IEEE Control Systems Letters paper presented at
2023 American Control Conference (ACC)
San Diego, CA, USA. May 31 - June 2, 2023

Physics-Informed Machine Learning for Modeling and Control of Dynamical Systems

Truong X. Nghiem[1], Ján Drgoňa[2], Colin Jones[3], Zoltan Nagy[4], Roland Schwan[3], Biswadip Dey[5], Ankush Chakrabarty[6], Stefano Di Cairano[6], Joel A. Paulson[7], Andrea Carron[8], Melanie N. Zeilinger[8] Wenceslao Shaw Cortez[2], and Draguna L. Vrabie[2]

Abstract— **Physics-informed machine learning (PIML) is a set of methods and tools that systematically integrate machine learning (ML) algorithms with physical constraints and abstract mathematical models developed in scientific and engineering domains. As opposed to purely data-driven methods, PIML models can be trained from additional information obtained by enforcing physical laws such as energy and mass conservation. More broadly, PIML models can include abstract properties and conditions such as stability, convexity, or invariance. The basic premise of PIML is that the integration of ML and physics can yield more effective, physically consistent, and data-efficient models. This paper aims to provide a tutorial-like overview of the recent advances in PIML for dynamical system modeling and control. Specifically, the paper covers an overview of the theory, fundamental concepts and methods, tools, and applications on topics of: 1) physics-informed learning for system identification; 2) physics-informed learning for control; 3) analysis and verification of PIML models; and 4) physics-informed digital twins. The paper is concluded with a perspective on open challenges and future research opportunities.**

I. Introduction

Modern engineering systems generate large amounts of data either via sensors in the physical world or via simulation of virtual environments. The increased storage and computational power of the underlying hardware and communication infrastructure paved the way for the use of data-driven algorithms. Machine learning (ML) methods leverage a large amount of data to achieve remarkable success, especially in areas such as games, speech recognition, or image processing. These recorded successes especially occur in the areas where there is an abundance of data and where the underlying processes have hard-to-discover governing laws and are driven by non-obvious fundamental principles. In such cases, ML shows strong capabilities in learning non-obvious relations that allow to achieve the desired tasks if a sufficient amount of representative data is available.

Due to such successes, a number of investigations have started involving the application of ML to other domains, including physical systems, such as automotive [1], aerospace [2], process control [3], energy systems [4], and robotics [5]. However, these traditional engineering applications are governed by fundamental principles and physical constraints that have been studied for centuries and are thus better understood or well-known. Some examples include conservation laws of mass or energy, fundamental laws of motion or electromagnetism, ranges of constants such as efficiencies and physical dimensions or gravitational constants. For physical systems, there may be expectations that ML models will satisfy such principles and constraints. While the prediction accuracy of ML methods may be as good in the computer science domains, the lack of satisfaction of fundamental principles and underlying physics and safety constraints represent a significant limitation to actual deployment of ML models in real-world applications.

Properly enforcing physical laws and constraints in ML may result in several benefits, affecting training, performance, and trust of the ML result such as: (i) reduced data requirement of the ML model due to learning in lower dimensional, manifolds; (ii) higher precision and improved generalization since the underlying physical principles will be satisfied and the errors related to violating them will be avoided; (iii) increased interpretability and trust by the application engineers by upholding the known domain principles. However, most of the standard ML approaches cannot leverage physical principles and constraints, resulting in a limited real-world impact of ML in the safety-critical domains [6].

The cautious adoption of modern ML methods in real-world engineering domains is a direct result of the research priorities in the historical developments within the computer science domains. More specifically, ML methods have been primarily developed in domains related to human aspects and behaviors, such as vision or language recognition, where the safety requirements are less strict. Furthermore, those domains consist of underlying physical principles that are still partially unexplained or extremely complex to describe and hence hard to be embedded directly in the ML algorithms.

In recent years, researchers from both computer science and engineering domains have recognized the potential of

[1]School of Informatics, Computing, and Cyber Systems, Northern Arizona University, Flagstaff, AZ 86011, USA truong.nghiem@nau.edu
[2]Pacific Northwest National Laboratory, Richland, WA, USA {jan.drgona, w.shawcortez, draguna.vrabie}@pnnl.gov
[3]EPFL, Switzerland {colin.jones, roland.schwan}@epfl.ch
[4]The University of Texas at Austin, USA nagy@utexas.edu
[5]Siemens Corporation, Technology, Princeton, NJ 08540, USA biswa-dey@ieee.org
[6]Mitsubishi Electric Research Laboratories, Cambridge, MA 02139, USA. achakrabarty@ieee.org, dicairano@ieee.org
[7]The Ohio State University, Columbus, OH 43210, USA paulson.82@osu.edu
[8]ETH Zurich, Switzerland carrona,mzeilinger@ethz.ch
This material is based upon work supported by the National Science Foundation under Grant No. 2138388 and Grant No. 2238296.

bridging this gap by developing new methods that combine algorithms and tools from machine learning with well-known engineering models and principles. This quest to extend machine learning to account for the physical principles and constraints of the underlying process gave rise to the so-called Physics Informed Machine Learning (PIML), also referred to as Scientific Machine Learning (SciML). Related surveys in this area include PIML methods for partial differential equations (PDEs) [7] or generic dynamical systems [8, 9]. However, to the author's best knowledge, there is no tutorial overview of the PIML methods for the control of dynamical systems. This paper aims to bridge this gap. The rest of the paper is structured as follows, section II. reviews the landscape of PIML methods for modeling and control of dynamical systems, including PIML methods for system identification, control design, and formal verification. Section III. briefly summarizes current challenges and opportunities. While section IV. concludes the paper.

II. Landscape of Physics-Informed Machine Learning Methods for Modeling and Control

In this paper, we define PIML as follows.

Definition II.1. PIML is a set of methods and tools that systematically integrate recent advancements in machine learning algorithms with mathematical models developed in various scientific and engineering domains. As opposed to pure data-driven methods that assume no existence of prior domain knowledge, PIML models can be trained from additional information obtained by enforcing constraints such as symmetries, causal relationships, or conservation laws. More broadly, PIML models can include abstract properties and conditions such as stability, convexity, or invariance from domains such as dynamical systems and control theory.

A. Physics-informed learning for system identification

One of the first applications of PIML in control is to learn dynamical models for a physical system given some prior knowledge and time series data. Conventional dynamical system and control theories usually use *white-box* modeling to develop models from physical principles and system specifications, which often require extraordinary effort and expertise. On the other hand, purely data-driven approaches employ *black-box* modeling, which learns models purely from data without making any prior physical assumptions, relying on ML techniques such as neural networks (NNs) and regression trees (RT). While *gray-box* modeling and physics-informed machine learning (PIML) both aim to bridge the gap between the white-box and black-box modeling paradigms by integrating physical laws into models, they take different approaches. *Gray-box* modeling starts from the same physical principles used in white-box modeling that are subsequently simplified to obtain a reduced-order model structure, which can be used to fit data to identify its parameters. Therefore the *gray-box* approach still requires significant expert knowledge and often ignores nonlinear phenomena, causing accuracy loss. On the other hand, PIML starts from the black-box modeling end and embeds prior knowledge of the system's physics into ML methods to yield models that are more interpretable, robust, accurate, and physically consistent for generalization tasks while maintaining the relative ease of use of *black-box* ML methods. This section reviews the main PIML approaches for incorporating physics into the data-driven modeling of dynamical systems. Namely, physics-informed model architecture, physics-informed loss function design, or a combination of the two [7, 10].

Definition II.2. Model architecture represents an interconnected set of computational nodes that form the resulting computational graph defining the mathematical expression defining the ML model. Architectures of ML models are often defined as composite functional forms or block diagrams.

1) Structural priors in discrete-time models: Discrete-time dynamical models are represented by a general class of state-space models (SSM) given as

$$x_{k+1} = f_{\text{SSM}}(x_k, u_k) \tag{1}$$

with $x_k \in \mathbb{R}^{n_x}$ and $u_k \in \mathbb{R}^{n_u}$ being system states and control inputs, respectively. Most of the structural PIML approaches are concerned with designing the functional form of (1) to offer increased expressivity compared to classical linear SSM while satisfying desired properties.

Recent examples include neural SSM [11, 12], deep Koopman models [13, 14], Hammerstein-Wiener neural models [15, 16], graph neural network-based (GNN) time-stepper models [17, 18], or SINDY-type methods based on sparse regression of candidate basis functions [19, 20]. Further examples include architectures such as Non-Autonomous Input-Output Stable Nets (NAIS-Nets) with guaranteed asymptotic stability of its forward pass dynamics based on stable matrix factorization of the state dynamics, joint learning of the Koopman model representation and state observer while ensuring observability [21], or so-called Gumbel Graph Networks (GGN) for modeling network dynamics [22]. While authors in [23] introduced physically consistent NN inspired by resistance-capacitance (RC) networks applied to modeling building thermal dynamics.

2) Structural priors in continuous-time models: Continuous-time models are, in general, represented by a set of ordinary differential equations (ODE) given as

$$\frac{dx}{dt} = f_{\text{ODE}}(x, u) \tag{2}$$

with $\mathbf{x} \in \mathbb{R}^{n_x}$ and $\mathbf{u} \in \mathbb{R}^{n_u}$ being system states and control inputs, respectively. Neural ODEs [24] have been proposed as black-box counterparts to white-box ODEs for data-driven modeling of dynamical systems. In NODEs, the right-hand side (RHS) of the differential equation (2) is approximated by deep neural network $NN(\mathbf{x}, \theta)$ with trainable parameters θ. However, purely black-box NODEs may require prohibitively large training data and may struggle with generalization and physical laws. Researchers have been looking at ways to design the structure of the RHS in (2) to balance conflicting criteria such as the expressivity of black-box models and the physical consistency of white-box models.

Examples include incorporated stability-promoting priors in NODEs via spectral element method [25], multiple-shooting integration schemes [26, 27], neural delay differential euquations [28], or extensions supporting uncertainty quantification (UQ) such as Bayesian NODEs [29], stochastic NODEs [30], or jump stochastic NODEs capable of handling discrete events [31]. Most recently, *universal differential equations* (UDE) [32] have been proposed as an extension to NODEs to allow for a systematic combination of white-box components and black-box components. Specific examples of PIML structured continuous models include the use of UDEs for modeling networked dynamical systems [33], or continuous graph neural networks (CGNNs) [34] also called graph neural ordinary differential equations (GDEs) [35]. Related approaches include *energy-based* model architectures such as Hamiltonian [36, 37] or Lagrangian [38, 39] neural networks, and their various extensions such as graph Hamiltonian neural networks (HNN) [40], Hamiltonian dynamics with dissipative forces [41], HNN with explicit constraints [42], or non-canonical Hamiltonian systems [43]. Such approaches have also been extended to learn dynamics of rigid body systems with contacts and collisions [44] or to learn Lagrangian dynamics from high-dimensional video data [45]. Others have proposed jointly learning Neural Lyapunov functions together with the forward dynamics model [46]. The main advantage of these *energy-based* approaches is that they can satisfy conservation laws or enforce properties such as stability or dissipativity by design.

3) Matrix factorizations: Linear algebra components are ubiquitous in machine learning, including weights in neural networks, operators such as Koopman or Perron-Frobenius, or Jacobian matrices of differential equations. Many matrix factorization methods exist for designing desired properties such as sparsity, symmetry, positive definiteness, eigenvalue placement, or invertibility. Naturally, researchers have exploited various matrix factorizations in weights of deep neural networks [47], with examples including Perron-Frobenius [48], orthogonal [49, 50], spectral [51], symplectic [37, 52], anti-symmetric [53], Gershgorin disc [54], and Schur Decomposition [55] weights. Let's consider spectral factorization [51] as an example. In this approach, the weight matrix $\mathbf{W}$ is parametrized by the components of singular value decomposition (SVD) method as follows,

$$\Sigma = \operatorname{diag}(\lambda_{\max} - (\lambda_{\max} - \lambda_{\min}) \cdot \sigma(s)) \tag{3a}$$

$$\mathbf{W} = \mathbf{U}\Sigma\mathbf{V} \tag{3b}$$

Where Σ is a matrix of singular values, $\lambda_{\min}$ and $\lambda_{\max}$ represent lower and upper bounds for singular values, σ is a sigmoid activation function, s is a vector of trainable singular value parameters, and $\mathbf{U}$ and $\mathbf{V}$ are trainable orthogonal matrices that can be either designed via Householder reflectors [51] or via soft constraint penalties [56]. Authors in [57] used the matrix factorizations to enforce dissipativity in neural SSM for modeling building thermal dynamics.

4) PIML loss functions: This approach incorporates physics through *learning biases* by imposing appropriate

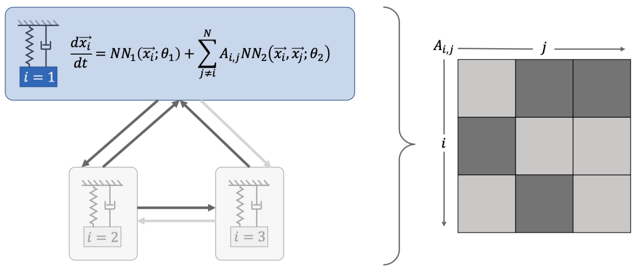
$\frac{d\vec{x_i}}{dt} = NN_1(\vec{x_i};\theta_1) + \sum_{j\neq i}^{N} A_{i,j} NN_2(\vec{x_i},\vec{x_j};\theta_2)$
$i = 1$
$i = 2$
$i = 3$
$A_{i,j}$
j
i

Fig. 1: Methodology of the universal differential equation for learning components of networked dynamical systems [33].

penalties during learning [7]. In its general form, the composite physics-informed loss function $\mathscr{L}$ is defined as

$$\mathscr{L} = \sum_{i}^{n} \ell^{i}_{\text{data}} + \sum_{j}^{m} \ell^{j}_{\text{physics}} \tag{4}$$

where ℓ^{i}_{data} represents n number of data-driven objective terms, $\ell^{j}_{\text{physics}}$ define m number of physics-based regularization terms. Examples include using soft penalties on eigenvalues or singular values of the neural network weights [58, 59] to promote stability, promoting boundedness, and smoothness [56], using penalties to regularize black-box GNN components [60], regularizing error and stiffness estimates in NODEs for improved accuracy and speed [61], and penalizing the violations of the Lyapunov stability conditions as additional loss term in training NODEs [62]. Others have proposed Jacobian regularization [63] for promoting stability, or using surrogate loss functions for faster training of NODEs [64].

5) Case Study: Incorporating physics in the model via structural priors: This case study is adopted from [33] and demonstrates the use of universal differential equations (UDE) [32] for data-driven modeling of networked dynamical systems. The method illustrated in Fig. 1. Specifically, the UDE model used in [33] is given as follows,

$$\frac{dx_i}{dt} = NN_1\left(x_i;\theta_1\right) + \sum_{j\neq i}^{N} \mathbf{A}_{i,j} NN_2\left(x_i, x_j;\theta_2\right) \tag{5}$$

The system dynamics is defined by the interaction of N number of nodes represented by state vectors $x_i(t) \in \mathbb{R}^{n_x}$ where $i \in \mathbb{N}_1^N$ represents node index. Here $NN_1(x_i,\theta_1) : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ and $NN_2(x_i,\theta_2) : \mathbb{R}^{2n_x} \to \mathbb{R}^{n_x}$ are neural networks modeling node and interaction dynamics, respectively. The trainable adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ defines the connectivity between nodes, i.e., $A_{i,j} = 1$ means node i is connected to node j. The equation 5 compactly represented as $F(x;\Theta)$ with lumped trainable parameters $\Theta = \{\theta_1, \theta_2, \mathbf{A}\}$, can be solved with standard numerical ODE solvers,

$$x\left(t_{end}\right) = \texttt{ODESolve}\left(F\left(x;\Theta\right), x_0, t_0, t_{end}\right). \tag{6}$$

In general, the gradients of $F(x;\Theta)$ can be computed in two ways via the adjoint sensitivity method as used in NODE or via automatic differentiation of the computational graph of the unrolled ODE solver. The loss function is formulated as

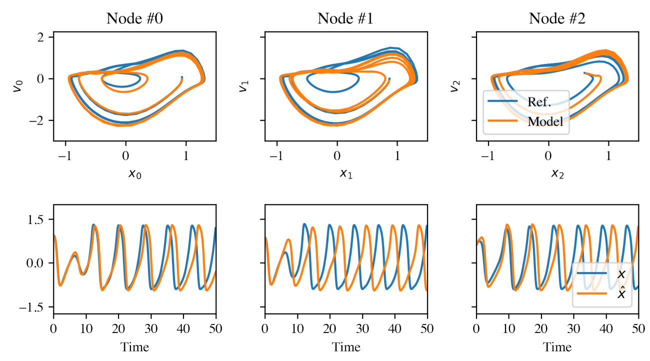

Fig. 2: Prediction performance of the UDE model 5 (orange) on never-before-seen networked system (blue).

regularized mean squared error (MSE) between predictions and measurements, given as,

$$\mathscr{L}(\Theta, \mathbf{A}) = \frac{1}{m}||\hat{\mathbf{X}} - \mathbf{X}||_2^2 + \alpha||\mathbf{A}||_1, \tag{7}$$

where $\mathbf{X}$ and $\hat{\mathbf{X}}$ are measured and predicted state trajectories, respectively. Scalar m represents a number of samples. The second term represents ℓ_1 penalty for promoting sparsity in the adjacency matrix $\mathbf{A}$.

The authors in [33] demonstrate the utility of this UDE modeling approach in a case study with coupled nonlinear oscillators. The benefit of the UDE model in the form (5) is improved generalization and interoperability, as opposed to purely black-box NODE [24]. This is because the structure of (5) is closer to the governing physics of a general class of networked systems. To demonstrate their utility, the authors deploy the trained models on networked systems with never-before-seen topology. Fig. 2 compares the phase portraits and time series of the never-before-seen system against the predictions generated by the UDE model 5 trained on data from a networked system with different topology. What is being demonstrated are the generalization capabilities of the UDE model 5 to qualitatively reproduce physically plausible transient behavior, including the reconstruction of limit cycle attractors outside of the distribution of the training data.

B. *Physics-informed learning for control*

PIML models have been widely used in model-based control methods, such as model predictive control (MPC) or model-based reinforcement learning (RL). This is typically done by combining physics-based priors with various regression methods to improve the control performance. Subsequently, with the spread of learning methods embedded with uncertainty quantification (UQ) measures, robust and stochastic model-based controllers have been developed to exploit the uncertainty information provided by these PIML models. Including UQ methods makes the controller cautious and is of great significance where only a limited amount of data is available. More recently, the use of PIML has been extended to different areas such as learning control Lyapunov functions, tuning model-based controllers, designing dual control strategies, developing safe control frameworks, and learning explicit control policies. PIML has also found applications in differentiable programming-based methods, which allows the designer to embed physical models and constraints while training a controller using automatic differentiation-based (AD) solvers, e.g., with physics-informed priors in actor-critics and model-based RL methods.

1) PIML dynamics models for MPC: Data-driven models have been widely used in MPC; see [65] for a comprehensive review of these methods up to 2020. With recent advances, many approaches have been proposed for incorporating PIML and physical priors into MPC. These methods employ learned PIML dynamic models (Section II-A) in the MPC optimization formulation, therefore implicitly leveraging physical priors through the models. Authors in [66] used structured neural SSM in MPC to control the nonlinear behavior of a robotic hand in cutting tasks. In [67] a physics-informed NN is used with MPC in order to control a robotic arm. Others have demonstrated the utility of using the sparse identification of nonlinear dynamics (SINDY) method for learning models for MPC, in the so-called SINDY-MPC framework [68]. The use of ML models in MPC with guarantees can be traced back to [69, 70]. The authors propose a robust control method that uses two models: the first is a simple physics-based linear model that accounts for the safety of the system used to ensure constraint satisfaction, while the second is an ML model used to maximize the performance of the controller. A different approach is presented in [71, 72], where a Gaussian process (GP) is used to learn the error dynamics to improve the physics-based model and hence the controller performance. Another MPC algorithm that uses kernel-based methods is presented in [73]. While the work in [74] demonstrates the utility of even a simple physics-informed ARMAX model for building control tasks using MPC.

2) Learning Lyapunov and Barrier functions: (Control) Lyapunov functions are crucial tools for determining the stability properties of dynamical systems, quantifying the domain of attraction and the robustness to perturbation, designing controllers, or designing terminal ingredients for MPC. However, methods for synthesizing Lyapunov functions in closed-form for nonlinear dynamical systems, even with known models, are generally not available. Early approaches for learning Lyapunov functions with NNs date back to the 90s [75]. Recently, different authors [46, 76] introduced neural Lyapunov functions, representing neural architectures that satisfy Lyapunov function properties by design. Subsequently, authors in [76] have used these neural Lyapunov functions as verification tools for safe learning-based controllers, while [77] used neural Lyapunov function for online tuning of MPC parameters, and in [78] neural Lyapunov function was trained in conjunction with neural control policy. Another example is [79], where NNs are employed to learn compositional Lyapunov functions. This work takes inspiration from recent methods that solve high-dimensional PDEs using NNs by exploiting suitable structural properties. As Lyapunov functions can also be represented by PDEs, PIML methods for learning PDEs can also be used to learn Lyapunov functions. Control barrier functions

(CBF) recently became a very popular computationally efficient method for safe learning-based control. However, similarly to Lyapunov functions, CBFs are in general, hard to design analytically for general nonlinear dynamical systems. To alleviate this problem, different authors have proposed learning CBFs from data. In [80], use NNs to approximate the signed distance functions and subsequently use second-order cone programming to synthesize the control policy. Authors in [81] utilize imitation learning to learn CBFs from expert demonstrations of safe trajectories. While others present architectures for constructing neural barrier functions [82] for multi-agent control, and robust neural Lyapunov-barrier [83] functions for safe nonlinear control.

3) Differentiable Control: These methods leverage automatic differentiation (AD) for computing gradients of the physics model or optimal control problem that can be used as part of the learning algorithm. An example is to use AD for computing the backward gradients of the underlying MPC optimization problem. In principle, this can be done in two ways, by differentiating the KKT conditions constructed analytically [84] or by unrolling the computational graph of the MPC problem [85, 86]. The advantage of the methods based on differentiable programming (DP) [87] is that they allow for end-to-end training of different components of the optimal control problem. Thanks to its versatility, this method has been used to learn system dynamical models [88], weighting factors of the objective functions [84], neural control policies [89]–[91], or safety filters based on differentiable projections [92], and differentiable control barrier function [93, 94]. Most recent applications of differentiable control include autonomous vehicles [95], robotics [96, 97], building control [98, 99], traffic flow [100], epidemic processes [101], or visuomotor control via differentiable rendering [102]. Naturally, physical priors can be incorporated into the MPC formulation through the system model, which will be taken into account when the optimization problem is differentiated. Most recently, several differentiable optimization and control libraries have emerged in the open-source domain [103]–[106] The DP approach has also been used to develop differentiable physics models to control partial differential equations (PDEs) [107, 108].

4) PIML for safe data-driven control: Another use of PIML models in combination with control-like algorithms is in safety frameworks. Given a safety-critical system, i.e., a dynamical system subject to state and input constraints, a safety framework can certify whether a proposed control input is safe to apply or not. Whenever a proposed input is unsafe, the safety framework can propose an alternative safe control input, which usually is as close as possible to the proposed one. For instance, [109]–[111] present safety frameworks that use an MPC like structure to determine the input that satisfies input and state constraints and is as close as possible to the proposed potentially unsafe input. Safety frameworks are also known under other names such as active set reachability [112], Safety Handling Exploration with Risk Perception Algorithm (SHERPA) [113], and model predictive safety filter [114]. Other approaches use control barrier functions [115, 116] and reachability analysis [24, 117] to ensure safety. A unified approach that brings predictive safety filters and control barrier functions, called predictive control barrier functions, has been recently presented [118]

Safety frameworks have also been proposed for learning-based control, where machine learning (ML) is used to learn either the system model or the control law. [119] uses an MILP formulation to train NN-based controllers to satisfy input constraints, safety constraints, and stability conditions. [120] uses differentiable projections to enforce Lyapunov stability conditions while minimizing a performance objective-based loss function (e.g., LQR) during training of a NN-based control law. Cautious control methods employ the uncertainty of the learned model to ensure safety constraints and become less conservative when the model is updated [72].

5) Physics-informed RL: PIML has been used in RL to improve its accuracy and physical consistency. Model-based reinforcement learning methods rely on a model of the environment and its dynamics with which the agent interacts. The models can be generated via high-fidelity physics-based simulations or learned from data. Deep-RL methods employ NNs to generate data-driven models. Approaches similar to those presented in the MPC section can be used to ensure physically consistent models. For example, [121] uses a physically consistent NN [23] to learn a dynamical model of a building, which is then used with a deep RL agent. In [122], a model-based policy search method, called PILCO, is developed where prior physics knowledge can be introduced in a GP model. Physics-informed RL schemes informed by the underlying dynamics have been employed to tackle aircraft conflict resolution problems [123] and power system optimization problems [124, 125]. Recent works in [126, 127] provide a systematic framework for using RL to tune the parameters of optimization-based MPC policies. This RL-MPC framework combines the advantages of both approaches, namely the flexibility of RL with constraints satisfaction and stability of MPC. It has recently found applications in building energy system control [128].

6) Case study: Physics-informed safety filters for data-driven control: The work [109] shows how to deploy RL algorithms on safety-critical systems, i.e., systems subject to state and input constraints, without violating constraints. The proposed approach makes use of predictive safety filters, namely, an optimization-based algorithm that receives the proposed control input and decides, based on the current system state, if it can be safely applied to the real system or if it has to be modified otherwise. Given a dynamical system of the form (1), the predictive safety filter solves each time-step the following optimization problem,

$$\begin{aligned} \min_{x,u}\ & \|u_L - u_k\| \\ \text{s.t. } & x_{k+1} = f(x_k, u_k), \\ & x_k \in \mathbb{X}, \quad u_k \in \mathbb{U}, \quad x_N \in \mathscr{S}^t, \quad x_0 = x(t), \end{aligned} \tag{8}$$

where u_L is the input proposed by a potentially unsafe controller, e.g., an RL algorithm and $\mathscr{S}^t$ is a terminal invariant set for the dynamical system $f(x_k, u_k)$. The predictive safety filter minimizes the difference between the proposed input and a safe input in order to be as less invasive as possible. The input u_k is then applied. While the aforementioned approach relies on a perfect knowledge of the system dynamics, in practice a perfect model is never available. The predictive safety filter can deal with the uncertain system by exploiting robust and stochastic model predictive control techniques. A parametric model $x_{k+1} = f(x_k, u_k, \theta)$ is considered where θ is an unknown parameter vector that is either bounded or has a known prior distribution $p(\theta)$ with mean $\mathbb{E}[\theta]$. In the aforementioned paper, the safety filter is applied to the swing-up of an inverted pendulum and the control of a quad-copter. In both cases, unsafe controllers are employed and the safety filter demonstrates how it is possible to satisfy constraints.

C. Analysis and verification of ML models

In addition to system identification and control design, PIML has been incorporated into the analysis and verification of dynamical systems. Firstly, consider a learning structure, (e.g., NN, GP) on which PIML methods typically rely on. Properties of these structures, such as input-to-state stability, input-output bounds, and estimation of Lipschitz constants, are important to determine how these structures behave in a closed-loop setting. Secondly, learned system dynamics would ideally have inherent physical properties, e.g., passivity, stability, and invariance, from the physical systems they are approximating. Guarantees that such physical properties hold can be beneficial for future control design in addition to ensuring a better representation of the physical system itself. Finally, learning-based control policies may lack guarantees of closed-loop stability or invariance. Verification methods, including sampling-based and reachability, can be used to guarantee such desirable properties of the closed-loop system. In the following, we survey the literature with respect to analysis and verification methods developed for PIML.

1) Analysis and verification of learned system dynamics: These are methods for analyzing and verifying properties of learned models of dynamical systems in the open-loop setting. They can be categorized by the class of system properties addressed as follows.

a) Stability: The stability and attractors of recurrent neural networks (RNNs) have been studied in continuous time in [129]. Authors in [130, 131] study Lyapunov spectra and Lyapunov exponents of RNNs. Authors in [132] empirically study stability of deep neural architectures in the context of solving forward-backward stochastic differential equations. [47] analyzes the dissipativity of autonomous neural dynamics. [133] proposes a method for synthesis of Lyapunov NNs while providing formal guarantees-using satisfiability modulo theories (SMT). [79] defines deep neural network (DNN) structure to approximate compositional Lyapunov functions for a system where the approximation error is dependent on the number of neurons in the network. [134] uses mixed-integer programming in a sampling-based Lyapunov function verification method for piece-wise linear NN approximations of nonlinear autonomous systems for estimating their region of attraction. [135] uses Koopman operator theory and sample-based methods to learn a neural network-based Lyapunov function and region of attraction of a given open-loop system. [136] uses GP-based approximate dynamic programming with a sample-based method to learn a Lyapunov function and region of attraction of an open loop system. [137] analyzes the Input-to-State (ISS) stability of long short-term memory (LSTM) networks by recasting them in the state space form. The authors in [138, 139] use contraction analysis to design an implicit model structure that allows for a convex parameterization of stable RNN models. [140] uses non-Euclidean contraction theory to establish well-posedness, contraction, and l_∞-Lipschitz properties of implicit NNs.

b) Lipschitz properties: [141] shows that determining the Lipschitz constant of a NN is an NP-hard problem and proposes algorithms to estimate the Lipschitz constant. Methods for data-driven Lipschitz estimation for controller design and safe policy iteration in ADP are proposed in [142, 143]. In [144], the authors propose a method for training deep feedforward NNs with bounded Lipschitz constants. [145] poses the Lipschitz constant estimation problem for deep neural networks as a semidefinite program (SDP). [146] shows how to train continuous-time RNNs with constrained Lipschitz constants to guarantee stability. [147] develops neural Lipschitz observers with guaranteed performance.

c) Robustness and other safety properties: The authors in [148] use the data Jacobian matrix to analyze the behavior of deep neural networks by means of their singular values. [149] develops an analyzer for deep convolutional networks with ReLU activations based on an over-approximation that can guarantee robustness. [150] proposes a method of verification for binarized NNs using existing boolean satisfiability solvers, which is tested for adversarial robustness to l_∞ perturbations. [151] uses mixed-integer linear programming (MILP) formulations to analyze robustness of NNs.

2) Analysis and verification of learned control policies: These methods analyze and verify properties of learned control policies in the closed-loop setting. They are categorized by the considered class of properties as follows.

a) Stability: [69] provides deterministic stability guarantees for learning-based MPC (LBMPC) based on linear state space models. The authors in [152] propose sampling-based probabilistic performance guarantees for approximate MPC policies based on the Hoeffding inequality [153]. [154] uses semidefinite programming to certify stability for model-based RL policies. [155] uses Lyapunov stability verification to certify model-based RL policies. [156] uses GP to estimate the region of attraction of a closed-loop system. Recently, [157] uses sample-based δ-covers of system domains to ensure input-to-state stability to a set of a closed-loop system when an NN is used as a state estimator or a closed-loop control law. [158] develops a neural contraction metric, i.e., an NN approximation of a contraction metric, which, coupled with a learning-based controller, ensures exponential convergence to a target trajectory. [159] guarantees stability by combining the worst-case approximation error of the NN controller with its Lipschitz constant, utilizing an MILP framework. [160] proposes a framework for the stability verification of

MILP representable control policies. [161] simultaneously learns an NN controller and Lyapunov function, guided by MILP stability verification, which either verifies stability or gives counter-examples that can help improve the candidate controller and the Lyapunov function. [162] learns an MPC policy and a "dual policy," which enables them to keep a check on the approximated MPC's optimality online during the control process to filter out suboptimal control inputs and invoke a backup controller with a bounded probability.

b) Invariance and other safety properties: [163] uses constrained zonotopes and reachability analysis to determine reachable sets for safety verification of NN controllers. [81] uses sampled trajectories of a closed-loop system to learn the control barrier function, which can be parameterized by an NN and can ensure invariance of the safe set. [119] uses an MILP formulation to perform an output range analysis of a trained NN controller to guarantee constraint satisfaction and asymptotic stability of the NN controller.

3) Case study: Verification of a NN controller for a DC-DC power converter: The following section provides a case study on the verification of a NN controller for a DC-DC power converter. To carry out this verification, we employ the EVANQP framework, developed by [160], which utilizes MILPs as its underlying verification technique. By utilizing this framework, we not only demonstrate the closed-loop stability of the NN policy, but we can also validate the satisfaction of constraints in closed-loop operation. The example is adopted from [160, Section V].

The use of neural network (NN) policies in power converter applications is primarily motivated by the desire to achieve comparable performance to that of an optimal MPC policy while requiring lower computational complexity and memory. Optimal MPC policies are often too computationally demanding to be run in real-time on low-cost microprocessor hardware, making NN policies an attractive alternative. Despite the successful deployment of an NN policy on real hardware by [164], their approach lacks guarantees of stability and constraint satisfaction. In the following, we address these shortcomings. We will focus on the approach described in [160] where the baseline policy $\psi_1(\cdot)$ represented by a robust MPC is approximated by a NN policy $\psi_2(\cdot)$. Specifically, we consider the robust Tube MPC approach proposed by [165], which is robust against additive input disturbances in the set $\mathbb{W} = \{w \in \mathbb{R}^m \mid \|w\|_\infty \leq \hat{\gamma}\}$. By verifying that the NN policy has a worst-case approximation error,

$$\gamma = \max_{x \in \mathbb{X}} \|\psi_1(x) - \psi_2(x)\|_\infty, \tag{9}$$

over a bounded polytopic set $\mathbb{X}$ that is smaller than $\hat{\gamma}$, we can prove that the NN policy $\psi_2(\cdot)$ is asymptotically stable in closed-loop and satisfies constraints on the feasible region of the robust MPC policy $\psi_1(\cdot)$ [160, Theorem 1]. Notably, this is achieved by reformulating (9) as a MILP, with the NN policy and the optimal solution map of the robust MPC scheme as MILP constraints. A wide range of candidate policies including ReLU NNs, optimal solution maps of parametric QPs, and MPC policies can be exactly represented using MILP constraints. We refer to such functions as MILP-representable, as they can be expressed exactly using linear equality and inequality constraints with both continuous and binary decision variables. The MILP-representable functions are piecewise linear in nature, enabling their exact representation using MILP constraints.

The model of the DC-DC converter is linearized and discretized, giving us the following two-state $x = (i_L, v_O)$ (current and voltage), and one-input (duty cycle) linear system

$$x^+ = Ax + Bu = \begin{bmatrix} 0.971 & -0.010 \\ 1.732 & 0.970 \end{bmatrix} x + \begin{bmatrix} 0.149 \\ 0.181 \end{bmatrix} u.$$

For the robust MPC we assume the uncertain dynamics

$$x^+ = Ax + Bu + Bw,$$

with disturbance set $\mathbb{W} = \{w \in \mathbb{R} \mid |w| \leq 0.1\}$. We formulate the robust MPC controller

$$\begin{aligned} \min_{\mathbf{z},\mathbf{v}} \quad & \sum_{i=0}^{N-1} \left(\|z_i - x_{\text{eq}}\|_Q^2 + \|v_i - u_{\text{eq}}\|_R^2 \right) + \|z_N - x_{\text{eq}}\|_P^2 \\ \text{s.t.} \quad & \forall i = 0, \dots, N-1, \\ & z_{i+1} = Az_i + Bv_i, \\ & z_i \in \mathbb{X} \ominus \mathscr{E}, \quad v_i \in \mathbb{U} \ominus K\mathscr{E} \\ & z_N \in \mathbb{X}_N, \quad x(0) \in z_0 \oplus \mathscr{E}, \end{aligned}$$

with steady-state $x_{\text{eq}} = \begin{bmatrix} 0.05 & 5 \end{bmatrix}^T$, and $u_{\text{eq}} = 0.3379$, state constraints $\mathbb{X} = \left\{x \in \mathbb{R}^2 \mid 0 \leq x_1 \leq 0.2, 0 \leq x_2 \leq 7\right\}$, input constraints $\mathbb{U} = \{u \in \mathbb{R} \mid 0 \leq u \leq 1\}$, terminal set $\mathbb{X}_N$, $\mathscr{E}$ the minimum robust invariant set with respect to a linear feedback gain K. The control law is then given by $\psi_1(x) = K(x - z_0^\star(x)) + v_0^\star(x)$. To approximate the robust Tube MPC, we employ a neural network (NN) with 2 hidden layers and 50 neurons each. A saturation layer is added at the end to ensure that the input is clipped between -1 and 1. We use 5000 samples of the Tube MPC uniformly distributed in the feasible region for training the NN, following standard techniques. The NN controller learns an approximation of the MPC policy by minimizing a least-squares loss function. The resulting NN controller and the original Tube MPC can be visualized in Figure 3. We employ the EVANQP framework to describe the MPC formulation and NN controller and solve the resulting Mixed-Integer Linear Program (MILP) to determine the worst-case approximation error. Our analysis yields a value of $\gamma = 0.073$. As we designed our controller to be robust for input perturbations with a maximum magnitude of 0.1, our results demonstrate that the NN controller satisfies constraints and converges asymptotically to the steady-state.

D. Learning from physics-informed digital twin simulations

Advancements in computing and the wide availability of modeling toolkits have yielded high-fidelity simulation software (an essential component of so-called "digital twins" [166]) across a range of domains. Simulation has become a critical tool for researchers to perform experiments in a scalable, safe, and repeatable manner. Data generated from digital twins can complement, or offer a powerful

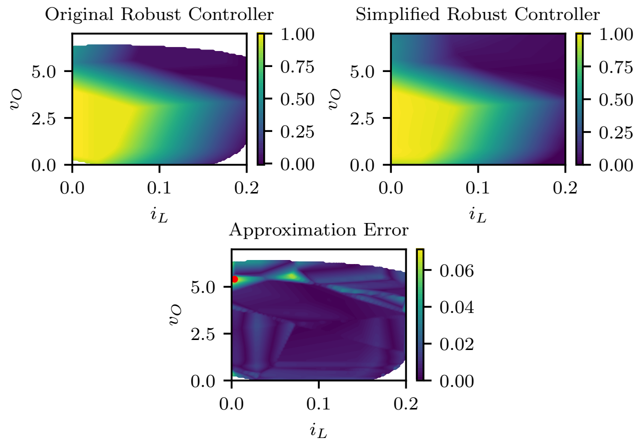

Fig. 3: Original, approximate/simplified robust control policy (top), and approximation error to the original robust MPC control policy (bottom). Taken from [160].

alternative to field experiments, which are typically time-consuming, and often require extensive guardrails to ensure expensive equipment/personnel are not subject to harm or are difficult to repeat. Some high-fidelity simulators include Modelica (cyber-physical systems [167], building energy systems [168]), CESM (climate models [169]), STK (spacecraft [170]), OpenFOAM (fluid flow [171]), and ChainQueen (soft robotics [172]), to name a few.

In the prior section on physics-informed system identification, data was assumed to have been obtained from the system under consideration, and an incomplete mathematical representation of the system dynamics is available from domain knowledge (or 'physics'). In this subsection, the 'physics' is integrated within the data via the data-generating source: the high-fidelity simulator. However, because the simulators comprise software modules that require high modeling complexity and are typically strongly interconnected to one another, directly considering analytical forms of these simulators is unwieldy and impractical. It is generally easier to construct machine learning tools that can directly use data generated by the simulators.

1) Controller design using simulation data: To ensure that a high-fidelity simulator can be effectively utilized for controller design tasks, it must first be calibrated with real-world data generated by the system of interest. This calibration step is important for ensuring the underlying assumptions and parameter values have been adequately selected to best mimic the true system. However, the complex representation of most high-fidelity simulators implies that the model calibration step can be challenging to solve [173, 174], especially compared to alternative black-box modeling approaches (e.g., deep neural networks) that do not require prior physical knowledge. A natural question arising from this comparison is: Why not directly work with standard machine learning algorithms and forego the high-fidelity simulator altogether? In the context of real-world engineering systems, the answer almost always boils down to a lack of accurate system data. For example, deep learning methodologies have become mainstream solutions in a variety of important applications, such as natural language processing and image classification; however, they often require massive amounts of high-quality labeled training data to surpass human performance. A key advantage of high-fidelity simulators is that they can be calibrated using substantially less training data, which is a direct consequence of the constraints imposed by the underlying first-principles models [175]. Once calibrated, the simulator can then be treated as a *data-generation source* that can be more confidently extrapolated outside of observed training instances. This data generation feature is useful for building control-relevant (or reduced-order) models [176, 177], which are critical components of modern model-based control algorithms, such as MPC, that can handle constrained nonlinear systems with strong multivariate interactions.

No matter the selected structure, a controller will still depend on design (or tuning) parameters that can strongly affect closed-loop performance and safety. Historically, these tuning parameters have been selected using a combination of heuristics and/or trial-and-error experimentation on the true system. To reduce the required testing and validation time, it has been recently proposed to perform controller tuning using the high-fidelity simulator. However, the tuning process is not straightforward due to the computationally expensive nature of high-fidelity simulators. As such, there has also been significant interest in the development of efficient data-driven automated calibration (or auto-tuning) strategies. In particular, Bayesian optimization (BO) has emerged as a powerful approach for handling these types of auto-tuning problems due to its ability to handle expensive black-box functions corrupted by random noise [178]–[180]. Several recent works have demonstrated the promise of BO for auto-tuning of MPC [181]–[184] and other complex control structures [185, 186]. Many interesting extensions of BO have also been pursued in the context of auto-tuning, including multi-objective [187] and robust [188, 189] formulations.

2) Improving generalization of learners via simulations: A unique opportunity afforded to us by the use of physics-informed simulation tools comes from the ability to *generate useful data* for analysis and design. This generation phase is typically not assumed in most current PIML research, where the more standard assumption is that data and domain knowledge pertaining to the 'target' system under consideration is available. Contrarily, digital twins comprise parameterized components whose physical parameters or physics-informed structure can be modified in software to generate data from multiple 'source' systems that are similar to the target system but not necessarily identical. The implication in modeling and control is that we can use leverage this multi-source dataset to evaluate performance on a range of similar dynamical systems and embed this data into the design pipeline for a target system from which only a few data points are available. Two classes of ML algorithms are naturally suited to learning from multi-source data: (i) meta-learning (also referred to as few-shot learning) and (ii) transfer learning.

a) Meta-learning: In meta-learning, two objective functions are typically used in the training phase: an outer-loop loss function for learning commonalities among the multi-source systems and an inner-loop loss function for quickly adapting to a new system with minimal data [190]. At inference, the outer-loop parameters are used to initialize the network, and a few inner-loop iterations are deemed sufficient for adaptation to the target system. Importantly, the learner structure does not change for the inference task, and the meta-learning algorithm learns to optimize the learning process itself; that is, for a classification problem, the meta-learner may not infer a classification output, but instead may infer a set of hyper-parameters for a classifier network such as loss function parameters or parameters relating to the neural architecture itself [191]. GP models have been used recently in order to meta-learn predictive models for MPC [192] by using dynamic trajectory data from similar systems, and neural networks have been used to meta-learn adaptive control policies in [193] for robotics. Some meta-learning algorithms do not require re-training or bilevel optimization for adaptation. Instead, they adapt based on contextualization; that is, with the same inputs, the inference changes due to contextual inputs additionally provided to the network. These context-based meta-learners, such as neural processes and deep kernel networks, have also been investigated recently for parameter learning [194, 195] using physics-informed simulators of building energy systems.

b) Transfer and multi-fidelity learning: Conversely, transfer learning relies on learning good representations from the multi-source dataset. At inference, the final network architecture is different from the network that was pre-trained, with the head of the network being inherited from the trained network, while the penultimate layers are altered to fit the learning task on the target system. Furthermore, the task performed by both the network architectures, e.g., classification, are identical. So far, the utility of transfer learning in modeling and control has mainly been demonstrated in energy systems [196, 197] where control policies or neural network-based surrogate thermal dynamics models are constructed by using simulations of buildings situated in various climate zones and exhibiting a wide range of architectures.

A generalized version of the transfer learning problem also arises in controller auto-tuning. Although auto-tuning strategies applied to the high-fidelity simulator can compensate for the error between the control-relevant model and the simulator, they implicitly assume the error between the simulator and the true system is negligible, which is not always the case in practice. One way to address this additional source of error is through the framework of *multi-fidelity optimization* wherein we assume access to a family of information sources controlled by a collection of "fidelity" parameters that can be generally continuous or discrete. In the simplest case, we would have one binary fidelity parameter that denotes two distinct but correlated levels, i.e., the simulator and the true system. Due to the general structure of the BO framework, several multi-fidelity BO methods have been developed in the literature [198]–[201], with some being applied directly to the controller auto-tuning problem [194, 202]. The main advantage of these types of multi-fidelity optimization methods is their ability to efficiently reconcile discrepancies between the high-fidelity simulator and the true system, which can allow for a significant reduction in the amount of testing required on the true system of interest.

3) Sim2Real in Reinforcement Learning: Model-free controllers, e.g. a class of (deep) reinforcement learning algorithms can potentially obtain near-optimal control policies without the need for a mathematical model. This is typically done by the control agent interacting with its environment (or plant) and learning to associate the states to advantageous control actions. The exploration-exploitation trade-off governs this interaction: in order for the agent to gather new knowledge, it needs to explore unknown effects of its actions by choosing them, e.g., randomly at times. This feature has three drawbacks for controlling real-world systems. For one, for large state-space systems, the agent may never see all possible states and/or all possible state-action combinations, especially those that may occur infrequently. Second, the amount of data needed for convergence may be large, and consequently the learning times prohibitively long. Third, the agent may choose actions that violate safety constraints, which may not be permissible in certain plants.

To overcome this issue the sim2real paradigm has been introduced, i.e., the notion that the agent is (pre-)trained in a physically accurate simulator before the deployment on the real system. This way, the agent can be provided with the equivalent of decades of experience and a large variety of potential states. The sim2real paradigm has been successfully explored by developing a variety of simulators for robotics [203, 204] and general purpose physics [205, 206], and demonstrating applications mainly in robotics [207]–[210] and automotive [1]. Of course, the physics fidelity of the simulators can determine the success and quality of the subsequent deployment, and also its applicability for sim2real. If it is computationally too expensive to run extensive simulation models, they are not suitable to be used in pre-training learning controllers. Here, PIML approaches can be very advantageous in providing physically accurate, yet computationally efficient environments, which can greatly improve the quality of the learning process.

E. Case Study: Violation-aware Bayesian optimization for energy consumption minimization of HVAC with constraints

We consider the problem of safely tuning set-points of an HVAC system as introduced in [211], also referred to as a vapor compression system (VCS). A VCS typically consists of a compressor, a condenser, an expansion valve, and an evaporator. Physics-based models of these systems can be formulated as large sets of nonlinear differential-algebraic equations (DAE) to simulate VCS dynamics. To inject realistic refrigerant dynamics and fluid flow, these models contain software blocks that exhibit significant numerical complexity. This motivates directly using data from VCS digital twin simulations to estimate energy consumption under different operating conditions, to assign set-points to the VCS actuators using data-driven, black-box optimization methods such as Bayesian optimization [178]. A high-fidelity digital twin of

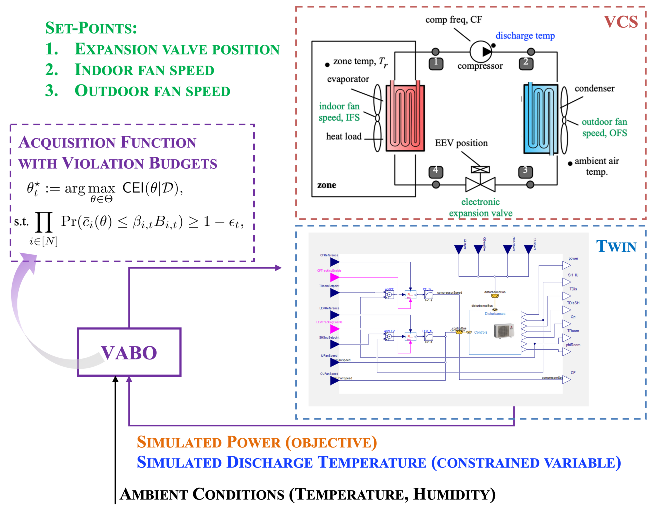
SET-POINTS:
1. EXPANSION VALVE POSITION
2. INDOOR FAN SPEED
3. OUTDOOR FAN SPEED
ACQUISITION FUNCTION WITH VIOLATION BUDGETS
$\theta_t^\star := \arg\max_{\theta\in\Theta} \mathsf{CEI}(\theta|\mathcal{D})$,
s.t. $\prod_{i\in[N]} \Pr(\bar{c}_i(\theta) \leq \beta_{i,t}B_{i,t}) \geq 1-\epsilon_t$,
VABO
VCS
comp freq, CF
discharge temp
compressor
zone temp, T_r
evaporator
indoor fan speed, IFS
heat load
EEV position
condenser
outdoor fan speed, OFS
ambient air temp.
zone
electronic expansion valve
TWIN
SIMULATED POWER (OBJECTIVE)
SIMULATED DISCHARGE TEMPERATURE (CONSTRAINED VARIABLE)
AMBIENT CONDITIONS (TEMPERATURE, HUMIDITY)

Fig. 4: Best feasible solution's power, discharge temperature, cumulative violation cost and the three set-points' evolution. Components of the figure have been taken from [211] and [212].

the VCS was constructed using Modelica [213]; see Fig. 4. Physics-based models of the compressor, expansion valve, accumulator, and both heat exchangers were interconnected. The final DAE model comprises 12114 equations; further modeling details can be found in [214].

Recklessly changing VCS actuator set-points can drive the system into operating modes that reduce the reliability or lifespan of the system. To avoid these harmful effects, one can add several constraints during the tuning process. One such constraint is the compressor discharge temperature; as high temperatures can result in the breakdown of refrigerant oils and increase wear and tear, shortening the product's lifespan. Furthermore, high temperatures are connected to high pressures, which may cause mechanical fatigue in refrigerant pipes. An advantage is that small constraint violations over a short period of time are acceptable. Authors in [211] proposed a violation-aware Bayesian optimization (VABO) to minimize energy use in the VCS while trading off constraint violations. The feedback loop is closed from compressor frequency to room temperature, leaving the set of three tunable set points as the expansion valve position and the indoor/outdoor fan speeds. The effects of these set points on power and discharge temperature are not easy to model, and no simple closed-form representation exists.

Authors in [211] report that the energy use induced by VABO decreases slightly faster than generic constrained BO [215] (cBO) and significantly faster than safe BO [216]. At the same time, the method manages the violation cost well under the violation cost budget. cBO incurs large discharge temperatures at many iterations because it makes large adjustments to the expansion valve position while maintaining the indoor fan speed at a low value, a combination that is not penalized during exploration. VABO reduces the energy by about 9% compared to the most power-efficient initial safe set-points. Authors also observed that large discharge temperature violations are entirely possible without violation awareness, as demonstrated by cBO. Safe BO tends to waste a lot of evaluations to enlarge the safe set, which leads to slow convergence; conversely, VABO implicitly encodes the safe set exploration into the acquisition function and only enlarges the safe set when necessary for optimization, while keeping the violation risk small.

III. CHALLENGES AND OPPORTUNITIES OF PIML FOR CONTROL

This section reflects on open challenges and opportunities of PIML methods from various perspectives, including data and prior knowledge requirements, computational demands, safety and performance guarantees, availability of software tools and learning materials, as well as new promising application domains. Opportunities of PIML methods in control include:

1) Modeling of human behaviors in interactive human-autonomous systems (e.g., autonomous vehicle in mixed traffic). Autonomous system physics is known, but human reaction needs to be learned from real data to model human-in-the-loop systems at scale.
2) Modeling of high dimensional and distributed physics in multi-physics systems. An example includes combustion processes where physics is impossible to model compactly by first principles, but there may be good surrogate models that retain core physical properties.
3) Generation of structured PIML controllers for multi-component systems with awareness of interconnections of the sub-components. This would lead to improved interpretability and allow for the localization of failures.
4) Constructing surrogate PIML models for the hard-to-optimize physics-based optimization problems or cost functions. This could significantly speed up the solution of hard optimization problems using cheaper surrogates.
5) Synthesis of explicit model predictive control policies for large-scale systems leading to a significant reduction in online computational requirements. Thus allowing for the execution of complex control policies in applications with limited communication bandwidths and small sampling rates or enabling deployment on edge devices.
6) Providing safety and performance guarantees for a broad class of learning-based control methods. Particularly applicable to applications with time-varying dynamics that require online adaptive processes to cope with constant changes in a safe manner.
7) Dealing with sim2real gap and allowing for automated tuning of PIML controllers from simulation and experimental data.
8) Integration of multi-modal inputs into modern control systems. Examples include a combination of video and audio signals with physical measurements processed for downstream decision-making by advanced controls.

Open challenges for PIML methods in control include:

1) How to quantify the uncertainty and modeling errors for PIML-based models?

2) How to quantify minimal data requirements for training PIML models and controllers?
3) How to effectively select representative training data for sampling-based PIML approaches?
4) How to automate the training and hyperparameter tuning process of PIML models?
5) How to avoid training failures of PIML models getting stuck in local optima. Can we obtain convergence guarantees for certain classes of PIML models?
6) How to guarantee stability and safety of a real-world system in closed-loop with PIML-based controllers in the presence of noise and plant-model mismatch?
7) How can verification methods for PIML be scaled up for large-scale or networked systems?
8) How to reduce the computational requirements of high-fidelity digital twins without sacrificing accuracy?

IV. Conclusions

In the last two decades, the use of machine learning (ML) methods revolutionized a range of industries, from retail, advertisement, entertainment, healthcare, finance, digital arts, and social networks, to surveillance. Although highly diverse, these applications have some common denominators, which is that their ML systems are primarily designed for pattern recognition from multi-modal data sources. However, as we move towards real-world engineering systems with humans-in-the-loop, such as autonomous vehicles, collaborative robotics, process control of chemical plants, or power grid, the primary focus is steered toward optimization and control of these dynamical systems with guarantees of safe operation. In recent years, physics-informed machine learning (PIML) has emerged as a class of methods that systematically combine data-driven ML with physics-based modeling and numerical solvers from engineering.

This tutorial paper provides an overview of the most recent PIML methods applied to the modeling and control of dynamical systems. Specifically, PIML techniques for system identification include structural priors in the architecture of the ML model, matrix factorizations, and physics-informed loss functions. PIML approaches to control cover learning dynamics models for model predictive control (MPC), learning Lyapunov and barrier functions, differentiable-programming-based control, safe data-driven control, and physics-informed reinforcement learning (RL). Obtaining safety and performance guarantees of PIML models and control policies requires rigorous tools verifying properties such as stability, robustness, Lipschitz properties, invariance, and other safety considerations such as constraint satisfaction. The paper also presents methods combining ML with high-fidelity digital twin models suitable for controller design and tuning via meta and transfer learning and dealing with sim2real gap in RL. Each major category of PIML methods is accompanied by a representative tutorial case study.

The use of PIML methods in control presents us with new exciting opportunities that include applications in systems with human-in-the-loop, multi-scale and multi-physics systems, and providing benefits such as improved interpretability and modularity, scalability to large-scale complex systems, safety guarantees for adaptive data-driven systems, or integration of multi-modal input data in the control systems.

However, several open questions remain that need to be addressed before the adoption of these methods in real-world applications. These include uncertainty quantification (UQ), data requirements and efficient sampling strategies, automated training, hyperparameter optimization, convergence guarantees, scalability of the verification methods, and computational requirements of high-fidelity physics simulators.

Acknowledgements

PNNL authors are supported by the U.S. Department of Energy, through the Energy Efficiency and Renewable Energy, Building Technologies Office under the "Advancing Market-Ready Building Energy Management by Cost-Effective Differentiable Predictive Control" project. PNNL is a multi-program national laboratory operated for the U.S. Department of Energy (DOE) by Battelle Memorial Institute under Contract No. DE-AC05-76RL0-1830. T.X.N. is supported by the National Science Foundation under Grant No. 2138388 and Grant No. 2238296. EPFL authors are supported by the Swiss National Science Foundation under the NCCR Automation project, grant agreement 51NF40_180545.

References

[1] B. Balaji, S. Mallya, S. Genc, S. Gupta, L. Dirac, V. Khare, G. Roy, T. Sun, Y. Tao, B. Townsend, E. Calleja, S. Muralidhara, and D. Karuppasamy, "Deepracer: Autonomous racing platform for experimentation with sim2real reinforcement learning," in *ICRA 2020*, 2020.
[2] S. L. Brunton, J. Nathan Kutz, K. Manohar, A. Y. Aravkin, K. Morgansen, J. Klemisch, N. Goebel, J. Buttrick, J. Poskin, A. W. Blom-Schieber, T. Hogan, and D. McDonald, "Data-driven aerospace engineering: Reframing the industry with machine learning," *AIAA Journal*, vol. 59, no. 8, pp. 2820–2847, 2021.
[3] R. Nian, J. Liu, and B. Huang, "A review on reinforcement learning: Introduction and applications in industrial process control," *Computers & Chemical Engineering*, vol. 139, p. 106886, 2020.
[4] P. L. Donti and J. Z. Kolter, "Machine learning for sustainable energy systems," *Annual Review of Environment and Resources*, vol. 46, no. 1, pp. 719–747, 2021.
[5] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, "Solving rubik's cube with a robot hand," *CoRR*, vol. abs/1910.07113, 2019.
[6] G. Dulac-Arnold, D. J. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," *CoRR*, vol. abs/1904.12901, 2019.
[7] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, Jun. 2021.
[8] R. Wang, "Physics-guided deep learning for dynamical systems: A survey," *CoRR*, vol. abs/2107.01272, 2021.
[9] C. Legaard, T. Schranz, G. Schweiger, J. Drgoňa, B. Falay, C. Gomes, A. Iosifidis, M. Abkar, and P. Larsen, "Constructing neural network based models for simulating dynamical systems," *ACM Comput. Surv.*, vol. 55, no. 11, feb 2023.
[10] C. Rackauckas, A. Edelman, K. Fischer, M. Innes, E. Saba, V. B. Shah, and W. Tebbutt, "Generalized physics-informed learning through language-wide differentiable programming," in *AAAI Spring Symposium: MLPS*, 2020.
[11] R. G. Krishnan, U. Shalit, and D. Sontag, "Structured inference networks for nonlinear state space models," in *AAAI*, 2017.
[12] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski, "Deep state space models for time series forecasting," in *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., 2018, pp. 7785–7794.

[13] E. Yeung, S. Kundu, and N. Hodas, “Learning deep neural network representations for Koopman operators of nonlinear dynamical systems,” in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 4832–4839.

[14] B. Lusch, J. N. Kutz, and S. L. Brunton, “Deep learning for universal linear embeddings of nonlinear dynamics,” *Nature Communications*, vol. 9, no. 1, p. 4950, Nov. 2018.

[15] Jeen-Shing Wang and Yi-Chung Chen, “A hammerstein-wiener recurrent neural network with universal approximation capability,” in *2008 IEEE International Conference on Systems, Man and Cybernetics*, Oct 2008, pp. 1832–1837.

[16] O. P. Ogunmolu, X. Gu, S. B. Jiang, and N. R. Gans, “Nonlinear systems identification using deep dynamic neural networks,” *CoRR*, vol. abs/1610.01439, 2016.

[17] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. A. Riedmiller, R. Hadsell, and P. W. Battaglia, “Graph networks as learnable physics engines for inference and control,” *CoRR*, vol. abs/1806.01242, 2018.

[18] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, “Neural relational inference for interacting systems,” in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 10–15 Jul 2018, pp. 2688–2697.

[19] S. L. Brunton, J. L. Proctor, and J. N. Kutz, “Discovering governing equations from data by sparse identification of nonlinear dynamical systems,” *Proceedings of the National Academy of Sciences*, vol. 113, no. 15, pp. 3932–3937, mar 2016.

[20] ——, “Sparse identification of nonlinear dynamics with control (sindyc),” *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 710–715, 2016, 10th IFAC Symposium on Nonlinear Control Systems NOLCOS 2016.

[21] S. Vijayshankar, A. Chakrabarty, P. Grover, and S. Nabi, “Co-design of reduced-order models and observers from thermo-fluid data,” *IFAC Journal of Systems and Control*, vol. 19, p. 100181, 2022.

[22] Z. Zhang, Y. Zhao, J. Liu, S. Wang, R. Tao, R. Xin, and J. Zhang, “A general deep learning framework for network reconstruction and dynamics learning,” *Applied Network Science*, vol. 4, no. 1, nov 2019.

[23] L. Di Natale, B. Svetozarevic, P. Heer, and C. Jones, “Physically consistent neural networks for building thermal modeling: Theory and analysis,” *Applied Energy*, vol. 325, p. 119806, 2022.

[24] M. Chen and C. J. Tomlin, “Hamilton–jacobi reachability: Some recent theoretical advances and applications in unmanned airspace management,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 333–358, 2018.

[25] A. Quaglino, M. Gallieri, J. Masci, and J. Koutník, “SNODE: Spectral discretization of neural ODEs for system identification,” 2020.

[26] S. Massaroli, M. Poli, S. Sonoda, T. Suzuki, J. Park, A. Yamashita, and H. Asama, “Differentiable multiple shooting layers,” *CoRR*, vol. abs/2106.03885, 2021.

[27] E. M. Turan and J. Jäschke, “Multiple shooting for training neural differential equations on time series,” *IEEE Control Systems Letters*, vol. 6, pp. 1897–1902, 2022.

[28] A. Schlaginhaufen, P. Wenk, A. Krause, and F. Dörfler, “Learning stable deep dynamics models for partially observed or delayed dynamical systems,” *CoRR*, vol. abs/2110.14296, 2021.

[29] R. Dandekar, V. Dixit, M. Tarek, A. Garcia-Valadez, and C. Rackauckas, “Bayesian neural ordinary differential equations,” *CoRR*, vol. abs/2012.07244, 2020.

[30] P. Kidger, J. Foster, X. Li, H. Oberhauser, and T. J. Lyons, “Neural sdes as infinite-dimensional gans,” *CoRR*, vol. abs/2102.03657, 2021.

[31] J. Jia and A. R. Benson, “Neural jump stochastic differential equations,” *CoRR*, vol. abs/1905.10403, 2019.

[32] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, and A. Edelman, “Universal differential equations for scientific machine learning,” Jan. 2020.

[33] J. Koch, Z. Chen, A. Tuor, J. Drgona, and D. Vrabie, “Structural inference of networked dynamical systems with universal differential equations,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 33, no. 2, p. 023103, 2023.

[34] L.-P. A. C. Xhonneux, M. Qu, and J. Tang, “Continuous graph neural networks,” in *Proceedings of the 37th International Conference on Machine Learning*, ser. ICML’20. JMLR.org, 2020.

[35] M. Poli, S. Massaroli, J. Park, A. Yamashita, H. Asama, and J. Park, “Graph neural ordinary differential equations,” *CoRR*, vol. abs/1911.07532, 2019.

[36] S. Greydanus, M. Dzamba, and J. Yosinski, “Hamiltonian neural networks,” *CoRR*, vol. abs/1906.01563, 2019.

[37] Y. D. Zhong, B. Dey, and A. Chakraborty, “Symplectic ode-net: Learning hamiltonian dynamics with control,” in *International Conference on Learning Representations*, 2020.

[38] M. Lutter, C. Ritter, and J. Peters, “Deep lagrangian networks: Using physics as model prior for deep learning,” *CoRR*, vol. abs/1907.04490, 2019.

[39] M. D. Cranmer, S. Greydanus, S. Hoyer, P. W. Battaglia, D. N. Spergel, and S. Ho, “Lagrangian neural networks,” *CoRR*, vol. abs/2003.04630, 2020.

[40] A. Sanchez-Gonzalez, V. Bapst, K. Cranmer, and P. Battaglia, “Hamiltonian graph networks with ode integrators,” 2019.

[41] Y. D. Zhong, B. Dey, and A. Chakraborty, “Dissipative symODEN: Encoding hamiltonian dynamics with dissipation and control into deep learning,” in *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2019.

[42] M. Finzi, K. A. Wang, and A. G. Wilson, “Simplifying hamiltonian and lagrangian neural networks via explicit constraints,” *CoRR*, vol. abs/2010.13581, 2020.

[43] S. Eidnes, A. J. Stasik, C. Sterud, E. Bøhn, and S. Riemer-Sørensen, “Pseudo-hamiltonian neural networks with state-dependent external forces,” *Physica D: Nonlinear Phenomena*, vol. 446, p. 133673, 2023.

[44] Y. D. Zhong, B. Dey, and A. Chakraborty, “Extending lagrangian and hamiltonian neural networks with differentiable contact models,” in *Advances in Neural Information Processing Systems*, 2021.

[45] Y. D. Zhong and N. Leonard, “Unsupervised learning of lagrangian dynamics from images for prediction and control,” in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 10 741–10 752.

[46] G. Manek and J. Z. Kolter, “Learning stable deep dynamics models,” *CoRR*, vol. abs/2001.06116, 2020.

[47] J. Drgona, A. Tuor, S. Vasisht, and D. Vrabie, “Dissipative deep neural dynamical systems,” *IEEE Open Journal of Control Systems*, vol. 1, pp. 100–112, 2022.

[48] A. Tuor, J. Drgona, and D. Vrabie, “Constrained neural ordinary differential equations with stability guarantees,” *arXiv preprint arXiv:2004.10883*, 2020.

[49] Z. Mhammedi, A. Hellicar, A. Rahman, and J. Bailey, “Efficient orthogonal parametrisation of recurrent neural networks using householder reflections,” in *International Conference on Machine Learning*, 2017, pp. 2401–2409.

[50] K. Jia, S. Li, Y. Wen, T. Liu, and D. Tao, “Orthogonal deep neural networks,” *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[51] J. Zhang, Q. Lei, and I. S. Dhillon, “Stabilizing gradients for deep neural networks via efficient SVD parameterization,” *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[52] E. Haber and L. Ruthotto, “Stable architectures for deep neural networks,” *Inverse Problems*, vol. 34, no. 1, p. 014004, 2017.

[53] B. Chang, M. Chen, E. Haber, and E. H. Chi, “AntisymmetricRNN: A dynamical system view on recurrent neural networks,” in *International Conference on Learning Representations*, 2019.

[54] M. Lechner, R. Hasani, D. Rus, and R. Grosu, “Gershgorin loss stabilizes the recurrent neural network compartment of an end-to-end robot learning scheme,” in *2020 International Conference on Robotics and Automation (ICRA). IEEE*, 2020.

[55] G. Kerg, K. Goyette, M. P. Touzel, G. Gidel, E. Vorontsov, Y. Bengio, and G. Lajoie, “Non-normal recurrent neural network (nnRNN): learning long time dependencies while improving expressivity with transient dynamics,” in *Advances in Neural Information Processing Systems*, 2019, pp. 13 613–13 623.

[56] E. Skomski, S. Vasisht, C. Wight, A. Tuor, J. Drgoňa, and D. Vrabie, “Constrained block nonlinear neural dynamical models,” in *2021 American Control Conference (ACC)*, 2021, pp. 3993–4000.

[57] J. Drgoňa, A. R. Tuor, V. Chandan, and D. L. Vrabie, “Physics-constrained deep learning of multi-zone building thermal dynamics,” *Energy and Buildings*, vol. 243, p. 110992, 2021.

[58] N. B. Erichson, M. Muehlebach, and M. W. Mahoney, “Physics-informed Autoencoders for Lyapunov-stable Fluid Flow Prediction,” May 2019.

[59] G. Shi, X. Shi, M. O’Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, “Neural lander: Stable drone landing control using learned dynamics,” in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 9784–9790.

[60] S. Seo and Y. Liu, "Differentiable physics-informed graph networks," *CoRR*, vol. abs/1902.02950, 2019.

[61] A. Pal, Y. Ma, V. Shah, and C. V. Rackauckas, "Opening the blackbox: Accelerating neural differential equations by regularizing internal solver heuristics," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 139. PMLR, 18–24 Jul 2021, pp. 8325–8335.

[62] I. D. J. Rodriguez, A. D. Ames, and Y. Yue, "Lyanet: A lyapunov framework for training neural odes," *CoRR*, vol. abs/2202.02526, 2022.

[63] C. Finlay, J.-H. Jacobsen, L. Nurbekyan, and A. M. Oberman, "How to train your neural ODE: The world of Jacobian and kinetic regularization," 2020.

[64] J. Kelly, J. Bettencourt, M. J. Johnson, and D. Duvenaud, "Learning differential equations that are easy to solve," 2020.

[65] L. Hewing, K. 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, no. 1, pp. 269–296, 2020.

[66] I. Lenz, R. A. Knepper, and A. Saxena, "Deepmpc: Learning deep latent features for model predictive control," in *Robotics: Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015*, 2015.

[67] J. Nicodemus, J. Kneifl, J. Fehr, and B. Unger, "Physics-informed neural networks-based model predictive control for multi-link manipulators," 2021.

[68] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Sparse identification of nonlinear dynamics for model predictive control in the low-data limit," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 474, no. 2219, p. 20180335, 2018.

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

[70] P. Bouffard, A. Aswani, and C. Tomlin, "Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 279–284.

[71] A. Carron, E. Arcari, M. Wermelinger, L. Hewing, M. Hutter, and M. N. Zeilinger, "Data-driven model predictive control for trajectory tracking with a robotic arm," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3758–3765, 2019.

[72] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using gaussian process regression," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2736–2743, 2020.

[73] E. T. Maddalena, P. Scharnhorst, Y. Jiang, and C. N. Jones, "KPC: Learning-based model predictive control with deterministic guarantees," in *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, ser. Proceedings of Machine Learning Research, vol. 144. PMLR, 07 – 08 June 2021, pp. 1015–1026.

[74] F. Bünning, B. Huber, A. Schalbetter, A. Aboudonia, M. Hudoba de Badyn, P. Heer, R. S. Smith, and J. Lygeros, "Physics-informed linear regression is competitive with two machine learning methods in residential building mpc," *Applied Energy*, vol. 310, p. 118491, 2022.

[75] Y. Long and M. Bayoumi, "Feedback stabilization: Control lyapunov functions modelled by neural networks," in *Proceedings of 32nd IEEE Conference on Decision and Control*. IEEE, 1993, pp. 2812–2814.

[76] S. M. Richards, F. Berkenkamp, and A. Krause, "The Lyapunov neural network: Adaptive stability certification for safe learning of dynamic systems," *CoRR*, vol. abs/1808.00924, 2018.

[77] M. Mittal, M. Gallieri, A. Quaglino, S. S. M. Salehian, and J. Koutník, "Neural lyapunov model predictive control: Learning safe global controllers from sub-optimal examples," *arXiv preprint arXiv:2002.10451*, 2020.

[78] S. Mukherjee, J. Drgoňa, A. Tuor, M. Halappanavar, and D. Vrabie, "Neural lyapunov differentiable predictive control," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 2097–2104.

[79] L. Grüne, "Computing Lyapunov functions using deep neural networks," *Journal of Computational Dynamics*, vol. 8, no. 2, p. 131, 2021.

[80] K. Long, C. Qian, J. Cortés, and N. Atanasov, "Learning barrier functions with memory for robust safe navigation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4931–4938, 2021.

[81] A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni, "Learning control barrier functions from expert demonstrations," in *2020 59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 3717–3724.

[82] Z. Qin, K. Zhang, Y. Chen, J. Chen, and C. Fan, "Learning safe multi-agent control with decentralized neural barrier certificates," in *International Conference on Learning Representations*, 2021.

[83] C. Dawson, Z. Qin, S. Gao, and C. Fan, "Safe nonlinear control using robust neural lyapunov-barrier functions," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 164. PMLR, 08–11 Nov 2022, pp. 1724–1735.

[84] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable mpc for end-to-end planning and control," in *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc., 2018.

[85] M. Okada, L. Rigazio, and T. Aoshima, "Path integral networks: End-to-end differentiable optimal control," *CoRR*, vol. abs/1706.09597, 2017.

[86] M. Pereira, D. D. Fan, G. N. An, and E. A. Theodorou, "Mpc-inspired neural network policies for sequential decision making," *CoRR*, vol. abs/1802.05803, 2018.

[87] M. Innes, A. Edelman, K. Fischer, C. Rackauckas, E. Saba, V. B. Shah, and W. Tebbutt, "A differentiable programming system to bridge machine learning and scientific computing," *CoRR*, vol. abs/1907.07587, 2019.

[88] S. East, M. Gallieri, J. Masci, J. Koutník, and M. Cannon, "Infinite-horizon differentiable model predictive control," *ArXiv*, vol. abs/2001.02244, 2020.

[89] J. Drgoňa, K. Kiš, A. Tuor, D. Vrabie, and M. Klaučo, "Differentiable predictive control: Deep learning alternative to explicit model predictive control for unknown nonlinear systems," *Journal of Process Control*, vol. 116, pp. 80–92, 2022.

[90] J. Drgoňa, S. Mukherjee, A. Tuor, M. Halappanavar, and D. Vrabie, "Learning stochastic parametric differentiable predictive control policies," 2022.

[91] D. Tabas and B. Zhang, "Safe and efficient model predictive control using neural networks: An interior point approach," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 1142–1147.

[92] P. L. Donti, M. Roderick, M. Fazlyab, and J. Z. Kolter, "Enforcing robust control guarantees within neural network policies," *CoRR*, vol. abs/2011.08105, 2020.

[93] W. Xiao, T.-H. Wang, M. Chahine, A. Amini, R. Hasani, and D. Rus, "Differentiable control barrier functions for vision-based end-to-end autonomous driving," 2022.

[94] S. Yang, S. Chen, V. M. Preciado, and R. Mangharam, "Differentiable safe controller design through control barrier functions," *IEEE Control Systems Letters*, vol. 7, pp. 1207–1212, 2023.

[95] P. Karkus, B. Ivanovic, S. Mannor, and M. Pavone, "Diffstack: A differentiable and modular control stack for autonomous vehicles," in *6th Annual Conference on Robot Learning*, 2022.

[96] J. Xu, T. Chen, L. Zlokapa, M. Foshey, W. Matusik, S. Sueda, and P. Agrawal, "An End-to-End Differentiable Framework for Contact-Aware Robot Design," in *Proceedings of Robotics: Science and Systems*, Virtual, July 2021.

[97] J. Degrave, M. Hermans, J. Dambre, and F. Wyffels, "A differentiable physics engine for deep learning in robotics," *Frontiers in Neurorobotics*, vol. 13, 2016.

[98] B. Chen, Z. Cai, and M. Bergés, "Gnu-rl: A precocial reinforcement learning solution for building hvac control using a differentiable mpc policy," in *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, ser. BuildSys '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 316–325.

[99] J. Drgoňa, A. Tuor, E. Skomski, S. Vasisht, and D. Vrabie, "Deep learning explicit differentiable predictive control laws for buildings," *IFAC-PapersOnLine*, vol. 54, no. 6, pp. 14–19, 2021, 7th IFAC Conference on Nonlinear Model Predictive Control NMPC 2021.

[100] S. Son, Y.-L. Qiao, J. Sewall, and M. C. Lin, "Differentiable hybrid traffic simulation," *ACM Transactions on Graphics (TOG)*, 2022.

[101] T. Asikis, L. Böttcher, and N. Antulov-Fantulin, "Neural ordinary differential equation control of dynamics on graphs," *Phys. Rev. Res.*, vol. 4, p. 013221, Mar 2022.

[102] K. M. Jatavallabhula, M. Macklin, F. Golemo, V. Voleti, L. Petrini, M. Weiss, B. Considine, J. Parent-Levesque, K. Xie, K. Erleben, L. Paull, F. Shkurti, D. Nowrouzezahrai, and S. Fidler, "gradsim: Differentiable simulation for system identification and visuomotor control," *International Conference on Learning Representations*, 2021.

[103] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and Z. Kolter, "Differentiable convex optimization layers," in *Advances in Neural Information Processing Systems*, 2019.

[104] P. Gradu, J. Hallman, D. Suo, A. Yu, N. Agarwal, U. Ghai, K. Singh, C. Zhang, A. Majumdar, and E. Hazan, "Deluca–a differentiable control library: Environments, methods, and benchmarking," *Differentiable Computer Vision, Graphics, and Physics in Machine Learning (Neurips 2020 Workshop)*, 2020.

[105] L. Pineda, T. Fan, M. Monge, S. Venkataraman, P. Sodhi, R. T. Chen, J. Ortiz, D. DeTone, A. Wang, S. Anderson, J. Dong, B. Amos, and M. Mukadam, "Theseus: A Library for Differentiable Nonlinear Optimization," *Advances in Neural Information Processing Systems*, 2022.

[106] A. Tuor, J. Drgona, M. Skomski, J. Koch, Z. Chen, S. Dernbach, C. M. Legaard, and D. Vrabie, "NeuroMANCER: Neural Modules with Adaptive Nonlinear Constraints and Efficient Regularizations," 2022.

[107] K. Um, R. Brand, Y. R. Fei, P. Holl, and N. Thuerey, "Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS'20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[108] P. Holl, N. Thuerey, and V. Koltun, "Learning to control pdes with differentiable physics," in *International Conference on Learning Representations*, 2020.

[109] K. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, p. 109597, 2021.

[110] B. Tearle, K. Wabersich, A. Carron, and M. N. Zeilinger, "A predictive safety filter for learning-based racing control," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7635–7642, 2021.

[111] A. P. Vinod, S. Safaoui, A. Chakrabarty, R. Quirynen, N. Yoshikawa, and S. Di Cairano, "Safe multi-agent motion planning via filtered reinforcement learning," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 7270–7276.

[112] T. Gurriet, M. Mote, A. D. Ames, and E. Feron, "An online approach to active set invariance," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 3592–3599.

[113] T. Mannucci, E.-J. van Kampen, C. de Visser, and Q. Chu, "Safe exploration algorithms for reinforcement learning controllers," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 4, pp. 1069–1081, 2018.

[114] K. Wabersich and M. N. Zeilinger, "Linear model predictive safety certification for learning-based control," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 7130–7135.

[115] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*, 2019, pp. 3420–3431.

[116] W. S. Cortez, J. Drgona, A. Tuor, M. Halappanavar, and D. Vrabie, "Differentiable predictive control with safety guarantees: A control barrier function approach," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 932–938.

[117] A. Chakrabarty, C. Danielson, S. D. Cairano, and A. Raghunathan, "Active learning for estimating reachable sets for systems with unknown dynamics," *IEEE Transactions on Cybernetics*, vol. 52, no. 4, pp. 2531–2542, 2022.

[118] K. Wabersich and M. N. Zeilinger, "Predictive control barrier functions: Enhanced safety mechanisms for learning-based control," *IEEE Transactions on Automatic Control*, pp. 1–1, 2022.

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

[120] P. L. Donti, M. Roderick, M. Fazlyab, and J. Z. Kolter, "Enforcing robust control guarantees within neural network policies," in *International Conference on Learning Representations*, 2021, pp. 1–26.

[121] L. Di Natale, B. Svetozarevic, P. Heer, and C. N. Jones, "Near-optimal deep reinforcement learning policies from data for zone temperature control," 2022.

[122] M. P. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ser. ICML'11. Madison, WI, USA: Omnipress, 2011, p. 465–472.

[123] P. Zhao and Y. Liu, "Physics informed deep reinforcement learning for aircraft conflict resolution," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 8288–8301, 2022.

[124] Y. Du, Q. Huang, R. Huang, T. Yin, J. Tan, W. Yu, and X. Li, "Physics-informed evolutionary strategy based control for mitigating delayed voltage recovery," *IEEE Transactions on Power Systems*, vol. 37, no. 5, pp. 3516–3527, 2022.

[125] W. Cai, H. N. Esfahani, A. B. Kordabad, and S. Gros, "Optimal management of the peak power penalty for smart grids using mpc-based reinforcement learning," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 6365–6370.

[126] S. Gros and M. Zanon, "Data-driven economic nmpc using reinforcement learning," *IEEE Transactions on Automatic Control*, vol. 65, no. 2, pp. 636–648, 2020.

[127] M. Zanon and S. Gros, "Safe reinforcement learning using robust mpc," *IEEE Transactions on Automatic Control*, vol. 66, no. 8, pp. 3638–3652, 2021.

[128] 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.

[129] H. Zhang, Z. Wang, and D. Liu, "A comprehensive review of stability analysis of continuous-time recurrent neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 7, 2014.

[130] R. Engelken, F. Wolf, and L. Abbott, "Lyapunov spectra of chaotic recurrent neural networks," *arXiv preprint arXiv:2006.02427*, 2020.

[131] R. Vogt, M. Puelma Touzel, E. Shlizerman, and G. Lajoie, "On lyapunov exponents for rnns: Understanding information propagation using dynamical systems tools," *Frontiers in Applied Mathematics and Statistics*, vol. 8, 2022.

[132] B. Güler, A. Laignelet, and P. Parpas, "Towards robust and stable deep learning algorithms for forward backward stochastic differential equations," *33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada*, 2019.

[133] A. Abate, D. Ahmed, M. Giacobbe, and A. Peruffo, "Formal synthesis of lyapunov neural networks," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 773–778, 2021.

[134] S. Chen, M. Fazlyab, M. Morari, G. J. Pappas, and V. M. Preciado, "Learning region of attraction for nonlinear systems," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 6477–6484.

[135] S. A. Deka, A. M. Valle, and C. J. Tomlin, "Koopman-based neural lyapunov functions for general attractors," 2022.

[136] A. Lederer and S. Hirche, "Local asymptotic stability analysis and region of attraction estimation with gaussian processes," in *IEEE Conference on Decision and Control*, 2019, pp. 1766–1771.

[137] F. Bonassi, E. Terzi, M. Farina, and R. Scattolini, "Lstm neural networks: Input to state stability and probabilistic safety verification," in *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, ser. Proceedings of Machine Learning Research, vol. 120. PMLR, 10–11 Jun 2020, pp. 85–94.

[138] M. Revay and I. Manchester, "Contracting implicit recurrent neural networks: Stable models with improved trainability," in *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, ser. Proceedings of Machine Learning Research, vol. 120. PMLR, 10–11 Jun 2020, pp. 393–403.

[139] M. Revay, R. Wang, and I. R. Manchester, "A convex parameterization of robust recurrent neural networks," *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1363–1368, 2021.

[140] S. Jafarpour, A. Davydov, A. Proskurnikov, and F. Bullo, "Robust implicit networks via non-Euclidean contractions," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 9857–9868.

[141] K. Scaman and A. Virmaux, "Lipschitz regularity of deep neural networks: Analysis and efficient estimation," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 3839–3848.

[142] A. Chakrabarty, D. K. Jha, and Y. Wang, "Data-driven control policies for partially known systems via kernelized lipschitz learning," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 4192–4197.

[143] A. Chakrabarty, D. K. Jha, G. T. Buzzard, Y. Wang, and K. G. Vamvoudakis, "Safe approximate dynamic programming via kernelized lipschitz estimation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 405–419, 2020.

[144] P. Pauli, A. Koch, J. Berberich, P. Kohler, and F. Allgöwer, "Training robust neural networks using lipschitz bounds," *IEEE Control Systems Letters*, vol. 6, pp. 121–126, 2022.

[145] M. Fazlyab, M. Morari, and G. J. Pappas, "Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming," *IEEE Transactions on Automatic Control*, vol. 67, no. 1, pp. 1–15, 2022.

[146] N. B. Erichson, O. Azencot, A. Queiruga, L. Hodgkinson, and M. W. Mahoney, "Lipschitz recurrent neural networks," in *International Conference on Learning Representations*, 2021.

[147] A. Chakrabarty, A. Zemouche, R. Rajamani, and M. Benosman, "Robust data-driven neuro-adaptive observers with lipschitz activation functions," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 2862–2867.

[148] S. Wang, A. rahman Mohamed, R. Caruana, J. A. Bilmes, M. Philipose, M. Richardson, K. Geras, G. Urban, and Özlem Aslan, "Analysis of deep neural networks with extended data jacobian matrix," in *ICML*, 2016, pp. 718–726.

[149] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev, "Ai2: Safety and robustness certification of neural networks with abstract interpretation," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 3–18.

[150] K. Jia and M. Rinard, "Efficient Exact Verification of Binarized Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1782–1795.

[151] V. Tjeng, K. Y. Xiao, and R. Tedrake, "Evaluating robustness of neural networks with mixed integer programming," in *International Conference on Learning Representations*, 2019.

[152] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer, "Learning an approximate model predictive controller with guarantees," *IEEE Control Systems Letters*, vol. 2, no. 3, pp. 543–548, 2018.

[153] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 13–30, 1963.

[154] M. Jin and J. Lavaei, "Stability-certified reinforcement learning: A control-theoretic perspective," *IEEE Access*, vol. 8, pp. 229 086–229 100, 2020.

[155] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 908–919.

[156] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause, "Safe learning of regions of attraction for uncertain, nonlinear systems with Gaussian processes," in *IEEE Conference on Decision and Control*, 2016, pp. 4661–4666.

[157] M. Marchi, J. Bunton, B. Gharesifard, and P. Tabuada, "Safety and Stability Guarantees for Control Loops With Deep Learning Perception," *IEEE Control Systems Letters*, vol. 6, pp. 1286–1291, 2022.

[158] H. Tsukamoto, S.-J. Chung, J.-J. Slotine, and C. Fan, "A Theoretical Overview of Neural Contraction Metrics for Learning-based Control with Guaranteed Stability," in *IEEE Conference on Decision and Control*, 2021, pp. 2949–2954.

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

[160] R. Schwan, C. N. Jones, and D. Kuhn, "Stability verification of neural network controllers using mixed-integer programming," 2022, arXiv:2206.13374.

[161] H. Dai, B. Landry, L. Yang, M. Pavone, and R. Tedrake, "Lyapunov-stable neural-network control," in *Robotics: Science and Systems XVII, Virtual Event, July 12-16, 2021*, 2021.

[162] X. Zhang, M. Bujarbaruah, and F. Borrelli, "Near-optimal rapid mpc using neural networks: A primal-dual policy learning framework," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 5, pp. 2102–2114, 2021.

[163] Y. Zhang and X. Xu, "Safety Verification of Neural Feedback Systems Based on Constrained Zonotopes," 2022.

[164] E. T. Maddalena, M. W. F. Specq, V. L. Wisniewski, and C. N. Jones, "Embedded PWM predictive control of DC-DC power converters via piecewise-affine neural networks," *IEEE Open Journal of the Industrial Electronics Society*, vol. 2, pp. 199–206, 2021.

[165] D. Mayne, M. Seron, and S. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, vol. 41, no. 2, pp. 219–224, 2005.

[166] S. A. Niederer, M. S. Sacks, M. Girolami, and K. Willcox, "Scaling digital twins from the artisanal to the industrial," *Nature Computational Science*, vol. 1, no. 5, pp. 313–320, 2021.

[167] P. Fritzson, "Modelica—A cyber-physical modeling language and the OpenModelica environment," in *2011 7th International Wireless Communications and Mobile Computing Conference*. IEEE, 2011, pp. 1648–1653.

[168] M. Wetter, W. Zuo, T. S. Nouidui, and X. Pang, "Modelica buildings library," *Journal of Building Performance Simulation*, vol. 7, no. 4, pp. 253–270, 2014.

[169] J. E. Kay, C. Deser, A. Phillips, A. Mai, C. Hannay, G. Strand, J. M. Arblaster, S. Bates, G. Danabasoglu, J. Edwards *et al.*, "The Community Earth System Model (CESM) large ensemble project: A community resource for studying climate change in the presence of internal climate variability," *Bulletin of the American Meteorological Society*, vol. 96, no. 8, pp. 1333–1349, 2015.

[170] S. McCamish and M. Romano, "Simulation of relative multiple spacecraft dynamics and control with MATLAB-SIMULINK and Satellite Tool Kit," in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, 2007, p. 6805.

[171] H. Jasak, A. Jemcov, Z. Tukovic *et al.*, "OpenFOAM: A C++ library for complex physics simulations," in *International workshop on coupled methods in numerical dynamics*, vol. 1000. IUC Dubrovnik Croatia, 2007, pp. 1–20.

[172] Y. Hu, J. Liu, A. Spielberg, J. B. Tenenbaum, W. T. Freeman, J. Wu, D. Rus, and W. Matusik, "Chainqueen: A real-time differentiable physical simulator for soft robotics," in *2019 International conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6265–6271.

[173] J. A. Paulson, E. A. Buehler, and A. Mesbah, "Arbitrary polynomial chaos for uncertainty propagation of correlated random variables in dynamic systems," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3548–3553, 2017.

[174] J. A. Paulson, M. Martin-Casas, and A. Mesbah, "Fast uncertainty quantification for dynamic flux balance analysis using non-smooth polynomial chaos expansions," *PLoS Computational Biology*, vol. 15, no. 8, p. e1007308, 2019.

[175] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational physics*, vol. 378, pp. 686–707, 2019.

[176] J. Ma, P. Mahapatra, S. E. Zitney, L. T. Biegler, and D. C. Miller, "D-rm builder: A software tool for generating fast and accurate nonlinear dynamic reduced models from high-fidelity models," *Computers & Chemical Engineering*, vol. 94, pp. 60–74, 2016.

[177] 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*. IEEE, 2018, pp. 6307–6313.

[178] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[179] 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.

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

[181] D. Piga, M. Forgione, S. Formentin, and A. Bemporad, "Performance-oriented model learning for data-driven MPC design," *IEEE Control Systems Letters*, vol. 3, no. 3, pp. 577–582, 2019.

[182] J. A. Paulson and A. Mesbah, "Data-driven scenario optimization for automated controller tuning with probabilistic performance guarantees," *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1477–1482, 2020.

[183] 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, no. 3, pp. 243–250, 2021.

[184] Q. Lu, R. Kumar, and V. M. Zavala, "MPC controller tuning using Bayesian optimization techniques," *arXiv preprint arXiv:2009.14175*, 2020.

[185] M. Fiducioso, S. Curi, B. Schumacher, M. Gwerder, and A. Krause, "Safe contextual Bayesian optimization for sustainable room temperature PID control tuning," *arXiv preprint arXiv:1906.12086*, 2019.

[186] 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, no. 1, pp. 1032–1042, 2021.

[187] G. Makrygiorgos, A. D. Bonzanini, V. Miller, and A. Mesbah, "Performance-oriented model learning for control via multi-objective

bayesian optimization," *Computers & Chemical Engineering*, vol. 162, p. 107770, 2022.

[188] 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*. IEEE, 2021, pp. 3633–3639.

[189] J. A. Paulson, G. Makrygiorgos, and A. Mesbah, "Adversarially robust Bayesian optimization for efficient auto-tuning of generic control structures under uncertainty," *AIChE Journal*, vol. 68, no. 6, p. e17591, 2022.

[190] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.

[191] T. Elsken, B. Staffler, J. H. Metzen, and F. Hutter, "Meta-learning of neural architectures for few-shot learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 12 365–12 375.

[192] E. Arcari, M. V. Minniti, A. Scampicchio, A. Carron, F. Farshidian, M. Hutter, and M. N. Zeilinger, "Bayesian multi-task learning MPC for robotic mobile manipulation," 2022.

[193] S. M. Richards, N. Azizan, J.-J. Slotine, and M. Pavone, "Adaptive-control-oriented meta-learning for nonlinear systems," *arXiv preprint arXiv:2103.04490*, 2021.

[194] S. Zhan, G. Wichern, C. Laughman, A. Chong, and A. Chakrabarty, "Calibrating building simulation models using multi-source datasets and meta-learned Bayesian optimization," *Energy and Buildings*, vol. 270, p. 112278, 2022.

[195] A. Chakrabarty, "Few-shot closed-loop performance optimization with Bayesian meta-learning," in *Proc. IEEE Conf. Dec. and Control (CDC)*, 2022, p. *To appear*.

[196] Y. Chen, Y. Zheng, and H. Samuelson, "Fast adaptation of thermal dynamics model for predictive control of hvac and natural ventilation using transfer learning with deep neural networks," in *2020 American Control Conference (ACC)*, 2020, pp. 2345–2350.

[197] S. Xu, Y. Wang, Y. Wang, Z. O'Neill, and Q. Zhu, "One for many: Transfer learning for building HVAC control," in *Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, ser. BuildSys '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 230–239.

[198] M. Poloczek, J. Wang, and P. Frazier, "Multi-information source optimization," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[199] 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.

[200] J. Song, Y. Chen, and Y. Yue, "A general framework for multi-fidelity Bayesian optimization with gaussian processes," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 3158–3167.

[201] J. Wu, S. Toscano-Palmerin, P. I. Frazier, and A. G. Wilson, "Practical multi-fidelity Bayesian optimization for hyperparameter tuning," in *Uncertainty in Artificial Intelligence*. PMLR, 2020, pp. 788–798.

[202] 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.

[203] M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, R. Mottaghi, J. Salvador, D. Schwenk, E. VanderBilt, M. Wallingford, L. Weihs, M. Yatskar, and A. Farhadi, "RoboTHOR: An Open Simulation-to-Real Embodied AI Platform," in *CVPR*, 2020.

[204] S. James, Z. Ma, D. Rovick Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, 2020.

[205] C. Gan, J. Schwartz, S. Alter, D. Mrowca, M. Schrimpf, J. Traer, J. De Freitas, J. Kubilius, A. Bhandwaldar, N. Haber, M. Sano, K. Kim, E. Wang, M. Lingelbach, A. Curtis, K. Feigelis, D. M. Bear, D. Gutfreund, D. Cox, A. Torralba, J. J. DiCarlo, J. B. Tenenbaum, J. H. McDermott, and D. L. K. Yamins, "Threedworld: A platform for interactive multi-modal physical simulation," 2020.

[206] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem, "Brax – a differentiable physics engine for large scale rigid body simulation," 2021.

[207] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra, "Sim2real predictivity: Does evaluation in simulation predict real-world performance?" *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6670–6677, oct 2020.

[208] N. Yokoyama, S. Ha, and D. Batra, "Success weighted by completion time: A dynamics-aware evaluation criteria for embodied navigation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[209] J. Truong, D. Yarats, T. Li, F. Meier, S. Chernova, D. Batra, and A. Rai, "Learning navigation skills for legged robots with learned robot embeddings," 2020.

[210] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[211] W. Xu, C. N. Jones, B. Svetozarevic, C. R. Laughman, and A. Chakrabarty, "VABO: Violation-aware Bayesian optimization for closed-loop control performance optimization with unmodeled constraints," in *2022 American Control Conference (ACC)*. IEEE, 2022, pp. 5288–5293.

[212] A. Chakrabarty, C. Danielson, S. A. Bortoff, and C. R. Laughman, "Accelerating self-optimization control of refrigerant cycles with bayesian optimization and adaptive moment estimation," *Applied Thermal Engineering*, vol. 197, p. 117335, 2021.

[213] Modelica Association, "Modelica specification, Version 3.4," 2017.

[214] H. Qiao, V. Aute, and R. Radermacher, "Transient modeling of a flash tank vapor injection heat pump system - Part I: Model development," *Int. J. Refrigeration*, vol. 49, pp. 169–182, 2015.

[215] J. R. Gardner, M. J. Kusner, Z. E. Xu, K. Q. Weinberger, and J. P. Cunningham, "Bayesian optimization with inequality constraints," in *ICML*, vol. 2014, 2014, pp. 937–945.

[216] F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe controller optimization for quadrotors with gaussian processes," in *2016 IEEE Int. Conf. on Robot. and Automat. (ICRA)*. IEEE, 2016, pp. 491–496.