Worst-case Nonlinear Regression with Error Bounds

Alberto Bemporad, *Fellow, IEEE*

***Abstract*— We propose an active-learning method for nonlinear minimax regression. Given a nonlinear function that can be arbitrarily evaluated over a compact set, we fit a surrogate model, such as a feedforward neural network, by minimizing the maximum absolute approximation error. To handle the nonsmoothness of this worst-case loss, we introduce a smooth L_∞ approximation that enables efficient gradient-based training. The training set is iteratively enriched by querying points of largest error via global optimization. We also derive constant and input-dependent worst-case error bounds over the entire input domain. The approach is validated on approximations of nonlinear functions and nonconvex sets, uncertain models of nonlinear dynamics, and explicit model predictive control laws. A Python library is available at `https://github.com/bemporad/maxfit`.**

***Index Terms*— Worst-case nonlinear regression, L_∞-regression, minimax regression, neural networks, uncertainty bounds, approximate explicit model predictive control.**

I. Introduction

Function approximation methods are fundamental in control systems engineering, both to obtain control-oriented models of dynamical systems and to simplify complex nonlinear control laws for online implementation. They can be used either to learn models or control laws directly from data, or to simplify a known model or control function. In both cases, the typical approach is to collect a sufficiently large set of samples and solve a nonlinear regression problem, for example by training a feedforward neural network (NN) by minimizing the mean squared error (MSE), possibly under regularization on the model parameters.

In control applications, however, it is often crucial to quantify a bound on the approximation error between the true function, when known, and the learned model. Such bounds are needed, for example, in robust control design to characterize modeling errors treated as additive disturbances, and when approximating control laws to guarantee closed-loop stability and constraint satisfaction despite approximation errors. This issue is particularly relevant for approximating model predictive control (MPC) laws to reduce online computational effort, typically using NNs [11], [16], [23]. However, most existing methods do not guarantee bounds on the worst-case approximation error (WCE) between the exact MPC law and its approximation. The recent contribution [1] addresses this problem by exploiting Lipschitz continuity, assuming the availability of a sufficiently large training dataset.

In system identification for robust control, set-membership methods have been used to obtain models with guaranteed bounded error predictions, including nonlinear models [21], often under assumptions on the system dynamics. Similar ideas have been applied to approximate nonlinear MPC laws in [10] under Lipschitz continuity assumptions. Gaussian processes [25] offer an alternative by providing models with quantified uncertainty, but they rely on restrictive model classes and yield only probabilistic guarantees.

Robustness in nonlinear regression has been studied in the statistics literature mainly to *mitigate* the effect of outliers in training data [13]. Here, instead, our goal is to minimize and quantify the WCE over *all* possible inputs in a given set. This problem is related to uncertainty quantification and conformal prediction [2], which typically provide probabilistic confidence intervals rather than worst-case guarantees [5]. Achieving the latter requires different techniques; for example, in [12] semidefinite programming is used to certify the robustness of NN predictions against input uncertainties.

The standard workflow to obtain nonlinear models with bounded prediction errors is to first identify a model and then estimate its approximation error. However, a small *average* error, such as a low MSE, does not necessarily imply a small *worst-case* error. This motivates methods that directly minimize the WCE during training and reliably quantify it afterward. *Minimax regression*, also known as *Chebyshev regression* or *L_∞ estimation* [24, Ch. 7], addresses this need by minimizing the maximum deviation between the true function and its approximation over a dataset or a domain. Despite its potential for robust system identification and control, it has received limited attention in the machine learning and control communities, with most existing results restricted to linear models. This is partly due to the nonsmooth optimization problems it entails and to the strong dependence of the solution quality on the choice of the training samples.

A. Contribution

In this paper, we consider fitting a parametric model, such as an NN, to a known nonlinear function using an ℓ_∞ loss over a finite set of samples, with the goal of minimizing the WCE over a compact input set. To enable the use of efficient quasi-Newton methods such as L-BFGS [18] together with automatic differentiation, we replace the nonsmooth maximum operator with a smooth (but arbitrarily accurate) approximation and solve the resulting problem. We reduce the dependence of the WCE on training samples by proposing an active-learning strategy that iteratively adds points of maximum approximation error, identified via global optimization.

A key difference with respect to active-learning methods for nonlinear regression and classification, which rely on empirical acquisition functions to select potentially informative data [7], [28], is that here the true function can be evaluated at arbitrary inputs during the acquisition process to explicitly identify a point of maximum error. The approach is therefore closely related to corner-case detection via numerical optimization [19].

In addition, we propose techniques to quantify the WCE over the entire input set after training, in the form of either constant or input-dependent upper and lower bounds on the approximation error. Finally, we show how the proposed approach can be used to learn sets (e.g., minimally conservative convex inner approximations of given nonconvex sets), uncertain (generally nonlinear) models of nonlinear dynamical systems (e.g., derived from physical principles),

The author is with the IMT School for Advanced Studies, Piazza San Francesco 19, Lucca, Italy. Email: `alberto.bemporad@imtlucca.it`.
This work was funded by the European Union (ERC Advanced Research Grant COMPACT, No. 101141351). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

and approximate explicit MPC laws with guaranteed bounds on the control error introduced by the approximation.

B. Notation

We denote by $[x]_+ = \max(x, 0)$ the positive part of $x \in \mathbb{R}$ and by $\max(v) = \max_{i=1,\dots,n}(v_i)$ the maximum component of a vector $v \in \mathbb{R}^n$. Given vectors $v_i \in \mathbb{R}^{n_i}$, $i = 1, \dots, N$, $n_i \geq 1$, we denote by $v = \text{col}(v_1, \dots, v_N) \in \mathbb{R}^n$ their vertical concatenation. Throughout the paper, with a slight abuse of notation, we define the sign function as $\text{sign}(x) = -1$ if $x \leq 0$ and $\text{sign}(x) = 1$ if $x > 0$.

II. Minimax Nonlinear Regression

Given a nonlinear function $f : \mathcal{X} \to \mathbb{R}$ over a compact set $\mathcal{X} \subset \mathbb{R}^n$ and a class of functions $\mathcal{F}$, our goal is to find a simpler function $\hat{f} : \mathcal{X} \to \mathbb{R}$, $\hat{f} \in \mathcal{F}$, that minimizes the WCE between $f(x)$ and $\hat{f}(x)$ over $\mathcal{X}$, and to quantify such an error.

Let θ be the vector of parameters identifying the function $\hat{f}$ within $\mathcal{F}$, $\theta \in \mathbb{R}^{n_\theta}$. For example, θ can collect the weights and bias terms of an NN, the coefficients of a polynomial or other basis functions, etc. We want to solve the following problem:

$$\theta^\star \in \arg\min_\theta \left(r(\theta) + \max_{x \in \mathcal{X}} |f(x) - \hat{f}(x;\theta)| \right) \tag{1}$$

where $r(\theta)$ is a regularization term, such as an ℓ_2 or ℓ_1 regularizer, or a penalty introduced to promote certain properties that $\hat{f}$ should have.

Solving Problem (1) poses two main difficulties: (i) it is a bilevel optimization problem, due to the inner maximization problem over $\mathcal{X}$; and (ii) the inner objective function is non-smooth due to the maximum operator, which makes applying very efficient quasi-Newton methods, such as L-BFGS [18] more difficult.

To address the first issue, we consider an initial training dataset $(x_1, y_1), \dots, (x_{N_0}, y_{N_0})$ of samples, where $x_k \in \mathcal{X}$ and $y_k = f(x_k)$, and replace $\mathcal{X}$ with $\{x_k\}_{k=1}^{N_0}$, so that solving the inner maximization problem simply amounts to finding the maximum of a finite number of terms. Accordingly, Problem (1) is replaced by

$$\theta^\star \in \arg\min_\theta \left(r(\theta) + \max_{k=1,\dots,N_0} |f(x_k) - \hat{f}(x_k;\theta)| \right). \tag{2}$$

A relevant problem addressed in this paper is how to actively learn a minimal number of samples x_k so that the solution of (2) provides a solution to (1).

We address the second issue by using the following smooth approximation of the max operator (cf. [22]):

$$\theta^\star \in \arg\min_\theta \Big(r(\theta) + \frac{1}{\gamma} \log\Big(\sum_{k=1}^{N_0} \big(e^{\gamma(y_k - \hat{f}(x_k;\theta))} + e^{\gamma(\hat{f}(x_k;\theta) - y_k)} \big) \Big) \Big). \tag{3}$$

Proposition 1: The minimization problem (3) has the same set of minimizers as (2) in the limit $\gamma \to +\infty$.

Proof: See Appendix A. ∎

After obtaining the model $\hat{f}(\cdot;\theta^\star)$ by solving (3), we can compute the WCE $\bar{e}^\star$ over the entire set $\mathcal{X}$ by solving the following problem

$$\bar{e}^\star = \max_{x \in \mathcal{X}} |f(x) - \hat{f}(x;\theta^\star)| \tag{4}$$

by using derivative-free global optimization methods [15], [26].

The main idea for enriching the dataset used in problem (2) is to add the obtained maximizer $x^\star$ and the corresponding value $y^\star = f(x^\star)$ to the training dataset and repeat the procedure until the maximum error $\bar{e}^\star$ is below a given threshold ϵ_{err}, or a preallocated computational budget is exhausted. In the latter case, as the WCE may not decrease at each iteration, the model $\hat{f}(x;\theta)$ that has provided the smallest WCE during the iterations is selected as the final model. The overall procedure is summarized in Algorithm 1.

As in most active-learning schemes, we start from an initial number N_0 of samples x_k purely based on their location in the input space, such as randomly from the uniform distribution over $\mathcal{X}$, or by using Latin hypercube sampling (LHS) [20], or by gridding $\mathcal{X}$ uniformly. Note that as N_0 increases, fewer active-learning steps are expected to be required to achieve a given WCE $\bar{e}^\star \leq \epsilon_{\text{err}}$, although the training problem (3) becomes more complex at each step.

Algorithm 1 Worst-case nonlinear regression with active learning

Input: Function f to approximate, model class $\mathcal{F}$, number N_0 of initial samples to collect, maximum number M of active-learning steps, desired error threshold ϵ_{err}, smoothness parameter γ, regularization function r, input set $\mathcal{X}$.

1. Generate initial dataset $(x_1, y_1), \dots, (x_{N_0}, y_{N_0})$, $y_k = f(x_k)$;
2. **Set** $N \leftarrow N_0$;
3. **Repeat**:
   3.1. Solve problem (3) and get optimal model parameters θ_N;
   3.2. Set $x_{N+1} \leftarrow$ global optimizer of problem (4);
   3.3. Set $y_{N+1} \leftarrow f(x_{N+1})$, $e_N \leftarrow |y_{N+1} - \hat{f}(x_{N+1};\theta_N)|$;
   3.4. $N \leftarrow N + 1$;
4. **Until** $e_{N-1} \leq \epsilon_{\text{err}}$ or $N = N_0 + M$;
5. **Set** $i^\star \leftarrow \arg\min_{k=N_0,\dots,N-1} e_k$, $\bar{e}^\star \leftarrow e_{i^\star}$, and $\theta^\star \leftarrow \theta_{i^\star}$;
6. **End**.

Output: Predictor $\hat{f}(x;\theta^\star)$ and error bound $\bar{e}^\star = \max_{x \in \mathcal{X}} |f(x) - \hat{f}(x;\theta^\star)|$.

Note that the objective function in (2) can be generalized to the following MSE-regularized objective:

$$r(\theta) + \max_k |f(x_k) - \hat{f}(x_k;\theta)| + \frac{\nu}{N} \sum_{k=1}^{N} \big(f(x_k) - \hat{f}(x_k;\theta) \big)^2 \tag{5}$$

where ν is a small positive scalar weight that discourages overfitting the model to the actively-learned corner-case samples.

A. Imposing a functional form on a set

Assume that we want to impose that $\hat{f}(x;\theta) \equiv w(x)$ for all x in a given set $\mathcal{G} = \{x : \ g(x) \leq 0, \ h(x) = 0\}$, where $w : \mathcal{X} \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^{n_g}$, and $h : \mathbb{R}^n \to \mathbb{R}^{n_h}$ are given functions. In Section V-B we will see an example with w affine and $\mathcal{G}$ polyhedral. Consider the indicator function $\delta : \mathbb{R}^n \to \{0, 1\}$ of $\mathcal{G}$, i.e., $\delta(x) = 1$ if $x \in \mathcal{G}$ and $\delta(x) = 0$ otherwise, and either its piecewise affine (PWA) approximation

$$\hat{\delta}(x;\beta) = \max\left(1 - \beta \max(\text{col}(g(x), \pm h(x), 0)), 0\right) \tag{6a}$$

or its approximation

$$\hat{\delta}(x;\beta) = e^{-\beta \left(\sum_{i=1}^{n_g} [g_i(x)]_+ + \sum_{j=1}^{n_h} |h_j(x)| \right)} \tag{6b}$$

where $\beta > 0$. Clearly, $\hat{\delta}(x;\beta) = 1$, $\forall x \in \mathcal{G}$. Moreover, for $x \notin \mathcal{G}$, $\lim_{\beta \to +\infty} \hat{\delta}(x;\beta) = 0$. Then, we define the model

$$\hat{f}(x;\theta) = \hat{\delta}(x;\beta) w(x) + (1 - \hat{\delta}(x;\beta)) \hat{v}(x;\theta) \tag{7}$$

where $\hat{v} \in \mathcal{F}$. Note that this changes the model class to which $\hat{f}$ belongs. The parameter β can be either fixed or trainable; in the latter case, we assume β is a component of θ.

B. Saturation

Assume that we want to bound the output of the model in the hyper-box $y_{\min} \leq y \leq y_{\max}$. For example, we may know that the codomain of f is contained in such a hyper-box. To this end, we consider the saturation function

$$\mathrm{sat}(y; y_{\min}, y_{\max}) = \min(\max(y, y_{\min}), y_{\max}) \tag{8a}$$

and its smooth approximation (cf. [8, Section 3.B]):

$$\sigma_\eta(y; y_{\min}, y_{\max}) = y_{\max} + \frac{1}{\eta} \log\left(\frac{1 + e^{-\eta(y - y_{\min})}}{1 + e^{-\eta(y - y_{\max})}}\right) \tag{8b}$$

(all arithmetic operations are component-wise) where η is either a fixed or a further trainable parameter, $\eta > 0$.

Proposition 2: Let $y_{\min} < y_{\max}$ component-wise and $\eta > 0$. Then, the function σ_η in (8b) is (component-wise) strictly monotonically increasing with respect to y, continuous, satisfies $y_{\min} \leq \sigma_\eta(y; y_{\min}, y_{\max}) \leq y_{\max}$, tends to $y_{\min}$ for $y \to -\infty$ and to $y_{\max}$ for $y \to +\infty$, and is such that $\sigma_\eta\left(\frac{y_{\min}+y_{\max}}{2}; y_{\min}, y_{\max}\right) = \frac{y_{\min}+y_{\max}}{2}$, $\forall \eta > 0$. Moreover, it approaches the saturation function $\mathrm{sat}(y; y_{\min}, y_{\max})$ asymptotically as $\eta \to +\infty$.

Proof: See Appendix B. ∎

Finally, we define the model

$$\hat{f}(x; \theta) = \hat{s}(\hat{v}(x; \theta); y_{\min}, y_{\max}) \tag{9}$$

where $\hat{v} \in \mathcal{F}$ and $\hat{s}$ is either sat in (8a) or σ_η in (8b) (we assume that η is a component of θ in case it is trainable). Note that this also changes the model class $\hat{f}$ belongs to. Saturation can also be combined with the approach detailed in Section II-A by setting $\hat{f}(x; \theta) = \hat{s}((1 - \hat{\delta}(x; \beta))\hat{v}(x; \theta) + \hat{\delta}(x; \beta)w(x); y_{\min}, y_{\max})$.

C. Inexact global optimization

The error bound $\bar{e}^\star$ returned by Algorithm 1 is guaranteed to be an upper bound under the assumption that the global optimization problem (4) is solved exactly. Under mild assumptions on f and $\hat{f}$, the following proposition shows that we can still guarantee an error bound when running the `direct` global optimization algorithm [14], [15] for a sufficiently large but finite number of iterations T.

Proposition 3: Assume $f : \mathcal{X} \to \mathbb{R}$ and $\hat{f}(\cdot; \theta^\star) : \mathcal{X} \to \mathbb{R}$ have Lipschitz constants $K_f > 0$ and $K_{\hat{f}} > 0$, respectively. For any given tolerance $\delta > 0$, there exists T large enough such that, after running the global optimizer algorithm [15] on problem (4) for T iterations, we get

$$\max_{x \in \mathcal{X}} |f(x) - \hat{f}(x; \theta^\star)| \leq \bar{e}_T + \delta \tag{10}$$

where $\bar{e}_T$ is the best value returned after T iterations.

Proof: Let $g : \mathcal{X} \to \mathbb{R}$, with $g(x) \triangleq |f(x) - \hat{f}(x; \theta^\star)|$. For any $x, z \in \mathcal{X}$, we get $|g(x) - g(z)| = \big||f(x) - \hat{f}(x; \theta^\star)| - |f(z) - \hat{f}(z; \theta^\star)|\big| \leq |(f(x) - \hat{f}(x; \theta^\star)) - (f(z) - \hat{f}(z; \theta^\star))| \leq |f(x) - f(z)| + |\hat{f}(x; \theta^\star) - \hat{f}(z; \theta^\star)| \leq (K_f + K_{\hat{f}})\|x - z\|_2$, where the first inequality follows from the reverse triangle inequality, and so g is Lipschitz continuous with constant $K_g \leq K_f + K_{\hat{f}}$.

Let $x^\star \in \arg\max_{x \in \mathcal{X}} g(x)$ and set $\epsilon = \delta / K_g$ (if $K_g = 0$ then g is constant and the result is trivial, since $\bar{e}_T = \bar{e}^\star$). By the everywhere-dense sampling property of DIRECT [14], the algorithm eventually samples a point x_τ, $\tau \leq T$, with $\|x^\star - x_\tau\|_2 \leq \epsilon$. Hence, $\bar{e}_T \geq g(x_\tau)$, and by Lipschitz continuity of g, $g(x_\tau) \geq g(x^\star) - K_g\|x^\star - x_\tau\|_2 \geq \bar{e}^\star - K_g\epsilon = \bar{e}^\star - \delta$. ∎

Note that solving problem (3) (in general nonconvex) suboptimally only affects the quality of the model $\theta^\star$, not the validity of the error bound $\bar{e}^\star$.

III. Input-dependent error bounds

Algorithm 1 provides the global, symmetric, and uniform bound $\bar{e}^\star$ for $f(x) - \hat{f}(x; \theta^\star)$ over $\mathcal{X}$, where $\theta^\star$ collects the optimal model coefficients determined by Algorithm 1 (possibly including $\beta^\star$ from (7) and $\eta^\star$ from (9)). Here, we want to obtain bounds in the more general and therefore less conservative form

$$-e^\star_{\min}(x) \leq f(x) - \hat{f}(x; \theta^\star) \leq e^\star_{\max}(x), \ \forall x \in \mathcal{X}.$$

A. Constant asymmetric error bounds

Asymmetric uniform bounds, i.e., lower and upper bounds that are constant over $\mathcal{X}$, can be computed as

$$\begin{aligned} e^\star_{\min}(x) \equiv \bar{e}^\star_{\min} &\triangleq \max_{x \in \mathcal{X}} \left([\hat{f}(x; \theta^\star) - f(x)]_+\right) \\ e^\star_{\max}(x) \equiv \bar{e}^\star_{\max} &\triangleq \max_{x \in \mathcal{X}} \left([f(x) - \hat{f}(x; \theta^\star)]_+\right) \end{aligned} \tag{11}$$

by global optimization similarly to (4). Clearly, these are tighter than the symmetric bound $\bar{e}^\star$ returned by Algorithm 1, since $-\bar{e}^\star \leq -\bar{e}^\star_{\min} \leq f(x) - \hat{f}(x; \theta^\star) \leq \bar{e}^\star_{\max} \leq \bar{e}^\star$. The result of Proposition 3 can be immediately extended to the bounds in (11).

B. Symmetric input-dependent error bounds

Consider a set $\mathcal{E}$ of positive functions $\varepsilon : \mathbb{R}^n \times \mathbb{R}^{n_\psi} \to \mathbb{R}$ parameterized by a vector $\psi \in \mathbb{R}^{n_\psi}$, $\varepsilon(x; \psi) > 0$, $\forall x \in \mathcal{X}$, $\forall \psi \in \mathbb{R}^{n_\psi}$. We consider the problem of learning a function ε that upper bounds $|f(x) - \hat{f}(x; \theta^\star)|$ as tightly as possible. To this end, we want to solve the following training problem

$$\begin{aligned} \psi^\star = \arg\min_\psi &\rho_\psi(\psi) + \frac{1}{N}\sum_{k=1}^{N} \mu(\varepsilon(x_k; \psi)) \\ &\text{s.t. } \varepsilon(x_k; \psi) \geq |e_k|, \ k = 1, \ldots, N \end{aligned} \tag{12}$$

where $e_k = y_k - \hat{f}(x_k; \theta^\star)$ is the prediction error at sample k, $\mu : \mathbb{R}_+ \to \mathbb{R}_+$ is a monotonically increasing function (e.g., $\mu(\varepsilon) = \alpha\varepsilon$, $\mu(\varepsilon) = \alpha\varepsilon^2$) that attempts to squeeze the interval $[-\varepsilon(x_k; \psi), \varepsilon(x_k; \psi)]$ as much as possible, under the regularization ρ_ψ on the parameters of function ε, and $N \leq N_0 + M$ is the number of available samples collected by Algorithm 1. We call ε an *envelope* function, due to the constraint in (12).

To relax the problem into an unconstrained nonlinear program, we rewrite the constraints in (12) into their equivalent form

$$\max\left(0, \max_{k=1,\ldots,N} (|e_k| - \varepsilon(x_k; \psi))\right) = 0 \tag{13}$$

and, by introducing an approximation of the maximum violation constraint in (13) similar to the one used in (3), we relax (13) into the following unconstrained nonlinear programming problem

$$\min_\psi \rho_\psi(\psi) + \frac{1}{N}\sum_{k=1}^{N} \mu(\varepsilon(x_k; \psi)) + \frac{1}{\gamma_\varepsilon} \log\left(1 + \sum_{k=1}^{N} e^{\gamma_\varepsilon(|e_k| - \varepsilon(x_k; \psi))}\right). \tag{14}$$

Note that, by Proposition 1, the last term in (14) enforces constraint (13) in the limit $\gamma_\varepsilon \to +\infty$, being a smooth version of $\max(0, \max_{k=1,\ldots,N}(|e_k| - \varepsilon(x_k; \psi)))$.

Any positive model $\varepsilon(x; \psi)$ can be used to learn the envelope functions. In this paper, we focus on NN models with two hidden layers

$$\begin{aligned} \varepsilon(x; \psi) = &W_3 a^+(W_2 \cdot a(W_1 \cdot x + b_1) + b_2) + b_3 \\ &W_3 \geq 0, \ b_3 > 0 \end{aligned} \tag{15}$$

where a^+ is a nonnegative activation function, such as the ReLU or sigmoid function, $a : \mathbb{R}^n \to \mathbb{R}^m$ is any activation function, $W_1 \in$

$\mathbb{R}^{n_1\times n}$, $W_2\in\mathbb{R}^{n_2\times n_1}$, $W_3\in\mathbb{R}^{1\times n_2}$ are weight matrices, and $b_1\in\mathbb{R}^{n_1}$, $b_2\in\mathbb{R}^{n_2}$, $b_3\in\mathbb{R}$ are bias terms. The transformation a^+ and the component-wise constraints on W_3, b_3 ensure that $\varepsilon(x;\psi)>0$ for all $x\in\mathbb{R}^n$, with $\psi=\mathrm{col}(W_1,W_2,W_3,b_1,b_2,b_3)$. In particular, by setting a^+ and a as ReLU functions we get PWA envelopes ε.

To get a global upper-bound function on $\mathcal{X}$, we compute the following scaling factor κ via global optimization:

$$\kappa^\star=\max_{x\in\mathcal{X}}\left(\frac{|f(x)-\hat f(x;\theta^\star)|}{\varepsilon(x;\psi^\star)}\right) \tag{16}$$

and define the bounding functions

$$e^\star_{\min}(x)=e^\star_{\max}(x)=\min(\bar e^\star,\kappa^\star\varepsilon(x;\psi^\star)). \tag{17}$$

Note that (16) is closely related to split conformal prediction [2], where the scaling factor $\kappa^\star$ is determined on a calibration dataset rather than by optimizing over the entire set $\mathcal{X}$, both for constant (homoscedastic) and input-dependent (heteroscedastic) bounds. In the latter case, our approach can also leverage additional calibration data when fitting $\varepsilon(x;\psi)$, so as to limit conservativeness when making the envelope function valid over the entire $\mathcal{X}$ in (17).

C. Asymmetric input-dependent error bounds

To further reduce the conservativeness of the bounds, the approach described above can be extended to fit *asymmetric* input-dependent upper and lower error bounds. Let us consider two different envelope functions $\varepsilon^u(x;\psi_u)$ and $\varepsilon^\ell(x;\psi_\ell)$, with $\psi_u,\psi_\ell\in\mathbb{R}^{n_\psi}$, to learn simultaneously by minimizing

$$\begin{aligned}(\psi_u^\star,\psi_\ell^\star)=\arg\min &\rho_\psi(\psi_u)+\rho_\psi(\psi_\ell)\\ &+\frac{1}{N}\sum_{k=1}^N\mu(\varepsilon^u(x_k;\psi_u)+\varepsilon^\ell(x_k;\psi_\ell))\\ \text{s.t. }&\max\Big(0,\max_{k=1,\dots,N}(e_k-\varepsilon^u(x_k;\psi_u)),\\ &\max_{k=1,\dots,N}(-e_k-\varepsilon^\ell(x_k;\psi_\ell))\Big)=0.\end{aligned} \tag{18}$$

Similarly to the envelope learning problem in (12), we can relax the problem into the unconstrained nonlinear program

$$\begin{aligned}&\min_{\psi_u,\psi_\ell}\frac{1}{N}\sum_{k=1}^N\mu(\varepsilon^u(x_k;\psi_u)+\varepsilon^\ell(x_k;\psi_\ell))+\rho_\psi(\psi_u)+\rho_\psi(\psi_\ell)\\ &+\frac{1}{\gamma_\varepsilon}\log\left(1+\sum_{k=1}^N\left(e^{\gamma_\varepsilon(e_k-\varepsilon^u(x_k;\psi_u))}+e^{\gamma_\varepsilon(-e_k-\varepsilon^\ell(x_k;\psi_\ell))}\right)\right)\end{aligned} \tag{19}$$

where the last term penalizes a smooth approximation of the maximum violation of the constraint in (18). Analogously to (4), after finding the optimal parameters $\psi_u^\star$ and $\psi_\ell^\star$, we determine two scaling factors $\kappa_u^\star$ and $\kappa_\ell^\star$, respectively, by solving the global optimization problems

$$\kappa_u^\star=\max_{x\in\mathcal{X}}\frac{[f(x)-\hat f(x;\theta^\star)]_+}{\varepsilon^u(x;\psi_u^\star)},\quad \kappa_\ell^\star=\max_{x\in\mathcal{X}}\frac{[\hat f(x;\theta^\star)-f(x)]_+}{\varepsilon^\ell(x;\psi_\ell^\star)} \tag{20}$$

to get the upper and lower bound functions on $\mathcal{X}$

$$\begin{aligned}e^\star_{\min}(x)&=\min(\bar e^\star,\kappa_\ell^\star\varepsilon^\ell(x;\psi_\ell^\star))\\ e^\star_{\max}(x)&=\min(\bar e^\star,\kappa_u^\star\varepsilon^u(x;\psi_u^\star))\end{aligned} \tag{21}$$

Note that, by replacing $\mu(\varepsilon^u(x;\psi_u)+\varepsilon^\ell(x;\psi_\ell))$ with $\mu(\varepsilon^u(x;\psi_u))+\mu(\varepsilon^\ell(x;\psi_\ell))$, the envelope functions ε^u and ε^ℓ can be learned independently. Note also that any function $\hat f_s$ of x whose values lie in $[\hat f(x;\theta^\star)-e^\star_{\min}(x),\hat f(x;\theta^\star)+e^\star_{\max}(x)]$ for all $x\in\mathcal{X}$ is a valid predictor of $f(x)$; in particular, the uncertainty interval can be symmetrized by simply redefining the centered predictor

$$\hat f^s(x)=\hat f(x;\theta^\star)+\frac{e^\star_{\max}(x)-e^\star_{\min}(x)}{2} \tag{22a}$$

along with the symmetric error bounding function

$$e^{\star s}_{\min}(x)=e^{\star s}_{\max}(x)=\frac{e^\star_{\max}(x)+e^\star_{\min}(x)}{2} \tag{22b}$$

although this changes the model class to which $\hat f$ belongs.

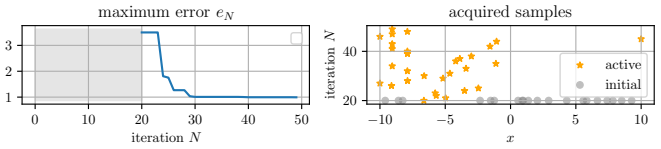

Fig. 1: Execution of Algorithm 1 on Example 1: maximum error e_N (left plot), initialization (gray area), samples acquired (right plot).

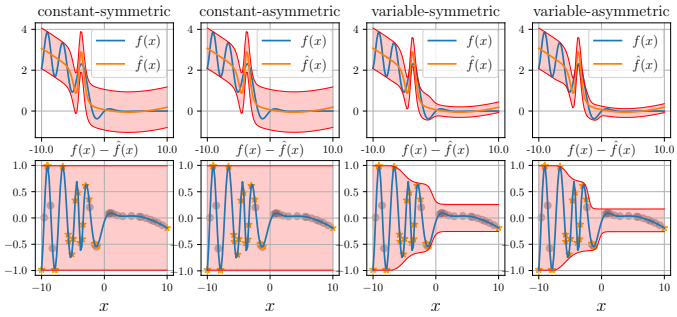

Fig. 2: Uncertainty bounds obtained on Example 1.

Example 1: Consider the scalar function $f(x)=\big(\sin((x-\frac{x^2}{10}))+(\frac{x}{10})^3-\frac{4x}{10}\big)\frac{e^{-x}}{1+e^{-x}}$ with $x\in[-10,10]$. We want to find an approximation of f within the very simple class $\mathcal{F}$ of NNs with two layers of 2 and 1 neurons, respectively, and tanh activation function. We run Algorithm 1 with $N_0=20$, $M=30$, $\epsilon_{\rm err}=0.001$, $\gamma=10$, $\nu=0$, and the regularization function $r(\theta)=10^{-8}\|\theta\|^2$. Figure 1 shows how the maximum prediction error $e_N=|f(x)-\hat f(x;\theta)|$ on collected training data decreases during the execution of Algorithm 1, reaching its minimum after around $N=30$ samples. The resulting model $\hat f(x;\theta^\star)$ is shown in Figure 2 along with the uniform and input-dependent bounds $e^\star_{\min}(x)$ and $e^\star_{\max}(x)$. The input-dependent bounds are also NNs in $\mathcal{F}$, also trained with ℓ_2-regularization $\rho_\psi(\psi)=10^{-6}\|\psi\|_2^2$, $\mu(\varepsilon)=10^{-3}\varepsilon^2$, $\gamma_\varepsilon=100$.

IV. Constraint Satisfaction

Assume that we are interested in replacing a given constraint $f(x)\le 0$ with a sufficiently simpler constraint $\bar f(x)\le 0$, such that

$$\bar f(x)\le 0\implies f(x)\le 0,\ \forall x\in\mathcal{X} \tag{23}$$

in the least conservative way, i.e., minimizing the size of the set $\{x\in\mathcal{X}:\bar f(x)>0,\ f(x)\le 0\}$ of feasible points for f that are declared infeasible by $\bar f$. The actual values taken by $f(x)$ are not relevant for (23), as long as the sign of $f(x)$ is correctly predicted by $\bar f(x)$, i.e., $\mathrm{sign}(\bar f(x))=-1\implies\mathrm{sign}(f(x))=-1$, $\forall x\in\mathcal{X}$. Therefore, we apply the worst-case regression approach developed in Section II with $f(x)$ replaced by $s(f(x))$ and $\hat f(x;\theta)$ by $s(\hat f(x;\theta))$, where $s:\mathbb{R}\to\mathbb{R}$ is a smooth approximation of the sign function. A possible choice is $s(x)=\tanh(\eta_s x)$, where $\eta_s>0$ is a given parameter (e.g., $\eta_s=10$). Note that s is only used during training and then removed after determining the optimal parameter vector $\theta^\star$.

After stopping the active-learning process and having fixed the optimal parameter vector $\theta^\star$, we need to make sure to construct $\bar{f}(x)$ such that it cannot be negative when $f(x) > 0$, so as to guarantee that condition (23) is satisfied. To this end, we compute the smallest value of $\hat{f}(x;\theta^\star)$ when the condition $f(x) > 0$ holds (i.e., $\text{sign}(f(x)) = 1$), by solving the following global optimization problem:

$$\Delta f \triangleq \min_{x\in\mathcal{X}} \left(\frac{1}{2}(1+\text{sign}(f(x)))\hat{f}(x;\theta^\star)\right) \tag{24}$$

and define $\bar{f}(x)$ as in the next proposition.

Proposition 4: Let Δf be the solution of (24) and define

$$\bar{f}(x) \triangleq \hat{f}(x;\theta^\star) - \Delta f + \epsilon_f \tag{25}$$

for an arbitrarily small number $\epsilon_f > 0$. Then, condition (23) is satisfied.

Proof: Assume, by contradiction, that there exists a vector x such that $\bar{f}(x) \le 0$ and $f(x) > 0$. Then,

$$\begin{aligned} 0 \ge \bar{f}(x) &= \hat{f}(x;\theta^\star) - \Delta f + \epsilon_f \\ &\ge \hat{f}(x;\theta^\star) - \hat{f}(x;\theta^\star) + \epsilon_f = \epsilon_f \end{aligned}$$

which contradicts the assumption $\epsilon_f > 0$. Hence, (23) must hold. ■

Note that, by selecting $\hat{f}$ as the composition of a strictly-increasing function $\hat{f}_m : \mathbb{R} \to \mathbb{R}$ with another function $\hat{f}_c : \mathbb{R}^n \to \mathbb{R}$, parameterized by θ_m and θ_c, respectively, $\hat{f}(x;\theta) = \hat{f}_m(\hat{f}_c(x;\theta_c);\theta_m)$, the resulting constraint (25) is equivalent to the constraint

$$\hat{f}_c(x;\theta_c^\star) \le \hat{f}_m^{-1}\left(\Delta f - \epsilon_f;\theta_m^\star\right). \tag{26}$$

In particular, if $\hat{f}_c$ is convex, such as an input-convex NN [27], the constraint in (26) is convex. In the special case $\hat{f}_c = \max_{i=1,\dots,n_f}(A_i x - b_i)$, we get the convex polyhedral set $\{x : Ax \le b - (\epsilon_f - \Delta f)\mathbf{1}\}$.

V. Modeling and Controller Approximations

A. Discrete-time uncertain models

We are given the continuous-time nonlinear model of a dynamical system

$$\begin{aligned} \dot{\xi}(t) &= F(\xi(t),u(t)) \\ \tau(t) &= G(\xi(t),u(t)) \end{aligned} \tag{27}$$

where $\xi(t) \in \mathbb{R}^{n_\xi}$ is the state vector and $u(t) \in \mathbb{R}^{n_u}$ the control input vector, derived, for example, from first-principles. Our goal is to learn a discrete-time uncertain model of the form

$$\begin{aligned} \xi(t+T_s) &= \hat{F}(\xi(t),u(t);\theta) + v_\xi(t) \\ \tau(t) &= \hat{G}(\xi(t),u(t);\theta) + v_\tau(t) \end{aligned} \tag{28}$$

where $T_s > 0$ is the sampling time, $\hat{F} : \mathbb{R}^{n_\xi} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_\xi}$ and $\hat{G} : \mathbb{R}^{n_\xi} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_\tau}$ are models parameterized by θ, and we treat the modeling errors $v_\xi(t) \in \mathbb{R}^{n_\xi}$ and $v_\tau(t) \in \mathbb{R}^{n_\tau}$ as unknown disturbance vectors affecting the state update and output equations, respectively, $v_\xi(t) \in \mathcal{V}_\xi$ and $v_\tau(t) \in \mathcal{V}_\tau$, where $\mathcal{V}_\xi \subset \mathbb{R}^{n_\xi}$ and $\mathcal{V}_\tau \subset \mathbb{R}^{n_\tau}$ are unknown hyper-boxes. Our goal is to learn $\hat{F}, \hat{G}$ along with the disturbance sets $\mathcal{V}_\xi, \mathcal{V}_\tau$. For example, if $\hat{F}, \hat{G}$ are linear models, we would get a conservative uncertain discrete-time linear state-space model of (27) with unknown additive disturbances whose bounds are known. Similarly, if $\hat{F}, \hat{G}$ are NNs with ReLU activation functions, we would get an uncertain PWA discrete-time model of (27).

We can solve the above problem by applying the worst-case regression approach of Section II component-wise to the functions F and G, with $x = \text{col}(\xi(t),u(t))$ and $y = f(x) = \xi_j(t+T_s)$ for each state $j = 1,\dots,n_\xi$, and $y = f(x) = \tau_j(t)$ for each output $j = 1,\dots,n_\tau$. Note that evaluating $\xi(t+T_s)$ requires integrating the continuous-time model (27) from t to $t+T_s$ with initial condition $\xi(t)$ and constant input $u(t)$. The uncertainty bounds for each component of F and G can then be obtained by solving (11) and combined to form the hyper-boxes $\mathcal{V}_\xi$ and $\mathcal{V}_\tau$.

B. Approximate mpQP and explicit linear MPC

Multiparametric quadratic programming (mpQP) problems [9]

$$\begin{aligned} z^\star(x) = \arg\min_z \ & \frac{1}{2}z'Qz + (Fx+p)'z \\ \text{s.t. } & Az \le Bx + b, \quad x \in \mathcal{X} \end{aligned} \tag{29}$$

arise in explicit model predictive control (MPC) of linear time-invariant systems [6]. The solution $z^\star(x)$ is a PWA function of the parameter vector $x \in \mathbb{R}^n$ over a polyhedral partition of the parameter set $\mathcal{X} \subseteq \mathbb{R}^n$ in N_r regions, and can be computed offline [9]. As the number of regions N_r grows considerably with the number of constraints, we want to learn a surrogate model $\hat{f}$ of selected components $f(x) \triangleq [I\ 0\ \dots\ 0]z^\star(x)$ of the solution $z^\star$ over $\mathcal{X}$.

A typical reference-tracking formulation used in applications is

$$\begin{aligned} \min_z \ & \sum_{k=0}^{N-1} (\tau_{k+1}-r)'Q_\tau(\tau_{k+1}-r) + \Delta u_k' Q_{\Delta u}\Delta u_k + \rho_2\zeta^2 + \rho_1\zeta \\ \text{s.t. } & \xi_{k+1} = \bar{A}\xi_k + \bar{B}u_k, \ k = 0,\dots,N-1 \\ & \tau_k = \bar{C}\xi_k, \ k = 1,\dots,N \\ & u_k = u_{k-1} + \Delta u_k, \ k = 0,\dots,N-1 \\ & u_{\min} \le u_k \le u_{\max}, \ k = 0,\dots,N_u-1 \\ & \Delta u_{\min} \le \Delta u_k \le \Delta u_{\max}, \ k = 0,\dots,N_u-1 \\ & \tau_{\min} - V_{\min}\zeta \le \tau_k \le \tau_{\max} + V_{\max}\zeta, \ k = 1,\dots,N_c \\ & \Delta u_k = 0, \ k = N_u,\dots,N-1, \ \zeta \ge 0 \end{aligned} \tag{30}$$

where k is the prediction step, $\Delta u_k \in \mathbb{R}^{n_u}$ are the input increments, $(\bar{A},\bar{B},\bar{C})$ is a state-space realization of the model of the controlled process $\xi(t+1) = \bar{A}\xi(t) + \bar{B}u(t)$, $\tau(t) = \bar{C}\xi(t)$, matrix Q_τ is the weight on tracking errors, $Q_\tau = Q_\tau' \succeq 0$, $Q_\tau \in \mathbb{R}^{n_\tau \times n_\tau}$ and $Q_{\Delta u}$ the weight of control increments, $Q_{\Delta u} = Q_{\Delta u}' \succ 0$, $Q_{\Delta u} \in \mathbb{R}^{n_u \times n_u}$; N the prediction horizon, N_u the number of free control moves, N_c the number of output constraints in prediction; $\zeta \in \mathbb{R}$ is a nonnegative slack variable used to soften output constraints via the given ECR (equal concern relaxation) nonnegative vectors $V_{\min}, V_{\max} \in \mathbb{R}^{n_\tau}$, and is penalized with weights $\rho_2, \rho_1 \ge 0$.

Problem (30) can be transformed into the mpQP (29) by defining $x = \text{col}(\xi_0, r, u_{-1})$, where $\xi_0 = \xi(t) \in \mathbb{R}^{n_\xi}$ is the current state vector, $r = r(t) \in \mathbb{R}^{n_\tau}$ is the current output reference value, and $u_{-1} = u(t-1) \in \mathbb{R}^{n_u}$ is the last control input applied to the process, $z = \text{col}(u_0,\dots,u_{N_u-1},\zeta)$ is the sequence of future control moves optimized over the prediction horizon and slack, and $y = f(x)$ is the optimal value of u_0, i.e., the control input $u(t)$ that gets applied to the process.

Let us assume $n_u = 1$ for simplicity of notation; in the case $n_u > 1$ we treat the approximation problem component-wise, i.e., we learn a surrogate model $\hat{f}_i$ for each component $i = 1,\dots,n_u$ of the optimal solution $f(x)$, using the approaches developed in Section II.

Assuming $u_{\min} \le u_{\max}$, $\Delta u_{\min} \le \Delta u_{\max}$, Problem (29) is feasible for all $x \in \mathcal{X}$ and the matrix Q is positive definite, so that the solution $z^\star(x)$ exists and is unique. When no constraints are active at the optimal solution $z^\star(x)$, we have that $f(x)$ coincides with the unconstrained solution

$$w(x) = -[1\ 0\ \dots\ 0]Q^{-1}(Fx+p) \tag{31}$$

for all and only the values of x satisfying

$$H_0 x \le K_0, \quad H_0 \triangleq -AQ^{-1}F - B, \ K_0 \triangleq b + AQ^{-1}p. \tag{32}$$

As we wish the surrogate model $\hat{f}$ to coincide with the unconstrained solution (31) when the active set is empty, we consider the approximations of the indicator function defined in (6) with $g(x) = \bar{H}_0 x - \bar{K}_0$ and $n_h = 0$, where $\{x : \bar{H}_0 x \le \bar{K}_0\}$ is a minimal hyperplane representation of $\{x : H_0 x \le K_0\}$. Moreover, to enforce the constraints on the first control input u_0, we also consider the saturation functions (8) with $y_{\min} = \max(u_{\min}, u_{-1} + \Delta u_{\min})$, $y_{\max} = \min(u_{\max}, u_{-1} + \Delta u_{\max})$.

The approximation error can be seen as an input disturbance $\delta_u \in \mathbb{R}^{n_u}$, $|\delta_u| \le \bar{e}^\star$, affecting the control input $u(t)$ applied to the process. Hence, a possible use of Algorithm 1 is to start from a robust MPC design with given uncertainty $|\delta_u| \le \bar{\delta}_u$, evaluate its approximation, and verify that $\bar{e}^\star \le \bar{\delta}_u$.

VI. Numerical Examples

All numerical tests were run in Python 3.11 with the `maxfit` package on a MacBook Pro with Apple M4 Max (16 CPU cores), using `direct` [14], [15] for global optimization (absolute tolerance 10^{-8}, relative tolerance 10^{-5}, max 2000 evaluations). Unless otherwise specified, $\gamma = 10$ and $r(\theta) = 10^{-8}\|\theta\|^2$.

A. Nonlinear function

We want to approximate the Gaussian function $f(x) = e^{-30((x_1-0.5)^2+(x_2-0.5)^2)}$ on the box $\mathcal{X} = [0,1]^2$ with a PWA one and quantify the WCE. We run Algorithm 1 with an initial dataset of $N_0 = 100$ training points, for $M = 50$ active-learning steps. We consider the class $\mathcal{F}$ of NN models $\mathcal{N}(x;\theta)$ with two layers of 10 and 5 neurons, respectively, while for the input-dependent bound functions $\mathcal{N}(x;\psi)$ we consider two layers of 20 and 10 neurons, respectively, with leaky-ReLU activation function $a(x) = \max(x, 0.1x)$ in both. The CPU time is 114.3 s for running Algorithm 1 (with 106.0 s spent on training the surrogate and 7.8 s in global optimization) and 6.4 s to compute an asymmetric input-dependent confidence interval as described in Section III-C, with $\rho_\psi(\psi) = 10^{-8}\|\psi\|^2$.

The results are shown in Figure 3. Note that, although the MSE is considerably smaller than the WCE, minimizing the WCE, even if related (cf. [24, Th. 1.3]), does not minimize the MSE, as expected.

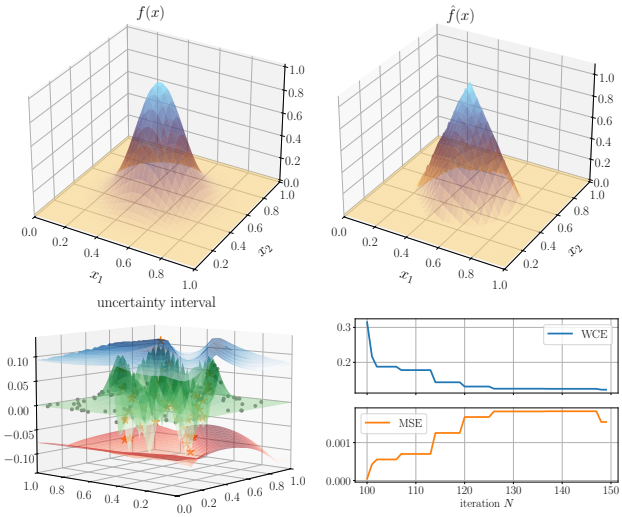

Fig. 3: Gaussian function. Top-left: function $f(x)$; Top-right: learned leaky-ReLU $\hat{f}(x;\theta)$; Bottom-left: pointwise error $e(x) = f(x) - \hat{f}(x;\theta)$, envelope $[e^\star_{\min}(x), e^\star_{\max}(x)]$, actively-learned samples (orange stars) and initial samples (gray dots); Bottom-right: WCE and MSE over active-learning iterations of Algorithm 1.

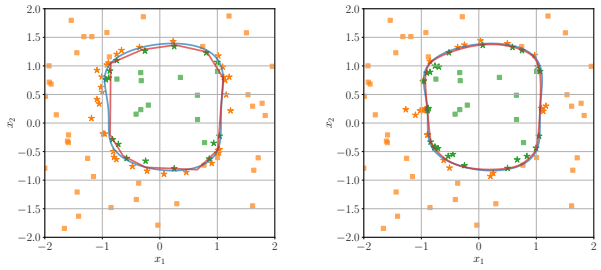

Fig. 4: Convex inner approximation of the nonconvex set $\mathcal{S}$ (blue): polyhedron (left) and input-convex NN (right). Feasible points (green), infeasible points (orange), initial training points (boxes), actively-learned points (stars).

B. Convex inner approximation of a nonconvex set

We want to find a convex inner approximation of the nonconvex set $\mathcal{S} = \{x \in \mathbb{R}^2 : x_1^2 + x_2^4 + \frac{1}{3}x_1^3 - x_2^3 - \frac{1}{2}x_2 \le 1\}$, with $x_1 \in [-2, 2]$, $x_2 \in [-2, 2]$. Following the approach described in Section IV, we run Algorithm 1 with $N_0 = 50$ for $M = 50$ active-learning steps, using $\eta = 10$ and the regularization function $r(\theta) = 10^{-4}\|\theta\|^2$. We consider two different model classes $\mathcal{F}$ for $\hat{f}$: (i) a convex PWA function $\hat{f}(x;\theta) = \max_{i=1\dots n_f} A_i x - b_i$, where $n_f = 10$ and $\theta = \mathrm{col}(A_1^\top, \dots, A_{n_f}^\top, b)$ collects the learnable parameters, which provides the convex polyhedral set $\{x \in \mathbb{R}^2 : Ax \le b - \Delta f + \epsilon_f\}$; (ii) an input-convex NN with 2 hidden layers of 8 neurons each, linear bypass, and softplus activation function $a : \mathbb{R} \to \mathbb{R}$, $\hat{f}(x;\theta) = W_3^2 a(W_2^2 a(V_1 x + b_1) + V_2 x + b_2) + V_3 x + b_3$, where $(\cdot)^2$ denotes component-wise square and ensures nonnegative weights and, therefore, convexity [27]. The given set $\mathcal{S}$ and the convex inner approximations obtained with the two model classes are shown in Figure 4. They are computed in 52.6 s (45.7 s for training, 6.4 s for global optimization) and 156.8 s (146.3 s training / 9.6 s global optimization), respectively, resulting in $\Delta f = -0.07817$ for the polyhedron and $\Delta f = -0.03612$ for the input-convex NN. In both cases, by Proposition 4, any $x \in [-2, 2]^2$ such that $\hat{f}(x;\theta) - \Delta f < 0$ implies that $x \in \mathcal{S}$.

C. Uncertain discrete-time model

We consider the following continuous-time nonlinear model of a pendulum with friction and nonlinear stiffness:

$$\dot{\xi}_1(t) = \xi_2(t)$$
$$\dot{\xi}_2(t) = \frac{1}{J}(u(t) - b\xi_2(t) - mg\ell_c \sin(\xi_1(t)) - k_1\xi_1(t) - k_3\xi_1(t)^3)$$

where $\xi_1(t)$ is the angular position (rad), $\xi_2(t)$ the angular velocity (rad/s), and $u(t)$ the applied torque (Nm), with parameters $J = 0.05$ kgm^2, $b = 0.08$ Nms/rad, $m = 1$ kg, $g = 9.81$ m/s^2, $\ell_c = 0.15$ m, $k_1 = 2$ Nm/rad, $k_3 = 5.0$ Nm/rad^3. We want to learn a discrete-time uncertain model of the form (28) with sampling time $T_s = 0.1$ s for the range $\xi_1(t) \in [-\pi, \pi]$ rad, $\xi_2(t) \in [-5, 5]$ rad/s, $u(t) \in [-2, 2]$ Nm. Algorithm 1 is run with $x = \mathrm{col}(\xi(t), u(t))$ starting from $N_0 = 100$ samples, generated by LHS, for $M = 50$ active-learning steps.

We consider the class $\mathcal{F}$ of shallow NN models $\mathcal{N}(x;\theta)$ with 10 neurons and ReLU activation function plus a linear function of u to approximate the model given by integrating the continuous-time model from t to $t + T_s$ with initial condition $\xi(t)$ and constant input $u(t)$ with Heun's method from the `diffrax` package [17]. The results are shown in Figure 5 for the first state. The CPU

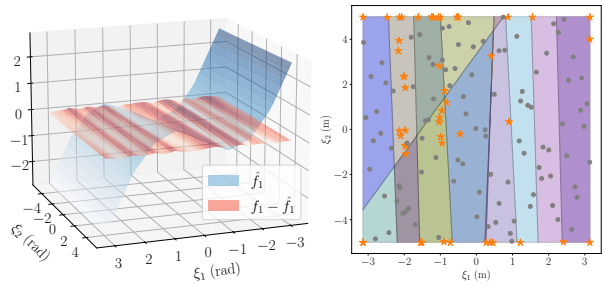

Fig. 5: Pendulum example: discrete-time ReLU mapping for $\xi_1(k+1)$ and approximation error (left), and regions of PWA equivalent of ReLU network (right) at $u = 0$ Nm.

time for running Algorithm 1 is 152.1 s for the first state (105.4 s for training, 45.8 s for global optimization) and 176.3 s for the second state (118.4 s for training, 72.2 s for global optimization). The resulting additive uncertainty bounds are $\mathcal{V}_\xi = [-0.0810, 0.0810] \times [-2.6368, 2.6368]$, corresponding to a maximum relative error of 2.82% for the first state and 5.67% for the second state.

D. Multiparametric quadratic programming

We consider a randomly-generated mpQP problem in the form (29) with $x \in \mathbb{R}^2$, $z \in \mathbb{R}^{10}$, 30 linear inequality constraints, and bound constraints $-z_{\min} \le z_i \le z_{\max}$, $i = 1, \dots, 10$, taking the first component $y = z_1$ of the optimal solution as the value to approximate. The exact mpQP solution $f(x)$, computed using the mpQP solver [3], has 162 polyhedral regions. We consider the class $\mathcal{F}$ of NN models $\mathcal{N}(x;\theta)$ with 5 neurons in each layer, linear bypass in each layer, and ReLU activation function. In order to have that the unconstrained solution (31) of the QP problem is exactly represented by $\hat{f}$ in the corresponding region (32), we set

$$\begin{aligned}\hat{f}(x;\theta) = \min(\max(&\hat{\delta}(x;\beta)\left(-[1\ 0\ \dots\ 0]Q^{-1}(Fx+p)\right)\\ &+(1-\hat{\delta}(x;\beta))\mathcal{N}(x;\theta), z_{\min}), z_{\max})\end{aligned} \tag{33}$$

where we used the saturation function sat as in (8a) with $z_{\min} = -1$, $z_{\max} = 1$, and defined $\hat{\delta}(x;\beta)$ from (6a) as $\hat{\delta}(x;\beta) = \max\left(1 - \beta \max\left(H_0 x - K_0, 0\right), 0\right)$, with H_0, K_0 as in (32) and β is a trainable parameter.

We initialize Algorithm 1 with $N_0 = 20$ samples for $M = 30$ active-learning iterations. Each sample is generated by solving the associated QP problem using the solver [4]. The total CPU time is 158.0 s (147.2 s for training, 10.4 s for global optimization, 7.6 s to compute the asymmetric input-dependent confidence interval), with bound functions $\mathcal{N}(x;\psi)$ also in $\mathcal{F}$. The obtained results are shown in Figure 6. Note that, as shown in the bottom-right plot of Figure 6, the approximation error is close to zero in the region where no constraints of the mpQP are active, as expected.

E. Approximate linear MPC

Consider the problem of controlling the non-minimum phase system with transfer function $G(s) = (s-0.5)/(s^2+0.4s+1)$ from the input u to the output τ, under constraints $-\Delta u_{\max} \le \Delta u(t) \le \Delta u_{\max}$, $\Delta u_{\max} = 0.5$, and $-\tau_{\max} \le \tau(t) \le \tau_{\max}$, $\tau_{\max} = 1.2$, for tracking setpoints $r(t) \in [-1, 1]$. We consider the linear MPC formulation (30) with prediction horizon $N = N_u = N_c = 20$, and weights $Q_\tau = 1$, $Q_{\Delta u} = 0.1$, $\rho_2 = 100$, $\rho_1 = 0$. The resulting mpQP problem (29) has $x \in \mathbb{R}^4$ as parameter vector (2 states, 1 reference, 1 previous input), $z \in \mathbb{R}^{21}$ as optimization vector (future control moves and slack), and 80 linear inequalities.

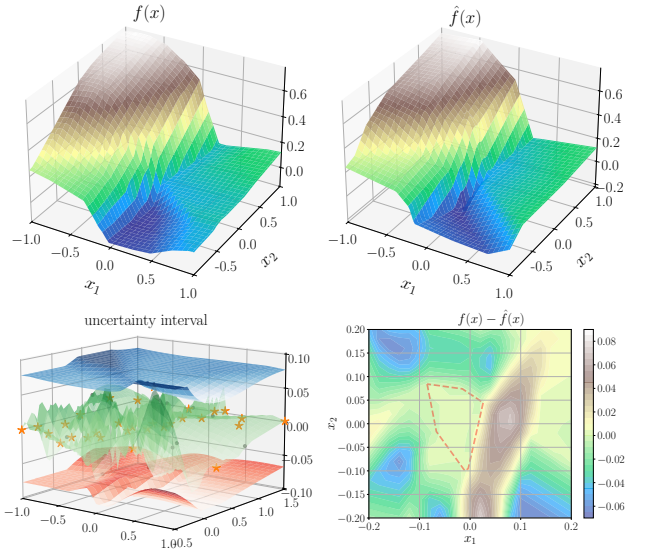

Fig. 6: mpQP example. Top-left: exact QP solution $f(x)$; Top-right: learned ReLU $\hat{f}(x;\theta)$; Bottom-left: pointwise error $e(x) = f(x) - \hat{f}(x;\theta)$ and envelope $[e^\star_{\min}(x), e^\star_{\max}(x)]$; Bottom-right: level sets of $f(x) - \hat{f}(x)$ around the origin and unconstrained-solution region (32) of mpQP problem (dashed-red polygon).

We want to approximate the resulting MPC control law $y = f(x)$ using the approach of Section V-B with an NN model $\mathcal{N}(x;\theta)$ as in (33), except that we use two layers of 20 and 10 neurons, respectively, replace $z_{\min} = -\Delta u_{\max} + u_{-1}$, $z_{\max} = \Delta u_{\max} + u_{-1}$ in the saturation function to enforce input increment constraints, and collect samples to minimize the worst-case approximation error. To this end, we initialize Algorithm 1 with $N_0 = 1000$ samples generated by LHS in $\mathcal{X} = \{x : [-3\ \ -3\ \ -1\ \ -2.5]^\top \le x \le [3\ 3\ 1\ 2.5]^\top\}$, which is an over-estimation of the region of interest for the system based on closed-loop MPC simulations for different reference excitations, for $M = 1000$ active-learning iterations, with regularization $r(\theta) = 10^{-4}\|\theta\|^2$ and $\nu = 10^{-4}$ in (5). Each sample is generated by solving the associated QP problem using the solver [4]. The results, obtained in 2233.3 s (1947.7 s spent on training, 280.9 s in global optimization), are shown in Figure 7. The resulting WCE is $\bar{e}^\star = 6.4602 \cdot 10^{-2}$, while the MSE is $2.8812 \cdot 10^{-4}$.

For comparison, we train the same model *without active learning* using 10^5 randomly-generated samples, with both MSE loss and the loss (3). In the first case, we obtain an MSE = $6.2401 \cdot 10^{-4}$ but a considerably larger WCE $\bar{e} = 0.3989$ over $\mathcal{X}$, while with the loss (3) we get an MSE = $5.2893 \cdot 10^{-4}$ and still a large WCE = 0.3982. The models were trained in 69.81 s and 79.29 s, respectively, using the `jax-sysid` package [8] from 10 different random initializations. This highlights that (i) judging the quality of the approximation based on MSE alone can be misleading, and (ii) an active-learning approach is important when minimizing the WCE.

VII. Conclusions

We have proposed a general active-learning framework for training surrogate models with guaranteed, and possibly minimal, worst-case error bounds, applicable to a broad class of modeling and control tasks. The guaranteed properties of the error bounds critically rely on the global optimizer's ability to find the true global maximum of the approximation error; since global optimization scales poorly with the problem dimension, this may limit the applicability of the

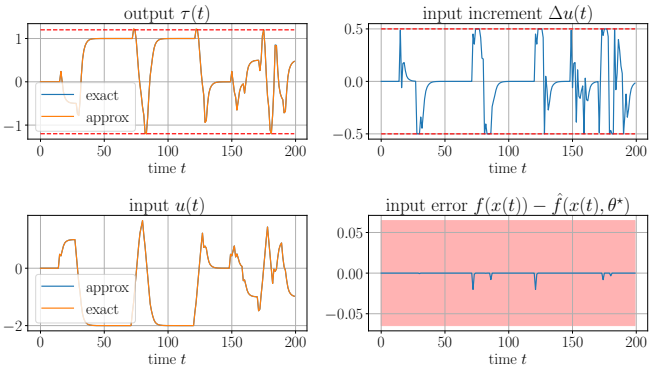

Fig. 7: Linear MPC example: closed-loop simulations under exact (QP-based) MPC control and approximate MPC $\hat{f}(x, \theta^\star)$. The bottom-right plot shows the pointwise error $e(x) = f(x) - \hat{f}(x; \theta^\star)$ and symmetric uniform bound $[-\bar{e}^\star, \bar{e}^\star]$ returned by Algorithm 1.

approach to small-scale problems, unless dedicated global optimizers are used for specific problem classes.

References

[1] H. Alsmeier, L. Theiner, A. Savchenko, A. Mesbah, and R. Findeisen. Imitation learning of MPC with neural networks: Error guarantees and sparsification. In *IEEE 63rd Conference on Decision and Control*, pages 4777–4782, Milano, Italy, 2024.

[2] A.N. Angelopoulos and S. Bates. Conformal prediction: A gentle introduction. *Foundations and Trends in Machine Learning*, 16(4):494–591, 2023.

[3] D. Arnström and D. Axehill. A high-performant multi-parametric quadratic programming solver. 2024. arXiv preprint 2404.05511.

[4] D. Arnström, A. Bemporad, and D. Axehill. A dual active-set solver for embedded quadratic programming using recursive LDL$^{\mathrm{T}}$ updates. *IEEE Transactions on Automatic Control*, 67(8):4362–4369, 2022.

[5] R.F. Barber, E.J. Candès, A. Ramdas, and R.J. Tibshirani. Predictive inference with the jackknife+. *The Annals of Statistics*, 49(1):486–507, 2021.

[6] A. Bemporad. Explicit model predictive control. In J. Baillieul and T. Samad, editors, *Encyclopedia of Systems and Control*, pages 744–751. 2021.

[7] A. Bemporad. Active learning for regression by inverse distance weighting. *Information Sciences*, 626:275–292, May 2023.

[8] A. Bemporad. An L-BFGS-B approach for linear and nonlinear system identification under ℓ_1 and group-lasso regularization. *IEEE Transactions on Automatic Control*, 70(7):4857–4864, 2025.

[9] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

[10] M. Canale, L. Fagiano, and M. Milanese. Fast nonlinear model predictive control via set membership approximation: an overview. In *Proc. 17th IFAC World Congress*, pages 12165–12170. Seoul, Korea, July 2008.

[11] S. Chen, K. Saulnier, N. Atanasov, D. Lee, V. Kumar, G. Pappas, and M. Morari. Approximating explicit model predictive control using constrained neural networks. pages 1520–1527. American Control Conference, 2018.

[12] 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*, 67(1):1–15, 2022.

[13] P.J. Huber and E.M. Ronchetti. *Robust Statistics*. Wiley, Hoboken, NJ, 2nd edition, 2009.

[14] D.R. Jones and J.R.R.A. Martins. The DIRECT algorithm: 25 years later. *Journal of Global Optimization*, 79(3):521–566, 2021.

[15] D.R. Jones, M. Schonlau, and W.J. Matthias. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

[16] B. Karg and S. Lucia. Efficient representation and approximation of model predictive control laws via deep learning. *IEEE Transactions on Cybernetics*, 50(9):3866–3878, 2020.

[17] P. Kidger. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.

[18] D.C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[19] D. Masti, F. Fabiani, G. Gnecco, and A. Bemporad. Counter-example guided inductive synthesis of control Lyapunov functions for uncertain systems. *IEEE Control Systems Letters*, 7:2047–2052, 2023.

[20] M.D. McKay, R.J. Beckman, and W.J. Conover. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[21] M. Milanese and C. Novara. Set membership identification of nonlinear systems. *Automatica*, 40(6):957–975, 2004.

[22] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

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

[24] M.J.D. Powell. *Approximation Theory and Methods*. Cambridge University Press, Cambridge, UK, 1981.

[25] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

[26] L.M. Rios and N.V. Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293, 2013.

[27] M. Schaller, A. Bemporad, and S. Boyd. Learning parametric convex functions. 2025. arXiv preprint 2506.04183.

[28] B. Settles. Active learning. In *Synthesis Lectures on Artificial Intelligence and Machine Learning*, number 18. Morgan & Claypool Publishers, 2012.

Appendix

A. Proof of Proposition 1

Consider any given vector $\theta \in \mathbb{R}^{n_\theta}$, let $e_k(\theta) \triangleq y_k - \hat{f}(x_k; \theta)$, $\tilde{e}(\theta) \triangleq \max_k |e_k(\theta)|$, and let $\tilde{K}(\theta)$ be the set of indices k such that $|e_k(\theta)| = \tilde{e}(\theta)$. Denote by $\#\tilde{K}(\theta)$ the cardinality of $\tilde{K}(\theta)$ (the set $\tilde{K}(\theta)$ may contain more than one element, in general), and let $c \triangleq \#\tilde{K}$ if $\tilde{e} > 0$ or $c \triangleq 2\#\tilde{K}$ if $\tilde{e} = 0$.

By taking the limit $\gamma \to +\infty$ of the cost function in (3), without regularization and dropping the dependence on θ for simplicity of notation, we get $\lim_{\gamma\to+\infty} \frac{1}{\gamma} \log \left(\sum_{k=1}^{N} e^{\gamma e_k} + e^{-\gamma e_k} \right)$ $= \lim_{\gamma\to+\infty} \tilde{e} + \frac{1}{\gamma} \log \left(\sum_{k=1}^{N} \left(e^{\gamma(e_k - \tilde{e})} + e^{\gamma(-e_k - \tilde{e})} \right) \right) =$ $\lim_{\gamma\to+\infty} \tilde{e} + \frac{1}{\gamma} \log \left(c + \sum_{k \notin \tilde{K}(\theta)} \left(e^{\gamma(e_k - \tilde{e})} + e^{\gamma(-e_k - \tilde{e})} \right) \right)$ $= \tilde{e} = \max_k |e_k|$. ∎

B. Proof of Proposition 2

Consider the scalar case $y \in \mathbb{R}$ for simplicity and a generic $\eta > 0$. Clearly, by setting $\bar{y} \triangleq \frac{(y_{\max} - y_{\min})}{2}$, we get $\sigma_\eta \left(\frac{y_{\min} + y_{\max}}{2}; y_{\min}, y_{\max} \right) = y_{\max} + \frac{1}{\eta} \log \left(\frac{1 + e^{-\eta \bar{y}}}{1 + e^{\eta \bar{y}}} \right) =$ $y_{\max} + \frac{1}{\eta} \log \left(e^{-\eta \bar{y}} \right) = y_{\max} - \bar{y} = \frac{(y_{\max} + y_{\min})}{2}$. The derivative of $\sigma_\eta(y; y_{\min}, y_{\max})$ with respect to y is $\sigma'_\eta(y; y_{\min}, y_{\max})$ $= \frac{e^{\eta(y - y_{\min})}(1 - e^{-\eta(y_{\max} - y_{\min})})}{(1 + e^{\eta(y - y_{\min})})(1 + e^{\eta(y - y_{\max})})}$ whose denominator is positive. Since $y_{\max} > y_{\min}$ and $\eta > 0$, its numerator is also positive and, hence, $\sigma'_\eta(y; y_{\min}, y_{\max}) > 0$ for all $y \in \mathbb{R}$, which implies that $\sigma_\eta(y; y_{\min}, y_{\max})$ is monotonically strictly increasing with respect to y. Moreover, the limits of $\sigma_\eta(y; y_{\min}, y_{\max})$ in (8b) are $\lim_{y\to-\infty} \sigma_\eta(y; y_{\min}, y_{\max}) = y_{\max} + \frac{\log(e^{\eta(y_{\min} - y_{\max})})}{\eta} =$ $y_{\min}$, $\lim_{y\to+\infty} \sigma_\eta(y; y_{\min}, y_{\max}) = y_{\max} + \frac{\log(1)}{\eta} = y_{\max}$. If $\sigma_\eta(y; y_{\min}, y_{\max}) > y_{\max}$ or $\sigma_\eta(y; y_{\min}, y_{\max}) < y_{\min}$ were satisfied for some $y \in \mathbb{R}$, then σ would not be monotonically increasing with respect to y, which is a contradiction. Finally, for any $y \in (y_{\min}, y_{\max})$, it is immediate to verify that $\lim_{\eta\to\infty} \sigma'_\eta(y; y_{\min}, y_{\max}) = 1$. ∎