Stability Verification of Neural Network Controllers using Mixed-Integer Programming

Roland Schwan[1,2], Colin N. Jones[1], and Daniel Kuhn[2]

***Abstract*— We propose a framework for the stability verification of Mixed-Integer Linear Programming (MILP) representable control policies. This framework compares a fixed candidate policy, which admits an efficient parameterization and can be evaluated at a low computational cost, against a fixed baseline policy, which is known to be stable but expensive to evaluate. We provide sufficient conditions for the closed-loop stability of the candidate policy in terms of the worst-case approximation error with respect to the baseline policy, and we show that these conditions can be checked by solving a Mixed-Integer Quadratic Program (MIQP). Additionally, we demonstrate that an outer approximation of the stability region of the candidate policy can be computed by solving an MILP. The proposed framework is sufficiently general to accommodate a broad range of candidate policies including ReLU Neural Networks (NNs), optimal solution maps of parametric quadratic programs, and Model Predictive Control (MPC) policies. We also present an open-source toolbox in Python based on the proposed framework, which allows for the easy verification of custom NN architectures and MPC formulations. We showcase the flexibility and reliability of our framework in the context of a DC-DC power convertor case study and investigate the computational complexity.**

I. Introduction

MPC has been extremely successful in control applications for refineries and chemical plants [1], building control [2], the control of quadcopters [3], robotics [4], and power electronics [5], [6]. The main advantages of MPC are its versatility, stability, and ability to account for input and state constraints. With the transition from the process industry to robotics and power electronics, the sampling times have decreased from hours to only a few milli- or even microseconds. This is especially challenging if one wants to deploy controllers on embedded systems with low computational resources and limited memory.

Hence, ideally, one would like to perform all heavy computations offline and precompute the optimal control law $\psi^\star(\cdot)$ that maps any feasible state to an optimal control input. A well-known technique to compute $\psi^\star(\cdot)$ such an optimal control law is *explicit* MPC [7]. For MPC controllers with quadratic cost functions and linear dynamics, $\psi^\star(\cdot)$ is a piecewise affine function defined over a polyhedral partition of the state space. Thus, in contrast to *implicit* MPC, which computes the optimal control input online by solving a different optimization problem for each state, *explicit* MPC precomputes the optimal affine control policy across predefined polytopic regions of the state space. Unfortunately, the required number of polytopes explodes with the dimension of the state and the number of constraints, which render explicit MPC intractable for larger systems. Additionally, the online search for the polytope containing the current state may require excessive processing power or storage space. Although these computational challenges can be mitigated by using search trees [8] or hash tables [9], the required memory may remain too large [10], especially for embedded systems. Recent attempts to approximate explicit MPC directly either suffer from a curse of dimensionality [11], or they rely on expensive set projections [12], [13].

The limited scalability of explicit MPC and its variants has promoted interest in general function approximators of MPC policies, such as deep NNs [14]. Deep NNs are attractive because they can exactly represent predictive controllers for linear systems [15], while being relatively inexpensive during online inference [16]. Unfortunately, one loses the stability guarantees for the learned controllers. Statistical methods are one way to verify the stability of the learned controllers [17], with extensions to filter out severely suboptimal control inputs with high probability [18].

However, NNs are not the only viable function approximators. In fact, every continuous nonlinear control law can be represented as the minimizer mapping of a parametric convex program [19]. Even the solutions of parametric linear programs (LPs) can represent any continuous piecewise affine function [20]. Hence, parametric optimization problems give rise to implicit function approximators, and by leveraging the implicit function theorem, one can directly optimize and learn the underlying problem parameter settings. Approaches for calibrating parametric quadratic programs (QPs), general convex programs, and root finding problems are described in [21], [22], and [23] respectively.

NN control policies have been successfully employed in applications for controlling chemical plants [24], robotic arms [25] or DC-DC power electronic converters [6] at a significant increase in computational speed. Thus, NN based controllers can not only result in better control performance, because of a tighter control loop, but also in drastic savings of computational resources.

This work was supported by the Swiss National Science Foundation under the NCCR Automation project, grant agreement 51NF40_180545.

We thank Silvia Mastellone for valuable feedback on an earlier version of this paper, and Emilio Maddalena for providing the experimental setup of the DC-DC power convertor.

[1]Roland Schwan and Colin N. Jones are with the Automatic Control Lab, EPFL, Switzerland.

[2]Roland Schwan and Daniel Kuhn are with the Risk Analytics and Optimization Chair, EPFL, Switzerland.

{roland.schwan, colin.jones, daniel.kuhn}@epfl.ch

A. Related Works

The idea to approximate predictive controllers with NNs enjoys growing popularity. Some of the earliest work dates back to the 90s, when single hidden layer NN approximations were used to learn a nonlinear MPC policy [26]. More recently, plain vanilla NN architectures have been enhanced with parametric QP implicit layers [27]. However, closed-loop stability can also be guaranteed by projecting the output of the NN into a safety set [13]. This projection can be viewed as a "Safety Filter" [28]. However, a computationally expensive optimization problem has to be solved online.

The satisfaction of the closed-loop state/input constraints and stability requirements can also be verified by performing a reachability analysis using an MILP representation of the NN controller [29] or by over approximating the NN output bounds via semidefinite programming formulations [30].

Other approaches show closed-loop stability by learning and verifying Lyapunov functions [31], [32]. This leads to a learner/verifier pattern to find a suitable Lyapunov function. But this can be demanding, since at every step adversarial points have to be found (verifier), and a new Lyapunov function candidate must then be found, without guarantees of convergence.

Alternatively, one can guarantee stability by combining the worst-case approximation error of the NN controller with its Lipschitz constant, utilizing an MILP framework [33]. In this paper, we propose direct sufficient conditions for closed-loop stability, which can be verified by solving an MIQP. Compared to the work in [33] we show imperially that our approach is less conservative and that solving an MIQP does necessarily result in longer solve times.

B. Contributions

We introduce a general framework for the verification of NN controllers via mixed-integer programming (MIP) together with an open source toolbox.[1] Specifically, we introduce the concept of MILP representable verification problems which allows us to model a variety of common components like ReLU NNs, and parametric quadratic programs which include MPC policies. Given a baseline policy (e.g., an MPC) and an approximate policy (e.g., a NN), we propose two main approaches to verify closed-loop stability of the approximate policy

- by computing the worst-case approximation error. Utilizing robust MPC variants like Tube MPC [34], we can guarantee closed-loop stability by bounding the approximation error inside the disturbance set of the robust MPC scheme.
- by providing sufficient conditions on closed-loop stability, which can be directly formulated and checked by solving an MIQP. We show that our conditions are less conservative in practice, compared to previous methods, i.e., are able to verify the closed-loop stability of more approximate policies. Additionally, we show that an outer approximation of the stability region of the approximate control scheme can be calculated by solving MILPs.

We then exemplify the application of the proposed method on a case-study of a DC-DC power convertor using different NN architectures and MPC formulations. We compare our methods against other approaches and show the superiority of our formulation. Additionally, we provide numerical experiments to show the performance of our approach.

The toolbox is written in Python and allows the automatic transcription of the discussed verification problems as MIPs. It can directly import NN architectures from Pytorch [35], and parametric QPs from CVXPY [36]. This not only allows for a simplified problem formulation, but also a tight integration with the existing machine learning ecosystem.

Notation

We denote the set of real numbers by $\mathbb{R}$, the set of n-dimensional real-valued vectors by $\mathbb{R}^n$ and the set of $n \times m$-dimensional real-valued matrices by $\mathbb{R}^{n\times m}$. Furthermore, we denote the subspace of symmetric matrices in $\mathbb{R}^{n\times n}$ by $\mathbb{S}^n$ and the cone of positive semi-definite and definite matrices by $\mathbb{S}^n_+$ and $\mathbb{S}^n_{++}$, respectively. We use I_n to denote the n-dimensional identity matrix, $\mathbf{1}_n$ to denote the n-dimensional column vector of ones, and $\mathrm{diag}(\cdot)$ to represent the mapping that transforms a column vector to the corresponding diagonal matrix. Given two sets A and B, we denote their Minkowski sum as $A \oplus B \coloneqq \{a + b \mid a \in A, b \in B\}$ and their Pontryagin difference as $A \ominus B \coloneqq \{a \mid a \oplus B \subseteq A\}$. The interior of a set S is denoted by $\mathrm{int}(S)$.

II. MILP Representable Verification Problems

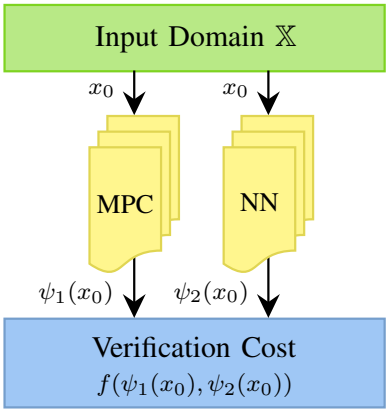

Fig. 1: Schematic of the verification approach.

A verification problem consists of a baseline control policy $\psi_1(x)$ (e.g., an MPC policy) and an approximate policy $\psi_2(x)$ (e.g., a NN) on a common input domain $\mathbb{X}$. The outputs of the policies then enter a verification cost function $f(\psi_1(x), \psi_2(x))$ that represents the objective of the verification problem. An overview of the proposed architecture is shown in Figure 1.

Definition 1: A function $\psi : \mathbb{X} \to \mathbb{U}$ with domain $\mathbb{X} \subseteq \mathbb{R}^n$ and range $\mathbb{U} \subseteq \mathbb{R}^m$ is MILP representable if there exists a

[1]The toolbox can be accessed under the following link: `https://github/PREDICT-EPFL/evanqp`

polyhedral set $P \subseteq \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^c \times \mathbb{R}^b$ such that $(x,u) \in \text{gr}(\psi)$ if and only if there exists $z \in \mathbb{R}^c$ and $\beta \in \{0,1\}^b$ such that $(x,u,z,\beta) \in P$.

A wide range of functions are MILP representable. In fact, one can show that the MILP representable functions are dense in the family of continuous functions on a compact domain $\mathbb{X}$ with respect to the supremum norm. Examples of MILP representable functions include the optimal solution map of parametric QPs (e.g., MPC policies), ReLU NNs, and piecewise affine functions. Additionally, compositions of MILP representable functions are also MILP representable.

Lemma 1: If two functions $\psi_i : \mathbb{X}_i \to \mathbb{U}_i,\ i = 1,2$ with $\mathbb{X}_2 \subseteq \mathbb{U}_1$ are both MILP representable, then the composition $\psi_1 \circ \psi_2 : \mathbb{X}_1 \to \mathbb{U}_2$ is MILP representable.

Proof: Let $P_1 \subseteq \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^{c_1} \times \mathbb{R}^{b_1}$ and $P_2 \subseteq \mathbb{R}^m \times \mathbb{R}^o \times \mathbb{R}^{c_2} \times \mathbb{R}^{b_2}$ be the polyhedral sets corresponding to the MILP representations of ψ_1 and ψ_2 respectively. Then, $(x_i,u_i) \in \text{gr}(\psi_i)$ if and only if there exist $z_i \in \mathbb{R}^{c_i}$ and $\beta_i \in \{0,1\}^{b_i}$ such that $(x_i,u_i,z_i,\beta_i) \in P_i$ for $i = 1,2$. To compute the composition of ψ_1 and ψ_2, we now set the output u_1 of ψ_1 equal to the input x_2 of ψ_2. We can define new variables $z = (z_1,z_2,u_1)$ and $\beta = (\beta_1,\beta_2)$. Hence, by construction, there exists a polyhedral set $P \subseteq \mathbb{R}^n \times \mathbb{R}^o \times \mathbb{R}^{m+c_1+c_2} \times \mathbb{R}^{b_1+b_2}$ such that $(x_1,u_2) \in \text{gr}(\psi_1 \circ \psi_2)$ if and only if $(x_1,u_2,z,\beta) \in P$, showing that the composition $\psi_1 \circ \psi_2$ is MILP representable. ∎

In order to formalize the verification problem for comparing the approximate control policy $\psi_2(\cdot)$ against the baseline policy $\psi_1(\cdot)$, define $\mathbb{X}$ as the relevant input domain of the control polices. For any property to be verified, we can pose the verification problem as the task of certifying the non-negativity of a function $f(\cdot)$ over the input domain $\mathbb{X}$:

$$0 \leq \min_{x_0 \in \mathbb{X}} f(\psi_1(x_0), \psi_2(x_0)). \tag{1}$$

Proposition 1: If $\mathbb{X}$ is a polyhedron and if ψ_1, ψ_2, and f are MILP representable, then the verification problem on the right-hand side of (1) can be posed as the following optimization problem, which is equivalent to an MILP:

$$\begin{aligned} \min_{\tau,x_0,u_1,u_2} \quad & \tau \\ \text{s.t.} \quad & x_0 \in \mathbb{X}, \\ & (x_0,u_1) \in \text{gr}(\psi_1) \\ & (x_0,u_2) \in \text{gr}(\psi_2) \\ & (u_1,u_2,\tau) \in \text{gr}(f). \end{aligned} \tag{2}$$

Proof: The equivalence of the optimization problem in (1) and (2) is immediate. In addition, one readily verifies that problem (2) is equivalent to an MILP because $\mathbb{X}$ is a polyhedron and f, ψ_1 and ψ_2 are MILP representable. ∎

In the next sections, we consider specific MILP representable functions relevant for control.

A. ReLU Neural Networks

We now demonstrate that NNs with ReLU activation functions are MILP representable. A fully-connected feed-forward NN $\psi : \mathbb{X} \subseteq \mathbb{R}^n \to \mathbb{U} \subseteq \mathbb{R}^m$ is a function representable as $\psi = \psi_\ell \circ \psi_{\ell-1} \circ \cdots \circ \psi_1$, where $\psi_i : \mathbb{X}_i \subseteq \mathbb{R}^{n_{i-1}} \to \mathbb{U}_i \subseteq \mathbb{R}^{n_i}$ corresponding to the i-th layer is defined through the componentwise maximum

$$\psi_i(z_{i-1}) = \text{ReLU}(z_{i-1}) = \max(0, W_i z_{i-1} + b_i) \tag{3}$$

for some weight matrix $W_i \in \mathbb{R}^{n_i \times n_{i-1}}$ and bias vector $b_i \in \mathbb{R}^{n_i}$, $i = 1,\ldots,\ell$, and where $n_0 = n$ is the input dimension and $n_\ell = m$ is the output dimension of the NN.

To prove that ψ is MILP representable, we use the piecewise linearity of the ReLU activation functions to represent them via linear constraints and binary decision variables indicating which piece is active [37], [38].

Lemma 2: If $\mathbb{X}$ is a compact polyhedron, then the ReLU NN $\psi : \mathbb{X} \to \mathbb{U}$ is MILP representable, and there exist constants $\underline{m}_i$, $\overline{m}_i$ such that

$$\text{gr}(\psi) = \left\{ (x,u) \;\middle|\; \begin{array}{l} \forall i = 1,\ldots,n-1, \\ \exists \beta_i \in \{0,1\}^{n_i}, \\ \exists z_i \in \mathbb{R}^{n_i} : \\ \quad z_0 = x, \\ \quad z_0 \in \mathbb{X}, \\ \quad z_i \geq 0 \\ \quad z_i \geq W_i z_{i-1} + b_i, \\ \quad z_i \leq W_i z_{i-1} + b_i \\ \qquad - \text{diag}(\overline{m}_i)(\mathbf{1} - \beta_i), \\ \quad z_i \leq \text{diag}(\underline{m}_i)\beta_i, \\ \quad u = W_n z_{n-1} + b_n \end{array} \right\}. \tag{4}$$

Proof: As $\mathbb{X}$ is bounded, there exist $\underline{x}, \overline{x} \in \mathbb{R}^n$ such that $\mathbb{X} \subseteq [\underline{x}, \overline{x}]$. Assume first that $\psi : \mathbb{X} \to \mathbb{U}$ is a single ReLU neuron defined through $u = \max(0,x)$. Then the graph of ψ can be represented via mixed-integer constraints as follows.

$$\text{gr}(\psi) = \left\{ (x,u) \;\middle|\; \begin{array}{l} \exists \beta \in \{0,1\}^n : \\ \quad x \in \mathbb{X}, \\ \quad u \geq 0, \\ \quad u \geq x, \\ \quad u \leq x - \text{diag}(\underline{x})(1-\beta), \\ \quad u \leq \text{diag}(\overline{x})\beta \end{array} \right\} \tag{5}$$

Here, the binary decision variable β indicates whether the neuron is active ($u = x$) or inactive ($u = 0$). Extending this reasoning to the full NN and given the element-wise upper and lower bounds $\underline{m}_i \leq W_i z_{i-1} + b_i \leq \overline{m}_i$ on the input of the i-th activation layer, we can use Lemma 1 to derive the graph representation (4), where the binary decision variables β_i correspond to layer i. The constants $\underline{m}_i$ and $\overline{m}_i$ always exist because $\mathbb{X}$ is compact, ψ_i is continuous for every $i = 1,\ldots,\ell$, and continuous images of compact sets are compact. ∎

Remark 1: The bounds $\underline{m}_i$ and $\overline{m}_i$ can be computed using interval arithmetic [39], zonotope propagation [40], or linear programming [38]. State-of-the-art MILP solvers such as Gurobi [41] are based on branch-and-cut methods. Hence, tightening the bounds $\underline{m}_i$ and $\overline{m}_i$ can reduce computation

times in practice since branches can be pruned more efficiently. If $\underline{m}_i$ and $\overline{m}_i$ can not be calculated exante, e.g., if $\mathbb{X}$ is unbounded, then $\underline{m}_i$ and $\overline{m}_i$ correspond to the usual "big-M" constants and have to be determined endogenously.

Remark 2: Note that Lemma 2 extends to NNs with general piecewise linear activation functions such as leaky ReLUs. Note also that the convex hull of $\mathrm{gr}(\psi)$ is a strict subset of the polyhedron obtained by relaxing $\beta_i \in [0,1]$ in (4). The approximation quality can be improved by adding valid cuts [42].

B. Box Saturation

Control policies are often subject to physical constraints. To enforce these constraints, one may project the output of an approximate policy to the feasible set. For example, the projection ψ onto the box $[\underline{x}, \overline{x}] \subseteq \mathbb{R}^m$ with $\underline{x} \leq \overline{x}$ can be viewed as a vector of element-wise projections of the form

$$\psi(x)_i = \begin{cases} \underline{x}_i \text{ if } u_i < \underline{x}_i, \\ u_i \text{ if } \underline{x}_i \leq u_i \leq \overline{u}_i, \\ \overline{x}_i \text{ if } u_i > \overline{x}. \end{cases} \tag{6}$$

Lemma 3: The projection ψ onto the box $[\underline{x}, \overline{x}] \subseteq \mathbb{R}^m$ is MILP representable.

Proof: The projection ψ defined through (6) can be rewritten in the following form using the ReLU function:

$$\psi(x) = \overline{x} - \mathrm{ReLU}(\overline{x} - (\mathrm{ReLU}(x - \underline{x}) + \underline{x})). \tag{7}$$

Hence, ψ can be recognized as a special instance of a ReLU NN, which is MILP representable according to Lemma 2. ■

C. Parametric QPs

We now show that the optimal solution mappings of parametric QPs are MILP representable. Specifically, we define $\psi(x)$ as the unique minimizer of the parametric QP

$$\begin{aligned} \min_{z} \quad & \frac{1}{2} z^T P z + (Qx + q)^T z \\ \text{s.t.} \quad & Az = Bx + b, \\ & Fz \leq Gx + g, \end{aligned} \tag{8}$$

with internal decision variable $z \in \mathbb{R}^{n_z}$ and matrices $P \in \mathbb{S}^{n_z}_{++}$, $Q \in \mathbb{R}^{n_z \times n}$, $q \in \mathbb{R}^{n_z}$, $A \in \mathbb{R}^{n_{\text{eq}} \times n_z}$, $B \in \mathbb{R}^{n_{\text{eq}} \times n}$, $b \in \mathbb{R}^{n_{\text{eq}}}$, $F \in \mathbb{R}^{n_{\text{ineq}} \times n_z}$, $G \in \mathbb{R}^{n_{\text{ineq}} \times n}$, and $g \in \mathbb{R}^{n_{\text{ineq}}}$.

Lemma 4: If the parametric QP (8) is feasible for every $x \in \mathbb{X}$, then its optimal solution mapping $\psi(x)$ is MILP representable.

Proof: Since the feasible set of problem (8) is a polyhedron, it admits a Slater point whenever it is non-empty. Since the objective function of (8) is also strictly convex, $z^\star(x)$ for $x \in \mathbb{X}$ is uniquely determined by the necessary and sufficient Karush-Kuhn-Tucker (KKT) conditions [43]

$$\text{Primal feasibility} \quad \lfloor Az = Bx + b, \quad Fz \leq Gx + g, \tag{9a}$$

$$\text{Stationarity} \quad \lfloor Pz + (Qx + q) + A^T \mu + F^T \lambda = 0, \tag{9b}$$

$$\text{Dual feasibility} \quad \lfloor \lambda \geq 0, \tag{9c}$$

$$\text{Complementarity} \quad \lfloor \lambda^T (Fz - Gx - g) = 0, \tag{9d}$$

where $\mu \in \mathbb{R}^{n_{\text{eq}}}$ and $\lambda \in \mathbb{R}^{n_{\text{ineq}}}$ are the Lagrange multipliers associated with the equality and inequality constraints, respectively. By introducing a binary decision variable β_i that evaluates to 0 if $F_i z < g_i$ and to 1 if $\lambda_i > 0$ for each $i = 1, \ldots, n_{\text{ineq}}$, we can linearize the complementarity condition as

$$\left.\begin{array}{l} 0 \leq \lambda_i \leq M \beta_i, \\ 0 \leq g_i - F_i z \leq M(1 - \beta_i) \end{array}\right\} \forall i = 1, \ldots, n_{\text{ineq}}, \tag{10}$$

where M is a suitable "big-M" constant [44]. Therefore, the graph of ψ can be represented as

$$\mathrm{gr}(\psi) = \left\{ (x, z) \;\middle|\; \begin{array}{l} \forall i = 1, \ldots, n_{\text{ineq}} \\ \exists \beta \in \{0,1\}^{n_{\text{ineq}}}, \\ \exists z \in \mathbb{R}^{n_z}, \\ \exists \lambda \in \mathbb{R}^{n_{\text{ineq}}}, \\ \exists \mu \in \mathbb{R}^{n_{\text{eq}}} : \\ \quad x \in \mathbb{X}, \\ \quad 0 = Az - Bx - b, \\ \quad 0 = Pz + (Qx + q) + A^T \mu \\ \qquad + F^T \lambda, \\ \quad 0 \leq \lambda_i \leq M \beta_i, \\ \quad 0 \leq g_i + (Gx)_i - F_i z \\ \qquad \leq M(1 - \beta_i) \end{array} \right\}. \tag{11}$$

■

Remark 3: Certain MILP solvers accept logical constraints and automatically transform them into "big-M" constraints or special-ordered set constraints, with the advantage that the constraints are optimally chosen by the solver, leading to faster convergence. Tight "big-M" bounds suitable for numerical purposes can also be calculated by solving auxiliary LPs. Details are relegated to Appendix I.

D. Piecewise Affine Functions over Polyhedral Sets

Piecewise affine functions defined over polyhedral sets lend themselves for modeling piecewise affine hybrid dynamical systems [45]. A continuous function $\psi : \mathbb{X} \to \mathbb{U}$ is called piecewise affine if there exist polyhedra $\mathbb{X}_i = \{x \in \mathbb{X} \mid F_i x \leq g_i\}$, $i \in \mathcal{I}$, with $\mathbb{X} = \bigcup_{i \in \mathcal{I}} \mathbb{X}_i$ as well as matrices $A_i \in \mathbb{R}^{m \times n}$ and vectors $c_i \in \mathbb{R}^n$, $i \in \mathcal{I}$, such that

$$\psi(x) = A_i x + c_i \quad \forall x \in \mathbb{X}_i, \; \forall i \in \mathcal{I}, \tag{12}$$

where $\mathcal{I}$ is a finite index.

Lemma 5: If $\mathbb{X}$ is a compact polyhedron, then the piecewise affine function defined in (12) is MILP representable.

Proof: By [45, Theorem 3.5] there exist binary variables $\beta_i \in \{0,1\}$, $i \in \mathcal{I}$, such that

$$F_i x_i \leq \beta_i g_i \quad \forall i \in \mathcal{I}, \tag{13a}$$

$$1 = \sum_{i \in \mathcal{I}} \beta_i,\ x = \sum_{i \in \mathcal{I}} x_i, \tag{13b}$$

$$\psi(x) = \sum_{i \in \mathcal{I}} (A_i x_i + \beta_i c_i). \tag{13c}$$

Note first that $\mathbb{X}_i \subseteq \mathbb{X}$ inherits compactness from $\mathbb{X}$. The constraint (13a) implies that $x_i = 0$ whenever $\beta_i = 0$. Indeed, if $\beta_i = 0$, then (13a) forces x_i to fall within the recession cone of the compact polyhedron $\mathbb{X}_i$, which coincides with the singleton $\{0\}$. If $\beta_i = 1$, on the other hand, then (13a) forces x_i to fall within $\mathbb{X}_i$. The graph of ψ, thus, admits the following MILP representation:

$$\mathrm{gr}(\psi) = \left\{ (x,u) \;\middle|\; \begin{array}{ll} \exists \beta_i \in \{0,1\} & \forall i \in \mathcal{I}, \\ \exists x_i \in \mathbb{R}^n & \forall i \in \mathcal{I}: \\ \quad x \in \mathbb{X}, & \\ F_i x_i \leq \beta_i g_i & \forall i \in \mathcal{I}, \\ \quad 1 = \sum_{j \in \mathcal{I}} \beta_j, & \\ \quad x = \sum_{j \in \mathcal{I}} x_j, & \\ \quad u = \sum_{j \in \mathcal{I}} (A_j x_j + \beta_j c_j) & \end{array} \right\}. \tag{14}$$

■

III. Approximation Error Computation

Consider now a verification problem of the form (1) with baseline policy $\psi_1(\cdot)$ and approximate policy $\psi_2(\cdot)$, and set the error function $f(\cdot)$ to the infinity norm $\|\cdot\|_\infty$. The worst-case error over a bounded polytopic input set $\mathbb{X}$ is given by

$$\gamma = \max_{x \in \mathbb{X}} \|\psi_1(x) - \psi_2(x)\|_\infty. \tag{15}$$

The infinity norm is a natural choice for measuring the mismatch between ψ_1 and ψ_2, and it is MILP representable.

Lemma 6: The infinity norm $f(t) = \|t\|_\infty$ is MILP representable on any bounded polytopic set $\mathbb{X} \subseteq \mathbb{R}^m$.

Instead of the infinity norm, one could also use the 1-norm to quantify the worst-case error in (15).

Corollary 1: The 1-norm $f(t) = \|t\|_1$ is MILP representable on any bounded polytopic set $\mathbb{X} \subseteq \mathbb{R}^m$.

The proofs of Lemma 6 and Corollary 1 are quite standard and can be found in Appendix II-A and II-B respectively.

Lemma 6 implies via Proposition 1 that if $\psi_1(x)$ and $\psi_2(x)$ are MILP representable, then the worst-case approximation error (15) can be computed by solving an MILP.

Remark 4: By solving $2m$ MILPs it is also possible to calculate a tight bounding box $[\underline{\gamma}, \overline{\gamma}] \subseteq \mathbb{R}^m$ that covers the approximation error $\psi_1(x) - \psi_2(x)$ for every $x \in \mathbb{X}$. Indeed, to compute the components of $\underline{\gamma}$, we solve m MILPs of the form

$$\begin{aligned} \underline{\gamma}_i = \min_{x,u_1,u_2} \quad & e_i^T (u_1 - u_2) \\ \text{s.t.} \quad & x \in \mathbb{X}, \\ & (x,u_1) \in \mathrm{gr}(\psi_1), \\ & (x,u_2) \in \mathrm{gr}(\psi_2), \end{aligned} \tag{16}$$

where $e_i \in \mathbb{R}^m$ is the i-th vector of the canonical basis. To compute the components of $\overline{\gamma}$, we simply convert the minimization to a maximization in (16).

A. Verification Against Robust MPC

We can use the worst-case approximation error (15) to verify an approximate MPC scheme against a robustly stable MPC policy. Let $\psi_1(x)$ be a control policy that is robust against input disturbances in the set $\mathbb{W} = \{w \in \mathbb{R}^m \mid \|w\|_\infty \leq \hat{\gamma}\}$. An example for such a control policy would be Tube MPC [34], which is also MILP representable as we will see in Section V-B. Additionally, let $\psi_2(x)$ be an approximation of $\psi_1(x)$. For example, $\psi_2(x)$ can be obtained by sampling the baseline policy $\psi_1(x)$ and learning the input/output map by training a NN on the samples. Then, we can verify the closed-loop stability of $\psi_2(x)$ as follows:

Theorem 1: The approximate control policy $\psi_2(x)$ is stable in closed-loop on a polyhedron $\mathbb{X}$ if (17) is satisfied. (17) is a MILP if $\psi_1(x)$ and $\psi_2(x)$ are MILP representable.

$$\begin{aligned} 0 \leq \min_{\tau,x,u_1,u_2} \quad & \hat{\gamma} - \tau \\ \text{s.t.} \quad & x \in \mathbb{X}, \\ & (x,u_1) \in \mathrm{gr}(\psi_1), \\ & (x,u_2) \in \mathrm{gr}(\psi_2), \\ & (u_1 - u_2, \tau) \in \mathrm{gr}(\|\cdot\|_\infty). \end{aligned} \tag{17}$$

Proof: The worst case approximation error between $\psi_1(x)$ and $\psi_2(x)$ over the domain $\mathbb{X}$ is given by γ as defined in (15). Hence, $\psi_2(x)$ applied to the real system deviates maximally by γ with respect to the infinity norm. Since the original policy $\psi_1(x)$ is robust against input disturbances of magnitude up to $\hat{\gamma}$ in respect to the infinity norm, the inequality $\gamma \leq \hat{\gamma}$ is sufficient for $\psi_2(x)$ to be stable. We can verify the inequality $\gamma \leq \hat{\gamma}$ by solving (17) where, at optionality, τ coincides with the worst case approximation error γ. If $\psi_1(x)$ and $\psi_2(x)$ are MILP representable, then it directly follows that (17) is also an MILP. ■

IV. Stability Verification for Approximate Linear MPC

In Section III-A, we verified the stability of the approximate policy against a robustly stable MPC policy by checking an inequality involving the worst-case approximation error. In this section, we show that we can also directly verify a Lyapunov decrease condition to show stability.

Assume we have a linear system

$$x^+ = Ax + Bu, \tag{18}$$

where the state $x \in \mathbb{R}^n$ and the input $u \in \mathbb{R}^m$ are constrained to lie in the polytopic sets $\mathbb{X} \subseteq \mathbb{R}^n$ and $\mathbb{U} \subseteq \mathbb{R}^m$, respectively, and we consider the following finite horizon MPC scheme

$$\begin{aligned} J^\star(x) := \min_{\mathbf{x},\mathbf{u}} \quad & J(\mathbf{x},\mathbf{u}) && \text{(19a)}\\ \text{s.t.} \quad & \forall i = 0,\dots,N-1, && \text{(19b)}\\ & x_{i+1} = Ax_i + Bu_i, && \text{(19c)}\\ & x_i \in \mathbb{X},\ u_i \in \mathbb{U}, && \text{(19d)}\\ & x_N \in \mathbb{X}_N, && \text{(19e)}\\ & x_0 = x, && \text{(19f)} \end{aligned}$$

with the cost function

$$J(\mathbf{x},\mathbf{u}) = V_N(x_N) + \sum_{i=0}^{N-1} \ell(x_i,u_i). \tag{20}$$

For ease of notation, we also henceforth use

$$\mathcal{F}(x) = \left\{ (\mathbf{x},\mathbf{u}) \;\middle|\; \begin{array}{l} \forall i = 0,\dots,N-1, \\ x_i \in \mathbb{X},\ u_i \in \mathbb{U}, \\ x_N \in \mathbb{X}_N,\ x_0 = x \end{array} \right\} \tag{21}$$

as a shorthand for the feasible set of problem (19).

Assumption 1: There exists a linear control law $v : \mathbb{R}^n \to \mathbb{R}^m$ such that the terminal set $\mathbb{X}_N \subseteq \mathbb{X}$ is a polytopic invariant set for the system $x^+ = Ax + Bv(x)$ with state and input constraints (19d) [46], and $V_N(x)$ is a Lyapunov function for the same system that decreases by one stage cost each time step, that is, $V_N(x^+) - V_N(x) \leq \ell(x, v(x))$ for every $x \in \mathbb{X}_N$.

A standard approach to satisfy Assumption 1 is to choose the classic stage cost

$$\ell(x_i,u_i) = x_i^T Q x_i + u_i^T R u_i, \tag{22}$$

where $Q \in \mathbb{S}_+^n$ and $R \in \mathbb{S}_{++}^m$, and the solution of the LQR problem as the Lyapunov function $V_N(x)$ and the LQR controller for $v(x)$. Then the control invariant set $\mathbb{X}_N$ can be calculated using the MPT toolbox [47], and the decrease condition for $V_N(x)$ is fulfilled. This can be directly verified by looking at the definition of the LQR problem, which solves the following problem:

$$\begin{aligned} V_N(x) = J_\infty^\star(x) := \min_{\mathbf{x},\mathbf{u}} \quad & \sum_{i=0}^{\infty} \ell(x_i,u_i) \\ \text{s.t.} \quad & x_{i+1} = Ax_i + Bu_i, \\ & x_0 = x. \end{aligned} \tag{23}$$

Clearly, if $v(x)$ is an optimal solution for the first input u_0 in (23), then $V_N(x^+) = V_N(Ax + Bv(x)) = V_N(x) + \ell(x, v(x))$. In addition, it is well known that $J^\star$ is a Lyapunov function for the system $x^+ = Ax + Bv(x)$ [48].

Assume we approximate the MPC scheme (19). We then get the approximate closed-loop system $x^+ = Ax+B\psi_2(x)$, where $\psi_2(x)$ is an MILP representable approximate MPC controller (e.g., a NN).

A. Sufficient Condition for Lyapunov Decrease

To verify stability, we will make use of the following sufficient condition for the approximate control law to be stabilizing for a given initial state set. The condition is similar as in [49], but has been extended to allow for the verification of asymptotic stability.

Lemma 7: Let Assumption 1 hold, and $J^\star(x)$ be the optimal value function of (19). Also, let $\epsilon > 0$ be a positive constant. If $\psi_2(x) : \mathbb{R}^n \to \mathbb{R}^m$ is a control law for the closed-loop system defined over the set $\mathbb{X}_0$, then $J^\star$ is also a Lyapunov function for $x^+ = Ax + B\psi_2(x)$ on a set $\mathbb{X}_0^{\psi_2} \subseteq \mathbb{X}_0$ if for all $x_0 \in \mathbb{X}_0$, there exists a feasible state/input sequence $(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) \in \mathcal{F}(x_0)$ such that $\tilde{u}_0 = \psi_2(x_0)$ and

$$J(\tilde{\mathbf{x}},\tilde{\mathbf{u}}) - J^\star(x_0) \leq \ell(\tilde{x}_0,\tilde{u}_0) - \epsilon\|\tilde{x}_0\|_2^2 \quad \forall x_0 \in \mathbb{X}_0^{\psi_2}, \tag{24}$$

with $\mathbb{X}_0^{\psi_2} = \{x_0 \in \mathbb{X}_0 \mid \tilde{u}_0 = \psi_2(x_0) \text{ and } \exists(\tilde{\mathbf{x}},\tilde{\mathbf{u}}) \in \mathcal{F}(x_0)\}$.

Proof: If the input/state sequence $(\psi_2(\tilde{x}_0), \tilde{u}_1, \dots, \tilde{u}_{N-1}, \tilde{x}_0, \dots, \tilde{x}_N)$ is feasible for (19) and satisfies (24) for all $\tilde{x}_0 \in \mathbb{X}_0^{\psi_2}$, then the shifted sequence $(\tilde{u}_1, \dots, \tilde{u}_{N-1}, v(\tilde{x}_N), \tilde{x}_1, \dots, \tilde{x}_N, \tilde{x}_{N+1})$ is also feasible, where $\tilde{x}_{N+1} = A\tilde{x}_N + Bv(\tilde{x}_N)$, and we used the assumption that $\mathbb{X}_N$ is invariant for the linear control law $v(\cdot)$. Evaluating the cost function at this sequence gives

$$\begin{aligned} J^\star(\tilde{x}_1) &\leq V_N(\tilde{x}_{N+1}) + \sum_{i=1}^{N} \ell(\tilde{x}_i,\tilde{u}_i) && \text{(25a)}\\ &= V_N(\tilde{x}_{N+1}) - V_N(\tilde{x}_N) + \ell(\tilde{x}_N,\tilde{u}_N) && \text{(25b)}\\ &\quad + V_N(\tilde{x}_N) + \sum_{i=0}^{N-1} \ell(\tilde{x}_i,\tilde{u}_i) && \text{(25c)}\\ &\quad - \ell(\tilde{x}_0,\psi_2(\tilde{x}_0)) && \text{(25d)}\\ &\leq J^\star(\tilde{x}_0) - \epsilon\|\tilde{x}_0\|_2^2 && \text{(25e)} \end{aligned}$$

for all $\tilde{x}_0 \in \mathbb{X}_0^{\psi_2}$. Equation (25b) is negative by the assumption that V_N decreases faster than the stage cost ℓ in the set $\mathbb{X}_N$ and (25c) is less than or equal to $J^\star(\tilde{x}_0) + \ell(\tilde{x}_0,\psi_2(\tilde{x}_0)) - \epsilon\|\tilde{x}_0\|_2^2$ because of (24). It follows that $J^\star(\tilde{x}_1) - J^\star(\tilde{x}_0) \leq -\epsilon\|\tilde{x}_0\|_2^2\ \forall \tilde{x}_0 \in \mathbb{X}_0^{\psi_2}$ and therefore $J^\star(x)$ is a Lyapunov function for the approximate closed-loop system $x^+ = Ax + B\psi_2(x)$. ■

The conditions in Lemma 7 are not easy to verify, but can be equivalently written as

$$\begin{aligned} &\forall x_0 \in \mathbb{X}_0^{\psi_2} : \exists(\tilde{\mathbf{x}},\tilde{\mathbf{u}}) \in \mathcal{F}(x_0) : \tilde{u}_0 = \psi_2(x_0), \\ &0 \leq \ell(\tilde{x}_0,\tilde{u}_0) - \epsilon\tilde{x}_0^T\tilde{x}_0 - J(\tilde{\mathbf{x}},\tilde{\mathbf{u}}) + J^\star(x_0) \end{aligned} \tag{26}$$

which we can be formulated as a min-max problem

$$0 \leq \min_{x_0 \in \mathbb{X}_0^{\psi_2}} \max_{\substack{(\tilde{\mathbf{x}},\tilde{\mathbf{u}}) \in \mathcal{F}(x_0) \\ \tilde{u}_0 = \psi_2(x_0)}} \ell(\tilde{x}_0,\tilde{u}_0) - \epsilon\tilde{x}_0^T\tilde{x}_0 - J(\tilde{\mathbf{x}},\tilde{\mathbf{u}}) + J^\star(x_0). \tag{27}$$

By replacing the last term with

$$J^\star(x_0) = \min_{(\mathbf{x},\mathbf{u}) \in \mathcal{F}(x_0)} J(\mathbf{x},\mathbf{u}) \tag{28}$$

and rearranging terms we get the problem

$$\begin{aligned} 0 \leq \min_{\substack{x_0 \in \mathbb{X}_0^{\psi_2} \\ (\mathbf{x},\mathbf{u}) \in \mathcal{F}(x_0)}} \quad & J(\mathbf{x},\mathbf{u}) \\ + \max_{\substack{(\tilde{\mathbf{x}},\tilde{\mathbf{u}}) \in \mathcal{F}(x_0) \\ \tilde{u}_0 = \psi_2(x_0)}} \quad & \ell(\tilde{x}_0,\tilde{u}_0) - J(\tilde{\mathbf{x}},\tilde{\mathbf{u}}) - \epsilon \tilde{x}_0^T \tilde{x}_0. \end{aligned} \tag{29}$$

A standard approach in robust optimization would be to dualize the max problem. Unfortunately, this leads to an optimization problem with a bilinear term in the objective, which can not be easily linearized. Empirically, we found that the resulting optimization problem is almost impossible to solve.

Instead, we write (29) it in terms of a bilevel optimization problem

$$\xi := \min_{\mathbf{x},\mathbf{u}} J(\mathbf{x},\mathbf{u}) + \ell(\tilde{x}_0,\tilde{u}_0) - J(\tilde{\mathbf{x}},\tilde{\mathbf{u}}) - \epsilon \tilde{x}_0^T \tilde{x}_0 \tag{30a}$$

$$\text{s.t. } x_0 \in \mathbb{X}_0^{\psi_2}, \tag{30b}$$

$$(\mathbf{x},\mathbf{u}) \in \mathcal{F}(x_0), \tag{30c}$$

$$(\tilde{\mathbf{x}},\tilde{\mathbf{u}}) = \arg\min_{\tilde{\mathbf{x}},\tilde{\mathbf{u}}} J(\tilde{\mathbf{x}},\tilde{\mathbf{u}}) + \epsilon \tilde{x}_0^T \tilde{x}_0 \tag{30d}$$

$$\text{s.t. } (\tilde{\mathbf{x}},\tilde{\mathbf{u}}) \in \mathcal{F}(x_0),$$

$$\tilde{u}_0 = \psi_2(x_0).$$

We can see that the conditions in Lemma 7 are fulfilled if and only if ξ is positive. Solving such bilevel problems can be quite challenging. In Section IV-D we propose to reformulate (30) as a mixed-integer program. Compared to dualizing the min-max problem (29), we found that the mixed-integer reformulations described in Section IV-D tend to be very solvable in practice. We believe that this is the case because the root relaxation is bounded due to only having bounded primal variables in the objective. Dualizing problem (29) introduces dual variables in the objective, which are generally unbounded.

Although $J^\star(x)$ is a Lyapunov function on the set $\mathbb{X}_0^{\psi_2}$, it is not sufficient to prove stability. The system must also be invariant on $\mathbb{X}_0^{\psi_2}$. This leads us to the following result:

Theorem 2: Let $\psi_2(0) = 0$. If the solution of (30) is $\xi \geq 0$, then there exists a set $\Omega_0^{\psi_2} \subseteq \mathbb{X}_0^{\psi_2}$ such that applying the controller $\psi_2(x_0)$ results in the closed-loop system converging asymptotically to the origin for all $x_0 \in \Omega_0^{\psi_2}$.

Proof: By construction, (30) satisfies condition (24) if and only if $\xi \geq 0$. Hence, $J^\star$ is a Lyapunov function for the system $x^+ = Ax + B\psi_2(x)$ where $x \in \mathbb{X}_0^{\psi_2}$. We are choosing $\Omega_0^{\psi_2} \subseteq \mathbb{X}_0^{\psi_2}$ to be the set on which the closed-loop is invariant. Thus, together with $J^\star(x)$ being a Lyapunov function and $\Omega_0^{\psi_2}$ being invariant, it follows that the system converges asymptotically to the origin for all $x_0 \in \Omega_0^{\psi_2}$. The condition $\psi_2(0) = 0$ ensures that the origin is actually an equilibrium point, and that $\Omega_0^{\psi_2}$ is not empty. ■

B. Direct Verification of Lyapunov Decrease

Instead of verifying the sufficient condition (24), we can also directly verify the Lyapunov decrease

$$J^\star(x^+) - J^\star(x) \leq -\epsilon \|x\|_2^2 \quad \forall x \in \mathbb{X}_0, \tag{31}$$

with $x^+ = Ax + B\psi_2(x)$. Hence, similar to the previous section, we can formulate the condition (31) as a bilevel optimization problem

$$\xi := \min_{\mathbf{x},\mathbf{u}} J(\mathbf{x},\mathbf{u}) - J(\tilde{\mathbf{x}},\tilde{\mathbf{u}}) - \epsilon \tilde{x}_0^T \tilde{x}_0 \tag{32a}$$

$$\text{s.t. } x_0 \in \mathbb{X}_0, \tag{32b}$$

$$(\mathbf{x},\mathbf{u}) \in \mathcal{F}(x_0), \tag{32c}$$

$$(\tilde{\mathbf{x}},\tilde{\mathbf{u}}) \in \tilde{\mathcal{F}}(x_0) \tag{32d}$$

where we define the feasible set of the inner optimization problem as

$$\begin{aligned} \tilde{\mathcal{F}}(x) = \arg\min_{\tilde{\mathbf{x}},\tilde{\mathbf{u}}} \; & J(\tilde{\mathbf{x}},\tilde{\mathbf{u}}) + \epsilon \tilde{x}_0^T \tilde{x}_0 \\ \text{s.t. } & (\tilde{\mathbf{x}},\tilde{\mathbf{u}}) \in \mathcal{F}(Ax + B\psi_2(x)) \end{aligned} \tag{33}$$

Note that compared to problem (30), the feasible set of (32) is actually an invariant set for the closed-loop system. Thus, we get the following result:

Theorem 3: Let $\tilde{\Omega}_0^{\psi_2} = \{x_0 \in \mathbb{X}_0 \mid (\mathbf{x},\mathbf{u}) \in \mathcal{F}(x_0), (\tilde{\mathbf{x}},\tilde{\mathbf{u}}) \in \tilde{\mathcal{F}}(x_0)\}$ be the feasible set of (32) for varying x_0. If the optimal value of (32) is $\xi \geq 0$ and $\psi_2(x_0) \in \mathbb{U}\ \forall x_0 \in \tilde{\Omega}_0^{\psi_2}$, then applying the controller $\psi_2(x_0)$ will result in the closed-loop system converging asymptotically to the origin and satisfies the state constraints $\mathbb{X}$ for all $x_0 \in \tilde{\Omega}_0^{\psi_2}$.

Proof: By construction, condition (31) is satisfied if and only if $\xi \geq 0$. Hence, it follows that $J^\star(x)$ is a Lyapunov function for the system $x^+ = Ax + B\psi_2(x)$ where $x \in \tilde{\Omega}_0^{\psi_2}$. With the constraint $\tilde{x}_0 = Ax_0 + B\psi_2(x_0)$, we ensure that $\tilde{\Omega}_0^{\psi_2}$ only contains x_0 such that when $\psi_2(x_0)$ gets applied to the system, it is inside the state/input feasible set of the MPC scheme (19). Since (19) satisfies the state constraints $\mathbb{X}$, together with the assumption that $\psi_2(x_0) \in \mathbb{U}\ \forall x_0 \in \tilde{\Omega}_0^{\psi_2}$, we get that $\tilde{\Omega}_0^{\psi_2}$ is invariant for the closed-loop system and $\tilde{\Omega}_0 \subseteq \mathbb{X}$. Hence, the system converges asymptotically to the origin and satisfies the state constraints in the set $\mathbb{X}$ for all $x_0 \in \tilde{\Omega}_0^{\psi_2}$. ■

Remark 5: Note that Theorem 3 does not require the MPC scheme (19) to have a terminal set, nor to be known stable, i.e., Assumption 1 does not have to hold. Hence, even MPC schemes that are not known to be stable a priori can be used, and asymptotic stability can still be verified because we verify the Lyapunov decrease condition of the MPC cost directly. Hence, if we chose $\psi_2(x)$ to be the same as the control policy of the solution of (19), we can verify the asymptotic stability of any MILP representable control law. In fact, this is very similar to the approach taken in [50], where the stability of MPC schemes is verified a posteriori using an MILP.

C. Outer Approximation of Stable Region

In Theorem 3 we have seen that the approximate controller $\psi_2(\cdot)$ is stable if $\xi \geq 0$ and satisfies state constraints on a set $\tilde{\Omega}_0^{\psi_2}$. Unfortunately, calculating this exact set presents a significant challenge, but a convex polyhedral outer approximation $\bar{\Omega}_0^{\psi_2}$ can be found. We can define the set $\bar{\Omega}_0^{\psi_2}$ as

$$\bar{\Omega}_0^{\psi_2} = \left\{x \in \mathbb{R}^n \mid C_{\text{stable}} x \leq c_{\text{stable}}\right\}, \tag{34}$$

where $C_{\text{stable}} \in \mathbb{R}^{n_{\text{stable}} \times n}$ and $c_{\text{stable}} \in \mathbb{R}^{n_{\text{stable}}}$ with n_{stable} being the number of hyperplanes in the polyhedron. Fixing C_{stable}, we solve n_{stable} bilevel optimizations problems

$$\begin{aligned} c^\star_{\text{stable}\,i} = \max_{\mathbf{x},\mathbf{u}} \; & C_{\text{stable}\,i} x_0 && \text{(35a)}\\ \text{s.t. } & x_0 \in \mathbb{X}, && \text{(35b)}\\ & (\mathbf{x},\mathbf{u}) \in \mathcal{F}(x_0), && \text{(35c)}\\ & (\tilde{\mathbf{x}},\tilde{\mathbf{u}}) \in \tilde{\mathcal{F}}(x_0), && \text{(35d)} \end{aligned}$$

for $i = 1, \dots, n_{\text{stable}}$. The chosen C_{stable} dictates the shape of the outer approximation of the polyhedron, and by solving (35) we find the smallest such polyhedron which encapsulates the set $\tilde{\Omega}_0^{\psi_2}$. A simple choice for C_{stable} and c_{stable} would be a box, i.e.,

$$C_{\text{stable}} = \begin{bmatrix} I_n \\ -I_n \end{bmatrix}, \quad c_{\text{stable}} = \mathbf{1}_{2n}. \tag{36}$$

D. MIP Formulation

Problems (30), (32) and (35) as stated in the previous sections can not be directly solved. Fortunately, we can reformulate them to fit into the verification framework introduced at the beginning. We note that the inner optimization problems (30d) and (32d) are QPs, and can be reformulated as

$$\begin{aligned} \tilde{z}(x_0, \psi_2(x_0)) = \arg\min_{\tilde{z}} \; & \frac{1}{2}\tilde{z}^T \tilde{P} \tilde{z} + \tilde{q}^T \tilde{z} \\ \text{s.t. } & \tilde{A}\tilde{z} + \tilde{B} \begin{bmatrix} x_0 \\ \psi_2(x_0) \end{bmatrix} = \tilde{b}, \\ & \tilde{F}\tilde{z} \le \tilde{g}, \end{aligned} \tag{37}$$

with $\tilde{z} = (\tilde{\mathbf{x}}, \tilde{\mathbf{u}})$ and appropriate matrices $\tilde{P}$, $\tilde{q}$, $\tilde{A}$, $\tilde{B}$, $\tilde{b}$, $\tilde{F}$, and $\tilde{g}$. Note that (37) is a positive definite parametric QP with parameters x_0 and $\psi_2(x_0)$. Hence, using the formulation introduced in Section II-C, we can rewrite (30) and (32) as

$$\begin{aligned} \xi := \min_{\substack{\mathbf{x},\mathbf{u},\tilde{u},\tilde{z} \\ \mu^L \lambda^L, \beta^L}} \; & J(\mathbf{x},\mathbf{u}) - \frac{1}{2}\tilde{z}^T \tilde{P} \tilde{z} - \tilde{q}^T \tilde{z} \\ \text{s.t. } & x_0 \in \mathbb{X}_0, \\ & (\mathbf{x},\mathbf{u}) \in \mathcal{F}(x_0), \\ & (x_0, \tilde{u}) \in \text{gr}(\psi_2), \\ & \tilde{A}\tilde{z} + \tilde{B} \begin{bmatrix} x_0 \\ \tilde{u} \end{bmatrix} = \tilde{b}, \\ & 0 = \tilde{P}\tilde{z} + \tilde{q} + \tilde{A}^T \mu^L + \tilde{F}^T \lambda^L, \\ & 0 \le \lambda_i^L \le M \beta_i^L, \\ & 0 \le \tilde{g}_i - \tilde{F}\tilde{z} \le M(1 - \beta_i^L), \\ & \lambda^L (\tilde{F}\tilde{z} - \tilde{g}) = 0, \end{aligned} \tag{38}$$

with μ^L and λ^L being the appropriate Lagrange multipliers of the respective QP, and β_i^L being the binary decision variables representing the active/inactive constraints. Assuming $\psi_2(x_0)$ is MILP representable (e.g., ReLU NN), problem (38) becomes an indefinite MIQP which can be solved with commercial solvers like Gurobi [41]. The problem (35) to find the outer approximation of the stable region $\bar{\Omega}_0^{\psi_2}$ is reformulated into something similar

$$\begin{aligned} c^\star_{\text{stable}\,i} = \max_{\substack{\mathbf{x},\mathbf{u},\tilde{u},\tilde{z} \\ \mu^L \lambda^L, \beta^L}} \; & C_{\text{stable}\,i} x_0 \\ \text{s.t. } & x_0 \in \mathbb{X}_0, \\ & (\mathbf{x},\mathbf{u}) \in \mathcal{F}(x_0), \\ & (x_0, \tilde{u}) \in \text{gr}(\psi_2), \\ & \tilde{A}\tilde{z} + \tilde{B} \begin{bmatrix} x_0 \\ \tilde{u} \end{bmatrix} = \tilde{b}, \\ & 0 = \tilde{P}\tilde{z} + \tilde{q} + \tilde{A}^T \mu^L + \tilde{F}^T \lambda^L, \\ & 0 \le \lambda_i^L \le M \beta_i^L, \\ & 0 \le \tilde{g}_i - \tilde{F}\tilde{z} \le M(1 - \beta_i^L), \\ & \lambda^L (\tilde{F}\tilde{z} - \tilde{g}) = 0, \end{aligned} \tag{39}$$

but is easier to solve since it is an MILP and not an indefinite MIQP.

Remark 6: Note that, because the objective function is indefinite, problem (38) is considerably harder to solve than a normal MILP or MIQP. Solvers only added support for such problems in recent years (e.g., introduced in Gurobi 9.0), by translating the indefinite cost into bilinear constraints, which are then solved using cutting planes and spatial branching.

V. Case Study: DC-DC Power Converter

In this section, we consider the DC-DC power converter from [6] as a case study. The original linear MPC controller is approximated via a piecewise affine NN. Evaluating the NN is considerably cheaper, allowing the approximate control policy to be deployed on a low cost microcontroller operating at 80MHz and with a sampling frequency of 10 kHz. Simulation results show the effectiveness of the approximate control in some cases, however, the work gives no guarantee of stability. Using the techniques introduced above, we can show that the NN controller is indeed stable for the closed-loop system, and give an outer approximation of the stability region.

Additionally, we design a tube MPC robust to input disturbances. The robust controller is then approximated using a NN. Using the formulation in Section III-A we can guarantee the closed-loop system controlled by the NN satisfies constraints and converges to the minimum robust invariant set.

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

$$x^+ = Ax + Bu = \begin{bmatrix} 0.971 & -0.010 \\ 1.732 & 0.970 \end{bmatrix} x + \begin{bmatrix} 0.149 \\ 0.181 \end{bmatrix} u. \tag{40}$$

A more detailed derivation of the model dynamics along with additional details on the physical system can be found in [6].

A. Nominal MPC

The problem of control design for tracking current and voltage references in the DC-DC converter can be formulated

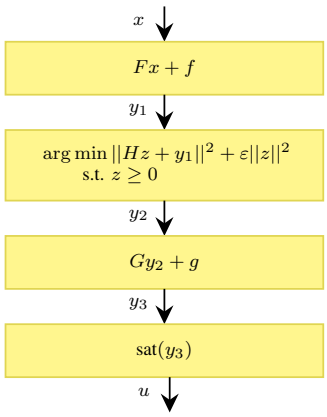

Fig. 2: The piecewise affine NN architecture.

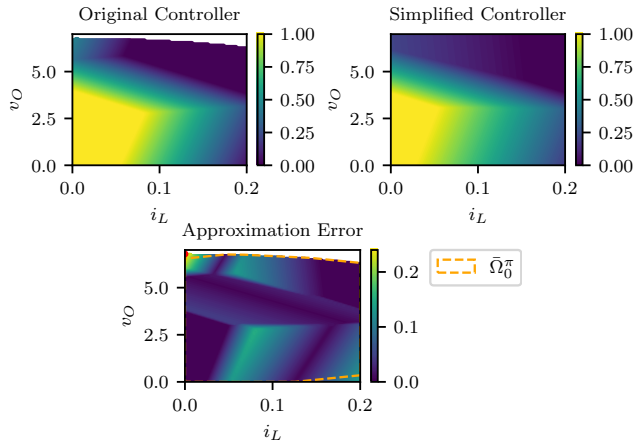

Fig. 3: Original, approximate/simplified control policy (top), and approximation error to the original MPC control policy with outer approximation of stability region (bottom).

as a linear MPC controller. We formulate the MPC controller which regulates to the steady-state $x_{\text{eq}} = \begin{bmatrix} 0.05 & 5 \end{bmatrix}^T$, and $u_{\text{eq}} = 0.3379$ as

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u}} \quad & \sum_{i=0}^{N-1} \left(\|x_i - x_{\text{eq}}\|_Q^2 + \|u_i - u_{\text{eq}}\|_R^2 \right) \\ & + \|x_N - x_{\text{eq}}\|_P^2 \\ \text{s.t.} \quad & \forall i = 0, \dots, N-1, \\ & x_{i+1} = Ax_i + Bu_i, \\ & \begin{bmatrix} 0 \\ 0 \end{bmatrix} \leq x_i \leq \begin{bmatrix} 0.2 \\ 7 \end{bmatrix}, \quad 0 \leq u_i \leq 1 \\ & x_N \in \mathbb{X}_N, \quad x_0 = x(0), \end{aligned} \tag{41}$$

with horizon length $N = 10$, $Q = \text{diag}(90, 1)$, $R = 1$, P the solution of the associated discrete-time algebraic Riccati equation, and $\mathbb{X}_N$ the system's maximal invariant set under the LQR policy. The resulting control policy can be seen in Figure 3.

We chose the same architecture for the approximate controller as in [6], which can be seen in Figure 2, with the appropriate dimensions of the layers fully determined by the size of the optimization variable $z \in \mathbb{R}^{n_z}$, which was chosen as $n_z = 3$. Note that this is not a classic NN architecture, since the second layer is the solution of a parametric QP. The reader is referred to [6] for a more detailed description and analysis of the architecture. This shows the flexibility of our toolbox, since we also allow more unconventional layer structures.

For the training data, 5000 samples of the original controller (41) uniformly distributed from the feasible region are taken, and the approximate controller is then trained using Adam [51] as the optimizer. To formulate the convex optimization layer, cvxpylayers [22] was used. The trained simplified controller and the absolute approximation error can be seen in Figure 3.

Using the formulation in Section III we find an absolute worst-case approximation error of $\gamma = 0.24$ at $i_L = 0$ and $v_O = 6.79$. Similarly, using both formulations in Sections IV-A and IV-B, by solving an indefinite MIQP respectively, we can verify the stability of the closed-loop system, which is the case for this example. With this, we have successfully verified the approximate control policy against the nominal MPC. Additionally, we solve (35) to get the outer approximation of the stability region $\bar{\Omega}_0^{\psi_2}$, which can be seen in Figure 3. Note that the region is smaller than the feasible region of the MPC controller. This is expected, because due to the approximation error, we can not expect to achieve the same stability region.

B. Robust Tube MPC

Starting from the MPC formulation in (41), we formulate a robust tube MPC [34]. We assume that we have uncertain dynamics

$$x^+ = Ax + Bu + Bw, \tag{42}$$

with disturbance set $\mathbb{W} = \{w \in \mathbb{R} \mid -0.1 \leq w \leq 0.1\}$. The objective is now to design a robust controller to withstand an input disturbance with amplitude 0.1. The input disturbance is typically associated to the distortion introduced by the switching nature of the converter, but here we use it to verify the stability of the approximate policy.

We extend the nominal MPC formulation (41) to the following robust tube MPC formulation

$$\begin{aligned} \min_{\mathbf{z},\mathbf{v}} \quad & \sum_{i=0}^{N-1} \left(\|z_i - x_{\text{eq}}\|_Q^2 + \|v_i - u_{\text{eq}}\|_R^2 \right) + \|z_N - x_{\text{eq}}\|_P^2 \\ \text{s.t.} \quad & \forall i = 0, \dots, N-1, \\ & z_{i+1} = Az_i + Bv_i, \\ & z_i \in \mathbb{X} \ominus \mathcal{E}, \quad v_i \in \mathbb{U} \ominus K\mathcal{E} \\ & z_N \in \mathbb{X}_N, \quad x(0) \in z_0 \oplus \mathcal{E}, \end{aligned} \tag{43}$$

where $\mathcal{E}$ is the minimum robust invariant set with respect to a linear feedback gain K, and the control law is given by $\psi_1(x) = K(x - z_0^\star(x)) + v_0^\star(x)$. Here K is the feedback gain and P is the solution of the Riccati equation associated with the discrete LQR problem. The policy is robustly stable for input disturbances in $\mathbb{W}$ [34, Theorem 1]. The horizon length was increased to $N = 20$ to have a bigger feasible

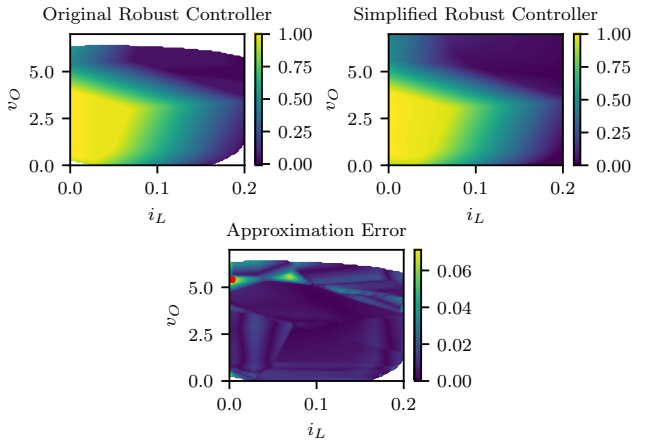

Fig. 4: Original, approximate/simplified robust control policy (top), and approximation error to the original robust MPC control policy (bottom).

region. Note that (43) is a pQP since the sets $\mathbb{X} \ominus \mathcal{E}$, $\mathbb{U} \ominus K\mathcal{E}$, and $\mathcal{E}$ are polyhedra that can be precalculated.

We then approximate the tube MPC with a NN with 2 hidden layers, 50 neurons each, and a saturation layer at the end to clip the input between -1 and 1. Similar as in Section V-A, 5000 samples of the tube MPC uniformly in the feasible region are taken and the NN is trained using standard techniques. The resulting NN controller together with the original tube MPC can be seen in Figure 4.

We apply the formulation from Section III to find the worst case approximation error, and get a value of $\gamma = 0.073$. Since we designed our controller to be robust for input perturbations with a maximum magnitude of 0.1, we have shown that the NN controller satisfies constraints and converges to the minimum robust invariant set in the feasible region of the tube MPC.

The set on which the closed-loop system is stable is directly given by the feasible set of the tube MPC, compared to the nominal MPC where we could only calculate an outer approximation of the stability region. Additionally, to verify stability, we only have to solve an MILP and not an indefinite MIQP, which might be considerably harder to solve. But for larger problem sizes, the solve times are not necessarily larger as we discuss in Section VI-B.

C. Experimental Validation

The approximate nominal controller in Section V-A has been implemented on an inexpensive STM32L476 microcontroller for a prototype of the described DC-DC power converter. The controller is running at a frequency of 10 kHz, with the execution time of the control law being between 22.0 μs and 27.5 μs. The closed-loop startup response can be seen in Figure 5. We can see that the constraints for both the voltage v_O and the current i_L are satisfied, and an excellent transient with a settling time of 2.33 ms. For more elaborate implementation details, the reader is referred to [6].

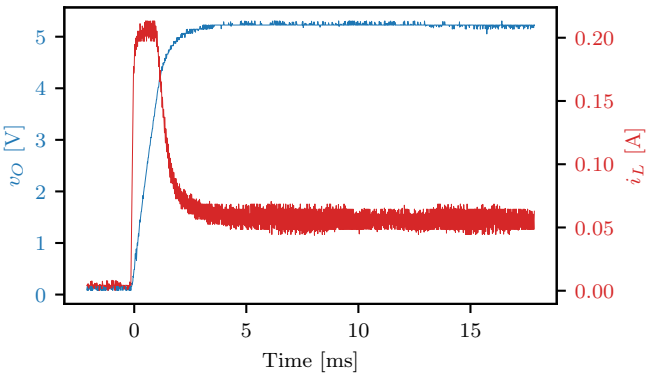

Fig. 5: Closed-loop startup response of the approximate MPC.

VI. Discussion

A. Comparison to Lipschitz Based Methods

We compare our method to the approach introduced in [33]. The authors provide the following two main results.

Lemma 8 ([33, Lemma 3.2]): There exist $\zeta_c > 0$ such that, if $\gamma < \zeta_c$, the LTI system in (18) with NN controller $\psi_2(x)$ converges in finite time to a neighborhood of the origin, for all $x_0 \in \Omega_c$, with $c := \max_a \{a \geq 0 \mid \Omega_a \subseteq \mathbb{X}\}$.

Theorem 4 ([33, Theorem 3.4]): Let Assumption 1 hold. There exist ζ_c and ϑ such that, if $\gamma < \zeta_c$ and $\mathcal{L}_\infty(\psi_1(x) - \psi_2(x), \mathbb{X}_N) < \vartheta$, and $b \geq 0$ can be chosen so that $\Omega_b \subseteq \mathbb{X}_N$, then the LTI system in (18) with NN controller $\psi_2(x)$ converges exponentially to the origin, for all $x_0 \in \Omega_c$, with $c := \max_a \{a \geq 0 \mid \Omega_a \subseteq \mathbb{X}\}$.

Here Ω_a denotes the a-sublevel set of $J^\star(x)$, and $\mathcal{L}_\alpha(F, \mathcal{S})$ is the local α-Lipschitz constant over some set $\mathcal{S} \subseteq \mathbb{R}^n$ for a given mapping $F : \mathbb{R}^n \to \mathbb{R}^m$.

Lemma 8 states that the NN policy is input-to-state-stable (ISS) if the worst case approximation error is smaller than a constant ζ_c. This does not guarantee asymptotic stability yet, but in Theorem 4 exponential convergence to the origin is shown if the Lipschitz constant of the approximation error is smaller than a constant ϑ.

We are not going in more detail on how to calculate these constants. The reader is referred to [33] for more details, but it turns out that these constants become small in practice. Since the NN almost surely does not approximate the baseline control policy arbitrary well in practice, the Lipschitz constant of the approximation error can grow significantly, especially if the NN is over parameterized. To demonstrate this, we consider a simple example of a double

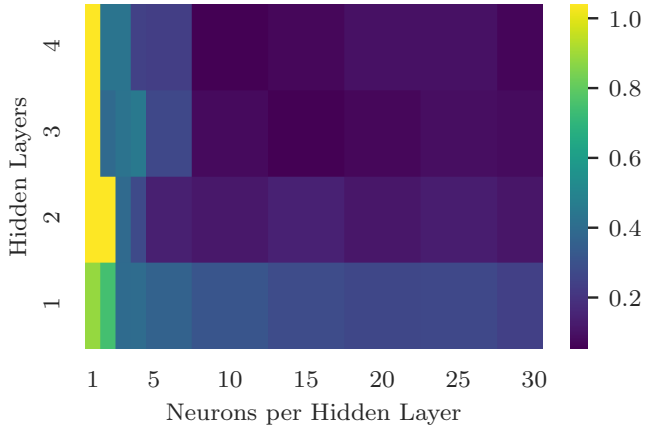

Fig. 6: Worst case approximation error γ.

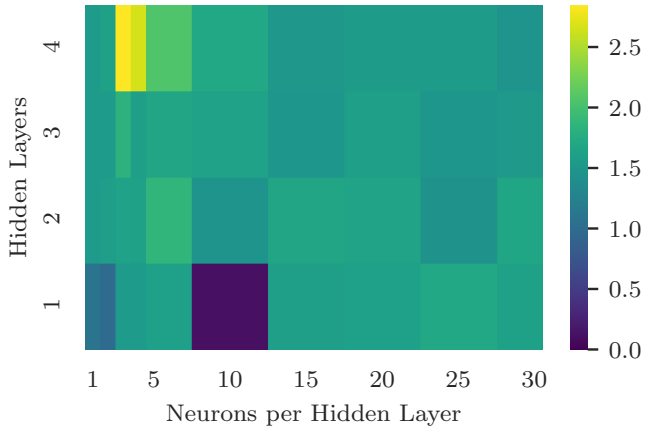

Fig. 7: Lipschitz constant $\mathcal{L}_\infty\left(\psi_1(x)-\psi_2(x), \mathbb{X}_N\right)$ of the approximation error.

integrator. We define the baseline MPC policy

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{u}} \quad & \sum_{i=0}^{N-1} \left(\|x_i\|_Q^2 + \|u_i\|_R^2 \right) + \|x_N\|_P^2 \\
\text{s.t.} \quad & \forall i = 0, \dots, N-1, \\
& x_{i+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_i + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_i, \\
& \begin{bmatrix} -10 \\ -10 \end{bmatrix} \leq x_i \leq \begin{bmatrix} 10 \\ 10 \end{bmatrix}, \quad -1 \leq u_i \leq 1 \\
& x_N \in \mathbb{X}_N, \quad x_0 = x(0),
\end{aligned}
\tag{44}
$$

with horizon length $N = 10$, $Q = \mathrm{diag}(1, 1)$, $R = 0.1$, P the solution of the associated discrete-time algebraic Riccati equation, and $\mathbb{X}_N$ the system's maximal invariant set under the LQR policy.

We estimate the largest sublevel set $\Omega_c \subseteq \mathbb{X}$ with $c = 180$, giving us $\zeta_{180} = 0.1673$. This will only guarantee ISS. To prove exponential stability, we estimate the largest sublevel $\Omega_b \subseteq \mathbb{X}_N$ with $b = 1.1$, giving us $\zeta_{1.1} = 0.0010$ and $\vartheta = 0.0436$.

For the training data, 1000 samples of the baseline controller (44) uniformly distributed from the feasible region are taken, and the approximate controller is then trained using L-BFGS for different hidden layers and neurons per hidden layer. The resulting worst case approximation error γ can be seen in Figure 6. Note that for sufficiently large NN architectures $\gamma < \zeta_{180}$, but $\gamma > \zeta_{1.1}$ for all architectures. Hence, with the approach from [33] it is possible to show ISS, but not exponential stability. In Figure 7 we also plot the Lipschitz constant $\mathcal{L}_\infty\left(\psi_1(x)-\psi_2(x), \mathbb{X}_N\right)$ of the approximation error. Note that it is generally orders of magnitude larger than ϑ, making it impossible to verify asymptotic stability even if the approximation error is small enough.

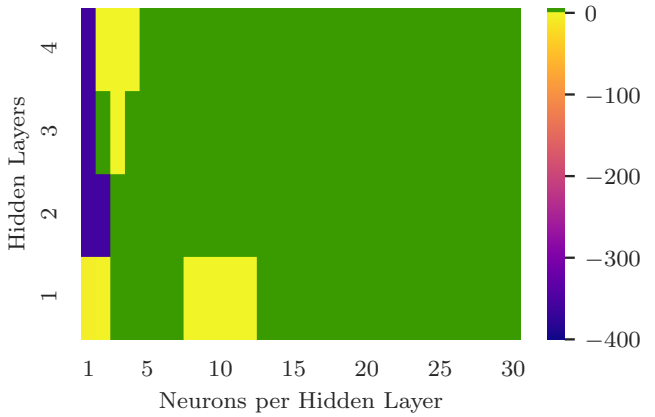

Fig. 8: Stability certificate ξ for solving (32). The green area shows successful verification of asymptotic stability.

As a comparison, we run our direct verification method by solving (32). The results can be seen in Figure 8. Note that we not only verify asymptotic stability, but did so even for NN policies with a very small number of hidden layers and neurons, where the method in [33] failed to show ISS.

B. Numerical Experiments

In this section, we show the numerical performance of our approach. All problems were solved using Gurobi 9.5 [41] using 32 Threads on a workstation with an AMD Ryzen Threadripper 3990X 4.3 GHz CPU and 32 GB of RAM.

We approximate and verify against MPC problem (41), and use a ReLU NN architecture as introduced in Section II-A to approximate the MPC policy. For the training, we collect 2000 samples from the MPC controller uniformly distributed on the feasible region, and the NN is trained using PyTorch [35] with ADAM [51] as the optimizer.

Here, we are not necessarily interested in the approximation accuracy of the NN, but more in the numerical performance and scaling of the to-be-solved optimization problems. Applying this method to a specific problem requires naturally more tuning since the verification will only succeed if the NN approximates the MPC policy closely enough, but the presented results will still give a quantitative insight in solve times and what can be expected if applied to a more specific problem.

In a first experiment, we fix the horizon of the MPC controller to be $N = 10$, and solve problem (17) for different NN architectures varying in the number of hidden

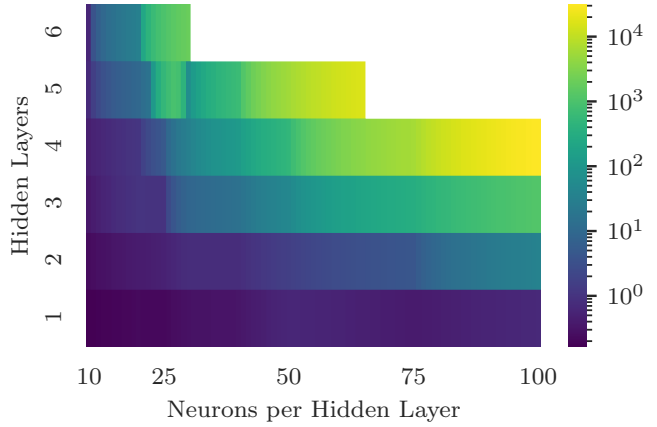

Fig. 9: Solve times of (17) in seconds with MPC horizon $N = 10$.

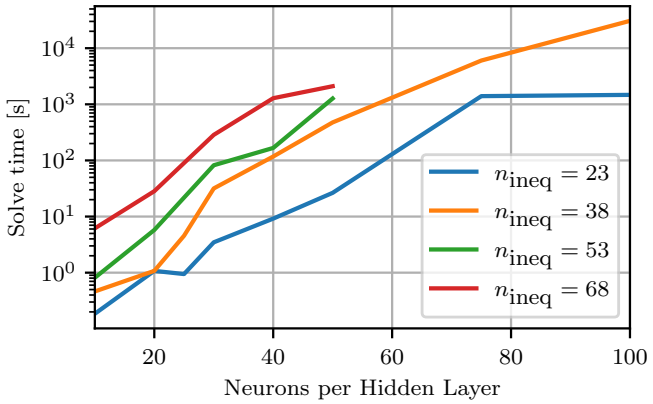

Fig. 10: Solve times of (17) in seconds for 4 hidden layers and MPC horizon lengths $N = 5, 10, 15, 20$.

layers and number of neurons per hidden layer. The resulting solve times can be seen in Figure 9. We can note that the computation time increases exponentially with the number of hidden layers and neurons per hidden layer. This is as expected, since each neuron adds a binary decision variable to the MILP, increasing the solve time exponentially. On the other hand, this also means that deeper architectures are preferred since the representational power of the NN grows faster with depth as compared to increasing the number of neurons per hidden layer, resulting in architectures which can be verified more quickly.

Next, we investigate the influence of the horizon length N of the MPC problem. For this, we fix the NN to have four hidden layers. Increasing the horizon increases the number of inequalities in the parametric QP. Hence, we expect the solve times to increase since we add a binary decision variable per inequality, which can be observed in Figure 10. The computation time increases with the number of inequalities and the number of neurons per hidden layer. Note that the complexity does not depend on the state dimension of the system, meaning that our method does not necessarily suffer from the curse of dimensionality.

As a final experiment, we compare the different methods

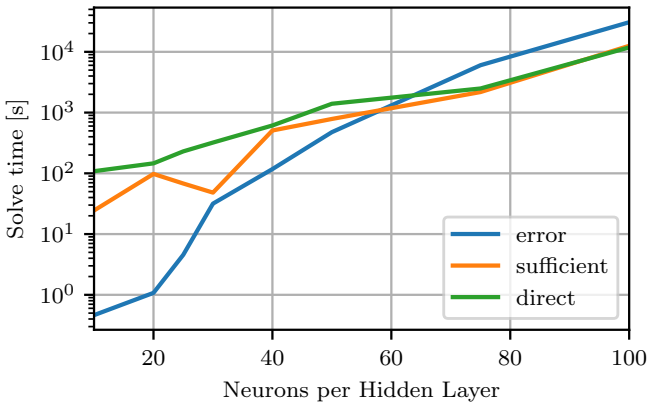

Fig. 11: Solve times of (17), (30), and (32) in seconds for 4 hidden layers and MPC horizon $N = 10$.

against each other. For this, we fix the horizon to $N = 10$ and the NN to have four hidden layers. We solve for the maximum absolute error, and verify the MPC approximation using the sufficient and direct methods introduced in Section IV by solving problems (17), (30), and (32) respectively. The resulting computation times can be seen in Figure 11. We can see that for small problems, solving for the maximum absolute error between the MPC and the NN is faster than the verification methods. But for bigger problems the verification problems are solved more quickly, although solving an indefinite MIQP is considered to be more difficult. Interestingly, the direct verification method is slower to solve than the sufficient method. Potentially, the cascading structure of the direct verification problem makes the problem harder to solve.

VII. Conclusions

We have introduced a flexible framework that allows one to formulate a variety of verification problems as MILPs or indefinite MIQPs. In particular, we have shown that not only NN with linear layers and ReLU activation functions are representable in such a framework, but also more complex layer structures such as parametric QPs and piecewise affine functions on polyhedral sets can be represented and combined.

Taking these formulations, we then constructed an MILP problem, which gives us the maximum approximation error between a given MPC and its NN approximation. Together with an MPC formulation robust in the input (ex. tube MPC), we can show constraint satisfaction and stability in the feasible region of the MPC controller. Alternatively, we provided indefinite MIQP formulations to directly verify stability and calculate an outer approximation of the stability region.

We compared our approach against a Lipschitz based method and showed that our approach outperforms it, being able to verify asymptotic stability in cases where the Lipschitz based method could only show ISS or could not show stability at all.

REFERENCES

[1] S. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, no. 7, pp. 733–764, 2003.

[2] F. Oldewurtel, A. Parisio, C. N. Jones, D. Gyalistras, M. Gwerder, V. Stauch, B. Lehmann, and M. Morari, "Use of model predictive control and weather forecasts for energy efficient building climate control," *Energy and Buildings*, vol. 45, pp. 15–27, 2012.

[3] M. W. Müller and R. D'Andrea, "A model predictive controller for quadrocopter state interception," in *European Control Conference*, 2013, pp. 1383–1389.

[4] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, 2018.

[5] P. Karamanakos, E. Liegmann, T. Geyer, and R. Kennel, "Model predictive control of power electronic systems: Methods, results, and challenges," *IEEE Open Journal of Industry Applications*, vol. 1, pp. 95–114, 2020.

[6] 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.

[7] A. Alessio and A. Bemporad, *A Survey on Explicit Model Predictive Control*. Springer, 2009, pp. 345–369.

[8] C. N. Jones, P. Grieder, and S. V. Raković, "A logarithmic-time solution to the point location problem for parametric linear programming," *Automatica*, vol. 42, no. 12, pp. 2215–2218, 2006.

[9] F. Bayat, T. A. Johansen, and A. A. Jalali, "Using hash tables to manage the time-storage complexity in a point location problem: Application to explicit model predictive control," *Automatica*, vol. 47, no. 3, pp. 571–577, 2011.

[10] M. Kvasnica and M. Fikar, "Clipping-based complexity reduction in explicit MPC," *IEEE Transactions on Automatic Control*, vol. 57, no. 7, pp. 1878–1883, 2012.

[11] A. Domahidi, M. N. Zeilinger, M. Morari, and C. N. Jones, "Learning a feasible and stabilizing explicit model predictive control law by robust optimization," in *IEEE Conference on Decision and Control and European Control Conference*, 2011, pp. 513–519.

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

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

[14] D. Psaltis, A. Sideris, and A. Yamamura, "A multilayered neural network controller," *IEEE Control Systems Magazine*, vol. 8, no. 2, pp. 17–21, 1988.

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

[16] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[17] 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.

[18] 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.

[19] M. Baes, M. Diehl, and I. Necoara, "Every continuous nonlinear control system can be obtained by parametric convex programming," *IEEE Transactions on Automatic Control*, vol. 53, no. 8, pp. 1963–1967, 2008.

[20] A. B. Hempel, P. J. Goulart, and J. Lygeros, "Every continuous piecewise affine function can be obtained by solving a parametric linear program," in *European Control Conference*, 2013, pp. 2657–2662.

[21] B. Amos and J. Z. Kolter, "OptNet: Differentiable optimization as a layer in neural networks," in *International Conference on Machine Learning*, 2017.

[22] 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.

[23] S. Bai, J. Z. Kolter, and V. Koltun, "Deep equilibrium models," in *Advances in Neural Information Processing Systems*, 2019.

[24] P. Kumar, J. B. Rawlings, and S. J. Wright, "Industrial, large-scale model predictive control with structured neural networks," *Computers & Chemical Engineering*, vol. 150, p. 107291, 2021.

[25] J. Nubert, J. Köhler, V. Berenz, F. Allgöwer, and S. Trimpe, "Safe and fast tracking on a robot manipulator: Robust MPC and neural network control," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3050–3057, 2020.

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

[27] E. Maddalena, C. da S. Moraes, G. Waltrich, and C. N. Jones, "A neural network architecture to learn explicit MPC controllers from data," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 11 362–11 367, 2020, 21th IFAC World Congress.

[28] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 269–296, 2020.

[29] 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.

[30] 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.

[31] H. Dai, B. Landry, M. Pavone, and R. Tedrake, "Counter-example guided synthesis of neural network lyapunov functions for piecewise linear systems," in *IEEE Conference on Decision and Control*, 2020, pp. 1274–1281.

[32] S. Chen, M. Fazlyab, M. Morari, G. J. Pappas, and V. M. Preciado, "Learning lyapunov functions for hybrid systems," in *International Conference on Hybrid Systems: Computation and Control*, 2021, pp. 1–11.

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

[34] 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.

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

[36] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[37] R. R. Bunel, I. Turkaslan, P. Torr, P. Kohli, and P. K. Mudigonda, "A unified view of piecewise linear neural network verification," in *Advances in Neural Information Processing Systems*, 2018.

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

[39] E. Wong and Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *International Conference on Machine Learning*, 2018, pp. 5286–5295.

[40] 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 *IEEE Symposium on Security and Privacy*, 2018, pp. 3–18.

[41] Gurobi Optimization, LLC, "Gurobi optimizer reference manual," 2022.

[42] R. Anderson, J. Huchette, W. Ma, C. Tjandraatmadja, and J. P. Vielma, "Strong mixed-integer programming formulations for trained neural networks," *Mathematical Programming*, vol. 183, no. 1, pp. 3–39, Sept. 2020.

[43] D. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1999.

[44] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[45] T. Marcucci and R. Tedrake, "Mixed-integer formulations for optimal control of piecewise-affine systems," in *International Conference on Hybrid Systems: Computation and Control*, 2019, p. 230–239.
[46] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.
[47] M. Herceg, M. Kvasnica, C. N. Jones, and M. Morari, "Multi-parametric toolbox 3.0," in *European Control Conference*, 2013, pp. 502–510.
[48] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.
[49] C. N. Jones and M. Morari, "Approximate explicit MPC using bilevel optimization," in *European Control Conference*, 2009, pp. 2396–2401.
[50] D. Simon and J. Löfberg, "Stability analysis of model predictive controllers using mixed integer linear programming," in *IEEE Conference on Decision and Control*, 2016, pp. 7270–7275.
[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

APPENDIX I
BOUND IMPROVEMENTS FOR PARAMETRIC QP FORMULATION

As we have seen in section II-C, the complementarity constraints of the KKT conditions have been formulated as "big-M" constraints (10). In practice, choosing M too large can lead to numerical issues; hence, ideally, we can calculate a finite bound beforehand [44]. Also, if the parametric QP is not the last layer, then bounds of the output y may even be required by the next layer.

Assuming we are given or have calculated upper and lower bounds for the input $\underline{x} \leq x \leq \overline{x}$, we can calculate

$$\begin{aligned} M_i^z = \max_{z,\lambda,\mu_{\text{eq}}} \quad & g_i - F_i z \\ \text{s.t.} \quad & \underline{x} \leq Dz \leq \overline{x}, \\ & Az = b, Fz \leq g, \\ & Pz + q + A^T\mu_{\text{eq}} + F^T\lambda = 0, \\ & \lambda \geq 0, \end{aligned} \tag{45}$$

for $i = 1, \ldots, n_{\text{ineq}}$ and

$$\begin{aligned} M_i^\lambda = \max_{z,\lambda,\mu_{\text{eq}}} \quad & \lambda_i \\ \text{s.t.} \quad & \underline{x} \leq Dz \leq \overline{x}, \\ & Az = b, Fz \leq g, \\ & Pz + q + A^T\mu_{\text{eq}} + F^T\lambda = 0, \\ & \lambda \geq 0, \end{aligned} \tag{46}$$

for $i = 1, \ldots, n_{\text{ineq}}$. Then we can take

$$M = \max(\max_i M_i^z, \max_i M_i^\lambda). \tag{47}$$

In case the optimization problems are unbounded, one may include the MILP constraints

$$0 \leq \lambda_i \leq M\beta_i, \tag{48a}$$
$$0 \leq g_i - F_i z \leq M(1 - \beta_i), \tag{48b}$$

again into the formulations (45) and (46). Although these MILPs run orders of magnitude faster than the actual verification problem, it may still take considerable time since we are solving $2n_{\text{ineq}}$ of them. In practice, these MILPs are not solved to optimality but are interrupted after a predefined timeout, and the best LP-Relaxation bound is used.

APPENDIX II
PROOFS

A. Proof of Lemma 6

Proof: Since the polytopic set $\mathbb{X}$ is bounded, there exists a bounded box $[\underline{t}, \overline{t}] \supseteq \mathbb{X}$. For every $i = 1, \ldots, m$ introduce a continuous decision variable $z_i \geq 0$ as well as a binary decision variable $\beta_i \in \{0, 1\}$ subject to the constraints

$$t_i \leq z_i \leq t_i + 2\overline{t}\beta_i, \tag{49a}$$
$$-t_i \leq z_i \leq -t_i - 2\underline{t}(1 - \beta_i). \tag{49b}$$

If $t_i > 0$, then (49b) implies that $\beta_i = 0$, and (49a) implies that $z_i = t_i$. If $t_i < 0$, on the other hand, then (49a) implies that $\beta_i = 1$, and (49b) implies that $z_i = -t_i$. If $t_i = 0$, finally, then (49) implies that $z_i = 0$ irrespective of β_i. In any case, we thus have $z_i = |t_i|$.

In order to express the infinity norm $f(t) = \max(z_1, \ldots, z_m)$ in terms of linear constraints, we introduce the continuous decision variables τ and $\tilde{z}_i$ for every $i = 1, \ldots, m$, as well as the binary decision variables $\tilde{\beta}_i \in \{0, 1\}$, $i = 1, \ldots, m$, subject to the constraints

$$1 = \sum_{i=1}^m \tilde{\beta}_i, \tag{50a}$$
$$\tau = \sum_{i=1}^m \tilde{z}_i, \tag{50b}$$
$$\tau \geq z_i \qquad \forall i = 1, \ldots, m, \tag{50c}$$
$$0 \leq \tilde{z}_i \leq \max(|\underline{t}|, |\overline{t}|)\tilde{\beta}_i \quad \forall i = 1, \ldots, m, \tag{50d}$$
$$\tilde{z}_i \leq z_i \qquad \forall i = 1, \ldots, m. \tag{50e}$$

The forcing constraints (50d) ensure that $\tilde{z}_i = 0$ whenever $\tilde{\beta}_i = 0$, whereas (50a) ensures that $\tilde{\beta}_i = 1$ for exactly one i. Hence, (50b) sets $\tau = \tilde{z}_i$ for the unique i with $\tilde{\beta}_i = 1$. By (50e), we thus have $\tau = \tilde{z}_i \leq z_i \leq \max(z_1, \ldots, z_m)$. The constraint (50c), finally, ensure the converse inequality $\tau \geq z_i \geq \max(z_1, \ldots, z_m)$. Hence, $\|t\|_\infty$ is MILP representable,

and its graph formulation can be expressed as

$$\mathrm{gr}(f) = \left\{ (t,\tau) \;\middle|\; \begin{array}{r} \forall i = 1,\dots,m, \\ \exists \beta, \tilde{\beta} \in \{0,1\}^m, \\ \exists z, \tilde{z} \in \mathbb{R}^m : \\ t \in \mathbb{X} \\ t_i \le z_i \le t_i + 2\bar{t}\beta_i, \\ -t_i \le z_i \\ \le -t_i - 2\underline{t}(1-\beta_i), \\ 1 = \sum_{j=1}^{m} \tilde{\beta}_j, \\ \tau = \sum_{j=1}^{m} \tilde{z}_j, \\ 0 \le \tilde{z}_i \le \max(|\underline{t}|, |\bar{t}|)\tilde{\beta}_i, \\ \tilde{z}_i \le z_i \le \tau \end{array} \right\}. \quad (51)$$

■

B. Proof of Corollary 1

Proof: The proof widely parallels that of Lemma 6. However, there is no need to introduce decision variables $\tilde{\beta}$ and $\tilde{z}$ because we can directly set

$$\tau = \|t\|_1 = \sum_{i=1}^{m} z_i. \quad (52)$$

■