dimensional integration, as it can significantly reduce the number of nodes in the full tensor grid without sacrificing accuracy (Gerstner and Griebel, 2003). This approach has been successfully demonstrated in the context of msMPC in Leidereiter et al. (2014).

In addition to sparse grid quadrature, optimization-based methods can be used to systematically locate nodes and weights in (20) so that the approximation error is minimized. This includes *moment-matching optimization* methods (Ryu and Boyd, 2015), which are a natural extension of Gaussian quadrature to the multivariate case, and *optimized stochastic collocation* methods (Sinsbeck and Nowak, 2015) that are a quasi-optimal procedure for minimizing the quadrature operator's error norm. Promising results have been demonstrated with these methods in, e.g., Paulson and Mesbah (2018b,a); Paulson et al. (2019), which motivate their potential use in msMPC.

Remark 8. Although MC or quadrature can be used to approximate the expectation, they may not provide an adequate approximation of the worst-case uncertainty value that can be important for guaranteeing closed-loop constraint satisfaction. In such cases, a hybrid approach that retains two separate sets of scenarios for the objective and constraints can be useful, as discussed in Rooney and Biegler (2003) for multiperiod design problems. Such a strategy, however, may result in significant online cost due to the complexity in locating the worst-case value of the uncertainty. One way to overcome this challenge is to apply a sensitivity-based prediction of the worst-case uncertainty value that has been shown to substantially reduce the size of the scenario tree in msMPC Holtorf et al. (2019).

4. Approximate LB-msMPC using deep learning

4.1. Neural network approximation of the control law

Even though there have been significant advances in algorithms and hardware over the past decade, solving the LB-msMPC problem defined in (17) in real time can be computationally challenging. This is further compounded by the inclusion of the GP prediction (17c), which requires the Cholesky decomposition $\Sigma^w(x_{k,j}, u_{k,j})^{1/2}$ to be computed at every node of the scenario tree – known to scale cubically with respect to the number of training points. To avoid the real-time solution of non-convex optimization problems, we instead look to develop a data-driven approximation of the implicit feedback control law (18). Neural network approximation of nonlinear MPC was originally proposed in Parisini and Zoppoli (1995), which used shallow networks with only one hidden layer that, according to the universal function approximation theorem, can approximate any function to any desired accuracy level under fairly mild conditions.

In this work, we focus on deep neural networks (DNNs) – with several hidden layers – as function approximators due to their enhanced approximation capabilities, as demonstrated in several recent works (Chen et al., 2018; Bonzanini et al., 2020a; Paulson and Mesbah, 2020; Zhang et al., 2019; Hertneck et al., 2018). Furthermore, recent theoretical results show that a DNN with a given size can exactly represent the MPC law for linear time-invariant systems (Karg and Lucia, 2018). The LB-msMPC problem (17) is a multiparametric optimization problem that depends on the current measured state (and potentially other parameters as noted in Remark 5). To construct a DNN approximation, a finite number of N_s feasible state samples $x^i \in \mathcal{X}_{\text{lb}}$ are selected and then N_s individual optimization problems are solved to obtain the corresponding optimal inputs $\kappa_{\text{lb}}(x^i) = u^\star_{0,1}(x^i)$. The training dataset can then be represented as

$$\mathcal{T} = \{(x^1, \kappa_{\text{lb}}(x^1)), \dots, (x^{N_s}, \kappa_{\text{lb}}(x^{N_s}))\}. \tag{21}$$

A standard feed-forward DNN with fully connected layers is defined as a function of the form

$$\mathcal{N}(x; \lambda) = \alpha_{L+1} \circ \beta_L \circ \alpha_L \circ \dots \circ \beta_1 \circ \alpha_1(x), \tag{22}$$

where L is the number of hidden layers, $\alpha_0(x) = W_0 x + b_0$ is an affine transformation of the input, $\alpha_l(\xi_{l-1}) = W_l \xi_{l-1} + b_l$ (with $\xi_l \in \mathbb{R}^H$ and H denotes the number of nodes per hidden layer) are affine transformations of the hidden layers for all $l \in \{1, \dots, L\}$, $\beta_l(\cdot)$ denotes the nonlinear activation functions for $l \in \{0, \dots, L-1\}$, and $\lambda = \{W_0, b_0, \dots, W_L, b_L\}$ denotes the collection of all unknown parameters in the network that consists of the weights W_l and biases b_l of the affine functions in each of the layers. Common choices for the activation function are rectified linear units (ReLU) and the sigmoid (hyperbolic tangent) function. Here we use ReLU activations due to their popularity in regression-based tasks.

For fixed values of the network dimensions L and H, the best approximation of the LB-msMPC control law (18) is defined as the one that minimizes a given loss function, such as the mean squared error (MSE), over the given training set $\mathcal{T}$

$$\lambda^\star = \underset{\lambda}{\text{argmin}} \frac{1}{N_s} \sum_{i=1}^{N_s} \| \kappa_{\text{lb}}(x^i) - \mathcal{N}(x^i; \lambda) \|^2. \tag{23}$$

The resulting deep learning-based LB-msMPC control law is denoted by $\kappa_{\text{dnn}}(x) = \mathcal{N}(x; \lambda^\star)$, which depends on H and L, as well as other "hyperparameters" in the training algorithm that must be selected before $\lambda^\star$ can be computed. A systematic approach for selecting these hyperparameters is described next.

4.2. Bayesian optimization for neural network hyperparameter selection

The goal of hyperparameter tuning in deep learning is to find the structure and training-related parameters of a DNN such that some loss function is minimized. In general, the network structure involves the number of layers and nodes, the activation function choice, and the various connections between layers. Examples of training-related parameters include the batch size, learning rate, and/or the decay rate.

One of the most commonly used approaches for selecting the hyperparameters is grid search wherein a separate network is trained by solving (23) for all possible hyperparameter values in a predefined set of options, e.g., Bergstra et al. (2011). This approach generally scales poorly with respect to the number of considered hyperparameters and thus is limited to a relatively small set of possibilities. A more sophisticated approach that is gaining popularity in machine learning is to tackle the hyperparameter selection problem using derivative-free optimization methods. In particular, Bayesian optimization (BO) (Snoek et al., 2012) has been shown to effectively minimize a given loss function, such as the MSE in (23), using a small number of objective function evaluations, and can straightforwardly handle a mixture of continuous and integer variables that often occur in training of DNNs.

The general notion underpinning BO is to approximate the objective function using a GP that is initially constructed from a small experimental design. The current GP is then used to decide which combination of hyperparameters is most promising by optimizing an acquisition function that balances exploration (searching the unknown parts of the function) and exploitation (evaluating near points that minimize the loss predicted by the GP) (Jalali et al., 2012). A new DNN is trained for the optimally chosen hyperparameter set to evaluate the loss function and this new data point is incorporated into the GP training data set. The GP is then updated given this new data point and the procedure repeats until some convergence criteria is met (or a maximum number of function evaluations is reached). The BO algorithm is illustrated in Fig. 2. A variety of acquisition functions can be used, including: (i) expected improvement, (ii) probability of improvement, and (iii) lower confidence bound. Interested readers are referred to Martinez-Cantin (2014) for additional details on the BO procedure.

4.3. Control-oriented generation of training data

The training dataset (21) must be generated before we can select the DNN weights and biases λ using (23) and the hyperparameters using BO. Ideally, the training samples $\{x^i\}_{i=1}^{N_s}$ should be selected such that N_s is as small as possible, while a desired accuracy level in the approximation of the control law is achieved. This requires defining the space over which the controller will be approximated – denoted by $\mathcal{X}$. We can generate samples in $\mathcal{X}$ using a variety of approaches, including MC sampling, quasi-random space filling sampling, and grid-based methods.

A natural choice for $\mathcal{X}$ is the working region of the controller $\mathcal{X}_{\text{lb}}$, but this requires knowledge of the domain of attraction of

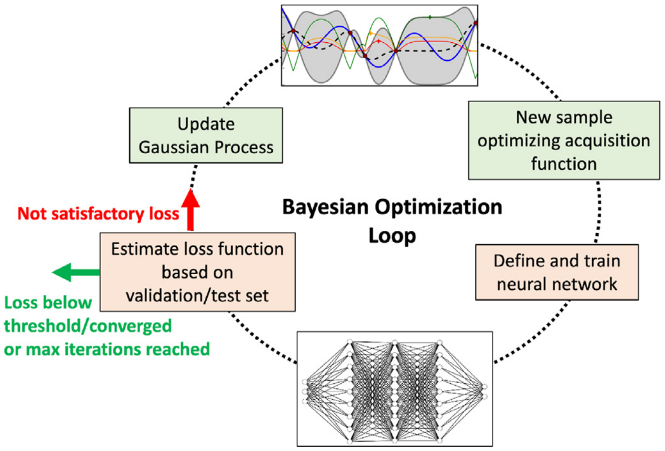

Update Gaussian Process
New sample optimizing acquisition function
Not satisfactory loss
Bayesian Optimization Loop
Estimate loss function based on validation/test set
Define and train neural network
Loss below threshold/converged or max iterations reached

Fig. 2. Illustration of the Bayesian optimization procedure for selection of DNN hyperparameters. Top figure in the loop shows a Gaussian process and corresponding acquisition functions, representing the concept of searching for the optimum, as described in Schneider et al. (2017).

the LB-msMPC that is difficult to obtain for general nonlinear systems. Furthermore, the set $\mathscr{X}_{\text{lb}}$ is unbounded whenever the operational constraints are softened, which is commonly done in practice to avoid infeasible optimization solutions. Assuming $\mathcal{Z}$ in (2) is a compact set, then an alternative is to select $\mathscr{X} = \mathcal{X}$ to be the set of states that are allowable during operation defined in (3). However, this assumes that the closed-loop system can be anywhere in the (potentially large) state constraint set, including values that are highly unlikely to be observed during closed-loop operation. This approach has been the focus of much of the recent work on DNN-based MPC, e.g., Chen et al. (2018); Bonzanini et al. (2020a); Paulson and Mesbah (2020); Hertneck et al. (2018), which we refer to as *open-loop training* since it neglects any potentially relevant information on the intended application of the controller.

Here, we propose an alternative method that looks to define a *control-oriented* representation of $\mathscr{X}$, so that the generated samples of the state are much more likely to be observed in actual closed-loop control implementations. The main idea is to focus on a subset of the state constraint space to learn a simpler DNN representation of κ_{lb} using significantly less training data. In particular, we propose to define the set $\mathscr{X}$ in terms of the following closed-loop tube of trajectories

$$\mathscr{X} = \textstyle\bigcup_{i=0}^{\mathrm{T}} X_i, \tag{24}$$

where X_i denotes the reachable set of states at time step i and T denotes the total number of time steps of interest. The reachable state sets can be defined as

$$X_{i+1} = f(X_i, \kappa_{\text{lb}}(X_i)) \oplus \mathcal{W}(X_i, \kappa_{\text{lb}}(X_i)), \tag{25}$$

for all $i \in \{0, \dots, T-1\}$ given an initial set of states X_0.

The key idea in the proposed approach is that $\mathscr{X}$ is defined by *closed-loop simulations* using the uncertain model $x^+ = f(x,u) + B_d(d(x,u) + v)$ as the best representation of the unknown true system (9). This serves as a proxy for the actual closed-loop system

$$x^+ = f(x, \kappa_{\text{lb}}(x)) + B_d(g(x, \kappa_{\text{lb}}(x)) + v), \tag{26}$$

that cannot be realized in practice because the LB-msMPC law is too expensive to evaluate in real-time. In other words, although it would be preferred to gather data from actual closed-loop control runs (26), this is not possible due to real-time limitations and instead we rely on an emulator of the system defined in terms of a GP model of the plant-model mismatch. It is important to note that the GP (14) is an explicit function of the system measurements at M different (x,u) pairs such that the simulated closed-loop tube (24) will approach the unknown tube defined in terms of (26) as $M \to \infty$.

Even using the GP-based uncertainty description, we still cannot explicitly compute $\mathscr{X}$ in (24) due to the underlying nonlinear dynamics. Instead, we generate random samples within $\mathscr{X}$ using the procedure summarized in Algorithm 1. First, an initial state is sampled from the pre-specified set X_0. The LB-msMPC law is then evaluated at this initial state by solving (17) numerically. A realization from the GP model is then drawn according to (16) that is projected onto the level set (19) to ensure the system evolution remains bounded. The control input and random disturbance values are then used to update the state of the system and the overall process is repeated until either the final time T or a target set X_{target} is reached. The target set is incorporated into the algorithm to account for the fact that one may want to stop a given simulation once a certain goal has been met (e.g., a setpoint has been reached). Furthermore, note that $\mathscr{X} \not\subset \mathcal{X}$ since constraint satisfaction has not been guaranteed in the design of κ_{lb} (see Remark 7). This can be mitigated by incorporating explicit backoffs into the design of the LB-msMPC law. Interested readers are referred to Paulson and Mesbah (2018a) for more a detailed discussion on explicit backoffs in the context of nonlinear MPC.

Algorithm 1 Control-oriented learning of DNN-based LB-msMPC with closed-loop simulation data.

Input: Nominal dynamics f, GP model of the uncertainty (15), number of simulations N_{sim}, set of initial states X_0, number of time steps T, and target set X_{target}.
Output: Training set $\mathcal{T}$.

```
for $i = 1$ to $N_{\text{sim}}$ do
    $x \leftarrow$ sample from the set~$X_0$
    for $j = 1$ to $T$ do
        $u \leftarrow \kappa_{\text{lb}}(x)$
        $\mathcal{T} \leftarrow \mathcal{T} \cup \{(x,u)\}$
        Use GP to evaluate $\mu^w(x,u)$ and $\Sigma^w(x,u)$
        $w \leftarrow$ sample from~$\mathcal{N}(\mu^w(x,u), \Sigma^w(x,u))$
        Project $w$ into level set $\mathcal{W}(x,u)$
        $x \leftarrow f(x,u) + w$
        if $x \in X_{\text{target}}$ then
            break
```

The set X_0 plays a key role in Algorithm 1. We generally expect that larger X_0 results in a larger $\mathscr{X}$ such that the controller must be approximated in a larger region. Thus, from a training point-of-view, it is desired to limit the size of X_0 as much as possible, which implies strong benefits are likely to be observed for batch, startup, and changeover control problems that consider a fairly limited window of initial states. In addition, the level sets (19) influence the growth of the tube over time – increasing α implies more uncertainty and thus increases the size of $\mathscr{X}$. This provides a mechanism to encode our belief in the GP prediction of the plant-model mismatch term that directly influences the size of the closed-loop tube $\mathscr{X}$. Lastly, we note that an important advantage of Algorithm 1 over open-loop training is that we can take advantage of warm starting to accelerate convergence of κ_{lb} evaluations for $i = 1, \dots, T$, as long as the successor state does not move significantly far away from the current state.

Remark 9. Algorithm 1 uses the nominal model, along with the GP-based uncertainty predictions, to perform closed-loop simulations. This control-relevant model can be straightforwardly replaced with any (potentially black-box) high-fidelity simulation of the process. The use of high-fidelity models for closed-loop validation and tuning of nonlinear MPC is discussed in Paulson and Mesbah (2018b).

5. Case study: atmospheric pressure plasma jet for biomaterials processing

We apply the proposed LB-msMPC and approximate LB-msMPC strategies for predictive control of a RF-excited atmospheric pressure plasma jet (APPJ) in Argon Gidon et al. (2018, 2019). APPJs are a class of widely used cold atmospheric plasma devices for surface processing and biomedical applications (Laroussi et al., 2012; Metelmann et al., 2018; Mani et al., 2015; Morent, 2013). A major challenge in model-based control of APPJs arises from the complex and nonlinear nature of the plasma dynamics, which generally cannot be described by first-principles models that are amenable to real-time computations (Gidon et al., 2017). Most model-based control strategies for APPJs rely on linear system identification models that inevitably exhibit a large plant-model mismatch (Gidon et al., 2018; 2019). This motivates the use of learning-based control such as the proposed LB-msMPC to enable safe and effective operation of APPJs, in particular for safety-critical applications in plasma medicine. In addition, the fast dynamics of plasmas necessitate fast control implementations on the millisecond timescale.

5.1. Problem formulation

A high-fidelity model of the RF-excited APPJ in Argon is used here as the plant model, which is described by a differential-algebraic system of equations (see Gidon et al., 2017 for details). Similarly to Gidon et al. (2018), we use subspace identification to obtain a control-relevant model suitable for describing the plasma dynamics within the operating range of interest. In particular, we apply a sequence of inputs to generate plant data, from which we identified a linear time-invariant (LTI) model

$$\begin{bmatrix} T_s \\ T_g \end{bmatrix}^+ = \begin{bmatrix} 0.42 & 0.68 \\ -0.06 & 0.26 \end{bmatrix}\begin{bmatrix} T_s \\ T_g \end{bmatrix} + \begin{bmatrix} 1.58 & -1.02 \\ 0.73 & 0.03 \end{bmatrix}\begin{bmatrix} P \\ q \end{bmatrix}, \tag{27}$$

where the states correspond to surface temperature T_s and gas temperature T_g, while the inputs correspond to the flowrate of Argon q and the applied power to the plasma P. The LTI model (27) is then augmented with the following nonlinear expression for the thermal dose delivered to the target surface measured in terms of cumulative equivalent minutes (CEM) (Gidon et al., 2017; Dewhirst et al., 2003)

$$\text{CEM}^+ = \text{CEM} + K^{(43-T_s)}\delta t, \tag{28}$$

where K is given by

$$K = \begin{cases} 0.5, & \text{if } T_s \geq 35\ ^\circ\text{C} \\ 0, & \text{otherwise.} \end{cases} \tag{29}$$

Note that the thermal dose CEM has the unit of minutes (min). As such, the control-relevant states and inputs are given by $x = [T_s, T_g, \text{CEM}]^\top$ and $u = [P, q]^\top$, respectively. Using the simulated data obtained from the high-fidelity plant model, we learn a GP model of the plant-model mismatch by using $M = 100$ measurements of the form (10), with additive noise $v \sim \mathcal{N}(0, 0.4^2 I_2)$.

The control objective is to achieve a specified thermal dose CEM_{sp}, which implies the following choice of stage and terminal costs

$$\ell(x,u) = 0, \quad \ell_f(x_N) = (\text{CEM}_N - \text{CEM}_{\text{sp}})^2. \tag{30}$$

In addition, the state and input are constrained to the following regions to ensure both safety of the APPJ during operation and reasonable accuracy of the model

$$\begin{bmatrix} 25\ ^\circ\text{C} \\ 20\ ^\circ\text{C} \end{bmatrix} \leq \begin{bmatrix} T_s \\ T_g \end{bmatrix} \leq \begin{bmatrix} 42.5\ ^\circ\text{C} \\ 80\ ^\circ\text{C} \end{bmatrix}, \tag{31}$$

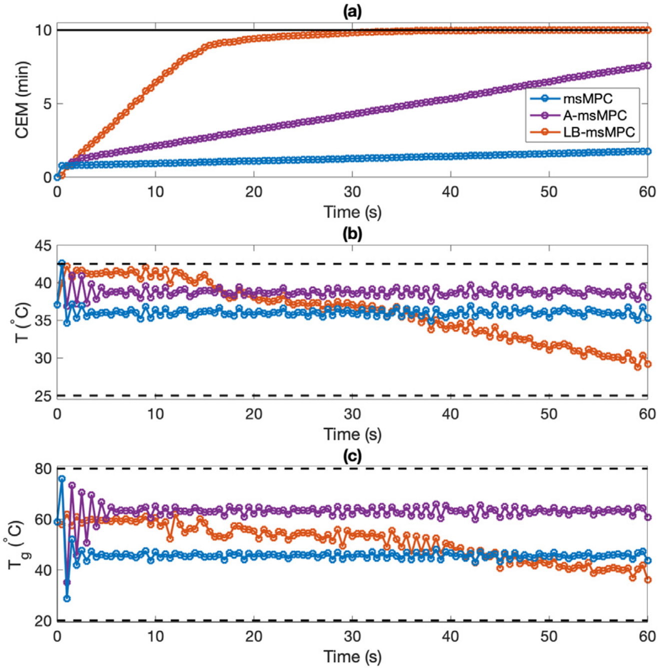


Fig. 3. Closed-loop simulations of different multi-stage MPC strategies: Worst-case msMPC (blue), Adaptive msMPC (purple), and LB-msMPC (red). (a–c) State profiles: Thermal dose CEM, surface temperature T, and gas temperature T_g. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$\begin{bmatrix} 1.5\ \text{W} \\ 1.0\ \text{slm} \end{bmatrix} \leq \begin{bmatrix} P \\ q \end{bmatrix} \leq \begin{bmatrix} 8\ \text{W} \\ 6\ \text{slm} \end{bmatrix}.$$

5.2. LB-msMPC for thermal dose delivery

We now demonstrate the effectiveness of the proposed LB-msMPC strategy described in Section 3, which is compared to two alternative formulations. The first is worst-case msMPC that uses fixed bounds derived offline according to the minimum and maximum values of $\mathcal{W}(x,u)$ in each dimension for all states and inputs satisfying (31), i.e., outer-bounding hyper-rectangle for the state- and input-dependent disturbance set. The second is a heuristic approach that we refer to as adaptive msMPC (A-msMPC) that avoids embedding the GP model within the optimal control problem, but instead updates the set of scenarios based on the most recent state and previous input values. This is similar to the idea recently proposed in Thombre et al. (2019) to update the scenario tree online using recent measurements.

All MPC problems are solved in Matlab using CasADi Andersson et al. (2019) and IPOPT Wächter and Biegler (2006) with a prediction horizon $N = 5$, robust horizon $N_r = 1$, i.e., 2 prediction steps, and maximum plasma treatment time of 60 s. A set of $3^2 = 9$ uncertainty scenarios are considered in each MPC formulation, as discussed in more detail below. The treatment must be stopped as soon as the target CEM value, i.e., $\text{CEM}_{\text{sp}} = 10$ min is reached. It is desired to deliver the target CEM as fast as possible, leading to a shorter treatment time. The performance of the three control strategies is summarized in Fig. 3.

We first examine worst-case msMPC as this is the baseline for our comparison. In this case, the scenarios correspond to all possible combinations of minimum, nominal, and maximum values $\{[w_{\min}]_i, 0, [w_{\max}]_i\}_{i=1}^2$, as estimated from the level sets $\mathcal{W}(x,u)$. The consideration of the full uncertainty range at every time step makes the controller conservative, as evidenced by the temperature trajectory staying well below its upper limit. This greatly re-

Table 1
Optimal hyperparameters of the DNN and corresponding ranges considered in the Bayesian optimization.

Hyperparameter	Range	Optimal value
L	[1,6]	3
H	[3,20]	18
μ	$[10^{-4}, 10^{-1}]$	4×10^{-2}
μ_r	$5 \times [10^{-3}, 10^{-1}]$	3.2×10^{-1}

stricts the growth of the CEM dose, which fails to reach the desired target by the end of the treatment time. This overly conservative performance is a direct consequence of the conservative estimates of the uncertainty bounds. The proposed LB-msMPC strategy, on the other hand, consistently updates the prediction of the uncertainty according to the GP model (14). Hence, the branching at each stage is given by (16) with $[\xi_k]_i \in \{-r^{-1/2}(\alpha), 0, r^{-1/2}(\alpha)\}$ for $\alpha = 0.99$. Since the scenario tree now explicitly captures the state- and input-dependence of the uncertainty, we observe a significant reduction in conservatism in Fig. 3. For example, the temperature trajectory is now able to push closer to its bound, which results in a faster increase in the delivered thermal dose CEM and thus a shorter treatment time. It is important to note that this faster treatment time did not come at the expense of constraint violations, as the LB-msMPC provides an accurate characterization of the plant-model mismatch through the GP model.

Lastly, we consider the A-msMPC strategy as a heuristic simplification of the LB-msMPC controller, which considers scenarios of the form $\{[w_{k,\min}]_i, 0, [w_{k,\max}]_i\}_{i=1}^2$ that are estimated at every time step k using the GP model. In particular, the most recent state measurement and a best guess for the current control input are substituted into the GP model to estimate the uncertainty bounds that are held constant over the prediction horizon. As such, A-msMPC has an online cost similar to worst-case msMPC, but some ability to account for state- and input-dependent plant-model mismatch. As shown in Fig. 3, although A-msMPC exhibits a considerably better closed-loop performance compared to msMPC, it fails to reach the desired CEM target within the maximum plasma treatment time of 60 s.

5.3. DNN-based approximation of LB-msMPC

The analysis in the previous section indicates that the LB-msMPC strategy offers the best closed-loop performance in terms of thermal dose delivery while respecting state constraints. We now construct a DNN approximation of the LB-msMPC control law, as discussed in Section 4. Closed-loop training data are generated according to Algorithm 1, with $T = 60$ s, $N_{\text{sim}} = 120$, and $X_0 = \{x_0\}$, where $x_0 = [37\ ^\circ\text{C}, 58.87\,^\circ\text{C}, 0\ \text{min}]^\top$. The Bayesian optimization (BO) loop, shown in Fig. 2, is used to select the hyperparameters of the DNN. In this case, we treat four hyperparameters as decision variables: the number of nodes H, the number of layers L, the adaptation parameter μ, and the initial rate of decrease of the adaptation parameter μ_r in the Levenberg-Marquardt training algorithm (Hagan and Menhaj, 1994). The optimal hyperparameter values and their corresponding bounds are shown in Table 1. Note that an MSE loss function and ReLU activation functions were chosen, and the BO was performed using `bayesopt` in Matlab. We avoid using a very large number of layers and nodes for two reasons. First, smaller neural networks have smaller memory requirements and, as a result, are more practical for embedded applications (Bonzanini et al., 2020a). Second, less complex networks alleviate overfitting. Fig. 4 depicts the best MSE error achieved during the BO iterations when using a total of 6000 training samples. The BO converges after 41 iterations. We expect that performing a grid search for selecting the hyperparameters, as commonly done, would require many more iterations. The DNN training converges after 132 epochs, in approximately 17 s on a MacBook with a 2.4 GHz Intel Core i9 CPU. The best performance was observed at epoch 92, where the validation MSE is 0.0182. We note that the training is rather fast since the DNN has a relatively simple structure, meaning it is not too deep.

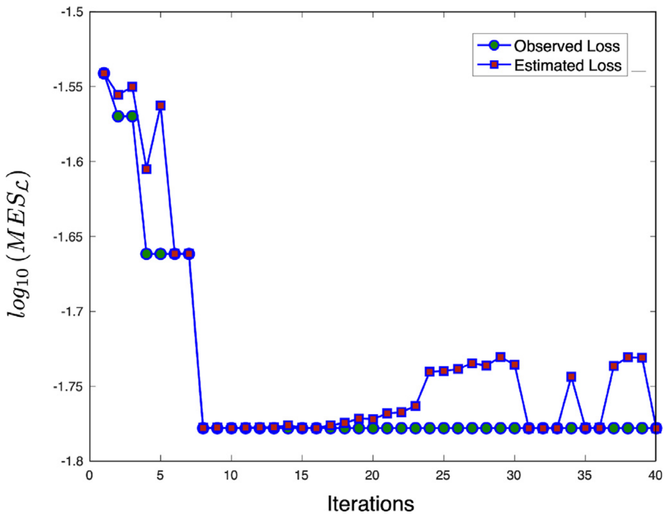


Fig. 4. Best mean squared error (MSE) loss of the κ_{dnn} control law approximation achieved in the Bayesian hyperparameter optimization. The green circles represent the actual loss evaluations (objective function), while the purple squares represent the estimated loss based on the GP surrogate model. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 2
Average performance of LB-msMPC and DNN-based LB-msMPC under different uncertainty realizations.

	κ_{lb}	κ_{dnn}
Online cost (ms)	177.5	40.3
Mean absolute error	–	0.1017

Fig. 5 compares the performance of the approximate control law κ_{dnn} to that of the LB-msMPC law κ_{lb}. We superimpose the results of 10 closed-loop simulations, each of which corresponding to different noise realizations. We observe that the closed-loop thermal dose CEM profile and the state profiles of the LB-msMPC controller and the DNN-based approximation are nearly identical. This can be attributed to training the DNN using closed-loop data. By examining the optimal input sequences, it is evident that κ_{dnn} closely matches that of κ_{lb}, with minor discrepancies that can be attributed to the DNN approximation error. It should be noted that this error propagates to future control actions. In other words, a slight difference on the first control action taken by the DNN controller leads to different measurements when applied to the system, and, subsequently, different control actions in the next step. As shown in the inset in Fig. 5b, neither the LB-msMPC controller nor its DNN-based approximation violate the surface temperature constraint, which suggests that both controllers can push the system to operate near its constraint. To evaluate the ability of κ_{dnn} in approximating the LB-msMPC law κ_{lb} throughout the relevant state space, $\mathscr{X}$, we run 50 closed-loop validation simulations over a course of $N_{\text{sim}} = 120$ steps, which yields a total of 6000 evaluations of the control law. Table 2 lists the mean average error (MAE) of the DNN-based control law κ_{dnn}. As expected, the MAE estimated via the closed-loop simulations is consistent with the MSE of the DNN, as reported in Fig. 4. Fig. 5 suggests that the greatest discrepancy between the two control laws is observed for

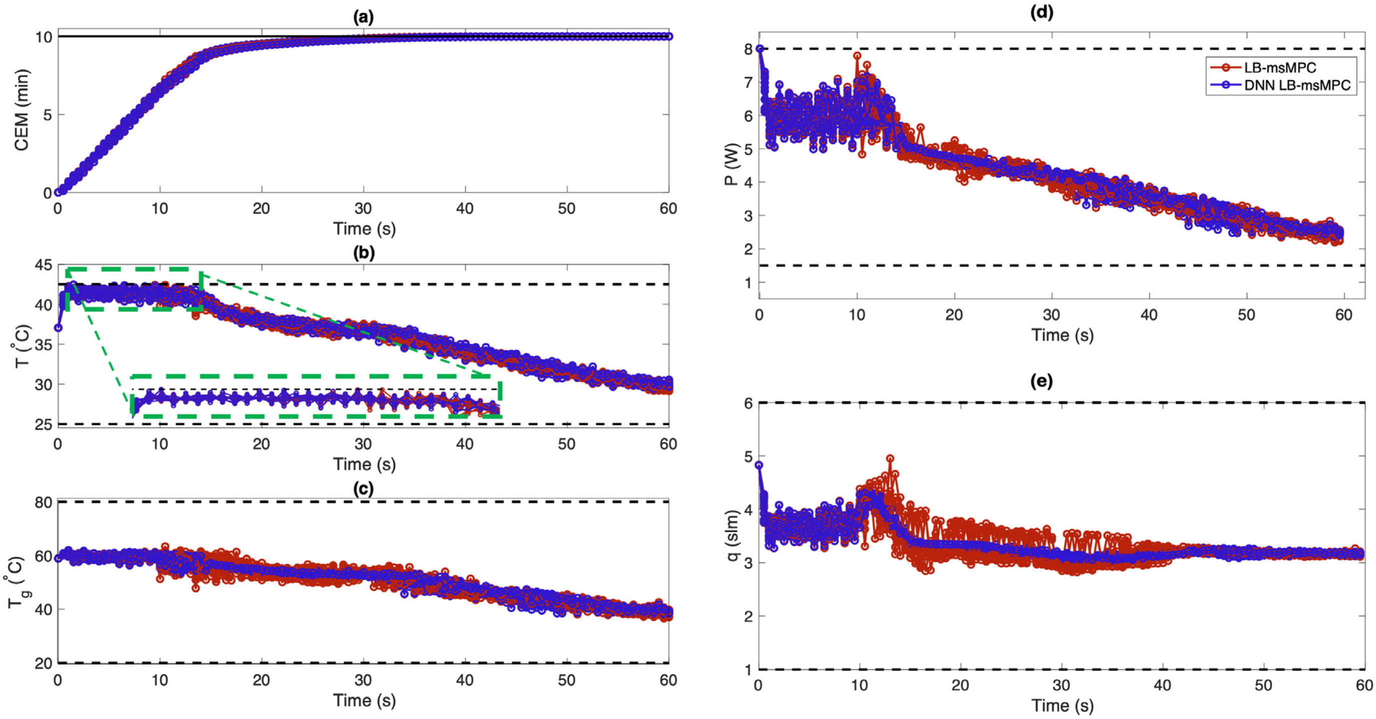


Fig. 5. Closed-loop simulations of LB-msMPC (red) and DNN-based approximate LB-msMPC (blue). (a–c) State profiles: Thermal dose CEM, surface temperature T, and gas temperature T_g. (d–e) Input profiles: Applied power P and Argon flow rate q. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the flow rate at time instants where CEM experiences the most abrupt changes as it gets close to the setpoint and, thus, the rate of change needs to abruptly decrease. Nevertheless, κ_{dnn} can be evaluated more than 4 times faster than κ_{lb} (see Table 2). The computation time reported for κ_{lb} is the evaluation time of the OCP (17), which includes the adaptation of the scenario tree in prediction using the GP. On the other hand, the computation time reported for κ_{dnn} includes the evaluation of the DNN, which has already been trained offline using closed-loop data. It should also be noted that the average computation time of LB-msMPC is expected to increase substantially with the problem complexity, whereas the computation time of the DNN-based control law mainly depends on the complexity of the DNN structure (Bonzanini et al., 2020a).

6. Conclusions and future work

We presented a learning-based multistage MPC strategy for uncertain nonlinear systems with hard-to-model and time-varying dynamics. The LB-msMPC strategy uses Gaussian processes to learn a state- and input-dependent description of plant-model mismatch and accordingly adapt the scenario tree in real-time. To address the rapid scaling of the computational complexity of the LB-msMPC with the number of branches and prediction horizon, the control law was approximated using DNNs to obtain a cheap-to-evaluate explicit control law. We presented an algorithm that uses a Gaussian process representation of plant-model mismatch to generate closed-loop data for training the DNN-based control law in the state-space regions most relevant to control. The effectiveness of the proposed LB-msMPC and approximate LB-msMPC strategies was demonstrated on a simulated cold atmospheric plasma system used for (bio)materials processing. We showed that the LB-msMPC strategy can effectively reduce conservativeness of worst-case msMPC. Furthermore, the DNN-based approximation of the LB-msMPC law can significantly reduce the online computational cost of control, while exhibiting an almost indistinguishable performance from that of the LB-msMPC strategy. Future work will focus on deploying such approximate learning-based MPC laws in resource-limited embedded systems, and provide constraint satisfaction guarantees using projected neural networks (Paulson and Mesbah, 2020).

Declaration of Competing Interest

The authors declare no conflict of interest.

Acknowledgments

This work was supported by the National Science Foundation under Grant 1839527. This material is also in part based upon work supported by the National Aeronautics and Space Administration (NASA) under grant number NNX17AJ31G. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NASA.

References

Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., Diehl, M., 2019. CasADi–a software framework for nonlinear optimization and optimal control. Math. Program. Comput. 11 (1), 1–36.

Aswani, A., Gonzalez, H., Sastry, S.S., Tomlin, C., 2013. Provably safe and robust learning-based model predictive control. Automatica 49 (5), 1216–1226.

Bemporad, A., Borrelli, F., Morari, M., et al., 2002. Model predictive control based on linear programming – the explicit solution. IEEE Trans. Autom. Control 47 (12), 1974–1985.

Bemporad, A., Morari, M., 1999. Robust model predictive control: a survey. In: Robustness in Identification and Control. Springer, London, UK, pp. 207–226.

Bergstra, J.S., Bardenet, R., Bengio, Y., Kégl, B., 2011. Algorithms for hyper-parameter optimization. In: Advances in Neural Information Processing Systems, pp. 2546–2554.

Bernardini, D., Bemporad, A., 2009. Scenario-based model predictive control of stochastic constrained linear systems. In: Proceedings of the Conference on Decision and Control held jointly with the Chinese Control Conference, Shanghai, China, pp. 6333–6338.

Bonzanini, A.D., Mesbah, A., 2020. Learning-based stochastic model predictive control with state-dependent uncertainty. In: Proceedings of Machine Learning Research, 120.

Bonzanini, A.D., Paulson, J.A., Graves, D.B., Mesbah, A., 2020. Toward safe dose delivery in plasma medicine using projected neural network-based fast approximate NMPC. In: Proceedings of IFAC World Congress, Berlin, Germany.

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: Proceedings of the IEEE Conference on Decision and Control. Jeju Island, Republic of Korea. Accepted

Burbidge, R., Rowland, J.J., King, R.D., 2007. Active learning for regression based on query by committee. In: Yin, H., Tino, P., Corchado, E., Byrne, W., Yao, X. (Eds.), Intelligent Data Engineering and Automated Learning - IDEAL 2007. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 209–218.

Caflisch, R.E., 1998. Monte carlo and quasi-monte carlo methods. Acta Numer. 1998, 1–49.

Chakrabarty, A., Dinh, V., Corless, M.J., Rundell, A.E., Żak, S.H., Buzzard, G.T., 2016. Support vector machine informed explicit nonlinear model predictive control using low-discrepancy sequences. IEEE Trans. Autom. Control 62 (1), 135–148.

Chen, S., Saulnier, K., Atanasov, N., Lee, D.D., Kumar, V., Pappas, G.J., Morari, M., 2018. Approximating explicit model predictive control using constrained neural networks. In: Proceedings of the American Control Conference, Milwaukee, WI, pp. 1520–1527.

Csekő, L.H., Kvasnica, M., Lantos, B., 2015. Explicit MPC-based RBF neural network controller design with discrete-time actual Kalman filter for semiactive suspension. IEEE Trans. Control Syst. Technol. 23 (5), 1736–1753.

Dewhirst, M.W., Viglianti, B., Lora-Michiels, M., Hanson, M., Hoopes, P., 2003. Basic principles of thermal dosimetry and thermal thresholds for tissue damage from hyperthermia. Int. J. Hyperth. 19 (3), 267–294.

Gerstner, T., Griebel, M., 2003. Dimension–adaptive tensor–product quadrature. Computing 71 (1), 65–87.

Gidon, D., Curtis, B., Paulson, J.A., Graves, D.B., Mesbah, A., 2018. Model-based feedback control of a kHz-excited atmospheric pressure plasma jet. IEEE Trans. Radiat. Plasma Med. Sci. 2 (2), 129–137.

Gidon, D., Graves, D.B., Mesbah, A., 2017. Effective dose delivery in atmospheric pressure plasma jets for plasma medicine: a model predictive control approach. Plasma Sources Sci. Technol. 26 (8), 085005.

Gidon, D., Graves, D.B., Mesbah, A., 2019. Predictive control of 2D spatial thermal dose delivery in atmospheric pressure plasma jets. Plasma Sources Sci. Technol. 28 (8), 085001.

Hagan, M.T., Menhaj, M.B., 1994. Training feedforward networks with the marquardt algorithm. IEEE Trans. Neural Netw. 5 (6), 989–993.

Hertneck, M., Köhler, J., Trimpe, S., Allgöwer, F., 2018. Learning an approximate model predictive controller with guarantees. IEEE Control Syst. Lett. 2, 543–548.

Hewing, L., Kabzan, J., Zeilinger, M.N., 2019. Cautious model predictive control using Gaussian process regression. IEEE Trans. Control Syst. Technol. 28 (6), 2736–2743.

Holtorf, F., Mitsos, A., Biegler, L.T., 2019. Multistage NMPC with on-line generated scenario trees: application to a semi-batch polymerization process. J. Process. Control 80, 167–179.

Jalali, A., Azimi, J., Fern, X. Z., 2012. Exploration vs. exploitation in Bayesian optimization. CoRR abs/1204.00471204.0047.

Karg, B., Alamo, T., Lucia, S., 2019. Probabilistic performance validation of deep learning-based robust NMPC controllers. arXiv preprint arXiv:1910.13906

Karg, B., Lucia, S., 2018. Efficient representation and approximation of model predictive control laws via deep learning. arXiv preprint arXiv:1806.10644

Kerrigan, E.C., Maciejowski, J.M., 2000. Soft constraints and exact penalty functions in model predictive control. In: Proceedings of the UKACC International Conference on Control, Cambridge, UK.

Kleywegt, A.J., Shapiro, A., Homem-de Mello, T., 2002. The sample average approximation method for stochastic discrete optimization. SIAM J. Optim. 12, 479–502.

Klintberg, E., Dahl, J., Fredriksson, J., Gros, S., 2016. An improved dual newton strategy for scenario-tree MPC. In: Proceedings of the Conference on Decision and Control, Las Vegas, NV, pp. 3675–3681.

Kocijan, J., Murray-Smith, R., Rasmussen, C.E., Girard, A., 2004. Gaussian process model based predictive control. In: Proceedings of the American Control Conference, Boston, MA, pp. 2214–2219.

Krishnamoorthy, D., Foss, B., Skogestad, S., 2019. A primal decomposition algorithm for distributed multistage scenario model predictive control. J. Process. Control 81, 162–171.

Krishnamoorthy, D., Skogestad, S., Jaschke, J., 2019. Multistage model predictive control with online scenario tree update using recursive Bayesian weighting. In: Proceedings of the European Control Conference, pp. 1443–1448.

Laroussi, M., Kong, M., Morfill, G., Stolz, W., 2012. Plasma Medicine: Applications of Low-Temperature Gas Plasmas in Medicine and Biology. Cambridge University Press.

Leidereiter, C., Potschka, A., Bock, H.G., 2014. Quadrature-based scenario tree generation for nonlinear model predictive control. IFAC Proc. Vol. 47 (3), 11087–11092.

Likar, B., Kocijan, J., 2007. Predictive control of a gas–liquid separation plant based on a Gaussian process model. Comput. Chem. Eng. 31 (3), 142–152.

Lucia, S., Finkler, T., Engell, S., 2013. Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty. J. Process. Control 23 (9), 1306–1319.

Mani, M., Lane, B., Donmez, A., Feng, S., Moylan, S., Fesperman, R., 2015. Measurement Science Needs for Real-time Control of Additive Manufacturing Powder Bed Fusion Processes. Technical Report.

Martinez-Cantin, R., 2014. Bayesopt: a Bayesian optimization library for nonlinear optimization, experimental design and bandits. J. Mach. Learn. Res. 15 (1), 3735–3739.

Mayne, D.Q., 2014. Model predictive control: recent developments and future promise. Automatica 50 (12), 2967–2986.

Mesbah, A., 2016. Stochastic model predictive control: an overview and perspectives for future research. IEEE Control Syst. 36, 30–44.

Mesbah, A., 2018. Stochastic model predictive control with active uncertainty learning: a survey on dual control. Annu. Rev. Control 45, 107–117.

Metelmann, H.R., Seebauer, C., Rutkowski, R., Schuster, M., Bekeschus, S., Metelmann, P., 2018. Treating cancer with cold physical plasma: on the way to evidence-based medicine. Contrib. Plasma Phys. 58 (5), 1–5.

Morent, R., 2013. Editorial: atmospheric pressure plasma polymerization. Open Plasma Phys. J. 7 (1), 6.

Ostafew, C.J., Schoellig, A.P., Barfoot, T.D., 2014. Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments. In: Proceedings of the IEEE International Conference on Robotics and Automation, Hong Kong, pp. 4029–4036.

Parisini, T., Zoppoli, R., 1995. A receding-horizon regulator for nonlinear systems and a neural approximation. Automatica 31 (10), 1443–1451.

Paulson, J.A., Martin-Casas, M., Mesbah, A., 2019. Optimal Bayesian experiment design for nonlinear dynamic systems with chance constraints. J. Process. Control 77, 155–171.

Paulson, J.A., Mesbah, A., 2018. Nonlinear model predictive control with explicit backoffs for stochastic systems under arbitrary uncertainty. IFAC-PapersOnLine 51 (20), 523–534.

Paulson, J.A., Mesbah, A., 2018. Shaping the closed-loop behavior of nonlinear systems under probabilistic uncertainty using arbitrary polynomial chaos. In: Proceedings of the IEEE Conference on Decision and Control, Miami Beach, FL, pp. 6307–6313.

Paulson, J.A., Mesbah, A., 2020. Approximate closed-loop robust model predictive control with guaranteed stability and constraint satisfaction. IEEE Control Syst. Lett. 4, 719–724.

Rakovic, S.V., Kerrigan, E.C., Mayne, D.Q., Lygeros, J., 2006. Reachability analysis of discrete-time systems with disturbances. IEEE Trans. Autom. Control 51 (4), 546–561.

Rasmussen, C.E., Williams, C.K.I., 2006. Gaussian Processes for Machine Learning.. MIT Press, Cambridge, MA.

Rooney, W.C., Biegler, L.T., 2003. Optimal process design with model parameter uncertainty and process variability. AIChE J. 49, 438–449.

Ryu, E.K., Boyd, S.P., 2015. Extensions of Gauss quadrature via linear programming. Found. Comput. Math. 15, 953–971.

Schneider, P.-I., Santiago, X.G., Rockstuhl, C., Burger, S., 2017. Global optimization of complex optical structures using Bayesian optimization based on Gaussian processes. In: Digital Optical Technologies 2017, 10335. International Society for Optics and Photonics, p. 103350O.

Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., de Freitas, N., 2016. Taking the human out of the loop: a review of Bayesian optimization. Proc. IEEE 104 (1), 148–175.

Sinsbeck, M., Nowak, W., 2015. An optimal sampling rule for nonintrusive polynomial chaos expansions of expensive models. Int. J. Uncertain. Quantif. 5 (3), 275–295.

Snoek, J., Larochelle, H., Adams, R.P., 2012. Practical Bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems, pp. 2951–2959.

Soloperto, R., Müller, M.A., Trimpe, S., Allgöwer, F., 2018. Learning-based robust model predictive control with state-dependent uncertainty. IFAC-PapersOnLine 51 (20), 442–447.

Subramanian, S., Lucia, S., Engel, S., 2015. Adaptive multi-stage output feedback NMPC using the extended Kalman filter for time varying uncertainties applied to a CSTR. IFAC-PapersOnLine 48 (23), 242–247.

Thangavel, S., Paulen, R., Engell, S., 2020. Adaptive multi-stage NMPC using sigma point principles. In: Proceedings of the European Control Conference, Saint Petersburg, Russia.

Thombre, M., Krishnamoorthy, D., Jäschke, J., 2019. Data-driven online adaptation of the scenario-tree in multistage model predictive control. IFAC-PapersOnLine 52 (1), 461–467.

Umlauft, J., Beckers, T., Hirche, S., 2018. Scenario-based optimal control for Gaussian process state space models. In: Proceedings of the European Control Conference, Limassol, Cyprus, pp. 1386–1392.

Wächter, A., Biegler, L.T., 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. 106 (1), 25–57.

Willett, R., Nowak, R., Castro, R.M., 2006. Faster rates in regression via active learning. In: Weiss, Y., Schölkopf, B., Platt, J.C. (Eds.), Advances in Neural Information Processing Systems 18. MIT Press, pp. 179–186.

Zhang, X., Bujarbaruah, M., Borrelli, F., 2019. Safe and near-optimal policy learning for model predictive control using primal-dual neural networks. In: Proceedings of the American Control Conference, Philadelphia, PA, pp. 354–359.