Fusion of Machine Learning and MPC under Uncertainty: What Advances Are on the Horizon?

Ali Mesbah, Kim P. Wabersich, Angela P. Schoellig, Melanie N. Zeilinger, Sergio Lucia, Thomas A. Badgwell, and Joel A. Paulson

***Abstract*— This paper provides an overview of the recent research efforts on the integration of machine learning and model predictive control under uncertainty. The paper is organized as a collection of four major categories: learning models from system data and prior knowledge; learning control policy parameters from closed-loop performance data; learning efficient approximations of iterative online optimization from policy data; and learning optimal cost-to-go representations from closed-loop performance data. In addition to reviewing the relevant literature, the paper also offers perspectives for future research in each of these areas.**

I. Introduction

Model predictive control (MPC) has become the prime technology for optimization-based control of constrained, multivariable systems [1], [2], with diverse applications ranging from manufacturing and energy systems to automotive systems and robotics [3]. A great deal of MPC research in the past two decades has focused on model uncertainty handling to realize safe and robust closed-loop MPC performance in spite of, for example, inadequate data for system identification, restrictive model classes, or presence of exogenous disturbances. While robust and stochastic MPC strategies [4]–[6] enable a systematic treatment of various sources of system uncertainty affecting the MPC performance, mainly to ensure constraint satisfaction with respect to certain disturbance or uncertainty classes, they follow a strict separation between the offline MPC design and closed-loop application during which the controller remains largely unchanged. Such a paradigm for uncertainty handling may severely limit the MPC performance and compromise constraint satisfaction for complex dynamical systems with hard-to-model and time-varying behavior, e.g., in situations where *a priori* system knowledge and/or data is unavailable.

The proliferation of machine learning (ML) and artificial intelligence (AI), combined with the availability of increased computational and sensing capabilities in modern control systems, has led to a rapidly growing interest in the use of learning and data-driven techniques for MPC [7]. While most research in this area has focused on automatic and data-driven adaptation of the prediction model or uncertainty description, the opportunities for machine learning extend far beyond model improvement and adaption.

In this paper, we give an overview of the recent research efforts on the fusion of ML and MPC under uncertainty and discuss perspectives for future research in this rapidly evolving field. To this end, we first provide a broad overview of MPC for uncertain systems in order to motivate and frame these efforts. Consider the following representation of a nominal (or *certainty equivalence*) MPC problem

$$\min_{U_t} \quad J_f(x_{N|t}, t+N) + \sum_{k=0}^{N-1} \ell(x_{k|t}, u_{k|t}, t+k), \tag{1a}$$

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

$$(x_{k|t}, u_{k|t}) \in \mathcal{Z}, \tag{1c}$$

$$x_{N|t} \in \mathcal{X}_f, \tag{1d}$$

$$x_{0|t} = x_t, \quad \forall k = 0, \ldots, N-1. \tag{1e}$$

Here, (1a) defines the total cost function that is composed of a sum of stage costs $\ell(\cdot)$ over the prediction horizon N and a terminal cost $J_f(\cdot)$. The recursive equation (1b) defines the predicted dynamics of the system in which a (potentially time-varying) model of the true system, denoted by $f(\cdot)$, is used to predict the future system state $x_{k|t}$ given changes in the control input $u_{k|t}$. The subscript $k|t$ on any variable is used to denote the prediction of the variable k-steps ahead of the current time index t. (1c) represents fairly general mixed path constraints on the predicted states and inputs. The set $\mathcal{Z}$ often decomposes into a form such as $\mathcal{Z} = \mathcal{X} \times \mathcal{U}$, where $x_{k|t} \in \mathcal{X}$ denotes the state constraints and $u_{k|t} \in \mathcal{U}$ denotes the control input constraints. The terminal constraint (1d) can be used to restrict the allowable values of the state at the end of the prediction horizon; the set $\mathcal{X}_f$ can be used to ensure constraint satisfaction is possible for time steps beyond the prediction horizon when properly chosen (see, e.g., [8] for a detailed discussion on set invariance in the context of MPC). Lastly, (1e) specifies the initial state condition in terms of the true state x_t. Here, we assume the true system state can be perfectly measured for simplicity of exposition, but there has been significant work on the more general *output feedback* MPC that requires an additional state estimation step.

The decision variables in the MPC problem (1) are the control inputs over the prediction horizon, denoted by $U_t =$

AM is with the Department of Chemical and Biomolecular Engineering at the University of California, Berkeley, CA 94720, USA.

KPW and MNZ are with the Institute for Dynamic Systems and Control, ETH Zürich, Zürich, CH-8092, Switzerland.

APS is with the Institute for Aerospace Studies, University of Toronto, Toronto, ON, M3H 5T6, Canada.

SL is with the Laboratory of Process Automation Systems, TU Dortmund, Dortmund, Germany.

TAB is the Chief Technology Officer at Collaborative Systems Integration, Austin, TX 78704, USA.

JAP is with the Department of Chemical and Biomolecular Engineering at The Ohio State University, Columbus, OH 43210, USA.

$\{u_{0|t}, u_{1|t}, \ldots, u_{N-1|t}\}$. Let $U_t^\star(x_t, t)$ denote the optimal control input sequence that solves (1) at the current time t. The key idea in MPC is to implement only the first element of this optimal input sequence, which will be re-planned once a new state value is measured. This procedure implicitly defines a feedback control law $\pi^{\text{MPC}}(x_t, t) = u_{0|t}^\star(x_t, t)$, where $u_{0|t}^\star(x_t, t)$ is the first element of $U_t^\star(x_t, t)$. Although π^{MPC} provides some degree of inherent robustness to system uncertainty due to feedback from x_t, it may not be sufficient to handle large uncertainties that may arise from errors in the model structure and/or parameters, or significant exogenous disturbances. Thus, as discussed above, there have been significant efforts to develop MPC strategies that are *by design* robust. Broadly speaking, these strategies modify the formulation (1) to include prediction models that explicitly incorporate uncertainty into the predictions.

A fairly general way to represent system uncertainties is in the form of stochastic variables [6]. Accordingly, the MPC of uncertain systems can take the general form of a closed-loop stochastic MPC (CL-SMPC) problem as follows

$$\min_{\Pi_t} \ \mathbb{E}\left\{ J_f(x_{N|t}, t+N) + \sum_{k=0}^{N-1} \ell(x_{k|t}, u_{k|t}, t+k) \right\}, \tag{2a}$$

$$\text{s.t. } x_{k+1|t} = f(x_{k|t}, u_{k|t}, t+k, w_{k|t}, \theta_t), \tag{2b}$$

$$u_{k|t} = \mu_{k|t}(x_{0|t}, \ldots, x_{k|t}), \tag{2c}$$

$$\Pr\left\{(x_{k|t}, u_{k|t}) \in \mathcal{Z}\right\} \geq \beta, \tag{2d}$$

$$\Pr\left\{x_{N|t} \in \mathcal{X}_f\right\} \geq \beta_f, \tag{2e}$$

$$w_{k|t} \sim P_{w_{k|t}}(\cdot | x_{k|t}, u_{k|t}, t+k), \quad \theta_t \sim P_{\theta_t}, \tag{2f}$$

$$x_{0|t} = x_t, \quad \forall k = 1, \ldots, N-1, \tag{2g}$$

where (2a) is similar to (1a), except we now must take an expectation $\mathbb{E}\{\cdot\}$ of the cost function with respect to the random predicted disturbance sequence $\{w_{0|t}, \ldots, w_{N-t|t}\}$ and the random variable θ_t that represents model parameter uncertainty; (2b) is the stochastic prediction model that defines the evolution of the random state over time; (2c) specifies the current control input in terms of a control law $\mu_{k|t}$, which is generally a function that can make use of all information in the form of state measurements up until the current time step; (2d) and (2e) are analogous to (1c) and (1d), respectively, but stated as *chance constraints* due to the inclusion of stochastic uncertainties; (2f) specifies the probability distributions of the predicted disturbance sequence and model parameter uncertainty; and the initial condition (2g) remains exactly the same as (1e).

The decision variables in the CL-SMPC problem are the *control policies* over the prediction horizon, denoted by $\Pi_t = \{\mu_{0|t}, \ldots, \mu_{N-1|t}\}$. Let $\Pi_t^\star(x_t, t)$ denote the optimal control policy sequence that solves (2) at the current time index t. Similarly to nominal MPC, we look to implement this implicitly defined feedback controller in a receding-horizon fashion, which induces the CL-SMPC control law

$$\pi^{\text{CL-SMPC}}(x_t, t) = \mu_{0|t}^\star(x_{0|t}; x_t, t), \tag{3}$$

where $\mu_{0|t}^\star$ is the first element of the optimal control policy sequence $\Pi_t^\star(x_t, t)$. Although CL-SMPC has the potential to greatly improve robustness to system uncertainties when compared to nominal MPC, it is not tractable to evaluate $\pi^{\text{CL-SMPC}}$ by solving (2) at every time index t. The two fundamental challenges in solving (2) are: (i) Π_t is composed of a sequence of functions that can arbitrarily map the states to control inputs, which are infinite dimensional objects that cannot be optimized over using readily available methods; and (ii) the probabilistic operators that appear in the cost and chance constraint functions involve high-dimensional integrals over generally nonlinear functions and support sets. In fact, robust and stochastic MPC strategies look to approximate (2) in view of these challenges (e.g., [4]–[6]). Many of these strategies are capable of providing robust or probabilistic stability and constraint satisfaction guarantees under certain modifications to (2), though these guarantees may be conservative due to the suboptimality of the introduced approximations. There is typically a natural tradeoff between the accuracy of the control policy approximation, as well as uncertainty propagation, and the resulting computational cost. Yet, there has been limited research on the relationship between the policy representation and the corresponding uncertainty propagation technique.

The focus of this paper is on how ML techniques can be effectively fused with MPC under uncertainty formulations that in essence approximate the CL-SMPC problem (2). ML can aid in the automated and data-driven adaptation or generation of different elements of these MPC formulations such that the closed-loop control performance with respect to that of the desired theoretical performance attained by exactly solving (2) is improved. The ways in which ML can assist the design of MPC under uncertainty are diverse. This paper is organized as a collection of contributions to discuss four major categories in learning-assisted MPC design under uncertainty:

1) **Learning model and uncertainty descriptions from system data (Section II):** The MPC performance and computational cost critically hinge on the availability of a sufficiently accurate and suitable model of system dynamics and its uncertainty descriptions for the application at hand. System data can be used to automatically adapt model descriptions during closed-loop operation, or in between operation episodes. Section II is led by Kim Wabersich, Angela Schoellig and Melanie Zeilinger.
2) **Learning control policy parameters from closed-loop performance data (Section III):** Approximate parameterizations of the general control policy in (2) typically involve several choices and parameters. These control design choices, along with the typical MPC tuning parameters, can be viewed as hyperparameters of the MPC under uncertainty formulations, which can be automatically selected using closed-loop performance data. Section III is led by Joel Paulson.
3) **Learning efficient approximations of iterative online**

optimization methods from policy data (Section IV): Approximations of the CL-SMPC control law (3) may still be computationally expensive to evaluate in real-time, particularly for fast-sampling systems. Furthermore, online optimization algorithms may have large working memory requirements that prevent their use in certain embedded control applications. Open- or closed-loop data of the implicitly-defined MPC law, generated offline, can be used to learn explicit and cheap-to-evaluate control laws. Section IV is led by Sergio Lucia.

4) **Learning optimal cost-to-go representations from closed-loop performance data (Section V):** The CL-SMPC problem (2) itself is an approximation of an underlying "true" stochastic optimal control problem, whose optimality conditions can be expressed as a stochastic dynamic program. Problem (2) has a close relationship with *Bellman's optimality conditions*, which constitute the basis for a large family of reinforcement learning (RL) methods. Not only can we interpret (2) in terms of a specific class of RL methods, we can actually combine MPC and RL together to potentially improve closed-loop performance over time. One such example would be using RL to determine a better cost-to-go representation using performance data generated in a closed-loop fashion. Section V is led by Thomas Badgwell, Joel Paulson, and Ali Mesbah.

In the remainder of this paper, we discuss these categories. We argue how the learning problem in each of these categories can be tackled using the "right" choice of data (e.g., system data or closed-loop performance data) and structure of a learnable model that can range from parametric data-driven representations to universal function approximators.

II. Learning System Model and Uncertainty Descriptions

Improved sensing and computational capabilities have led to a paradigm shift in model-based control from first-principles approaches that require relatively few measurements to fully data-enabled (grey-box) models. The corresponding developments extend the field of system identification [9] towards more complex model structures and towards robust or probabilistic models. They have showcased improved controller performance with reduced manual modeling effort. In particular, MPC has seen significant success in this regard as it provides systematic handling of constraints and allows for a principled integration of data-driven models. While we focus on models related to learning-based MPC research [7, Section 3], simulation environments can equally benefit from the presented methods for reliable tuning of model-free controllers and closed-loop system verification. An important aspect that we do not cover here is the design of experiments to obtain information-rich data from a system in a safe and efficient manner. While the application of open-loop input signals can be sufficient in case of linear systems [9], nonlinear systems typically require closed-loop excitation mechanisms, which is an active field of research, see, for example, [10], [11].

We consider discrete-time dynamical systems of the form

$$x(k+1) = f_{\mathrm{true}}(x(k), u(k), k, w(k), \theta_{\mathrm{true}}), \qquad (4)$$

where $x(k) \in \mathbb{R}^{n_x}$ is the system state and $u(k) \in \mathbb{R}^{n_u}$ is the applied input at time step k. The quantity $w(k)$ is a sequence of random variables with known bound or distribution, typically describing the effects of external systems and the environment such as wind or temperature, which can vary at every sampling time. The subscript 'true' in (4) highlights the *true* but often *unknown* quantities of the control system, which we categorize into structural uncertainties of the model f_{true} and parametric uncertainties of the parameters θ_{true}. Structural uncertainty can be induced, for example, by simplifications made when deriving first-principle models, such as linear friction models or neglected complex aerodynamic effects. Parametric uncertainties θ_{true} quantify uncertain system characteristics that are constant for all times. Such parametric uncertainties often occur in grey-box models, for example, in the form of mass or spring constants, or due to manufacturing tolerances of a system. Data-driven modeling techniques aim at reducing these sources of uncertainties with little manual effort using available measurements, as detailed in the following.

We assume access to a measured state and input sequence

$$X := [x(0), .., x(N-1)],\ U := [u(0), .., u(N-1)] \qquad (5)$$

of length N of system (4) in combination with some information about $w(k)$, depending on the specific learning technique. While we consider data in the form of (5) for simplicity, all methods presented in this part can equally handle episodic measurements, which include multiple resets of the system state. Importantly, the amount of information that (5) contains typically determines the resulting prediction accuracy. In the case of approximate linear systems, it is, e.g., possible to characterize sufficient conditions on system data for parameter identification through a so-called persistency of excitation condition. Corresponding input signals can be designed using, e.g., generalized binary noise or sum of sinusoids [12, Section 3.3.2]. More general, the challenge of gathering informative trajectories for model learning and control, including nonlinear system models, is considered in the field of dual control, see, e.g., [13] for an overview. In addition to measurements (5), most techniques can efficiently incorporate existing prior knowledge about the system in various forms as discussed in the corresponding sections.

Prediction models of (4) are decomposed into

$$f(x, u, k, w) = f_n(x, u, k, \theta_n) + f_l(x, u, k, w, \theta), \qquad (6)$$

containing a nominal model f_n with nominal parameters θ_n and a learning-based model f_l. In the following, we present learning methods to infer models of the form (6) from available data (5). While we briefly review so-called nominal models in Section II-A, the main focus will be on methods that allow for a principled treatment of the resulting model uncertainty, which is critical for performance and safety certificates. Thereby, we distinguish between model

learning schemes that have evolved from the extensively studied field of robust control theory in Section II-B and successful probabilistic learning formulations in Section II-C, which are typically more challenging to exploit in control.

Remark 1: Here, we consider access to full state measurements (5), whereas in practice often only output measurements $y(k) = g(x(k))$, $y(k) \in \mathbb{R}^{n_y}$ with $n_y < n_x$ are available. One approach to recover models of the form (6) is to define the system state as $x(k) = [y(k), y(k-1), .., y(k-m_y), u(k-1), u(k-2), .., u(k-m_u)]$, yielding an autoregressive model [9], [14], [15]. Furthermore, there is a large body of literature considering the identification of state-space models in combination with state estimators, see, for example, [16] in case of parametric uncertainties, and [17] addressing structural uncertainties. In addition, behavioral approaches [18], as briefly mentioned in Section II-A, handle output measurements naturally.

A. Nominal Models

Data-driven models, for which model uncertainty estimates are difficult to obtain or to integrate into the controller design explicitly, are often used as nominal models without a rigorous error quantification. Such models can nevertheless provide good predictive performance and are commonly employed in practical applications. Noteworthy examples include parametric models in the form of (recurrent) artificial neural networks in combination with inverse control [19] and MPC [20]. Nonparametric linear system descriptions based on behavioral system analysis gained significant traction in recent years for nominal data-driven trajectory predictions [18], including recent extensions toward robust [21], [22] and probabilistic [23] controller design.

B. Robust Models

Robust models are based on a bounded disturbance set assumption, that is, $w(k) \in \mathcal{W}$ with $\mathcal{W}$ bounded, and provide uncertainty estimates that include the true system dynamics (4) with probability one. To obtain a robust model (6), we distinguish between the case that f_l is known, leading to parametric set-membership estimation [24] of a possibly small set $\Theta \subseteq \mathbb{R}^{n_\theta}$ containing plausible system parameters θ, and nonparametric techniques [25] that directly estimate a set of functions $\mathcal{F}$ containing (4).

In the parametric uncertainty case, the goal is to compute possible parameter values θ that are consistent with the available system trajectories (5) according to

$$\Omega_k = \left\{ \theta \middle| \begin{array}{c} \forall j = 0, .., k-1, \ \exists w(j) \in \mathcal{W} \text{ s.t.:} \\ x(j+1) = f(x(j), u(j), j, w(j), \theta). \end{array} \right\}. \quad (7)$$

A corresponding nominal parameter value θ_n, which is often used in MPC for cost predictions, can then be selected by projecting a point estimate of θ_{true}, e.g., resulting from a least mean squares filter, onto the set Ω_k [26]. The main computational challenge related to (7) is the growing complexity of the set Ω with increasing k, which can be limited using approximations, see, for example, [27]. Most commonly available techniques consider linear [26], [28], [29] or nonlinear systems [30], [31] that are linear in parameters; that is, models of the form

$$f_l(x, u, k, w, \theta) = \Phi(x, u)\theta + w, \quad (8)$$

with $\Phi(x, u) = [x^\top, u^\top]$ in the case of linear systems.

In the presence of structural model uncertainty, nonparametric approaches allow for building the system model (6) directly based on available data (5). Thereby the idea is to exploit continuity properties of f_{true} w.r.t. x, u, and k such as $\|\nabla f_{\text{true}}\| \leq \epsilon$ to infer a feasible set of system realizations

$$\mathcal{F}_k = \left\{ f \middle| \begin{array}{c} \|\nabla f\| \leq \epsilon, \forall j = 0, .., k-1, \ \exists w(j) \in \mathcal{W} \text{ s.t.:} \\ x(j+1) = f(x(j), u(j), j, w(j), \theta). \end{array} \right\},$$

which contains f_{true} with probability one. Methods for obtaining strict uncertainty bounds on possible functions belonging to $\mathcal{F}_k$ include Lipschitz interpolation [32] or kinky inference [33]. A nominal function estimate f_n can then be selected to maximize the distance to the boundary of $\mathcal{F}_k$ subject to additional smoothness properties [34], [35].

C. Probabilistic Models

Probabilistic models rely on distributional information to mitigate the potential conservativeness of worst-case uncertainty bounds considered in the previous section. While being less conservative conceptually, most methods require simplifying assumptions such as normally distributed disturbances and prior distributions on parameters [6]. Similarly, as in the robust case, we distinguish between parametric and nonparametric approaches in the following. In the parametric case, we follow a Bayesian approach [36], [37], allowing us to incorporate prior knowledge about the parameter θ_{true} and disturbance $w(k)$ in the form of probability distributions $p(\theta_{\text{true}})$ and $p(w(k))$. Using the prior information, the posterior distribution of $p(\theta_{\text{true}}|X, U)$ given system trajectories (5) reads

$$p(\theta_{\text{true}}|X, U) = \frac{p(X|U, \theta_{\text{true}})p(\theta_{\text{true}})}{\int p(X|U, \tilde{\theta}_{\text{true}})p(\tilde{\theta}_{\text{true}})\mathrm{d}\tilde{\theta}_{\text{true}}}, \quad (9)$$

with likelihood $p(X|U, \theta_{\text{true}})$ depending on the noise distribution $p(w(k))$. While the computation of the posterior distribution (9) is generally difficult and often requires numerical approximations, see, for example, [38], a closed-form solution can be obtained for systems that are linear in their parameters (8) with Gaussian prior parameter distribution $p(\theta) = \mathcal{N}(\mu^\theta, \Sigma^\theta)$ and Gaussian noise $p(w(k)) = \mathcal{N}(\mu^w(k), \Sigma^w(k))$ [36], [37]. MPC formulations that exploit such models include [39]–[41].

Nonparametric probabilistic learning models are typically based on Gaussian process (GP) regression [36], [37], [42] due to their flexible yet computationally tractable properties. Assuming i.i.d. Gaussian noise $p(w(k)) = \mathcal{N}(0, \Sigma^w(k))$ and system dynamics of the form $x(k+1) = f_{\text{true}}(x(k), u(k)) + w(k)$, a GP regression model builds on the assumption that values of f_{true} for different arguments x, u are jointly Gaussian distributed according to a kernel function κ, expressing the covariance between function values. Based on the available system data (5), the joint distribution of observed data

points $X^+ = [x(1), .., x(N)]$ and function values at a test point x, u is

$$\begin{bmatrix} X^+ \\ f_{\text{true}}(x,u) \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K + I\sigma_w^2 & K_{x,u} \\ K_{x,u}^\top & \kappa([x,u],[x,u]) \end{bmatrix}\right),$$

with Gram matrix K based on data X, U in (5) using the kernel κ, where $K_{x,u}$ denotes the corresponding entries for the test point x, u. By conditioning the joint Gaussian distribution on the observations X^+ at X, U, we obtain a closed-form solution for the Gaussian posterior distribution $p(f_t(x,u)|X^+)$. Examples of MPC formulations based on GP regression include [14], [43]–[46].

It is important to note that closed-form solutions for prediction models presented in this section only exist for one-step-ahead predictions in the case of Gaussian distributions. One practical approach is to iteratively employ one-step predictions [47] to predict entire trajectories efficiently. However, as shown in [48]–[50], this can lead to severe prediction errors since it neglects occurring correlations across multiple time-steps. Tractable alternatives include sampling and linearization-based approximations [49], as well as memory [50] and multiple-step predictions [51].

D. Recent Developments and Future Research Directions

Motivated by the recent results in the field of data-driven behavioral system theoretic methods, a promising future direction is the consideration of input-output models by extending these techniques towards addressing process and measurement noise and nonlinear system classes, see, e.g., [18, Section 5.3.6]. Aside from behavioral approaches, robust and probabilistic system models primarily focus on noise-free system measurements (5). While this simplified setting is an essential first step, real-world applications commonly suffer from sensor noise and significant nonlinearities. Promising developments towards this practically relevant setting are outlined in [52], with critical open questions including a principled selection of prior knowledge and an automatic validation of probabilistic modeling assumptions. While the methods outlined in this section of the paper can be data efficient, they offer limited generalization properties for varying operating conditions. To this end, so-called meta-learning approaches have emerged, where the idea is to perform a pre-training step using data from different tasks, which enables fast adaptation to similar tasks and different conditions of the system environment [53]–[55].

III. Learning control policy parameters

As described in Section I, the CL-SMPC problem (2) is intractable in its general form and, thus, the ideal control law (3) cannot be directly implemented. Thus, a variety of approximations have been proposed to improve the tractability of (2), leading to various robust and stochastic MPC methods. Generally, these methods introduce additional "tuning" parameters $\gamma \in \Gamma \subset \mathbb{R}^{n_\gamma}$ into the controller. These tuning parameters differ from the internal optimization variables Y_t (i.e., decision variables) that are adapted at each time step via the solution to some optimization $\min_{Y_t \in \mathcal{Y}} L(Y_t; x_t, t, \gamma)$ for some overall cost function L and constraint set $\mathcal{Y}$. We denote the approximate control laws as

$$\hat{\pi}(x_t, t; \gamma) = g(Y_t^\star(x_t, t, \gamma)), \tag{10}$$

where $g(\cdot)$ is some function that maps the optimal value of the internal optimization variables $Y_t^\star(x_t, t, \gamma)$ to the current control input action. The control law (10) can represent a large class of nominal, robust, and stochastic MPC methods, common examples of which are summarized below.

Certainty-equivalence MPC (CE-MPC):

- Internal variables: Y_t represents the collection of the predicted nominal state and control input trajectories over the prediction horizon.
- Tuning parameters: γ could represent any set of parameters that appear in the cost function (e.g., weights in the stage or terminal costs), the prediction model (e.g., A and B matrices in a linear model $x_{k+1|t} = Ax_{k|t} + Bu_{k|t}$), or the constraints functions (e.g., back-off parameters) [56].

Tube-based MPC [5], [57], [58]:

- Internal variables: Y_t represents the collection of the predicted central path of the states and control inputs over the prediction horizon in addition to any adaptable parameters that appear in the shape of the tube.
- Tuning parameters: In addition to the tuning parameters for CE-MPC, γ could include parameters related to the design of the predicted tube (e.g., the controller gain matrix K used to control the evolution of the disturbance away from the nominal path).

Scenario-based (or multi-stage) MPC [59], [60]:

- Internal variables: Y_t represents the collection of the predicted state and control input trajectories for every disturbance sequence in the scenario tree.
- Tuning parameters: In addition to the tuning parameters for CE-MPC, γ could include parameters related to the definition of the scenario tree such as the number and location of the considered disturbance scenarios.

Moment-based Stochastic MPC [61]–[63]:

- Internal variables: Y_t represents the predicted moments (usually the mean and covariance matrix) of the random state and control input over the prediction horizon. It is common to use a parametrized predicted feedback control policy of the form $\mu_{k|t}(x_{k|t}) = K_{k|t}x_{k|t} + u_{k|t}$; if $K_{k|t}$ is optimized online then it must be included in Y_t (otherwise it may be treated as fixed, or it may be fully or partially included in the choice of γ).
- Tuning parameters: In addition to the tuning parameters for CE-MPC, γ could include any adaptable parameters that appear in the predicted control policy.

An important question in any MPC design is how to select the tuning parameters γ in order to realize a desired closed-loop performance under system uncertainties [64]. In this section, we discuss the opportunities that Bayesian optimization holds for automated tuning (or *auto-tuning*) of MPC controllers using closed-loop performance data. We also

elucidate the connection between policy-based reinforcement learning and the auto-tuning problem. Although our discussion focuses on MPC, these approaches can be applied for the design and tuning of generic controller structures (including schemes that involve logic or hierarchical representations).

A. Closed-loop Performance Indicators

The tuning parameters γ generally have a significant influence on the closed-loop performance. As opposed to selecting γ heuristically, we can optimize them based on a given performance metric in an *automated* fashion. For the control law (10), we define the closed-loop system as

$$x_{t+1} = f_{\text{true}}(x_t, \hat{\pi}(x_t, t; \gamma), t, w_t, \theta), \tag{11}$$

where f_{true} represents the true system dynamics that may not be known to us (available only in the form of a simulator or experiment that can be queried) and w_t and θ are the true disturbances and parameters, respectively. We represent the complete set of uncertain variables as $\boldsymbol{\omega} = \{x_0, \theta, w_0, w_1, \ldots, w_{T-1}\}$; we treat $\boldsymbol{\omega} \sim P_{\boldsymbol{\omega}}$ as being generated from some underlying probability distribution $P_{\boldsymbol{\omega}}$ that, similarly to the dynamics, may not be known. By recursively applying the closed-loop dynamics (11) over some (finite) number of time steps T, we can define a single closed-loop trajectory fully in terms of the tuning parameters γ and the complete set of uncertain variables $\boldsymbol{\omega}$

$$\begin{aligned} z(\gamma, \boldsymbol{\omega}) = \{&\theta, x_0, \hat{\pi}(x_0, 0; \gamma), w_0, \ldots, \\ &x_{T-1}, \hat{\pi}(x_{T-1}, T-1; \gamma), w_{T-1}, x_T\}. \end{aligned} \tag{12}$$

Accordingly, we can define a closed-loop performance measure to be any arbitrary function of the closed-loop history, i.e., $c(z(\gamma, \boldsymbol{\omega}))$. Some common examples for the closed-loop performance (for a given set of uncertainty realizations) would be the average cost over time, the worst-case or average constraint violation, an indicator cost for achieving the desired outcome, or some combination of these different measures. Since we do not have control and/or knowledge of the uncertainties $\boldsymbol{\omega}$, we must modify our definition of the overall control objective

$$J(\gamma) = \mathbb{E}_{\boldsymbol{\omega}}\{c(z(\gamma, \boldsymbol{\omega}))\}. \tag{13}$$

Here, an expectation operator is applied to the closed-loop performance measure (13) so that our cost is defined with respect to the average level of performance. Note that this is a general representation, as we can always convert expectations to probabilities using the indicator function $\mathbf{1}_A(x)$, which is equal to 1 if $x \in A$ and 0 otherwise. It is important to analyze the characteristics of the cost function (13) to determine a good method for solving the optimization problem that minimizes the cost

$$\min_{\gamma \in \Gamma} \; J(\gamma) := \mathbb{E}_{\boldsymbol{\omega}}\{c(z(\gamma, \boldsymbol{\omega}))\}. \tag{14}$$

In particular, $J(\gamma)$ is: (i) not known in closed-form and, thus, its derivative information may not be available, (ii) defined in terms of an expectation that cannot generally be evaluated exactly, and (iii) expensive to evaluate (either from a computation or experimentation point-of-view) since one must perform a minimum of one complete closed-loop evaluation to estimate $J(\gamma)$. Based on these characteristics, we refer to (14) as a stochastic "black-box" optimization problem. Although a wide variety of derivative-free optimization (DFO) methods have been proposed for tackling such problems, many of them are not viable for expensive objective functions since they require a large number of function evaluations.

Surrogate-based DFO methods rely on a *surrogate model* to more effectively guide the search process at every iteration. These methods have been shown to be effective at finding the near-global solution when an accurate surrogate model can be constructed [65]. A key question is how to select the "right" type of surrogate model when little is known about $J(\gamma)$. Recent work has focused on Gaussian process (GP) models since they are *probabilistic* (i.e., they provide an estimate of the uncertainty in the prediction) and *nonparametric* (i.e., they do not assume the function belongs to a space that can be finitely parametrized such as the space of polynomial functions of finite degree) [42]. Given a set of function evaluations, we can induce a GP model for $J(\gamma)$ by assuming it must follow a GP prior, which we can update with the available data using Bayes' rule. This results in the Bayesian optimization (BO) framework [66], [67], which can be thought of as a family of methods that combine a probabilistic surrogate model of the objective function (such as a GP) with an expected utility (or acquisition) function to sequentially select the next evaluation point for γ. By properly designing the acquisition function, BO systematically tradeoffs between exploration (i.e., evaluating the objective in new regions that are most unknown) and exploitation (i.e., evaluating the objective near the known best points) of the search space. Next, we provide an overview of how the BO framework can be applied to auto-tuning.

B. Bayesian Optimization for Auto-Tuning

BO for auto-tuning relies on the ability to collect closed-loop performance data for particular tuning parameter values. Since the expectation in (14) cannot be evaluated exactly, we must use some estimate of $J(\gamma)$. When $\boldsymbol{\omega}$ is high-dimensional, Monte Carlo (MC) sampling can be applied

$$y = \frac{1}{K} \sum_{i=1}^{K} c(z(\gamma, \boldsymbol{\omega}^{(i)})), \tag{15}$$

where $\mathbf{W} = \{\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(K)}\}$ is a set of K independently and identically distributed (i.i.d.) samples of the random uncertainty vector $\boldsymbol{\omega}$. The estimator (15) is unbiased for any positive integer value K, i.e., $\mathbb{E}_{\mathbf{W}}\{y\} = J(\gamma)$. We assume that the noise in the measurements of closed-loop performance is modeled as

$$y = J(\gamma) + \epsilon, \tag{16}$$

where $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$ is a zero-mean normal random variable with some variance σ_ϵ^2 that may not be known. Assume that we have n noisy objective function evaluations from

(16), which we denote by $y_{1:n} = \{y_1, \ldots, y_n\}$ computed at corresponding tuning parameter values $\gamma_{1:n} = \{\gamma_1, \ldots, \gamma_n\}$. By placing a GP prior over the objective function $J(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot,\cdot))$, $y_{1:n}$ must be jointly Gaussian with any predicted $J(\gamma)$ at some test point γ

$$\begin{bmatrix} y_{1:n} \\ J(\gamma) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(\gamma_{1:n}) \\ m(\gamma) \end{bmatrix}, \begin{bmatrix} K_n & k(\gamma_{1:n}, \gamma) \\ k(\gamma, \gamma_{1:n}) & k(\gamma,\gamma) \end{bmatrix} \right), \tag{17}$$

where $K_n = k(\gamma_{1:n}, \gamma_{1:n}) + I_n \sigma_\epsilon^2$ denotes the covariance matrix evaluated at the input training data points; the functions $m(\cdot)$ and $k(\cdot,\cdot)$ are overloaded to include element-wise operations across their inputs. By conditioning on the data $\{y_{1:n}, \gamma_{1:n}\}$, the posterior distribution of the predicted $J(\gamma)$ can be determined analytically as

$$J(\gamma) | y_{1:n}, \gamma_{1:n} \sim \mathcal{N}(\mu_n(\gamma), \sigma_n^2(\gamma)), \tag{18}$$

where

$$\mu_n(\gamma) = m(\gamma) + k(\gamma, \gamma_{1:n}) K_n^{-1} (y_{1:n} - m(\gamma_{1:n})), \tag{19a}$$
$$\sigma_n^2(\gamma) = k(\gamma,\gamma) - k(\gamma, \gamma_{1:n}) K_n^{-1} k(\gamma_{1:n}, \gamma). \tag{19b}$$

The mean function and covariance function are both typically parametrized by a set of unknown hyperparameters that must be calibrated to the available data $\{\gamma_{1:n}, y_{1:n}\}$ (along with the noise variance σ_ϵ^2) using maximum likelihood or maximum *a posteriori* estimation.

Given the predicted posterior distribution in (18), BO uses an *acquisition function* $\alpha_n : \Gamma \to \mathbb{R}$ that maps γ to a real number that quantifies the expected utility of spending a part of the exploration budget to evaluate $J(\gamma)$. The BO framework then repeats this process sequentially to decide where to perform the next evaluation, as summarized in Algorithm 1. Many different acquisition functions have been proposed, with the majority of them being roughly sortable into one of three categories: (i) improvement-based, (ii) optimistic, and (iii) information-based.

Algorithm 1 The Bayesian optimization (BO) framework

1: **Initialize:** Input space Γ; GP prior $m(\cdot)$ and $k(\cdot,\cdot)$; and maximum number of iterations N_b.
2: **for** $n = 0$ to $N_b - 1$ **do**
3: Construct GP surrogate model for $J(\gamma)$ given available data $\{y_{1:n}, \gamma_{1:n}\}$ via (18).
4: Maximize the acquisition function to find $\gamma_{n+1} = \operatorname{argmax}_{\gamma \in \Gamma} \alpha_n(\gamma)$.
5: Perform expensive closed-loop simulations to evaluate objective $y_{n+1} = J(\gamma_{n+1}) + \epsilon_{n+1}$ using (16).

The choice of α_n can have a significant effect on the performance of Algorithm 1 and, to the best of our knowledge, there is no systematic procedure for *a priori* selecting the "best" acquisition function. There have been several recent applications of BO to controller auto-tuning that have shown significant improvement over a variety of different alternatives (see, e.g., [68]–[71]). These works mainly use improvement-based acquisition functions, in particular *expected improvement*. Note that it is common practice in the BO literature for a practitioner to pick their favorite acquisition and keeps it fixed throughout the entire optimization process. An interesting direction for future work is to adaptively sample different acquisition functions at each iteration. One simple approach for doing this is described in [72] in which the performance of the randomly selected α_n is measured at each iteration and its probability of being selected in the future is increased if it improved the objective.

C. Important Extensions of BO-based Auto-Tuning

There are many unique properties that arise in controller auto-tuning that can be further exploited within the BO framework. We briefly outline some of these ideas below.

1) Input-dependent noise: The simplified representation of the noise model in (16) is only valid in the context of large K in (15) in accordance with the central limit theorem

$$y - J(\gamma) \quad \Rightarrow \quad \mathcal{N}\left(0, \frac{\sigma_J^2(\gamma)}{\sqrt{K}}\right), \tag{20}$$

where $\sigma_J^2(\gamma)$ denotes the variance of the objective function estimator (15) for any fixed γ and $\Rightarrow$ denotes convergence in distribution. Therefore, we can only ensure the additive normal assumption is satisfied for large K, which is typically not possible due to the expensiveness of closed-loop experiments. Due to the flexibility in the GP representation of the objective, we can incorporate more complicated noise models (usually at the cost of more expensive training and/or prediction steps). One such example are input-dependent (also known as heteroscedastic) noise models. Although multiple generalized GP models have been proposed in the literature, such as most likely GPs [73] and variational heteroscedastic GP [74], there has been limited discussion on their application within the BO framework.

2) Robust auto-tuning: One strategy for ensuring that the tuning parameters are robust with respect to the uncertainties is to formulate a quantile-based objective function; however, the MC estimate (15) is likely to have very high variance for small K due to the non-smooth indicator function. Since increasing K greatly increases the cost of a single function evaluation, we look to keep K as small as possible in practice. One approach that has recently been proposed for solving this issue is adversarially robust BO (ARBO) [75], which tackles a minimax problem of the form

$$\min_{\gamma \in \Gamma} \max_{\boldsymbol{\omega} \in \boldsymbol{\Omega}} \; c(z(\gamma, \boldsymbol{\omega})), \tag{21}$$

where $\boldsymbol{\Omega}$ is a compact set of possible uncertainty realizations. Notice that the main difference between (14) and (21) is that the expectation operator $\mathbb{E}_{\boldsymbol{\omega}}\{\cdot\}$ is replaced with a maximization operator $\max_{\boldsymbol{\omega} \in \boldsymbol{\Omega}}\{\cdot\}$. The solution to (21) ensures a stronger degree of robustness to the uncertainties while the maximization problem can be easier to handle in certain situations, especially when the distribution of $\boldsymbol{\omega}$ is unavailable. When the dimensionality of $\boldsymbol{\omega}$ is relatively small and we have control over the uncertainty sampling locations (such as when we have access to a high-fidelity closed-loop

simulator), we can *simultaneously* sample $\{\gamma, \boldsymbol{\omega}\}$ together, i.e., only one full closed-loop simulation is required at every iteration. Alternatively, when dealing with a large number of uncertainties, the max can be defined with respect to the most sensitive uncertainties while the others can be treated as the noise term in (16), as in traditional BO.

3) Multi-fidelity evaluations: The quality of the identified solution from BO is tied to the overall budget available N_b. When N_b is small, we can only obtain a limited amount of information about the objective function, which can severely limit our sequential selection of points. An alternative way to think about the auto-tuning problem is that we have access to a family of information sources denoted by $J(\gamma, s)$, where s is a collection of "fidelity" parameters that controls the accuracy of the objective. In the context of MPC auto-tuning, s could be related to many different quantities, including sampling time, tolerance of the numerical optimization routine, and accuracy of the model simulator. Increasing s corresponds to increasing the accuracy of the estimate of the objective; however, this increase in accuracy requires more computational cost. Given a fixed computational budget, we can adaptively select s and γ simultaneously to search for the minimum value of the true objective. Multi-fidelity BO has been considered in [76] and has been applied to MPC tuning in [77]. These multi-fidelity methods tend to outperform single fidelity BO, especially for problems with limited budget. However, the quality of the search process depends heavily on the selected low-fidelity representations of the objective function (both their cost and accuracy). Further research is needed to understand the impact of the quality of the multi-fidelity representation on performance in MPC auto-tuning.

4) Constraint handling and multi-objective problems: The majority of work on auto-tuning via BO considers a single objective. However, this is rarely the most natural representation in practice, as one must simultaneously consider multiple performance indicators (e.g., setpoint tracking error, constraint violation, and economic costs). The most straightforward way to deal with this problem is to lump all effects into a single objective such that $J(\gamma)$ is a weighted combination of many components. However, it can be difficult to select such weighted combinations since the different performance indicators are not on the same basis. One way to address this issue is to directly formulate the auto-tuning problem as a constrained BO problem [71], [78]. The main idea in these methods is to model the objective and each constraint with a separate (independent) GP model, all of which can be exploited in the search process. These approaches are best-suited for problems that naturally break down into a clear single objective with necessary constraints. However, this can be more generally interpreted as a multi-objective optimization (MOO) problem. The main goal of MOO is to systematically study the tradeoff between different objective functions of interest through the construction of an optimal Pareto front. BO methods for MOO have been investigated in the context of MPC auto-tuning, as discussed in [79].

D. Connecting Auto-tuning to Reinforcement Learning

The idea of using closed-loop performance data also constitutes the basis of many methods developed within the field of reinforcement learning (RL) [80]. As such, we can view the aforementioned BO-based auto-tuning strategies as a special case of the large family of policy-based RL methods. Policy gradient is one of the most popular RL methods for direct policy search with continuous control input spaces [81]. The main idea behind policy gradient algorithms is to adjust the control policy parameters γ in the negative direction of the cost gradient $\nabla_\gamma J(\gamma)$. Whenever we implement a randomized control policy $\pi_\gamma(u_t|x_t)$, which denotes the conditional probability of taking control input u_t given the current state x_t and fixed policy parameters γ, we can take advantage of the *policy gradient theorem* to obtain an estimate of this gradient [82]

$$\begin{aligned}\nabla_\gamma J(\gamma) &= \int c(z)\nabla_\gamma p_\gamma(z)dz, \\ &= \int c(z)\left[\frac{\nabla_\gamma p_\gamma(z)}{p_\gamma(z)}\right]p_\gamma(z)dz, \\ &= \mathbb{E}_z\left\{c(z)\nabla_\gamma \log p_\gamma(z)\right\},\end{aligned} \tag{22}$$

where z denotes the random closed-loop trajectory or path (12), whose distribution can be derived using the Markovian nature of the system

$$p_\gamma(z) = p(x_0)\prod_{t=1}^{T} p(x_{t+1}|x_t, u_t)\pi_\gamma(u_t|x_t). \tag{23}$$

In (22), unbiased estimates of the gradient of the cost J can be obtained by sampling from z, which can be done in a straightforward fashion for fixed γ. To see this, note that $\nabla_\gamma \log p_\gamma(z)$ can be simplified using (23)

$$\nabla_\gamma \log p_\gamma(z) = \sum_{t=1}^{T} \nabla_\gamma \log \pi_\gamma(u_t|x_t), \tag{24}$$

implying stochastic gradient descent optimization methods [83] can be readily applied as long as we can evaluate the cost and the gradient of the policy for different randomly drawn trajectories. Although the technical results above only hold for stochastic control policies, it has been shown that a similar idea holds for deterministic control policies such as MPC [81]. In particular, we can convert a deterministic policy to a stochastic one by adding a small amount of noise, e.g., $\hat{pi}(x_t;\gamma) + n$ where $n \sim \mathcal{N}(0, \sigma^2)$ with $\sigma^2 \to 0$. MPC-based control policies, under suitable regularity conditions, are almost everywhere differentiable with respect to some tuning parameters and, thus, policy gradient RL methods can be utilized for offline or online auto-tuning. See, e.g., [84]–[86] for some recent works in this direction.

Although policy gradient methods are expected to be more scalable than BO methods since they take advantage of gradient information, they often require many more samples to converge, especially when started from a poor initial guess. BO methods, on the other hand, are capable of globally searching the policy parameter space and can directly enforce constraints so that they can provide strong safety guarantees in an online setting. The latter is less straightforward when using policy gradient methods. Since BO and policy gradient

have complementary strengths and weaknesses, there are many opportunities to hybridize these approaches to achieve significant boosts in online safety and performance.

IV. Learning efficient approximations of iterative online optimization

The solution of the MPC problem, both the nominal MPC formulation (1) or a tractable reformulation of the CL-SMPC (2), is commonly computed using iterative optimization algorithms that search for a point that satisfies the first-order necessary conditions of optimality. Most iterative optimization algorithms for MPC can be divided between especially tailored for linear MPC and more general algorithms that can be used for nonlinear MPC.

For linear MPC, the resulting optimization problem is typically a convex quadratic program (QP), provided that the cost is chosen as a quadratic function with a positive definite Hessian and the model and constraints are linear. To this end, many QP solvers have been developed. The most popular choices are the use of active set solvers [87], [88] and interior point algorithms [89], amongst others [90]. In addition, for very fast optimization, especially under resource limitations, it can be advantageous to use first-order optimization methods that do not need to compute nor store the Hessian of the problem. A prominent algorithm is the fast gradient method [91], which was used for embedded MPC in [92]. To deal with constraints, fast gradient methods can be extended with an outer loop in an augmented Lagrangian schemes [93] [94], the alternating direction method of multipliers [95] and its variations such as the operator splitting method [96]. In addition, it is possible to significantly improve the performance of the linear algebra used in these algorithms if it is optimized at a very low level for the specific hardware, as done, e.g., in [97]. In the nonlinear case, many algorithms use the aforementioned ideas within sequential quadratic programming [98] or nonlinear interior point methods [99] especially tailored for embedded optimization, [100], [101]. An overview of algorithmic details can be found in [102].

An alternative approach to extending the application of MPC to fast and embedded systems is usually called explicit MPC. Explicit MPC takes advantage of the fact that the MPC problem, both in linear and nonlinear MPC, is a parametric optimization problem. This means that the optimal control input that is applied to the system is only a function of some parameters; in the case of the nominal MPC problem (1), this is the initial condition x_t. For a linear MPC problem with quadratic cost function, the solution of the parametric optimization problem is a piecewise affine function of the current state of the system, as shown in [103]. Explicit MPC precomputes the piecewise affine function that completely defines the MPC law, which can be written as

$$\mathcal{K}(x_{\text{init}}) = \begin{cases} K_1 x_{\text{init}} + g_1 & \text{if} \quad x_{\text{init}} \in \mathcal{R}_1, \\ \vdots & \\ K_{n_\text{r}} x_{\text{init}} + g_{n_\text{r}} & \text{if} \quad x_{\text{init}} \in \mathcal{R}_{n_\text{r}}, \end{cases} \tag{25}$$

with n_r regions, $K_i \in \mathbb{R}^{Nn_u \times n_x}$ and $g_i \in \mathbb{R}^{Nn_u}$. Each region $\mathcal{R}_i$ is described by a polyhedron

$$\mathcal{R}_i = \{x \in \mathbb{R}^{n_x} \mid Z_i x \leq z_i\} \quad \forall i = 1, \ldots, n_r, \tag{26}$$

where $Z_i \in \mathbb{R}^{c_i \times n_x}$, $z_i \in \mathbb{R}^{c_i}$ describes the c_i halfspaces $a_{i,j} x_{\text{init}} \leq b_{i,j}$ of the i-th region with $j = 1, \ldots, c_i$, $a_{i,j} \in \mathbb{R}^{1 \times n_x}$ and $b_{i,j} \in \mathbb{R}$. Then, instead of solving the optimization problem online, one divides the task in two steps. In the first step, the piecewise affine function (25) is computed offline and stored. In the second step, a point location problem is solved to find the region $\mathcal{R}_i$ in which the current state x_{init} lies. The same idea can also be used for certain robust MPC formulations [104].

The main drawback of explicit MPC is that the number of regions on which the control law is defined grows exponentially with the prediction horizon and the number of constraints. This can complicate the first step as it might not be possible to compute or store all the regions on which the MPC law is defined. In addition, this could make the point location problem too slow for embedded applications. There has been a significant research effort to alleviate such problems through the elimination of redundant regions [105], or using a smaller number of regions to describe the MPC law [106]. Yet, explicit MPC is typically used for small systems and short prediction horizons, especially in embedded applications with limited storage capabilities and computational power. These limitations become even more evident when system uncertainty must be considered in any tractable reformulation of the CL-SMPC problem (2). In such a case, it is generally necessary to resort to approximate solutions of the optimization, or the explicit MPC problem. While some approximate explicit MPC schemes have been proposed, including, e.g., the use of simplicial partitions [107], radial basis functions [108], machine learning and in particular deep neural networks have recently emerged as a powerful method for the deployment of complex stochastic and robust MPC algorithms on resource-limited hardware.

A. Approximate MPC via Machine Learning

Leveraging the fact that the MPC problem is a parametric program, it seems reasonable to use function approximation tools, as those used in modern machine learning, to efficiently approximate the implicit control law that results from the MPC problem. The idea is very simple: if, given an initial state x_{init}, it is possible to solve the MPC problem and obtain the optimal input u_0^*, will it be possible to generate many of these solutions offline in order to obtain an accurate approximation of the solution using standard supervised learning techniques?

This notion was already proposed in 1995 for nonlinear MPC by [109]. Recently, there has been a renewed interest in pursuing this idea. The main difference in recent works is the use of *deep* neural networks, instead of *shallow* neural networks. Recent successes of the machine learning community, especially in image recognition, as well as advances in the theoretical description of the representation capabilities of deep neural networks [110], [111], have motivated the use

of deep neural networks to perform efficient approximation of very complex functions, as those defined by the MPC problem.

A standard feedforward neural network is defined as a sequence of layers of neurons, which determines a function $\mathcal{N}: \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$ of the form

$$\mathcal{N}(x; \theta, M, L) = f_{L+1} \circ g_L \circ f_L \circ \cdots \circ g_1 \circ f_1(x), \quad (27)$$

where the input to the network is $x \in \mathbb{R}^{n_x}$ and the output of the network is $u \in \mathbb{R}^{n_u}$. We denote the input with x and the output with u because the network will be used to approximate the solution of the MPC problem. Thus, a network computes, for each input (i.e., current state of the system), a corresponding output of the network (i.e., approximate solution of the MPC problem). M is the number of neurons in each hidden layer and L is the number of hidden layers. If $L \geq 2$, $\mathcal{N}$ is described as a *deep* neural network and, if $L = 1$, as a *shallow* neural network. Each hidden layer consists of an affine function

$$f_l(\xi_{l-1}) = W_l \xi_{l-1} + b_l, \quad (28)$$

where $\xi_{l-1} \in \mathbb{R}^M$ is the output of the previous layer with $\xi_0 = x$. The second element of the neural network is a nonlinear activation function g_l. Common activation functions include the hyperbolic tangent ($\tanh$) or the rectifier linear units (ReLU), which computes the element-wise maximum between zero and the affine function of the current layer l

$$g_l(f_l) = \max(0, f_l). \quad (29)$$

The parameter $\theta = \{\theta_1, \ldots, \theta_{L+1}\}$ contains all the weights and biases of the affine functions of each layer

$$\theta_l = \{W_l, b_l\} \quad \forall l = 1, \ldots, L+1, \quad (30)$$

where the weights are

$$W_l \in \begin{cases} \mathbb{R}^{M \times n_x} & \text{if} \quad l = 1, \\ \mathbb{R}^{M \times M} & \text{if} \quad l = 2, \ldots, L, \\ \mathbb{R}^{n_u \times M} & \text{if} \quad l = L+1, \end{cases} \quad (31)$$

and the biases are

$$b_l \in \begin{cases} \mathbb{R}^{M} & \text{if} \quad l = 1, \ldots, L, \\ \mathbb{R}^{n_u} & \text{if} \quad l = L+1. \end{cases} \quad (32)$$

When a feedforward neural network only includes ReLU activation functions, which are piecewise affine, the network (27) describes a piecewise affine function and, thus, can describe the exact solution of the MPC problem in the linear case (25). The number of linear regions that can be described by a neural network with ReLU activation functions grows exponentially with the number of layers of the network (the depth of the network) [112]. The improved accuracy of neural networks with several layers, when compared to traditional one-layer networks with the same number of weights, has been shown, e.g., in [113].

To approximate the MPC law using neural networks, one needs to perform three main steps. The first one is data generation. In order to *train* a neural network, it is necessary to obtain pairs of an initial condition x_{init} and the corresponding optimal control action u_0^*, yielding a training set $\mathcal{D} = \{(x_{\text{init},1}, u_{0,1}^*), (x_{\text{init},2}, u_{0,2}^*), \ldots, (x_{\text{init},n_{\text{tr}}}, u_{0,n_{\text{tr}}}^*)\}$ with a total of n_{tr} data points. For high-dimensional systems, good strategies for sampling the state-space are very important to achieve a good accuracy with a reasonable number of data points. While one can generate new points by repeatedly solving the MPC problem for different initial conditions, obtaining a high approximation accuracy in high dimensions may require a prohibitively large number of data points. Random sampling of the state-space, or using a uniform grid, are usually effective in low-dimensional state-spaces. For more efficient sampling, one can use quasi-random sampling strategies [114], as well as sampling around closed-loop simulations of the system controlled by the MPC [115]–[117] to allow better approximation of the neighborhood of optimal trajectories, leading to reduced approximation errors in regions of interest. An alternative useful strategy includes the generation of additional data points based on sensitivities of the optimization problem [118].

The second step is the training of the neural network using the data generated in the first step. The training consists in finding the weights θ that minimize the approximation error. Typically, this is achieved by minimizing the mean squared error over the training points, that is,

$$\min_{\theta} \ \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} ||\mathcal{N}(x_{\text{init},i}; \theta, M, L) - u_{0,i}^*||^2. \quad (33)$$

The unconstrained minimization problem (33) is usually solved using stochastic gradient descent, or its variations. There are several tools available for simple and efficient formulation and training of neural networks, such as Tensorflow [119] or Pytorch [120].

The third step is the deployment of the approximate MPC law on suitable hardware. The choice of hardware architecture often involves a trade-off between cost, energy consumption, and required performance. A key advantage of the above-discussed approximations is that they are straightforward to deploy on many different platforms since their implementation only relies on matrix-vector multiplications and a simple nonlinear operation (i.e., the activation function). This facilitates application of fast MPC strategies on different embedded devices, including low-cost microcontrollers [113], [121] and field programmable gate arrays (FPGAs) [115], [122], [123].

B. Closed-loop Guarantees and Verification Strategies

Approximating the solution of an iterative online optimization introduces an additional approximation error into the closed-loop, beyond model uncertainty and disturbances considered in MPC of uncertain systems. The approximation error leads to asymptotic performance losses. Neural network-based approximate MPC strategies have been proposed that can realize offset-free tracking, despite the approximation error, via online solution of a small-scale target tracking optimization problem [123], using an online adaptive correction scheme [116], or adapting the weights of

the last layer [124]. Furthermore, the guarantees on constraint satisfaction, stability or closed-loop performance that are typically established for MPC of uncertain systems no longer hold for neural network-based MPC laws defined by the optimal parameters (33). Nevertheless, it is possible to deal with the approximation error in different ways. For example, one can consider the approximation error as an additional uncertainty, with assumed bounds or distributions, in a robust or stochastic MPC formulation in the first place [125]. After training the approximate controller, the assumptions about the approximation error should be verified (e.g., if the actual maximum approximation error is smaller than initially assumed) for the guarantees to hold.

There are several approaches to verifying if the closed-loop guarantees are satisfied. One can use probabilistic verification methods [126], [127], as used in [125], [128], [129]. Similar ideas can be used to achieve guarantees about the suboptimality caused by the approximation error [130], or any other closed-loop performance indicator [121]. Alternatively, if deterministic guarantees are desired, it is possible to *correct* the approximate input given by the network using a projection onto a safe set [129], [131]–[133]. In the linear case, the neural network-based controller, as well as the closed-loop that includes the linear dynamic system and the controller, can be represented as a mixed-integer linear program. This idea, often referred to as output-range analysis [134], can be used to perform *a priori* verification of the safety and stability of the closed-loop [124], or to compute rigorous approximation error upper bounds [135].

C. Future Research Directions

While the online computation time required for approximate controllers is very low, the amount of data needed to train a controller with small approximation error grows rapidly with the system dimension. Recent works show that the limitations are likely to be less severe than for explicit MPC [114]. Still, millions of *valid* data points might be necessary for systems of dimension larger than 20 states. Further research is needed to extend even more the possibilities of the approach, for example, by combining with efficient sampling strategies, or efficient solvers that can make use of very good initial guesses provided by the approximate controllers.

An additional open challenge is how to achieve efficient approximate controllers that can readily adapt to changing situations. If large amounts of data are necessary to train a good controller, one might want to avoid the regeneration and retraining of the controller once some changes occur in the system, or in the expected disturbances that affect the system. Combinations of ideas from the field of reinforcement learning, as proposed in [136], may be a promising future area of research.

V. Learning Optimal Cost-to-go Representations: How can Reinforcement Learning Help?

Reinforcement learning (RL) is a semi-supervised learning method in which an agent tries to learn the best way to accomplish a task through trial and error. Sutton and Barto [137] provide an excellent introduction to RL. Two independent research threads contributed to the development of RL: the psychology of animal learning and the mathematics of optimal control [137]. From animal psychology came insights into how animals learn to solve problems, and from optimal control came a test to determine when the best possible solution has been reached. From the control point-of-view, RL can be viewed as a solution approach to (stochastic) optimal control problems when the true system dynamics are unknown [80].

There are two main classes of RL methods: model-based and model-free. In model-based RL, previously observed data are used to learn a dynamic model, which can then be used to approximate the solution of the true underlying stochastic optimal control problem. This leads to strong parallels with the model learning discussion in Section II, as well as the use of receding- or rolling-horizon methods for deploying the controller (which are closely related to the notion of MPC). Model-free RL methods, on the other hand, are able to eliminate the model building step and, instead, learn a mapping from measurements (or observations) to control inputs (or actions). Model-free RL methods are commonly divided into two major categories: *policy search* and *approximate dynamic programming* (ADP). In Section III-D, we provided a brief overview of policy search methods, which look to directly optimize control policy parameters using data from previous closed-loop episodes. Although powerful, these methods may suffer from some degree of (unknown) sub-optimality since we are optimizing over parametrized policies that may not match the complex structure of the true optimal policy. ADP methods look to overcome this issue by directly approximating the optimality conditions for the true stochastic optimal control problem, which can be derived using Bellman's principle of optimality [138]–[140]. Below, we provide an overview of ADP methods and discuss how they can be useful in the context of MPC.

Let us distinguish the true system from the model using the following notation

$$x_{t+1} = f_{\text{true}}(x_t, u_t, w_t), \quad t = 0, 1, 2, \dots, \tag{34}$$

where the random uncertainty follows some true probability distribution $P_{\text{true}}(\cdot|x_t, u_t)$. To simplify notation regarding constraints, we assume that control inputs can only take values in a given subset $\mathbb{U}_{\text{true}}(x_t) \subset \mathcal{U}$ that can generally depend on the current state x_t; this allows us to implicitly consider, e.g., conditional chance constraints. Similarly to Section III, a cost function is defined over T time steps. However, the cost is broken down into the sum of stage costs and a terminal cost. As such, we look to optimize the following problem

$$\begin{aligned} \min_{\mu_0,\dots,\mu_{T-1}} \quad & \mathbb{E}\left\{ J_{f,\text{true}}(x_T) + \sum_{t=0}^{T-1} \ell_{\text{true}}(x_t, u_t) \right\}, \\ \text{s.t.} \quad & x_{t+1} = f_{\text{true}}(x_t, u_t, w_t), \\ & u_t = \mu_t(x_t) \in \mathbb{U}_{\text{true}}(x_t). \end{aligned} \tag{35}$$

There are some important differences between (35) and the CL-SMPC problem (2). Although both optimize over control policies, (35) is meant to operate over the full time horizon of interest T that is typically much greater than the finite-horizon N considered in (2) for computational tractability. Furthermore, (35) is expressed in terms of the true dynamics, cost, and constraint functions, which are never known exactly, whereas (2) is posed in terms of more computationally tractable models. It is also important to note that the true system is assumed to behave as a Markov process, which is an important assumption that must be satisfied to apply the principle of optimality. A system can always be represented as a Markov process through appropriate transformations, though this might be difficult to do in practice (e.g., time-invariant parameter uncertainties in a system can be handled by treating them as additional states [141]). The fact that the Markov process assumption need not be satisfied by the policy search methods discussed in Section III-D is often an overlooked advantage of those methods.

Starting with $J_T^\star(x) = J_{f,\text{true}}(x)$, we can recursively define the optimal *cost-to-go* functions as

$$J_t^\star(x) = \min_{u \in \mathbb{U}_{\text{true}}(x)} \mathbb{E}\left\{\ell_{\text{true}}(x,u) + J_{t+1}^\star(f_{\text{true}}(x,u,w))\right\}, \tag{36}$$

for $t = T-1, \ldots, 0$, where $J_t^\star(x)$ is the optimal cost for the subproblem when starting at state x at time t and following an optimal control policy for all future steps $t, \ldots, T-1$. The optimal cost $J^\star(x_0)$ that solves (35) is then obtained as the final step of the recursion (36), i.e., $J^\star(x_0) = J_0^\star(x_0)$. In the RL literature, it is common to express the right-hand side of (36) in terms of the so-called *optimal Q-factors*

$$Q_t^\star(x,u) = \ell_{\text{true}}(x,u) + \mathbb{E}\left\{J_{t+1}^\star(f_{\text{true}}(x,u,w))\right\}, \tag{37}$$

which represents the optimal cost for the subproblem when starting at state x and taking control action u at time t and following an optimal control policy for all remaining steps $t+1, \ldots, T-1$. From the definitions in (36) and (37), we can derive a simple relationship between the cost-to-go functions and the Q-factors

$$J_t^\star(x) = \min_{u \in \mathbb{U}_{\text{true}}(x)} Q_t^\star(x,u). \tag{38}$$

The main advantage of the Q-factor is that the optimal policy can be trivially recovered at any point in time

$$\mu_t^\star(x_t) \in \operatorname*{argmin}_{u \in \mathbb{U}_{\text{true}}(x_t)} Q_t^\star(x_t,u), \tag{39}$$

which only depends on the current state value. This summarizes the main idea of dynamic programming: recursively define the cost-to-go functions (or equivalently the Q-factors) by passing backward in time and then using these functions to evaluate an optimal policy according to (39).

If x_k is a discrete, scalar variable, then it may be possible to enumerate all states $x_k \in X$ when solving (36). However, if X is continuous (as is typically the case in most MPC applications), we cannot solve (36) exactly. ADP methods generally look to compute the Q-factors from data by assuming that the Q-factor is stationary, i.e., $Q_t(x,u) = Q(x,u)$ for all t and for some static function Q. Such stationarity does indeed arise in the case that the time horizon is infinite. Unfortunately, we cannot directly take the limit $T \to \infty$ in (35) without introducing further technicalities, so it is common practice in RL to consider a *discounted* cost problem

$$\begin{aligned} \min_{\mu_0, \mu_1, \ldots} \quad & \mathbb{E}\left\{\sum_{t=0}^{\infty} \alpha^t \ell_{\text{true}}(x_t,u_t)\right\}, \\ \text{s.t.} \quad & x_{t+1} = f_{\text{true}}(x_t,u_t,w_t), \\ & u_t = \mu_t(x_t) \in \mathbb{U}_{\text{true}}(x_t), \end{aligned} \tag{40}$$

where $\alpha \in (0,1)$ is a scalar called the discount factor. For α close to 1, the discounted cost is approximately equal to the average reward $\lim_{T\to\infty} \mathbb{E}\left\{\frac{1}{T}\sum_{t=0}^{T} \ell_{\text{true}}(x_t,u_t)\right\}$. Let $Q_\alpha^\star(x,u)$ denote the Q-factor for (40), we can then derive an infinite-horizon discounted version of dynamic programming that results in the same Q-factor on both sides

$$Q_\alpha^\star(x,u) = \ell_{\text{true}}(x,u) + \alpha \mathbb{E}\left\{\min_{u'} Q_\alpha^\star(f_{\text{true}}(x,u,w),u')\right\}. \tag{41}$$

The optimal policy is now stationary and can be derived *for all times* using the following simple formula

$$\pi^\star(x_t) \in \operatorname*{argmin}_{u \in \mathbb{U}_{\text{true}}(x_t)} Q_\alpha^\star(x_t,u). \tag{42}$$

The simplicity of the true optimal control policy representation is what makes Q-learning methods attractive; we only need to learn a "good" approximation of $Q_\alpha^\star$ to derive a "good" policy. However, as one might expect, it is not easy to learn the Q-factor, especially when a good system model is not available. The main idea behind Q-learning methods is to use sample trajectories to update an initial guess for the Q-factor as follows

$$\begin{aligned} \tilde{Q}_\alpha^{(\text{new})}(x_t,u_t) = (1-\eta)\tilde{Q}_\alpha^{(\text{old})}(x_t,u_t) \\ + \eta\left[\ell_{\text{true}}(x_t,u_t) + \alpha \min_{u'} \tilde{Q}_\alpha^{(\text{old})}(x_{t+1},u')\right], \end{aligned} \tag{43}$$

where $\tilde{Q}_\alpha^{(\text{old})}$ is an initial guess for the Q-factor, $\tilde{Q}_\alpha^{(\text{new})}$ is an updated version of the Q-factor given a sample starting from state x_t with control input $u_t \in \operatorname{argmin}_u \tilde{Q}_\alpha^{(\text{old})}(x_t,u)$ that is selected optimally from the guessed Q-factor, $x_{t+1} = f(x_t,u_t,w_t)$ is a realization of the successor state, and η is a step size or learning rate. The update rule in (43) forms the basis of most Q-learning algorithms [142], [143]. ADP methods that rely on estimating the cost-to-go functions are also widely used in RL. In particular, temporal difference learning algorithms are based on bootstrapping from a current estimate of the cost-to-go function [144]–[146]. We note that a distinct feature of ADP, not discussed here, is its ability to systematically trade-off between system exploration and exploitation via the so-called *dual control* mechanism (see [13] and the references therein). This can make ADP especially useful for adaptive control of time-varying systems.

We can connect ADP back to the CL-SMPC (2) through the optimal cost-to-go function. In particular, we can see that the optimal policy (39) is equal to the CL-SMPC law (3) whenever $N \geq 1$, $\alpha = 1$, $J_f(x) = J_\alpha^\star(x)$, and the assumed model, stage cost, constraints, and uncertainty perfectly match their true descriptions. These are obviously unrealistic assumptions that cannot be satisfied in practice. However, it can still be useful to take advantage of approximate cost-to-go functions $J_f(x) \approx J_\alpha^\star(x)$ when solving (2). Interested readers are referred to [147] for a more detailed discussion on the impact of cost-to-go approximations on the stability and feasibility of controllers derived using MPC principles. Various ways in which MPC and ADP can be integrated are also discussed in [148].

A practical algorithm for ADP, which has become increasingly popular, is deep RL (DRL) [149]. In essence, DRL uses deep neural networks to parameterize the policy and value functions and, thus, is particularly suitable for dealing with continuous state and action spaces. Although the amount of data required to train DRL algorithms and the ability to handle constraints are generally important considerations, it has been demonstrated that combining aspects of MPC with RL can improve the data efficiency, constraint handling and performance of RL (e.g., [150]–[152]). As argued in [153], a major research direction is how to systematically take advantage of *offline* approximation in value space and rollout in combination with *online* decision-making using (learning-based) MPC. Such a paradigm has been central to major successes of RL, such as AlphaGO [110], and has the potential to take advantage of the strengths of each method in a synergistic fashion. Additional research is needed to better understand the theoretical properties of such a hybrid control strategy and to develop improved algorithms that allow faster training of policy and value function representations.

VI. Conclusions

This paper discussed the recent advances and provided perspectives for further research in four major directions in which machine learning can aid in the design of model predictive control (MPC) under uncertainty. These directions included data-driven adaptation of the prediction model and/or uncertainty description in MPC; learning parameters of MPC laws in an automated manner (i.e., auto-tuning); approximating MPC laws with explicit, cheap-to-evaluate surrogates that are especially useful for embedded control applications; and the synergy between reinforcement learning and MPC, which can be central to the design of more practical reinforcement learning algorithms and providing insights into the theoretical properties of MPC. We emphasize that this is a rapidly evolving field and, thus, giving an exhaustive account of the literature was beyond the scope of this paper.

References

[1] M. Morari and J. H. Lee, "Model predictive control: Past, present and future," *Computers & Chemical Engineering*, vol. 23, no. 4-5, pp. 667–682, 1999.

[2] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*. Nob Hill Publishing Madison, 2017, vol. 2.

[3] J. H. Lee, "Model predictive control: Review of the three decades of development," *International Journal of Control, Automation and Systems*, vol. 9, no. 3, pp. 415–424, 2011.

[4] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

[5] B. Kouvaritakis and M. Cannon, *Model Predictive Control: Classical, Robust and Stochastic*. Springer International Publishing Switzerland, 2016.

[6] A. Mesbah, "Stochastic model predictive control: An overview and perspectives for future research," *IEEE Control Systems*, vol. 36, no. 6, pp. 30–44, 2016.

[7] 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.

[8] E. C. Kerrigan, "Robust constraint satisfaction: Invariant sets and predictive control," Ph.D. dissertation, University of Cambridge, 2001.

[9] L. Ljung, "System identification," in *Signal analysis and prediction*. Springer, 1998, pp. 163–173.

[10] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *arXiv preprint arXiv:2108.06266*, 2021.

[11] H. Mania, M. I. Jordan, and B. Recht, "Active learning for nonlinear system identification with guarantees," *arXiv preprint arXiv:2006.10277*, 2020.

[12] Y. Zhu, *Multivariable system identification for process control*. Elsevier, 2001.

[13] 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.

[14] M. Maiworm, D. Limon, J. M. Manzano, and R. Findeisen, "Stability of Gaussian process learning based output feedback model predictive control," in *Proceedings of the IFAC Conference on Nonlinear Model Predictive Control*, 2018, pp. 455–461.

[15] M. Lauricella and L. Fagiano, "Set membership identification of linear systems with guaranteed simulation accuracy," *IEEE Transactions on Automatic Control*, vol. 65, pp. 5189–5204, 2020.

[16] T. B. Schön, A. Wills, and B. Ninness, "System identification of nonlinear state-space models," *Automatica*, vol. 47, no. 1, pp. 39–49, 2011.

[17] A. Doerr, C. Daniel, M. Schiegg, N.-T. Duy, S. Schaal, M. Toussaint, and T. Sebastian, "Probabilistic recurrent state-space models," in *Proceedings of Machine Learning Research*, 2018, pp. 1280–1289.

[18] I. Markovsky and F. Dörfler, "Behavioral systems theory in data-driven analysis, signal processing, and control," *Annual Reviews in Control*, vol. 52, pp. 42–64, 2021.

[19] S. Zhou, M. K. Helwa, and A. P. Schoellig, "Deep neural networks as add-on modules for enhancing robot performance in impromptu trajectory tracking," *International Journal of Robotics Research*, vol. 39, pp. 1397–1418, 2020.

[20] A. Draeger, S. Engell, and H. Ranke, "Model predictive control using neural networks," *IEEE Control Systems Magazine*, vol. 15, no. 5, pp. 61–66, 1995.

[21] J. Berberich, J. Köhler, M. A. Müller, and F. Allgöwer, "Data-driven model predictive control with stability and robustness guarantees," *IEEE Transactions on Automatic Control*, vol. 66, pp. 1702–1717, 2020.

[22] L. Xu, M. S. Turan, B. Guo, and G. Ferrari-Trecate, "A data-driven convex programming approach to worst-case robust tracking controller design," *arXiv preprint arXiv:2102.11918*, 2021.

[23] J. Coulson, J. Lygeros, and F. Dörfler, "Distributionally robust chance constrained data-enabled predictive control," *IEEE Transactions on Automatic Control*, 2021, In Press.

[24] M. Milanese and A. Vicino, "Optimal estimation theory for dynamic systems with set membership uncertainty: An overview," *Automatica*, vol. 27, no. 6, pp. 997–1009, 1991.

[25] M. Milanese and C. Novara, "Set membership identification of nonlinear systems," *Automatica*, vol. 40, no. 6, pp. 957–975, 2004.

[26] M. Lorenzen, M. Cannon, and F. Allgöwer, "Robust MPC with recursive model update," *Automatica*, vol. 103, pp. 461–471, 2019.

[27] S. M. Veres, H. Messaoud, and J. P. Norton, "Limited-complexity model-unfalsifying adaptive tracking-control," *International Journal of Control*, vol. 72, pp. 1417–1426, 1999.

[28] M. Tanaskovic, L. Fagiano, R. Smith, and M. Morari, "Adaptive receding horizon control for constrained MIMO systems," *Automatica*, vol. 50, no. 12, pp. 3019–3029, 2014.

[29] E. Terzi, L. Fagiano, M. Farina, and R. Scattolini, "Learning multi-step prediction models for receding horizon control," in *Proceedings of the European Control Conference*, 2018, pp. 1335–1340.

[30] V. Adetola, D. DeHaan, and M. Guay, "Adaptive model predictive control for constrained nonlinear systems," *Systems & Control Letters*, vol. 58, no. 5, pp. 320–326, 2009.

[31] G. A. A. Gonçalves and M. Guay, "Robust discrete-time set-based adaptive predictive control for nonlinear systems," *Journal of Process Control*, vol. 39, pp. 111–122, 2016.

[32] G. Beliakov, "Interpolation of Lipschitz functions," *Journal of Computational and Applied Mathematics*, vol. 196, pp. 20–44, 2006.

[33] J.-P. Calliess, "Conservative decision-making and inference in uncertain dynamical systems," Ph.D. dissertation, University of Oxford, 2014.

[34] J. M. Manzano, D. Limón, D. M. de la Peña, and J. Calliess, "Output feedback MPC based on smoothed projected kinky inference," *IET Control Theory & Applications*, vol. 13, pp. 795–805, 2019.

[35] E. T. Maddalena and C. N. Jones, "Learning non-parametric models with guarantees: A smooth Lipschitz regression approach," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 965–970, 2020.

[36] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[37] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[38] B. P. Carlin and T. A. Louis, *Bayesian methods for data analysis*. CRC Press, 2008.

[39] M. Lorenzen, F. Dabbene, R. Tempo, and F. Allgöwer, "Stochastic MPC with offline uncertainty sampling," *Automatica*, vol. 81, pp. 176–183, 2017.

[40] C. D. McKinnon and A. P. Schoellig, "Experience-based model selection to enable long-term, safe control for repetitive tasks under changing conditions," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2018, pp. 2977–2984.

[41] K. P. Wabersich, L. Hewing, A. Carron, and M. N. Zeilinger, "Probabilistic model predictive safety certification for learning-based control," *IEEE Transactions on Automatic Control*, vol. 67, no. 1, pp. 176–188, 2021.

[42] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. The MIT Press, 2006.

[43] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Advances in Neural Information Processing Systems*, 2017, pp. 908–918.

[44] L. Hewing, A. Liniger, and M. N. Zeilinger, "Cautious NMPC with Gaussian process dynamics for miniature race cars," *Proceedings of the European Control Conference*, pp. 1341–1348, 2018.

[45] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," in *Proceedings of the IEEE Conference on Decision and Control*, Miami, 2018, pp. 6059–6066.

[46] A. D. Bonzanini, D. B. Graves, and A. Mesbah, "Learning-based SMPC for reference tracking under state-dependent uncertainty: An application to atmospheric pressure plasma jets for plasma medicine," *IEEE Transactions on Control Systems Technology*, vol. 30, pp. 611 – 624, 2022.

[47] A. Girard, C. E. Rasmussen, J. Quinonero-Candela, R. Murray-Smith, O. Winther, and J. Larsen, "Multiple-step ahead prediction for non linear dynamic systems–a Gaussian process treatment with propagation of the uncertainty," *Advances in Neural Information Processing Systems*, vol. 15, pp. 529–536, 2002.

[48] J. Umlauft, T. Beckers, and S. Hirche, "Scenario-based optimal control for Gaussian process state space models," in *Proceedings of the European Control Conference*, Limassol, Cyprus, 2018, pp. 1386–1392.

[49] L. Hewing, E. Arcari, L. P. Fröhlich, and M. N. Zeilinger, "On simulation and trajectory prediction with Gaussian process dynamics," in *Proceedings of Learning for Dynamics and Control*, 2020, pp. 424–434.

[50] T. Beckers and S. Hirche, "Prediction with approximated gaussian process dynamical models," *IEEE Transactions on Automatic Control*, 2021, In Press.

[51] N. Lambert, A. Wilcox, H. Zhang, K. S. Pister, and R. Calandra, "Learning accurate long-term dynamics for model-based reinforcement learning," in *Proceedings of the IEEE Conference on Decision and Control*, Austin, 2021, pp. 2880–2887.

[52] A. Wigren, J. Wågberg, F. Lindsten, A. G. Wills, and T. B. Schön, "Nonlinear system identification: Learning while respecting physical models using a sequential monte carlo method," *IEEE Control Systems Magazine*, vol. 42, no. 1, pp. 75–102, 2022.

[53] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," *arXiv preprint arXiv:1803.11347*, 2018.

[54] S. Saemundsson, K. Hofmann, and M. P. Deisenroth, "Meta reinforcement learning with latent variable Gaussian processes," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2018.

[55] E. Arcari, A. Carron, and M. N. Zeilinger, "Meta learning MPC using finite-dimensional gaussian process approximations," *arXiv preprint arXiv:2008.05984*, 2020.

[56] J. A. Paulson and A. Mesbah, "Nonlinear model predictive control with explicit backoffs for stochastic systems under arbitrary uncertainty," *IFAC-PapersOnLine*, vol. 51, pp. 523–534, 2018.

[57] L. Chisci, J. A. Rossiter, and G. Zappa, "Systems with persistent disturbances: predictive control with restricted constraints," *Automatica*, vol. 37, pp. 1019–1028, 2001.

[58] D. Q. Mayne, M. M. Seron, and S. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, vol. 41, pp. 219–224, 2005.

[59] D. Bernardini and A. Bemporad, "Scenario-based model predictive control of stochastic constrained linear systems," in *Proceedings of the IEEE Conference on Decision and Control*, Shanghai, 2009, pp. 6333–6338.

[60] 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, pp. 1306–1319, 2013.

[61] P. Hokayem, E. Cinquemani, D. Chatterjee, F. Ramponi, and J. Lygeros, "Stochastic receding horizon control with output feedback and bounded controls," *Automatica*, vol. 48, pp. 77–88, 2012.

[62] J. A. Paulson and A. Mesbah, "An efficient method for stochastic optimal control with joint chance constraints for nonlinear systems," *International Journal of Robust and Nonlinear Control*, vol. 29, pp. 5017–5037, 2019.

[63] J. A. Paulson, E. A. Buehler, R. D. Braatz, and A. Mesbah, "Stochastic model predictive control with joint chance constraints," *International Journal of Control*, vol. 93, pp. 126–139, 2020.

[64] J. A. Paulson and A. Mesbah, "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, 2018, pp. 6307–6313.

[65] A. I. Forrester and A. J. Keane, "Recent advances in surrogate-based optimization," *Progress in Aerospace Sciences*, vol. 45, pp. 50–79, 2009.

[66] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[67] P. I. Frazier, "A tutorial on Bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.

[68] D. Piga, M. Forgione, S. Formentin, and A. Bemporad, "Performance-oriented model learning for data-driven MPC design," *IEEE Control Systems Letters*, vol. 3, pp. 577–582, 2019.

[69] J. A. Paulson and A. Mesbah, "Data-driven scenario optimization for automated controller tuning with probabilistic performance guarantees," *IEEE Control Systems Letters*, vol. 5, pp. 1477–1482, 2020.

[70] M. Khosravi, V. N. Behrunani, P. Myszkorowski, R. S. Smith, A. Rupenyan, and J. Lygeros, "Performance-driven cascade controller tuning with Bayesian optimization," *IEEE Transactions on Industrial Electronics*, vol. 69, pp. 1032–1042, 2021.

[71] F. Sorourifar, G. Makrygirgos, A. Mesbah, and J. A. Paulson, "A data-driven automatic tuning method for MPC under uncertainty using constrained Bayesian optimization," *IFAC-PapersOnLine*, vol. 54, pp. 243–250, 2021.

[72] K. Kandasamy, K. R. Vysyaraju, W. Neiswanger, B. Paria, C. R. Collins, J. Schneider, B. Poczos, and E. P. Xing, "Tuning hyperparameters without grad students: Scalable and robust Bayesian opti-

misation with Dragonfly." *Journal of Machine Learning Research*, vol. 21, no. 81, pp. 1–27, 2020.

[73] K. Kersting, C. Plagemann, P. Pfaff, and W. Burgard, "Most likely heteroscedastic Gaussian process regression," in *Proceedings of the International Conference on Machine Learning*, 2007, pp. 393–400.

[74] M. Lázaro-Gredilla and M. K. Titsias, "Variational heteroscedastic Gaussian process regression," in *Proceedings of the International Conference on Machine Learning*, 2011, pp. 841–848.

[75] J. A. Paulson, G. Makrygiorgos, and A. Mesbah, "Adversarially robust Bayesian optimization for efficient auto-tuning of generic control structures under uncertainty," *AIChE Journal*, p. e17591, 2022.

[76] K. Kandasamy, G. Dasarathy, J. Oliva, J. Schneider, and B. Poczos, "Multi-fidelity Gaussian process bandit optimisation," *Journal of Artificial Intelligence Research*, vol. 66, pp. 151–196, 2019.

[77] F. Sorourifar, N. Choksi, and J. A. Paulson, "Computationally efficient integrated design and predictive control of flexible energy systems using multi-fidelity simulation-based Bayesian optimization," *Optimal Control Applications and Methods*, 2021, In Press.

[78] J. A. Paulson, K. Shao, and A. Mesbah, "Probabilistically robust Bayesian optimization for data-driven design of arbitrary controllers with Gaussian process emulators," in *Proceedings of the IEEE Conference on Decision and Control*, Austin, 2021, pp. 3633–3639.

[79] G. Makrygiorgos, A. D. Bonzanini, V. Miller, and A. Mesbah, "Performance-oriented model learning for control via multi- objective Bayesian optimization," *Computers & Chemical Engineering*, 2022, In Press.

[80] B. Recht, "A tour of reinforcement learning: The view from continuous control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 253–279, 2019.

[81] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proceedings of the International Conference on Machine Learning*, 2014, pp. 387–395.

[82] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in Neural Information Processing Systems*, vol. 12, 1999.

[83] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[84] S. Gros and M. Zanon, "Reinforcement learning for mixed-integer problems based on MPC," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 5219–5224, 2020.

[85] M. Zanon and S. Gros, "Safe reinforcement learning using robust MPC," *IEEE Transactions on Automatic Control*, vol. 66, no. 8, pp. 3638–3652, 2020.

[86] S. Gros and M. Zanon, "Reinforcement learning based on MPC and the stochastic policy gradient method," in *Proceedings of the American Control Conference*, New Orleans, 2021, pp. 1947–1952.

[87] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[88] R. Quirynen and S. Di Cairano, "PRESAS: Block-structured preconditioning of iterative solvers within a primal active-set method for fast model predictive control," *Optimal Control Applications and Methods*, vol. 41, no. 6, pp. 2282–2307, 2020.

[89] A. Domahidi, E. Chu, and S. Boyd, "ECOS: An SOCP solver for embedded systems," in *Proceedings of the European Control Conference*, Zurich, 2013, pp. 3071–3076.

[90] A. Bemporad, "A quadratic programming algorithm based on nonnegative least squares with applications to embedded model predictive control," *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 1111–1116, 2016.

[91] Y. E. Nesterov, "A method for solving the convex programming problem with convergence rate O(1/k^2)," in *Dokl. akad. nauk Sssr*, vol. 269, 1983, pp. 543–547.

[92] S. Richter, C. N. Jones, and M. Morari, "Computational complexity certification for real-time MPC with input constraints based on the fast gradient method," *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1391–1403, 2011.

[93] V. Nedelcu, I. Necoara, and Q. Tran-Dinh, "Computational complexity of inexact gradient augmented lagrangian methods: application to constrained MPC," *SIAM Journal on Control and Optimization*, vol. 52, no. 5, pp. 3109–3134, 2014.

[94] P. Zometa, M. Kögel, T. Faulwasser, and R. Findeisen, "Implementation aspects of model predictive control for embedded systems," in *Proceedings of the American Control Conference*, Montréal, 2012, pp. 1205–1210.

[95] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[96] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," in *Proceedings of the UKACC International Conference on Control*, 2018, pp. 339–339.

[97] G. Frison, D. Kouzoupis, T. Sartor, A. Zanelli, and M. Diehl, "BLASFEO: Basic linear algebra subroutines for embedded optimization," *ACM Transactions on Mathematical Software*, vol. 44, no. 4, pp. 1–30, 2018.

[98] R. Quirynen, M. Vukov, M. Zanon, and M. Diehl, "Autogenerating microsecond solvers for nonlinear MPC: a tutorial using ACADO integrators," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 685–704, 2015.

[99] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[100] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari, "FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs," *International Journal of Control*, vol. 93, no. 1, pp. 13–29, 2020.

[101] T. Englert, A. Völz, F. Mesmer, S. Rhein, and K. Graichen, "A software framework for embedded nonlinear model predictive control using a gradient-based augmented lagrangian approach (GRAMPC)," *Optimization and Engineering*, vol. 20, no. 3, pp. 769–809, 2019.

[102] H. J. Ferreau, S. Almér, R. Verschueren, M. Diehl, D. Frick, A. Domahidi, J. L. Jerez, G. Stathopoulos, and C. Jones, "Embedded optimization methods for industrial automatic control," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 13 194–13 209, 2017.

[103] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3 – 20, 2002.

[104] E. N. Pistikopoulos, N. P. Faísca, K. I. Kouramas, and C. Panos, "Explicit robust model predictive control," *IFAC Proceedings Volumes*, vol. 42, no. 11, pp. 243–248, 2009.

[105] T. Geyer, F. D. Torrisi, and M. Morari, "Optimal complexity reduction of polyhedral piecewise affine systems," *Automatica*, vol. 44, no. 7, pp. 1728–1740, 2008.

[106] J. Holaza, B. Takács, and M. Kvasnica, "Synthesis of simple explicit MPC optimizers by function approximation," in *Proceedings of the International Conference on Process Control*, Strbske Pleso, Slovakia, 2013, pp. 377–382.

[107] A. Bemporad, A. Oliveri, T. Poggi, and M. Storace, "Ultra-fast stabilizing model predictive control via canonical piecewise affine approximations," *IEEE Transactions on Automatic Control*, vol. 56, no. 12, pp. 2883–2897, 2011.

[108] L. Csko, M. Kvasnica, and B. Lantos, "Explicit MPC-based RBF neural network controller design with discrete-time actual Kalman filter for semiactive suspension," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 5, pp. 1736–1753, 2015.

[109] T. Parisini and R. Zoppoli, "A receding-horizon regulator for nonlinear systems and a neural approximation," *Automatica*, vol. 31, no. 10, pp. 1443–1451, 1995.

[110] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of GO with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, 2016.

[111] I. Safran and O. Shamir, "Depth-width tradeoffs in approximating natural functions with neural networks," in *International Conference on Machine Learning*, 2017, pp. 2979–2987.

[112] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Advances in Neural Information Processing Systems*, 2014, pp. 2924–2932.

[113] B. Karg and S. Lucia, "Deep learning-based embedded mixed-integer model predictive control," in *Proceedings of the European Control Conference*, 2018, pp. 2075–2080.

[114] S. W. Chen, T. Wang, N. Atanasov, V. Kumar, and M. Morari, "Large

scale model predictive control with neural networks and primal active sets," *Automatica*, vol. 135, p. 109947, 2022.

[115] S. Lucia, D. Navarro, B. Karg, H. Sarnago, and O. Lucia, "Deep learning-based model predictive control for resonant power converters," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 1, pp. 409–420, 2020.

[116] D. Krishnamoorthy, A. Mesbah, and J. A. Paulson, "An adaptive correction scheme for offset-free asymptotic performance in deep learning-based economic MPC," *IFAC-PapersOnLine*, vol. 54, pp. 584–589, 2021.

[117] B. Karg and S. Lucia, "Approximate moving horizon estimation and robust nonlinear model predictive control via deep learning," *Computers & Chemical Engineering*, vol. 148, p. 107266, 2021.

[118] D. Krishnamoorthy, "A sensitivity-based data augmentation framework for model predictive control policy approximation," *IEEE Transactions on Automatic Control*, 2021, In Press.

[119] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org.

[120] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 8024–8035.

[121] B. Karg, T. Alamo, and S. Lucia, "Probabilistic performance validation of deep learning-based robust NMPC controllers," *International Journal of Robust and Nonlinear Control*, vol. 31, no. 18, pp. 8855–8876, 2021.

[122] J. Chen, Y. Chen, L. Tong, L. Peng, and Y. Kang, "A backpropagation neural network-based explicit model predictive control for DC–DC converters with high switching frequency," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 8, no. 3, pp. 2124–2142, 2020.

[123] K. J. Chan, J. A. Paulson, and A. Mesbah, "Deep learning-based approximate nonlinear model predictive control with offset-free tracking for embedded applications," in *Proceedings of the American Control Conference*, New Orleans, 2021, pp. 3475–3481.

[124] B. Karg and S. Lucia, "Stability and feasibility of neural network-based controllers via output range analysis," in *Proceedings of the IEEE Conference on Decision and Control*, Nice, 2020, pp. 4947–4954.

[125] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer, "Learning an approximate model predictive controller with guarantees," *IEEE Control Systems Letters*, vol. 2, pp. 543–548, 2018.

[126] T. Alamo, R. Tempo, A. Luque, and D. Ramirez, "Randomized methods for design of uncertain systems: sample complexity and sequential algorithms," *Automatica*, vol. 52, pp. 160–172, 2015.

[127] R. Tempo, E. Bai, and F. Dabbene, "Probabilistic robustness analysis: explicit bounds for the minimum number of samples," *Systems & Control Letters*, vol. 30, pp. 237–242, 1997.

[128] M. Fazlyab, M. Morari, and G. J. Pappas, "Probabilistic verification and reachability analysis of neural networks via semidefinite programming," in *Proceedings of the IEEE Conference on Decision and Control*, Nice, 2019, pp. 2726–2731.

[129] B. Karg and S. Lucia, "Efficient representation and approximation of model predictive control laws via deep learning," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3866–3878, 2020.

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

[131] S. Chen, K. Saulnier, N. Atanasov, D. D. Lee, V. Kumar, G. J. Pappas, and M. Morari, "Approximating explicit model predictive control using constrained neural networks," in *Proceedings of the American Control Conference*, Milwaukee, 2018, pp. 1520–1527.

[132] J. A. Paulson and A. Mesbah, "Approximate closed-loop robust model predictive control with guaranteed stability and constraint satisfaction," *IEEE Control Systems Letters*, vol. 4, no. 3, pp. 719–724, 2020.

[133] A. D. Bonzanini, J. A. Paulson, D. B. Graves, and A. Mesbah, "Toward safe dose delivery in plasma medicine using projected neural network-based fast approximate nmpc," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 5279–5285, 2020.

[134] S. Dutta, S. Jha, S. Sankaranarayanan, and A. Tiwari, "Output range analysis for deep feedforward neural networks," in *NASA Formal Methods Symposium*. Springer, 2018, pp. 121–138.

[135] F. Fabiani and P. J. Goulart, "Reliably-stabilizing piecewise-affine neural network controllers," *arXiv preprint arXiv:2111.07183*, 2021.

[136] B. Karg and S. Lucia, "Reinforced approximate robust nonlinear model predictive control," in *Proceedings of the International Conference on Process Control*, Strbske Pleso, Slovakia, 2021, pp. 149–156.

[137] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[138] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, 2007, vol. 703.

[139] D. P. Bertsekas, "Approximate dynamic programming," 2008.

[140] J.-M. Lee and J. H. Lee, "Approximate dynamic programming strategies and their applicability for process control: A review and future directions," *International Journal of Control, Automation, and Systems*, vol. 2, no. 3, pp. 263–278, 2004.

[141] J. Skaf, S. Boyd, and A. Zeevi, "Shrinking-horizon dynamic programming," *International Journal of Robust and Nonlinear Control*, vol. 20, no. 17, pp. 1993–2002, 2010.

[142] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[143] J. N. Tsitsiklis, "Asynchronous stochastic approximation and Q-learning," *Machine Learning*, vol. 16, no. 3, pp. 185–202, 1994.

[144] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, no. 1, pp. 9–44, 1988.

[145] S. J. Bradtke and A. G. Barto, "Linear least-squares algorithms for temporal difference learning," *Machine Learning*, vol. 22, no. 1, pp. 33–57, 1996.

[146] D. P. Bertsekas and S. Ioffe, "Temporal differences-based policy iteration and applications in neuro-dynamic programming," *Lab. for Info. and Decision Systems Report LIDS-P-2349, MIT, Cambridge, MA*, 1996.

[147] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[148] J. H. Lee and W. Wong, "Approximate dynamic programming approach for process control," *Journal of Process Control*, vol. 20, no. 9, pp. 1038–1048, 2010.

[149] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, pp. 26–38, 2017.

[150] S. Kamthe and M. Deisenroth, "Data-efficient reinforcement learning with probabilistic model predictive control," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2018, pp. 1701–1710.

[151] J. Arroyo, C. Manna, F. Spiessens, and L. Helsen, "Reinforced model predictive control (RL-MPC) for building energy management," *Applied Energy*, vol. 309, p. 118346, 2022.

[152] T. H. Oh, H. M. Park, J. W. Kim, and J. M. Lee, "Integration of reinforcement learning and model predictive control to optimize semi-batch bioreactor," *AIChE Journal*, p. e17658, 2022.

[153] D. P. Bertsekas, "Newton's method for reinforcement learning and model predictive control," 2022.