Assured Neural Network Architectures for Control and Identification of Nonlinear Systems

James Ferlez* and Yasser Shoukry*

Abstract—In this paper, we consider the problem of automatically designing a Rectified Linear Unit (ReLU) Neural Network (NN) architecture (number of layers and number of neurons per layer) with the assurance that it is sufficiently parametrized to control a nonlinear system; i.e. control the system to satisfy a given formal specification. This is unlike current techniques, which provide no assurances on the resultant architecture. Moreover, our approach requires only limited knowledge of the underlying nonlinear system and specification. We assume only that the specification can be satisfied by a Lipschitz-continuous controller with a known bound on its Lipschitz constant; the specific controller need not be known. From this assumption, we bound the number of affine functions needed to construct a Continuous Piecewise Affine (CPWA) function that can approximate any Lipschitz-continuous controller that satisfies the specification. Then we connect this CPWA to a NN architecture using the authors’ recent results on the Two-Level Lattice (TLL) NN architecture; the TLL architecture was shown to be parameterized by the number of affine functions present in the CPWA function it realizes.

I. INTRODUCTION

Recent advances in theory and computation have facilitated wide-spread adoption of Rectified Linear Unit (ReLU) Neural Networks (NNs) in conventional feedback-control settings, especially for Cyber-Physical Systems (CPSs) [1]. However, this proliferation of NN controllers has also highlighted weaknesses in state-of-the-art NN design techniques, because CPS systems are often safety critical. In a safety-critical CPS, it is not enough to simply learn a NN controller, i.e. fit data: such a controller must also have demonstrable safety or robustness properties, usually with respect to closed-loop specifications on a dynamical system. Moreover, a meaningful safety specification for a CPS is often binary: either a controller makes the system safe or it doesn’t. This is in contrast to optimization-based approaches typical in Machine/Reinforcement Learning (ML/RL), where the goal is to optimize a particular cost or reward without any requirements imposed on eventual quantity in question. Examples of the former include stability about an equilibrium point or forward invariance of a particular set of safe states (e.g. for collision avoidance). Examples of the latter include minimizing the mean-squared fit error; minimizing regret; or maximizing a surrogate for a value function; etc.

The distinction between safety specifications and ML/RL objectives is relevant to more than just learning NN weights and biases, though: it is has special relevance to NN architecture design — i.e. deciding on the number of neurons and their connection (or arrangement) in the NN to be trained in the first place. Specifically, (binary) safety specifications beg existential questions about NN architectures in a way that conventional, optimization-focused ML/RL techniques do not: given a particular NN architecture for a controller, it is necessary to ask whether there is there any possible choice of weights and biases that achieve the desired safety specification. By contrast, conventional ML/RL type problems instead take an architecture as given, and attempt to achieve the best training error, reward, etc. subject to that implicit constraint. Thus, in typical ML/RL treatments, NN architectures merely adjust the final cost/reward, rather than leading to ill-posedness, such as can occur with a safety specification.

In this paper, we directly address the issue of whether a ReLU NN architecture is well-posed as a state-feedback controller for a given nonlinear system and closed-loop (safety) specification. That is we present a systematic methodology for designing a NN controller architecture that is guaranteed (or assured) to be able to meet a given closed-loop specification. Our approach provides such a guarantee contingent on the following properties of the system and specification:

(i) the nonlinear system’s vector field is Lipschitz continuous with known Lipschitz constants;

(ii) there exists a Lipschitz continuous controller that satisfies the closed-loop specification robustly, and the Lipschitz constant of that controller is known (although the controller itself need not be known); and

(iii) the conjectured Lipschitz continuous controller makes a compact subset of the state space positive invariant.

The need to assume the existence of a controller is primarily to ensure that the specification is well-posed in general — i.e., for any controller, whether it is a NN or not; we will elaborate on the robustness in (ii) subsequently. Importantly, subject to these conditions, our approach can design a NN controller architecture with the following assurance: there exists neuron weights/biases for that architecture such that it can exactly meet the same specification as the assured non-NN controller (albeit non-robustly). Moreover, our proposed methodology requires only the information described above, so it is applicable even without perfect knowledge of the underlying system dynamics, albeit at the expense of designing rather larger architectures.

The cornerstone of our approach is a special ReLU NN architecture introduced by the authors in the context of another control problem: viz. the Two-Level Lattice (TLL) ReLU NN architecture [2]. A TLL NN, like all ReLU NNs, instantiates a Continuous, Piecewise Affine (CPWA) function, and hence implements one of finitely many local linear functions at each point in its domain; thus, its domain, like that of any CPWA, can be partitioned into linear regions, each of which corresponds to a different local linear function. However,

*Department of Electrical Engineering and Computer Science, University of California, Irvine {jferlez,yshoukry}@uci.edu
This work was partially sponsored by the NSF awards #CNS-2002405 and CNS-2013824.

1The term “linear” here is somewhat of a misnomer: “affine” is more accurate. However, we use this terminology for consistency with the literature.
Unlike general ReLU NN architectures, a TLL NN exposes these local linear functions directly in the parameters of the network. As a consequence of this parameterization, then, TLL NNs are easily parameterized by their number of linear regions: after all, each linear region must instantiate at least one local linear function – see [2, Theorem 2]. In fact, this idea also applies to upper bounds on the number of regions desired in an architecture; see [2, Theorem 3]. TLL NNs thus have the special property that they can be used to connect a desired number of linear regions directly to a ReLU architecture.

Thus, to obtain an assured controller architecture, it is enough to obtain an assured upper-bound on the number of linear regions required of a controller. In this paper, we show that such an assured upper bound can be obtained using just the information assumed in (i) - (iii); i.e. primarily Lipschitz constants and bounds on the relevant objects. This bound is derived by counting the number of linear regions needed to linearly and continuously interpolate between regularly spaced “samples” of a controller with known Lipschitz constant.

Moreover, this core approach is relevant to end-to-end learning beyond just the design of controller architectures: it can also be used to obtain architectures that are guaranteed to represent the dynamics of a nonlinear system itself – i.e. assured architectures for system identification. Indeed, in this paper, we further show that contingent on information akin to (i) and (iii), it is possible to generate a ReLU architecture that is guaranteed to capture the essence of an unknown nonlinear control system. Specifically, we exhibit a methodology for designing an architecture to represent a nonlinear (controlled) vector field with the following assurance: if the ReLU vector field is sufficiently well trained on data from a compatible – but unknown – controlled vector field, then robustly controlling the ReLU dynamics to a specification will yield a controller that likewise controls the unknown dynamics to the same specification (albeit non-robustly). Providing a guaranteed architecture for system identification has unique value for end-to-end learning: because the ReLU control system can be used as a surrogate for the original nonlinear system, control design can be moved entirely from the unknown system to the known ReLU surrogate, the latter of which can be simulated instead. Furthermore, this system identification can be combined with a guaranteed controller architecture to do in silico ReLU control design.

The contributions of this paper can be summarized thusly:

I. A new notion of Abstract Disturbance Simulation (ADS) to formulate of robust specification satisfaction; ADS unifies and generalizes several related notions of simulation in the literature – see Section III-B.

II. A methodology to design a ReLU architecture that is assured to be able to control an unknown nonlinear system to meet a closed-loop specification; this is subject to the existence of a (likewise) unknown controller that robustly meets the same specification.

III. A methodology to design a ReLU architecture that can be used in system identification of an unknown nonlinear control system; this architecture, when adequately trained, is assured to be viable as a surrogate for the original nonlinear system in controller design.

A preliminary version of this paper appeared as [3]. Relative to [3], this paper has the following additional novel content: first, it uses new, dramatically improved techniques to obtain a smaller architecture than [3] (see Remark 3); second, it includes full proofs of every claim; and third, it contains the extension to system identification architectures noted above.

A. Related Work

The literature most directly relevant to this paper is work by the authors: AReN [2] and the preliminary version of this paper, [3]. The former contains an algorithm that generates an architecture assured to represent an optimal MPC controller. The AReN algorithm is fully automatic, and the assurances provided are the same as those for the referent MPC controller; but it is not very generalizable, given the restriction to MPC control. This paper and its preliminary version offer a significantly improved methodology. The architecture design presented herein is likewise fully automatic; however, it is generalizable to any Lipschitz control system, and it provides assurances for a wide variety specification captured by simulation relations.

The largest single class of NN architecture design algorithms is commonly referred to as Neural Architecture Search (NAS) algorithms. These algorithms essentially use an iterative improvement/optimization scheme to design a NN architecture, thus automating something like a “guess-train-evaluate” hyperparameter tuning loop. There is a large literature on NAS algorithms; [4] is a good summary. Typical NASs design architectures within some structured class of architectures such as: “chain” architectures (i.e. a sequence of fully connected layers) [5]; chain architectures with different layer types (e.g. convolution, pooling, etc. in addition to fully connected) [6], [7]; or mini-NN architectures that are replicated and interconnected to form larger networks [8]. They then update these architectures according to a variety of different mechanisms: RL formulations [9], [10], [8]; sequential decision problems formulations where actions corresponding to network morphisms [7]; Bayesian optimization formulations [11], [5]; and Neuro-evolutionary approaches (relatedly population dynamics or genetic algorithms) [12], [13]. Different evaluation mechanisms are used to evaluate the “quality” of current architecture iterate: lower fidelity models [8] or weight inheritance morphisms [4], [7] (a sort of transfer learning); learning curve extrapolation to estimate the final performance of an architecture before training has converged [6]; and one-shot models that agglomerate many architectures into a single large architecture that shares edges between individual architectures [14]. Notably these algorithms all share the same features: they are highly automatic, even accounting for the need to choose meta-hyperparameters; they are fairly general, since they are data-driven; however, they provide no closed-loop assurances.

At the opposite end of our assessment spectra are control-based methods for obtaining NN controllers with assurances; we regard these methods as implicit architecture design techniques, since exhibiting a NN controller serves as a direct validation of that controller’s architecture. Examples of these methods include: directly approximating a controller by a NN for non-affine systems [15]; adaptively learning NN controller weights to ensure Input-to-State stability for certain third-order affine systems [16], [17]; NN hybrid adaptive control for stabilizing uncertain impulsive dynamical systems [18]. These methods are based on the assertion that a function of interest can be approximated by a sufficiently large (usually shallow) NN: the size of this NN is explicit, which limits their effectiveness as architecture design methods. Even neglecting this shortcoming, these methods generally provide just one meaningful assurance (stability); they are not at all general,
since they are based on approximating a specific, hand-designed controller; and they are thus highly manual methods.

A subset of NN verification methods from the control system literature is related to the “guess-train-evaluate” architecture design iterations described above. In particular, some closed-loop NN verifiers provide additional dynamical information about how a NN controller fails to meet a specification. Examples include: using complementary analysis on NN-controlled linear systems, thus obtaining sufficient conditions for stability in terms of LMIs [19]; training and verifying a specific NN architecture as a barrier certificate for hybrid systems [20]; and using adversarial perturbation to verify NN control policies [21]. These methods can be assessed as follows: they are highly automatic (verifiers); they are of limited generalizability, since they require either specific models and/or detailed knowledge of the dynamical model; and each provides one and only one assurance (e.g. stability or a barrier certificate). A similar, but less applicable, subset of the control literature that consists of experimental work that suggests promising NN controller architectures. These works do not do automatic NN architecture design, nor do they contain verification algorithms of the type suggested above. Even so, they experimentally support using some conventional NN architectures as controllers [22]; and using Input Convex NNs (ICNNs) for controllers and system identification [23].

We also acknowledge prior work on system identification using NNs, although they do not emphasize architecture design. These methods suffer from a lack of assurances on the resultant NNs/architectures [24], [25], [26], [27].

Finally, subsequent to the publication of [3], we became aware of other works that use more or less what we describe as TLL NNs [28], [29]. The former is concerned with simplification of explicit MPC controllers rather than NN architecture design; the latter can be regarded as architecture design for a different application (although the architectures are less efficient than the ones presented here – see Remark 4).

II. PRELIMINARIES

A. Notation

We denote by \( \mathbb{N} \), \( \mathbb{R} \) and \( \mathbb{R}^+ \) the set of natural numbers, the set of real numbers and the set of non-negative real numbers, respectively. For a function \( f : A \to B \), let \( \text{dom}(f) \) return the domain of \( f \), and let \( \text{range}(f) \) return the range of \( f \). For \( x \in \mathbb{R}^n \), we will denote by \( \|x\| \) the max-norm of \( x \). Relatedly, for \( x \in \mathbb{R}^n \) and \( \varepsilon \geq 0 \) we will denote by \( B(x; \varepsilon) \) the open ball of radius \( \varepsilon \) centered at \( x \) as specified by \( \| \cdot \| \), and \( \overline{B}(x; \varepsilon) \) its closed-ball analog. Let \( \text{int}(X) \) denote the interior of a set \( X \subseteq \mathbb{R}^n \), and \( \text{bd}(X) = X \setminus \text{int}(X) \) denote its boundary. \( e_i \) will denote the \( i \)th column of the \( (n \times n) \) identity matrix, unless otherwise specified. Let \( \text{hull}(X) \) denote the convex hull of a set \( X \subseteq \mathbb{R}^n \), \( \|f\|_{\infty} \triangleq \text{ess sup}_{x \in \mathbb{R}^n} \|f(x)\| \), and \( \|f\|_X \) will denote the same but restricted to the set \( X \). The projection map over \( \mathbb{R}^n \) will be denoted by \( \pi_i : \mathbb{R}^n \to \mathbb{R} \), so that \( \pi_i(x) \) returns the \( i \)th component of the vector \( x \) (in the understood coordinate system). Finally, given two sets \( A \) and \( B \) denote by \( B^A \) the set of all functions \( f : A \to B \).

B. Dynamical Model

In this paper, we will assume an underlying, but not necessarily known, continuous-time nonlinear dynamical system specified by an ordinary differential equation (ODE): that is

\[ \dot{x}(t) = f(x(t), u(t)) \]  \hspace{1cm} (1)

where the state vector \( x(t) \in \mathbb{R}^n \) and the control vector \( u(t) \in \mathbb{R}^m \). Formally, we have the following definition:

**Definition 1** (Control System). A control system is a tuple \( \Sigma = (X, U, \mathcal{U}, f) \) where

- \( X \subseteq \mathbb{R}^n \) is the connected, compact subset of the state space with non-empty interior;
- \( U \subseteq \mathbb{R}^m \) is the compact set of admissible controls;
- \( \mathcal{U} \subseteq U \times \mathbb{R}^+ \) is the space of admissible open-loop control functions – i.e. \( v \in \mathcal{U} \) is a function \( v : \mathbb{R}^+ \to U \) and
- \( f : \mathbb{R}^n \times U \to \mathbb{R}^n \) is a vector field specifying the time evolution of states according to (1).

A control system is said to be (globally) Lipschitz if there exists constants \( K_x \) and \( K_u \) s.t. for all \( x, x' \in \mathbb{R}^n \) and \( u, u' \in \mathbb{R}^m \):

\[ \|f(x, u) - f(x', u')\| \leq K_x \|x - x'\| + K_u \|u - u'\|. \]  \hspace{1cm} (2)

In the sequel, we will primarily be concerned with solutions to (1) that result from instantaneous state-feedback controllers, \( \Psi : X \to U \). Thus, we use \( \zeta_{x_0, t} \) to denote the closed-loop solution of (1) starting from initial condition \( x_0 \) (at time \( t = 0 \)) and using state-feedback controller \( \Psi \). We refer to such a \( \zeta_{x_0, t} \) as a (closed-loop) trajectory of its associated control system.

**Definition 2** (Closed-loop Trajectory). Let \( \Sigma \) be a Lipschitz control system, and let \( \Psi : \mathbb{R}^n \to U \) be a globally Lipschitz continuous function. A closed-loop trajectory of \( \Sigma \) under controller \( \Psi \) and starting from \( x_0 \in X \) is the function \( \zeta_{x_0, t} : \mathbb{R}^+ \to X \) that solves the integral equation:

\[ \zeta_{x_0, t}(t) = x_0 + \int_0^t f(\zeta_{x_0, s}(\sigma), \Psi(\zeta_{x_0, s}(\sigma)))d\sigma. \]  \hspace{1cm} (3)

It is well known that such solutions exist and are unique under these assumptions [30].

**Definition 3** (Feedback Controllable). A Lipschitz control system \( \Sigma \) is feedback controllable by a Lipschitz controller \( \Psi : \mathbb{R}^n \to U \) if the following is satisfied:

\[ \Psi \circ \zeta_{x_0, t} \in \mathcal{U} \hspace{1cm} \forall x \in X. \]  \hspace{1cm} (4)

If \( \Sigma \) is feedback controllable for any such \( \Psi \), then we simply say that it is feedback controllable.

Because we’re interested in a compact set of states, \( X \), we consider only feedback controllers whose closed-loop trajectories stay within \( X \).

**Definition 4** (Positive Invariance). A feedback trajectory of a Lipschitz control system, \( \zeta_{x_0, t} \), is positively invariant if \( \zeta_{x_0, t}(t) \in X \) for all \( t \geq 0 \). A controller \( \Psi \) is positively invariant if \( \zeta_{x_0, t} \) is positively invariant for all \( x_0 \in X \).

For technical reasons, we will also need the following stronger notion of positive invariance.

**Definition 5** (\( \delta, \tau \) Positive Invariance). Let \( \delta, \tau > 0 \) and \( \text{edge}_\delta(X) \triangleq \{ x \in X \mid \text{int}(X) \cap B(x; \delta) \neq \emptyset \} \). Then a positively invariant controller \( \Psi \) is \( \delta, \tau \) positively invariant if

\[ \forall x_0 \in \text{edge}_\delta(X) \cdot \zeta_{x_0, t}(\tau) \in X \setminus \text{edge}_\delta(X) \]  \hspace{1cm} (5)

and \( \Psi \) is positively invariant with respect to \( X \setminus \text{edge}_\delta(X) \).

For a \( \delta, \tau \) positively invariant controller, trajectories that start \( \delta \)-close to the boundary of \( X \) will end up \( \delta \)-far away from that boundary after \( \tau \) seconds, and remain there forever after.
Finally, borrowing from [31], we define a \( \tau \)-sampled transition system embedding of a feedback-controlled system.

**Definition 6** (\( \tau \)-sampled Transition System Embedding). Let \( \Sigma = (\Sigma, \Pi, \tau, f) \) be a feedback controllable Lipschitz control system, and let \( \Psi : \mathbb{R}^m \rightarrow U \) be a Lipschitz continuous feedback controller. For any \( \tau > 0 \), the \( \tau \)-sampled transition system embedding of \( \Sigma \) under \( \Psi \) is the tuple \( \Sigma_{\tau}(\Pi_{\Psi}) = (\Sigma_{\tau}, \Pi_{\Psi}, \tau) \) where:

- \( \Sigma_{\tau} = X \) is the state space;
- \( \Pi_{\Psi} = \{ (\Psi \circ \zeta_{u}(\tau)) | u \in [0, \tau] \} \) is the set of open loop control inputs generated by \( \Psi \)-feedback, each restricted to the domain \([0, \tau]\); and
- \( \zeta_{u}(\tau) \subseteq X \times \Pi_{\Psi} \times X \) such that \( x \circ \zeta_{u}(\tau) \rightarrow x' \) iff both \( u = (\Psi \circ \zeta_{u}(\tau)) | \tau \) and \( x' = \zeta_{u}(\tau) \).

\( \Sigma_{\tau}(\Pi_{\Psi}) \) is thus a metric transition system [31].

**Definition 7** (Simulation Relation). Let \( S_1 = (X_1, U_1, \tau) \) and \( S_2 = (X_2, U_2, \tau) \) be two metric transition systems. Then we say that \( S_2 \) simulates \( S_1 \), written \( S_1 \preceq S_2 \), if there exists a relation \( \preceq \subseteq X_1 \times X_2 \) such that:

- \( \{ x_1 | (x_1, x_2) \in \preceq \} = X_1 \); and
- for all \( (x_1, x_2) \in \preceq \) we have \( x_1 \rightarrow x_1' \iff \exists x_2 \in X_2, (x_1', x_2) \in \preceq \land x_2 \rightarrow x_2' \).

### C. ReLU Neural Network Architectures

We will consider controlling the nonlinear system defined in (1) with a state-feedback neural network controller \( \mathcal{N} \):

\[
\mathcal{N} : X \rightarrow U
\]

where \( \mathcal{N} \) denotes a Rectified Linear Unit Neural Network (ReLU NN). Such a \( (K) \)-layer ReLU NN is specified by \( K \) layers (or just layers). A layer with \( i \) inputs and \( \sigma \) outputs is specified by a \( (\sigma \times 1) \) matrix of weights, \( W_i \), and a \( (1 \times 1) \) matrix of biases, \( b_i \), as follows:

\[
\begin{align*}
\mathcal{N}(x) &= (L_{\theta(1)} \circ L_{\theta(2)} \circ \cdots \circ L_{\theta(K)})(x), \\
\end{align*}
\]

where the max function is taken element-wise, and \( \theta \equiv (W, b) \) for brevity. Thus, a \( K \)-layer ReLU NN function is specified by \( K \) layer functions \( \{ L_{\theta(i)} : i = 1, \ldots, K \} \) whose input and output dimensions are composable: that is, they satisfy \( i_o = i_{o-1} \) for \( i = 2, \ldots, K \).

### Theorem 1 (Two-Level-Lattice (TLL) NN Architecture)

Let \( \Theta_i = ((W_{i,1}^1, b_{i,1}^1), \ldots, (W_{i,K}^n, b_{i,K}^n)) \), \( i = 1, 2 \). Then the parallel composition of \( \mathcal{N}_\Theta \) and \( \mathcal{N}_{\Theta_2} \) is a NN given by the parameter list:

\[
\Theta_1 \parallel \Theta_2 \equiv \left( \left( W_{i,1}^1, b_{i,1}^1 \right), \ldots, \left( W_{i,K}^n, b_{i,K}^n \right) \right).
\]

That is \( \Theta_1 \parallel \Theta_2 \) accepts an input of the same size as both \( \Theta_1 \) and \( \Theta_2 \), but has as many outputs as \( \Theta_1 \) and \( \Theta_2 \) combined.

### Corollary 1

Let \( f : \mathbb{R}^n \rightarrow \mathbb{R}^m \) be a CPWA function, each of whose \( m \) component CPWA functions are denoted by \( f_i : x \mapsto \pi_i(f(x)) \), and let \( N \) be an upper bound on the number of linear regions in each \( f_i \).

Then for \( \Theta_{\mathcal{N}} \) a TLL architecture representing a CPWA \( \mathcal{N} \), the \( m \)-fold parallel TLL architecture:

\[
\Theta_{\mathcal{N}}(\mathcal{N}) \equiv \Theta_{\mathcal{N}}(\mathcal{N}) \parallel \cdots \parallel \Theta_{\mathcal{N}}(\mathcal{N})
\]

has the property that \( f = \mathcal{N}_{\Theta_{\mathcal{N}}} \).

**Proof.** Apply Theorem 1 component-wise.

Finally, note that a ReLU NN function, \( \mathcal{N} \), is known to be a continuous, piecewise affine (CPWA) function consisting of finitely many linear segments. Thus, a function \( \mathcal{N} \) is itself necessarily globally Lipschitz continuous.

### D. Notation Pertaining to Hypercubes

Since the unit ball of the max-norm, \( || \cdot || \), on \( \mathbb{R}^n \) is a hypercube, we will make use of the following notation.

**Definition 11** (Face/Corner of a hypercube). Let \( C_n = [0, 1]^n \) be a unit hypercube of dimension \( n \). A set \( F \subseteq C_n \) is a \( k \)-dimensional face of \( C_n \) if there exists a set \( J \subseteq \{1, \ldots, n\} \) such that \( |J| = n-k \) and

\[
\forall x \in F . \bigwedge_{j \in J} (\pi_j(x) \in \{0, 1\}).
\]
Let $\mathcal{F}_n(C_n)$ denote the set of $k$-dimensional faces of $C_n$, and let $\mathcal{F}(C_n)$ denote the set of all faces of $C_n$ (of any dimension). A corner of $C_n$ is a 0-dimensional face of $C_n$. Furthermore, we will use the notation $F^k$ to denote an full-dimensional face (a 1-dimensional in this case) whose index set $J = \{i\}$ and whose projection on the $i^{th}$ coordinate is $k \in (0,1)$.

We extend these definitions by isomorphism to any other hypercubes in $\mathbb{R}^n$.

### III. Problem Formulation: NN Architectures for Control

We begin by stating the first main problem that we will consider in this paper: that of designing an assured NN architecture for nonlinear control (hereafter referred to as the controller architecture problem). Specifically, we wish to identify a ReLU architecture to be used as a feedback controller for the control system $\Sigma$; this architecture must further come with the assurance that there exist parameter weights for which the realized NN controller controls the architecture for nonlinear control (hereafter referred to as the controller architecture design problem). Our solution of this problem, then, actually incorporates ADS (see also [3]). The second innovation of this paper: a new simulation relation, Abstract Disturbance Simulation (ADS) (see also [3]). Simulating is a formal notion of specification satisfaction for metric transition systems; we call this relation as a formal notion of specification satisfaction for metric transition systems; we call this relation an abstract disturbance simulation.

The main assumption in Problem 1 is that there exists a controller $\Psi$ which satisfies the specification, $S_{spec}$. We use this assumption primarily to help ensure that the problem is well posed: indeed, it is known that there are nonlinear control problems for which no continuous controllers exits. Thus, this assumption is in some sense an essential requirement to formulate a well-posed controller architecture problem: for if there exists no such $\Psi$, then there could be no NN controller that satisfies the specification, either, since the latter also belongs to the class of Lipschitz continuous functions (modulo discrepancies in Lipschitz constants). In this way, the existence of a controller $\Psi$ also subsumes any possible conditions on the nonlinear system that one might wish to impose: stabilizability, for example.

That Problem 1 is more or less ill-posed without assuming the existence of a controller $\Psi$ (for some constant $K_{cont}$) suggests a natural solution to the problem. In particular, a NN architecture can be bootstrapped from this knowledge by simply designing an architecture that is sufficiently parameterized as to adequately approximate any such $\Psi$, i.e. any function of Lipschitz constant at most $K_{cont}$ (this is a preview of the approach we will subsequently use). Unfortunately, however, this approach also reveals a deficiency in the assumption associated with $\Psi$: controller approximation necessarily introduces instantaneous control errors relative to $\Psi$, and these errors can compound transition upon transition from the dynamics. As a consequence, the assumed information about $\Psi$ is actually not as immediately helpful as it appears. In particular, if $\Psi$ is not very robust, then the accumulation of such errors could make it impossible to prove that an (approximate) NN controller satisfies the same specification as $\Psi$, viz. (17).

This effect can be seen directly in terms of the simulation relations in (16) and (17). Take $x \in X_{\tau}$ and consider two transitions from $x$: one in $S_\tau(\Sigma_\Psi)$ given by $x \xrightarrow{\overline{\nu}_{\Psi}} x'$ and one in $S_\tau(\Sigma_{\Psi_0})$ given by $x \xrightarrow{\nu_{\overline{\Psi}_{\Theta}}} x''$. Note that if $\overline{\nu}_{\Psi}$ merely approximates $\Psi$, there will in general be a discrepancy between $x'$ and $x''$, i.e. $x' \neq x''$. Thus, although $x''$ necessarily has a simulating state in $S_{spec}$ by assumption, $x''$ need not have its own simulating state in $S_{spec}$. This follows because the simulation relation in Definition 7 can assert simulating states only through transitions (i.e. (6)), and there may be no transition $x \xrightarrow{\nu_{\overline{\Psi}_{\Theta}}} x''$ in $S_\tau(\Sigma_\Psi)$.

### B. Abstract Disturbance Simulation

Motivated by the observations above, we propose a new simulation relation as a formal notion of specification satisfaction for metric transition systems; we call this relation abstract disturbance simulation or ADS (see also [3]). Simulation by means of an ADS relation is stronger than ordinary simulation (Definition 7) in order to incorporate a notion of...
robustness. Thus, abstract disturbance simulation is inspired by – and is related to – both robust bisimulation [33] and especially disturbance bisimulation [34]. Crucially however, it abstracts those notions away from their definitions in terms of specific control system embeddings and explicit modeling of disturbance inputs. As a result, ADS can then be used in a generic context such as the one suggested by Problem 1.

Fundamentally, ADS still functions in terms of conventional simulation relations; however, it incorporates robustness by first augmenting the simulated system with “virtual” transitions, each of which has a target that is perturbed from the target of a corresponding “real” transition. In this way, it is conceptually similar to the technique used in [31] and [35] to define a quantized abstraction, where deliberate nondeterminism is introduced in order to account for input errors. As a result of these additional transitions, when a metric transition system is ADS simulated by a specification, this implies that the system robustly satisfies the specification relative to satisfaction merely by an ordinary simulation relation.

As a prerequisite for defining ADS, we introduce the following definition: it captures the idea of augmenting a metric transition system with virtual, perturbed transitions.

**Definition 12 (Perturbed Metric Transition System).** Let $S = (X, U, \rightarrow)$ be a metric transition system where $X \subseteq X_M$ for a metric space $(X_M, d)$. Then the δ-perturbed metric transition system of $S$, $\delta^S$, is a tuple $\delta^S = (X, U, \rightarrow)$ where the (altered) transition relation, $\rightarrow$, is defined as:

$$x_0 \rightarrow^u x' \text{ iff } \exists x'' \in X \text{ s.t. } d(x'', x') \leq \delta \text{ and } x \rightarrow^u x''.$$ (18)

Note that $\delta^S$ has identical states and input labels to $S$, and it also subsumes all of the transitions therein, i.e. $\rightarrow \subseteq \rightarrow$. However, as noted above, the transition relation for $\delta^S$ explicitly contains new nondeterminism relative to the transition relation of $S$; each additional nondeterministic transition is obtained by perturbing the target state of a transition in $S$.

Using this definition, we can finally define an abstract disturbance simulation between two metric transition systems.

**Definition 13 (Abstract Disturbance Simulation).** Let $S = (X_S, U_S, \rightarrow_S)$ and $T = (X_T, U_T, \rightarrow_T)$ be metric transition systems whose state spaces $X_S$ and $X_T$ are subsets of the same metric space $(X_M, d)$. Then $T$ abstract-disturbance simulates $S$ under disturbance $\delta$, written $S \preceq_{\text{AD}} T$ if there is a relation $R \subseteq X_S \times X_T$ such that:

1. for every $(x, y) \in R$, $d(x, y) \leq \delta$;
2. for every $x \in X_S$ there exists a pair $(x, y) \in R$; and
3. for every $(x, y) \in R$ and $x_0 \rightarrow^u x'$ there exists a $y' \rightarrow^v y$ such that $(x', y') \in R$.

**Remark 1.** $\preceq_{\text{AD}}$ corresponds with the usual notion of simulation for metric transition systems. Thus,

$$S \preceq_{\text{AD}} T \iff \delta^S \preceq_{\text{AD}} T.$$ (19)

**C. Main Controller Architecture Problem**

Using ADS for specification satisfaction, we can now state the version of Problem 1 that we will consider and solve as the main result of this paper.

**Problem 2 (Controller Architecture Design).** Let $\delta > 0$, $\tau > 0$ and $K_{\text{cont}} > 0$ be given. Let $\Sigma$ be a feedback controllable Lipschitz control system, and let $S_{\text{spec}} = (X_{\text{spec}}, U_{\text{spec}}, \rightarrow_{\text{spec}})$ be a transition system encoding for a specification on $\Sigma$.

Now, suppose that there exists a $\delta, \tau$ positively invariant Lipschitz-continuous controller $\Psi : \mathbb{R}^n \to U$ with Lipschitz constant $K_{\Psi} \leq K_{\text{cont}}$ such that:

$$S_\tau(\Sigma_{\text{spec}}) \preceq_{\text{AD}} S_\tau(\Sigma_\Psi).$$ (20)

Then the problem is to find a ReLU architecture, $\text{Arch}(\Theta)$, with the property that there exists values for $\Theta$ such that:

$$S_\tau(\Sigma_\Psi) \preceq_{\text{AD}} S_{\text{spec}}.$$ (21)

**Problem 2** is distinct from Problem 1 in two crucial ways. The first of these is the foreshadowed use of ADS for specification satisfaction with respect to $\Psi$. In particular, we now assume the existence of a controller, $\Psi$, that satisfies the specification up to some fixed robustness margin $\delta$, as captured by $\preceq_{\text{AD}}$. This will be the main technical facilitator of our solution, since it enables the design of a NN architecture around the (still unknown) controller $\Psi$.

However, Problem 2 also has a second additional assumption relative to Problem 1: the controller $\Psi$ must also be $\delta, \tau$ positive invariant with respect to the compact subset of states under consideration, $X$ – see Definition 5. This is a technically, but not conceptually, relevant assumption, and it is an artifact of the fact that we are confining ourselves to a compact subset of the state space. It merely ensures that those $\delta$ perturbations created internally to $\preceq_{\text{AD}}$ will lie entirely within $X$. $\delta, \tau$ invariance captures this by asserting that those states within $\delta$ of the boundary of $X$ (i.e. $\text{edge}_\delta(X)$) are “pushed” sufficiently strongly towards the interior of $X$ so that after $\tau$ seconds, they are no longer $\delta$ close to the boundary of $X$ (i.e. in $\text{edge}_\delta(X)$).

Thus, every transition in $S_\tau(\Sigma_{\Psi})$ starting from $\text{edge}_\delta(X)$ has a target in $X \setminus \text{edge}_\delta(X)$, and the $\delta$-perturbed version of $S_\tau(\Sigma_{\Psi})$ has no transitions with targets outside of $X$.

**IV. ReLU Architectures for Controlling Nonlinear Systems**

We are almost able to state main theorem of this paper: that is Theorem 2, which directly solves Problem 2. As a necessary prelude, though, we introduce the following two definitions. The first formalizes the distance between coordinate-wise upper and lower bounds of a compact set (Definition 14). The second formalizes the notion of a (rectangular) grid of points that is sufficiently fine to cover a compact set $X$ by sup-norm balls of a fixed size (Definition 15).

**Definition 14 (Extent of $X$).** The extent of a compact set $X$ is defined as:

$$\text{ext}(X) \triangleq \max_{k=1,\ldots,n} \max_{x \in X} \left| \max_{\pi_k(x)} \pi_k(x) - \min_{\pi_k(x)} \pi_k(x) \right|.$$ (22)

Indeed, the extent of a compact set may also be regarded as the smallest edge length of a hypercube that can contain $X$.

**Definition 15 ($\eta$-grid).** Let $\eta > 0$ be given, and let $X$ be compact and connected with non-empty interior. Then a set $X_\eta \subset X$ is an $\eta$-grid of $X$ if

- the set of $\eta/2$-balls, $X_{\text{part}} \triangleq \{B(x; \eta/2) | x \in X_\eta\}$, has the properties that:
  1. for all $x, x' \in X_\eta$ and $i = 1, \ldots, n$, there is an integer $k_i \in \mathbb{Z}$ such that $\pi_i(x) = \pi_i(x') + k_i \cdot \eta$; and
  2. $X \subseteq \bigcup_{x \in X_\eta} B(x; \eta)$.

Elements of $X_{\eta}$ will be denoted by bold-face font, i.e. $x \in X_{\eta}$. 
Remark 2. It is not the case that a compact and connected set with non-empty interior necessarily has an \( \eta \) grid for any arbitrary choice of \( \eta \); see Fig. 1 for an illustration.

![Fig. 1: Illustration of a set \( X \) and a choice of \( \eta \) for which no \( \eta \)-grid exists: any attempt to cover one "spike" with grid points will result in excluding the other. (Red dashed lines indicate \( \eta/2 \) balls, and sold circles indicate elements of \( X \).)](image)

Now we can state the main theorem of the paper.

**Theorem 2** (ReLU Architecture for Control). Let \( \delta > 0 \), \( \tau > 0 \) and \( K_{\text{cont}} > 0 \) be given, and let \( \Sigma \) and \( S_{\text{spec}} \) be as in the statement of Problem 2. Furthermore, suppose that there exists a \( \delta, \tau \) positively invariant Lipschitz continuous controller \( \Psi : \mathbb{R}^n \to U \) with Lipschitz constant \( K_{\Psi} \leq K_{\text{cont}} \) such that:

\[
S_{\tau}(\Sigma_{\Psi}) \leq_{\text{ADG}} S_{\text{spec}}. \tag{23}
\]

Finally, suppose that \( \mu > 0 \) such that:

\[
K_{\Psi} \cdot \mu \cdot \tau \cdot e^{(K_{\Psi} + 2K_{\text{cont}})\tau} < \delta. \tag{24}
\]

If \( \eta \leq \frac{\mu}{\xi K_{\text{cont}}} \) is such that there exists an \( \eta \)-grid of \( X \), then there exists an \( m \)-fold parallel TLL NN architecture \( \text{Arch}(\Theta_{n,N}^{\text{ReLU}}) \) of size (see Corollary 1):

\[
N \geq n! \left[ \frac{\text{ext}(X)}{\eta} + 2 \right]^n \tag{25}
\]

with the property that there exist values for \( \Theta_{n,N}^{\text{ReLU}} \) such that:

\[
S_{\tau}(\Sigma_{\Psi}, \Theta_{n,N}^{\text{ReLU}}) \leq_{\text{ADG}} S_{\text{spec}}. \tag{26}
\]

Remark 3. Note that the coefficient \( n! \) in (25) is significantly smaller than the analogous one exhibited in [3]; viz. \( \left( n! \cdot \sum_{k=1}^{n} \frac{2^k}{k!} \right) \). Furthermore, [3] also failed to account for the need to choose an \( \eta \)-grid.

The size of the architecture specified by Theorem 2 is effectively determined by the ratio \( \text{ext}(X)/\eta \) (for given state and control dimensions). This quantity has two specific connections to the assumptions in Problem 2. On the one hand, the maximum allowable \( \eta \) is set by \( \mu \), which is determined by the Lipschitz constants of the dynamics and the properties of the assumed controller, \( \Psi \). On the other hand, the specific character of state set itself influences the size the architecture, both explicitly by way of \( \text{ext}(X) \) and implicitly via the requirement that an \( \eta \)-grid exists for \( X \).

With regard to the first of these influences, (24) can be rearranged to provide the following upper bound for \( \eta \):

\[
\eta \leq \frac{1}{3K_{\text{cont}}K_{\Psi}} e^{-(K_{\Psi} + 2K_{\text{cont}})\tau} \cdot \left( \frac{\delta}{\xi} \right). \tag{27}
\]

Of unique interest is the influence of the unknown but asserted-to-exist controller, \( \Psi \). In particular, note that the robustness of \( \Psi \) has an intuitive effect on the size of the architecture. As the robustness of the asserted controller increases (for fixed \( \tau \)), i.e. as \( \delta \) increases, \( \eta \) is permitted to be larger, and the architecture can be correspondingly smaller. Of course as the asserted controller becomes less robust, i.e. as \( \delta \) decreases, the architecture must correspondingly become larger. This trade off makes intuitive sense in light of our eventual proof strategy: in effect, we will design an architecture that can approximate any potential \( \Psi \), so the more robust \( \Psi \) is, the less precisely the NN architecture must be able to approximate it – so a smaller architecture suffices (and vice versa). See Section IV-A.

With regard to the second influence, the extent of the set \( X \) has a straightforward and direct effect on the designed architecture: the “larger” the set is, the larger the architecture is required. However, the “complexity” of the set \( X \) indirectly influences the size of the architecture via the \( \eta \)-grid requirement. Indeed, if the boundary of \( X \) has many thin protuberances (see Fig. 1, for example), then an extremely fine \( \eta \) grid may be required to ensure that grid points are placed throughout \( X \). Crucially, this choice of \( \eta \) may need to be significantly smaller than the maximum allowable \( \eta \) computed via (24), as above – the result will be a significantly larger architecture than suggested by (24) alone. Unfortunately, this effect is difficult to quantify directly, given the variability in complexity of state sets. Nevertheless, given any connected compact set \( X \), there exists an \( \eta > 0 \) for which \( X \) has an \( \eta \) grid.

**Proposition 1.** Let \( X \subset \mathbb{R}^n \) be compact and connected with a non-empty interior. Then there exists an \( \eta > 0 \) such that there exists an \( \eta \) grid of \( X \).

Proof. We will consider dyadic \( \eta \) grids: that is grids based on \( \eta_k = 2^{-k} \), where the associated candidate grid is given by

\[
X_{\eta_k} \triangleq \{ x \in X \mid \exists z \in \mathbb{Z}, x = \frac{z}{2^k} \}. \tag{28}
\]

For convenience, define the following notation:

\[
H_{\eta_k} \triangleq \bigcup_{x \in X_{\eta_k}} B(x; \eta_{k}) \tag{29}
\]

Now because \( X \) is connected with non-empty interior, it is the case that \( \partial d(X) = \text{int}(X) \). Hence, for every \( x \in X \), there exists an \( K_x \) such that for all \( k \geq K_x \), \( x \in H_{\eta_k} \) (consider a truncation of the binary expansion of each coordinate of \( x \)). It follows that if there is a \( K \) such that \( K_x < K \) for all \( x \in X \), then the claim is proved (simply choose \( X_{\eta_k} \)).

Thus, suppose by contradiction that there is a divergent sequence \( \{x_k\} \) s.t. \( K_{x_k} < K_{x_{k+1}} \) for all \( k \in \mathbb{N} \). However, \( X \subset \mathbb{R}^n \) is compact, so \( \{x_k\} \) has a convergent subsequence \( \{x_{k_l}\} \) with limit \( x' \in X \); this subsequence retains the property that \( K_{x_{k_l}} < K_{x_{k_{l+1}}} \). Moreover, by the above claim, there is some \( K_{x'} \) such that \( x' \in H_{\eta_{k_l}} \) for all \( k \geq K_{x'} \).

Now we consider two cases: first that \( x' \in \overline{B(x; \eta_{K_{x'}})} \) for some \( x \), and second that \( x' \) is on the boundary of some \( B(x; \eta_{K_{x'}}) \). In the first case, we note that there exists an \( L \) such that for all \( \ell \geq L \), \( x_{k_{l}} \in B(x; \eta_{K_{x'}}) \), simply by the convergence of the subsequence to \( x' \). This is clearly a contradiction, since it implies that \( x_{k_{l}} \in H_{\eta_{K_{x'}}} \) for all \( \ell \geq L \).
On the other hand, suppose \( x' \) is on the boundary of some \( B(x; \eta_{K_x}) \), and suppose that \( x' \) belongs to a \( d \)-dimensional face of \( B(x; \eta_{K_x}) \) where \( 1 \leq d < n \) (if \( d = 0 \) then \( x' \) is itself a dyadic point, and the above argument applies directly, since \( x' \in X \eta_{K_x} \)). Let this face be denoted \( F_{x'} \). Then there exists a finite \( K_{x'} > K_x \) such that \( x' \in B(x; \eta_{K_x}) \) for some \( x = z/2^k \); use coordinate-wise binary expansions to find a dyadic point within the face \( F_{x'} \) that includes \( x' \) in the associated open \( \eta_{K} \) ball (this is possible since \( F_{x'} \) has at least one non- dyadic coordinate by the \( d \geq 1 \) assumption). This leads to the same contradiction as before, since a tail of the subsequence \( \{ \eta_{K_x} \} \) is eventually contained in this ball.

The remainder of this section is divided as follows. Section IV-A contains a proof sketch of Theorem 2, which divides the proof into two main intermediate steps. Section IV-B and Section IV-C thus contain the formal proofs for these intermediate steps. The overall formal proof of Theorem 2 then appears in Section IV-D.

A. Proof Sketch of Theorem 2

Our proof of Theorem 2 implements the following simple strategy. By assumption, a \( \Psi \) exists that satisfies the specification robustly (via ADS), and hence, we show that any suitably close approximation of \( \Psi \) will function as a controller that satisfies the specification as well (albeit non-robustly). This implies that we need only design a NN architecture with enough parameterization that it can approximate any possible \( \Psi \) that satisfies the conditions of the theorem.

There is, however, an important and non-obvious sequence of observations required to design such a NN architecture. To start, any such \( \Psi \) is Lipschitz continuous, so it is possible to uniformly approximate it by interpolating between its values taken on a uniform grid. Moreover, its Lipschitz constant has a known upper bound, so for a given approximation accuracy, the fineness of that grid can be chosen conservatively, and hence independent of any particular \( \Psi \). However, we show it is possible to interpolate between points over such a uniform grid using a CPWA that has a number of linear regions (Definition 9) proportional to the number of points in the grid – which we reiterate is independent of any particular \( \Psi \). This CPWA is in effect parameterized by the values it takes on those grid points, but in such a way that its number of linear regions is independent of those parameter values. This shows that an arbitrary \( \Psi \) can be approximated by a CPWA with a fixed number of linear regions; it remains to connect this to a NN architecture. Fortunately, the TLL NN architecture [2] can be used directly for this purpose by way of Theorem 1 [2, Theorem 7]; the result in question explicitly specifies a NN architecture that can implement any CPWA with a known, bounded number of linear regions.

Consequently, the proof of Theorem 2 can instead be decomposed into establishing the following two implications:

Step 1) “Approximate controllers satisfy the specification”:
There is an approximation accuracy, \( \mu \), and sampling period, \( \tau \), with the following property: if the unknown controller \( \Psi \) satisfies the specification (under \( \delta \) disturbance and sampling period \( \tau \)), then any controller – NN or otherwise – that approximates \( \Psi \) to accuracy \( \mu \) in \( ||\cdot||_X \) will also satisfy the specification (but under no disturbance). See Lemma 1.

Step 2) “Any controller can be approximated by a CPWA with the same fixed number of linear regions”:
If unknown controller \( \Psi \) has a Lipschitz constant \( K_\Psi \leq K_{\text{cont}} \), then \( \Psi \) can be approximated by a CPWA with a number of regions that depends only on \( K_{\text{cont}} \) and the approximation accuracy. See Corollary 2.

The conclusion of Theorem 2 then follows from Step 1 and Step 2 by means Theorem 1 [2, Theorem 7], since any CPWA with the same number of linear regions (or fewer) can be implemented exactly by a common TLL NN architecture.

Remark 4. Unlike our use of Theorem 1 to directly construct a single TLL, the architectures used in [29] consist of two successive TLL layers. The first is used to represent “basis” functions specified over overlapping regular polytopes (decomposable by a common set of simplexes [29, Figure 3.2]); and the second captures the sum of these basis functions. This approach leads to larger architectures compared to our direct approach: consider [29, Eq. (5.3)], which has exponentially many neurons in the number of linear regions. By contrast, the architecture of Theorem 2 has only polynomially many neurons as a function of the number of regions: i.e. as a function of \( N \), polynomially many min units are needed [2], so the max network has polynomially many neurons; and each min network is straightforwardly polynomial in \( N \).

B. Proof of Theorem 2, Step 1: Approximate Controllers Satisfy the Specification

The goal of this section is to choose constant \( \mu > 0 \) such that: for any \( \Psi \) satisfying the assumptions in Theorem 2, then any other controller \( \Upsilon \) with \( ||\Upsilon - \Psi||_X \leq \mu \) also satisfies the specification, i.e. \( S_\tau(\Sigma_\Upsilon) \preceq_{AD_0} S_{\text{spec}} \).

Specifically, we will prove the following lemma.

Lemma 1. Let \( \Sigma \) be as in Problem 2. Also, let \( \Psi : X \to \Upsilon \) be \( \delta \)-positively invariant w.r.t \( X \), and have Lipschitz constant at most \( K_{\text{cont}} \). Further suppose that \( \mu > 0 \) is such that:

\[
K_u \cdot \mu \cdot \tau \cdot e^{(K_u + K_{\text{cont}})\tau} < \delta.
\]

Then for any \( \Upsilon : X^n \to \mathbb{R}^m \) that is Lipschitz continuous with Lipschitz constant \( K_\Upsilon \), we have that

\[
||\Upsilon - \Psi||_X \leq \mu \implies S_\tau(\Sigma_\Upsilon) \preceq_{AD_0} S_{\tau}(\Sigma_{\Psi}).
\]

And hence if in addition \( S_\tau(\Sigma_{\Psi}) \preceq_{AD_0} S_{\text{spec}} \), then:

\[
S_\tau(\Sigma_\Upsilon) \preceq_{AD_0} S_{\text{spec}}.
\]

Our specific approach to prove this will be as follows. First, we choose \( \mu \) small enough such that a \( \tau \)-second flow of \( \zeta_{\Sigma_\Upsilon} \) doesn’t deviate by more than \( \delta \) from \( \zeta_{\Psi} \) over its duration, \( \tau \). This is accomplished by means of a Grönwall-type bound that uses \( ||\Upsilon - \Psi||_X \leq \mu \). That is assume \( \Psi \) is \( \delta \)-positively invariant, and use \( ||\Upsilon - \Psi||_X \leq \mu \) in the Grönwall inequality:

\[
\forall x \in X, ||\zeta_{\Sigma_\Upsilon}(\tau) - \zeta_{\Psi}(\tau)|| \leq K_u \cdot \mu \cdot \tau \cdot e^{(K_u + K_{\text{cont}})\tau} < \delta.
\]

Then we use this conclusion to construct the appropriate simulation relations that show:

\[
S_\tau(\Sigma_\Upsilon) \preceq_{AD_0} S_\tau(\Sigma_{\Psi}) \preceq_{AD_0} S_{\text{spec}}.
\]

First, we formally obtain the desired conclusion of (33) by way of the following proposition.

Proposition 2. Let \( \Sigma \) and \( \Psi \) be as in the statement of Lemma 1, and let \( \Upsilon : \mathbb{R}^m \to \mathbb{R}^m \) be Lipschitz continuous
with Lipschitz constant $K_T$. Also, suppose that $\mu > 0$ is such that:
\[ K_u \cdot \mu \cdot \tau \cdot e^{(K_x + K_u K_T)\tau} < \delta. \] (35)

If $||Y - \Psi||_X \leq \mu$, then for any $x \in X$:
\[ ||\zeta_{X,T}(\tau) - \zeta_x\Psi(\tau)|| \leq \delta. \] (36)

Proof. By definition, we have:
\[
\begin{align*}
  &\left|\int_0^t \left( \frac{f(\zeta_{X,T}(\sigma), \Psi(\zeta_{X,T}(\sigma))) - f(\zeta_x, \Psi(\zeta_x)))}{d\sigma} + K_u \cdot \|Y - \Psi\|_X \right) d\sigma \right| \\
  &\leq \int_0^t \|f(\zeta_{X,T}(\sigma), \Psi(\zeta_{X,T}(\sigma))) - f(\zeta_x, \Psi(\zeta_x)))\| d\sigma \\
  &\leq K_T \cdot \|\zeta_X - \zeta_x\| + K_u \cdot \|Y - \Psi\|_X.
\end{align*}
\] (37)

The first term is so bounded because $Y$ is assumed to be globally Lipschitz continuous on all of $\mathbb{R}^n$. In particular, the first term of (38) can be bounded using the global Lipschitz continuity of $\Psi$, which is $\zeta_x \Psi(t)$ in $X$ for all $t$ by the assumption of $(\delta, \tau)$ forward invariance of $X$: consequently, the $X$-restricted sup norm may be employed.

Thus, using (37) and (38), we obtain the bound
\[
\|\zeta_{X,T}(\tau) - \zeta_x\Psi(\tau)\| \leq \int_0^t \left( K_x + K_u K_T \right) \cdot \|\zeta_{X,T}(\sigma) - \zeta_x\Psi(\sigma)\| \\
+ K_u \cdot \|Y - \Psi\|_X d\sigma.
\] (39)

The mechanism for obtaining such a CPWA interpolation will ultimately be to “tile” the set $X \subseteq \mathbb{R}^n$ by $n$-simplexes$^3$, each of whose vertices are points from the interpolation grid. That is, we will choose these simplexes such that a full-dimensional face of any simplex will be shared exactly by one and only one other simplex. Each simplex can then be regarded as corresponding to a linear region, because the graph of (any) $\Psi$ on its $n + 1$ vertices defines an affine function over that simplex. Moreover, an affine function on one simplex is continuous with the affine function defined on any of its neighbors, because of the face-sharing property described above. Thus, this approach constructs an interpolation CPWA where each affine function is “activated” only on its respective simplex. Crucially, this construction leads to a CPWA with same number of linear regions, irrespective of the particular values of $\Psi$: the number of simplexes is determined entirely by the spacing of the uniform grid, which is in turn determined only by Lipschitz constants and the set $X$ itself. Hence, the grid can be specified by the problem parameters, and the values of $\Psi$ thereon can be considered as parameters. This is exactly a parameterized CPWA of the sort described above; the culmination of this approach is in Corollary 2.

1) Preliminaries: Before we state the main result of this section, we introduce the following notation to refer to those hypercubes that have at least one corner in $X_\eta$, but no elements of $X_\eta$ in their interior.

Definition 16 (Interpolation Hypercube). Let $X_\eta$ be an $\eta$-grid of $X$, and let $x \in X_\eta$. Then each vector $\rho \in \{-1, 1\}^n$ defines an interpolation hypercube relative to $x$, given by:
\[
C_{x, \rho} \triangleq B(x + \frac{\rho}{2} \cdot \sum_{i=1}^n \pi_i(\rho) \cdot e_i)
\] (41)

$^3$That is simplexes with $n + 1$ vertices.
and whose corners are
\[ \mathcal{F}_0(C_{\omega \rho}) = \{ x + \eta \cdot \sum_{i \in \mathbb{Z}} \pi_i(\rho) \cdot e_i | x \subseteq \{ 1, \ldots, n \} \}, \tag{42} \]
where \( e_i \) is the \( i \)th column of the \((n \times n)\) identity matrix.

Note that \( \bigcup_{\eta \in \{-1,1\}^n} C_{\omega \rho} = B(x; \eta) \). Fig. 3 is a cartoon that depicts interpolation hypercubes, among other things.

Interpolation hypercubes can have corners that lie outside of \( X \), and we will subsequently need to treat these corners separately. Hence the following notation for an extra corner.

**Definition 17** (Extra Corner). Let \( C_{\omega \rho} \) be an interpolation hypercube of an \( \eta \)-grid, \( X_\eta \). Then an **extra corner** of \( C_{\omega \rho} \) is any corner \( x \in \mathcal{F}_0(C_{\omega \rho}) \setminus X_\eta \). Denote the set of all extra corners of all interpolation hypercubes of \( X_\eta \) as
\[ \Delta_{X_\eta} = \bigcup_{x \in X_\eta} \mathcal{F}_0(C_{\omega \rho}) \setminus X_\eta. \tag{43} \]

Finally, we have the following proposition, which indicates the maximum number of interpolation hypercubes that are required to cover \( X \) (by tiling), given an \( \eta \)-grid, \( X_\eta \).

**Proposition 3.** Let \( X_\eta \) be an \( \eta \) grid for \( X \) as before. Then \( X \) is covered by the union of at most
\[ \left\lfloor \frac{\text{ext}(X)}{\eta} + 2 \right\rfloor^n \tag{44} \]
interpolation hypercubes.

**Proof.** A compact set \( X \) can be contained in a hypercube with edge-length \( \text{ext}(X) \), which can in turn be partitioned by at most \( \left\lfloor \frac{\text{ext}(X)}{\eta} \right\rfloor^n \) elements. Thus, \( |X_\eta| \leq \left\lfloor \frac{\text{ext}(X)}{\eta} \right\rfloor^n \) elements. It follows then that \( X \) can be covered by the inclusion of one interpolation hypercube per element of \( X_\eta \), plus one extra set of interpolation hypercubes at either extreme of each coordinate. \( \square \)

**Remark 5.** Note that an \( \eta \)-grid of \( X \) effectively specifies a tiling of no more than \( \left\lfloor \frac{\text{ext}(X)}{\eta} \right\rfloor^n + 2 \) hypercubes covering \( X \): i.e. the interpolation hypercubes cover \( X \), and a full dimensional face of any one of them is shared by one and only one interpolation hypercube.

2) **Parameterized CPWA for Approximation:** Now we can state the main results of this section. The first, **Lemma 2**, specifies the structure and properties of the parameterized CPWA that we will use. Thus, as a corollary, we have **Corollary 2**, which particularizes this parameterized CPWA to Step 2 in the proof of **Theorem 2**.

**Lemma 2.** Let \( \eta > 0 \) be fixed, and let \( X \) be compact. Also let \( X_\eta \) be an \( \eta \)-grid of \( X \) that is enumerated as \( X_\eta = \{ x_i \}_{i=1}^{\left\lfloor \frac{\text{ext}(X)}{\eta} \right\rfloor^n + 1} \).

Then there is a parameterized CPWA \( \Upsilon_\omega : X \to \mathbb{R} 
\]
with parameter vector \( \omega \in \mathbb{R}^{|X_\eta|} \) that has the following properties: i) for all \( i = 0, \ldots, |X_\eta| - 1 \) and \( j = 0, \ldots, n - 1 \),
\[ \pi_i(\omega) = \Upsilon_\omega(x_i); \tag{45} \]
i.e. \( \omega \) specifies the value of \( \Upsilon_\omega \) on the grid \( X_\eta \); ii) \( \Upsilon_\omega \) has at most
\[ n! \cdot \left\lfloor \frac{\text{ext}(X)}{\eta} + 2 \right\rfloor^n \tag{46} \]
linear regions no matter the value of \( \omega \); and
\[ \text{iii)} \ \text{let } \nu_{\omega, \rho} \triangleq \bigcup_{x \in \mathcal{F}_0(C_{\omega, \rho})} B(x; \eta); \text{ then for each interpolation hypercube } C_{\omega, \rho} \text{ of } X_\eta \text{ and each } x \in C_{\omega, \rho}, \]
\[ \min_{x' \in \mathcal{F}_0(C_{\omega, \rho}) \cap X_\eta} \Upsilon_\omega(x') \leq \Upsilon_\omega(x) \leq \max_{x' \in \mathcal{F}_0(C_{\omega, \rho}) \cap X_\eta} \Upsilon_\omega(x'). \tag{47} \]

We defer the proof of **Lemma 2** to Section IV-C3 in order to immediately state the conclusion of relevance for Step 2 in the proof of **Theorem 2**.

**Corollary 2.** Let \( K_{\text{cont}} > 0 \) be given, and let \( \mu > 0 \) and \( \eta \leq \frac{\mu}{K_{\text{cont}}} \) be as above. Also let \( \Psi : X \to \mathbb{R} \) be any Lipschitz continuous function with Lipschitz constant at most \( K_{\text{cont}} \).

If we use the parameterized CPWA from **Lemma 2** component-wise, with the \( j \)-th parameter vector specified by
\[ \omega^{(j)} = \omega^{(j)} \triangleq \prod_{i=0}^{\left\lfloor \frac{\text{ext}(x)}{\eta} \right\rfloor^n n} \{ \pi_j(\Psi(x_i)) \}, \tag{48} \]
then \( \Upsilon_{\omega^{(j)}} \triangleq \left( \Upsilon_{\omega^{(1)}} \cdots \Upsilon_{\omega^{(n)}} \right) \) agrees with \( \Psi \) on \( X_\eta \) and
\[ \| \Upsilon_{\omega^{(j)}} - \Psi \| \leq \mu. \tag{49} \]

Moreover, \( \Upsilon_{\omega^{(j)}} \) has at most \( n! \cdot \left\lfloor \frac{\text{ext}(X)}{\eta} + 2 \right\rfloor^n \) linear regions for each component-wise function, independent of the particular choice of \( \Psi \). Finally, \( \Upsilon_{\omega^{(j)}} \) has a Lipschitz constant of at most \( 3 \cdot K_{\text{cont}} \) on \( \bigcup_{x \in \mathcal{F}_0(C_{\omega, \rho}) \cap \mathcal{D}} C_{\omega, \rho} \subseteq X \).

**Proof.** This follows from **Lemma 2**, with the exception of (49), and the claim about the Lipschitz constant of \( \Upsilon_{\omega^{(j)}} \).

The claim (49) follows from (47) in **Lemma 2**, given that \( \eta \) was assumed chosen as \( \eta \leq \frac{\mu}{K_{\text{cont}}} \). In particular, we have that for any interpolation hypercube \( C_{\omega, \rho} \) and any \( x \in C_{\omega, \rho} \), there exists a \( j \in \{ 1, \ldots, n \} \) s.t.
\[ \| \Upsilon_{\omega^{(j)}}(x) - \Psi(x) \| = | \pi_j(\Psi(x)) - \Upsilon_{\omega^{(j)}}(x) | \leq \pi_j(\Psi(x)) - \Upsilon_{\omega^{(j)}}(x), \tag{50} \]
from here, we consider as separate cases both possible signs of \( \pi_j(\Psi(x)) - \Upsilon_{\omega^{(j)}}(x) \). However, for brevity, we write out only the case \( \pi_j(\Psi(x)) - \Upsilon_{\omega^{(j)}}(x) \geq 0 \), since the other case can be treated by similar arguments. In this case:
\[ \| \Upsilon_{\omega^{(j)}}(x) - \Psi(x) \| = | \pi_j(\Psi(x)) - \Upsilon_{\omega^{(j)}}(x) | \leq \pi_j(\Psi(x)) - \Upsilon_{\omega^{(j)}}(x), \tag{51} \]
where the (51) follows from **Lemma 2**, equation (47). Then we note that there exists an \( x \in \nu_{\omega, \rho} \cap X_\eta \) such that
\[ \min_{x' \in \mathcal{F}_0(C_{\omega, \rho}) \cap X_\eta} \Upsilon_{\omega^{(j)}}(x') = \Upsilon_{\omega^{(j)}}(x) = \pi_j(\Psi(x)) \tag{52} \]
by i) of **Lemma 2**. But by construction, \( x \in \nu_{\omega, \rho} \), so there exists an \( x'' \in \mathcal{F}_0(C_{\omega, \rho}) \) such that \( x \in B(x''; \eta) \). It follows that \( \| x - x'' \| \leq \| x - x'' \| + \| x'' - x \| \leq 2 \mu \). Thus, by the choice of \( \eta \leq \frac{\mu}{K_{\text{cont}}} \), \( K_{\text{cont}} \)-Lipschitz continuity of \( \Psi \) implies:
\[ \| \Upsilon_{\omega^{(j)}}(x) - \Psi(x) \| \leq \pi_j(\Psi(x)) - \Upsilon_{\omega^{(j)}}(x) \leq \| \Psi(x) - \Upsilon_{\omega^{(j)}}(x) \| \leq \frac{3}{4} \mu \leq \mu, \tag{53} \]
This concludes the proof for the case \( \pi_j(\Psi(x)) - \pi_j(\Upsilon_{\omega^{(j)}}(x)) \geq 0 \); the other case follows similarly.

Thus, it remains to show the claim about the Lipschitz constant of \( \Upsilon_{\omega^{(j)}} \). This ultimately follows from the observation that every affine function in \( \Upsilon_{\omega^{(j)}} \) interpolates between the values specified on the corners of some interpolation hypercube. Consequently, it suffices to show that the Lipschitz constant
of none of these affine functions exceeds the claimed value. Thus, let \( x, x' \in C_{xy} \) be arbitrary such that \( x \) and \( x' \) belong to the same simplex of the dissection \( D_{\alpha} \) (see Lemma 4). Now by linear interpolation between the values of \( \Psi_{xy} \) on that simplex, there exists for each coordinate \( i = 1, \ldots, m \) a pair of corners \( x_{\min}^i, x_{\max}^i \in F_0(C_{xy}) \) such that

\[
\pi_i(\Psi_{xy}(x_{\min}^i)) \leq \pi_i(\Psi_{xy}(x)) \leq \pi_i(\Psi_{xy}(x_{\max}^i)),
\]

and which holds for any \( x \) in the same simplex. Now define:

\[
\gamma_{\Delta} = \max_i [\pi_i(\Psi_{xy}(x_{\min}^i)) - \pi_i(\Psi_{xy}(x))] ;
\]

and

\[
i_{\Delta} = \arg \max_i [\pi_i(\Psi_{xy}(x_{\max}^i)) - \pi_i(\Psi_{xy}(x_{\min}^i))].
\]

Assuming without loss of generality that \( \gamma_{\Delta} \neq 0 \), then by linearity of \( \Psi_{xy} \) on the simplex containing \( x \) and \( x' \):

\[
\|\Psi_{xy}(x) - \Psi_{xy}(x')\| \leq \frac{\gamma_{\Delta}}{\eta} \|x - x'\|,
\]

assuming \( x_{\min}^i \) and \( x_{\max}^i \) are achieved on the closest corners of \( C_{xy} \), and \( x \rightarrow x' \) defines a line parallel to the \( i_{\Delta} \)th coordinate. On the other hand, we note that

\[
y_{\Delta} = \pi_i(\Psi_{xy}(x_{\min}^i)) - \pi_i(\Psi_{xy}(x_{\min}^i)) = \pi_i(\Psi_{xy}(x_{\min}^i)) - \pi_i(\Psi_{xy}(x_{\max}^i)) \leq \|\Psi_{xy}(x_{\min}^i) - \Psi_{xy}(x_{\max}^i)\|.
\]

Note now that the value of \( \Psi_{xy} \) on \( x_{\min}^i \) and \( x_{\max}^i \) is specified as the value of \( \Psi \) on some \( \eta \)-grid points no further than \( \eta \) away from each of those corners; recall the use of \( \nu, \rho, \vartheta \) in the statement of Lemma 2, iii). In particular, this implies that

\[
\exists x_{\min} \in X_{\eta} \cdot \|x_{\min}^i - x_{\min}\| \leq \eta \land \Psi_{xy}(x_{\min}^i) = \Psi(x_{\min})
\]

and likewise for \( x_{\max} \). Thus, we have the following:

\[
\|\Psi_{xy}(x_{\min}^i) - \Psi_{xy}(x_{\max}^i)\| = \|\Psi(x_{\max}) - \Psi(x_{\min})\| \leq K_{\rho} \cdot \|x_{\max} - x_{\min}\| \leq K_{\rho} \cdot 3 \cdot \eta
\]

which follow by Lipschitz continuity of \( \Psi \). Substituting (57) into (59) gives the bound on the Lipschitz constant of \( \Psi_{xy} \).

3) Proof of Lemma 2: Our approach to proving Lemma 2 will be as follows. On the domain \( X_{\eta} \), we will first define a function \( \Gamma_{\omega} : X_{\eta} \rightarrow \mathbb{R} \) as \( \Gamma_{\omega} : x_i \mapsto \pi_i(\omega) \). Then we will extend \( \Gamma_{\omega} \) as a CPWA to the union of all the interpolation hypercubes of \( X_{\eta} \), i.e. \( \bigcup_{x \in X_{\eta}, \rho \in (-1,1)^n} C_{xy} \). By this means, we obtain a parameterized CPWA that satisfies Lemma 2.

To accomplish this, we will first need to consider the problem of extending \( \Gamma_{\omega} \) from the corners of a specific interpolation hypercube to the rest of that interpolation hypercube itself. Importantly, however, this interpolation must be systematized in such a way that adjacent interpolation hypercubes have consistent interpolations on the non-trivial faces they have in common. Thus, we have the following two generic lemmas, which are the last facilitators we will need in order to prove Lemma 2. Lemma 3 describes the general methodology we undertake for doing this: namely, dividing the hypercube into simplices, each of which represents a linear region in the CPWA. Lemma 4 then considers a particular instance of this construction that has several unique properties; in particular, it is a consequence of Lemma 4 that will permit a globally consistent interpolation despite considering each interpolation hypercube individually.

**Lemma 3.** Let \( C_n = [0,1]^n \), and suppose that:

\[
\alpha : F_0(C_n) \rightarrow \mathbb{R}
\]

is a function defined on the corners of \( C_n \). Also suppose that \( D = \{s_i\}_{i=1}^{N} \) is a set of \( n \)-simplices such that: their vertices are contained in \( F_0(C_n) \); they together partition \( C_n \); and any non-trivial face of one simplex is shared exactly by one and only one other simplex. \( D \) is thus a special case of a dissection of \( C_n \).

Then \( D \) induces a uniquely defined CPWA function \( \Xi^n_D : C_n \rightarrow \mathbb{R} \) such that:

1) \( \forall x \in F_0(C_n), \Xi^n_D(x) = \alpha(x), \) i.e. \( \Xi^n_D \) extends \( \alpha \) to \( C \); and
2) \( \forall x \in C_n \)

\[
\min_{x \in F_0(C_n)} \alpha(x) \leq \Xi^n_D(x) \leq \max_{x \in F_0(C_n)} \alpha(x).
\]

**Proof.** This follows because each such simplex specifies \( n+1 \) points from graph(\( \alpha \)), so there is at least one hyperplane that interpolates between those points in \( \mathbb{R}^{n+1} \).

In particular, let \( V_{s_i} \) be the vertices of the simplex \( s_i \in D \). By assumption, \( V_{s_i} \subseteq F_0(C_n) \), so \( \alpha \) is defined on \( V_{s_i} \) for each \( s_i \). Thus, let \( G_{s_i} = \{(a_i, b_i) \mid a_i \in V_{s_i} \} \subset \text{graph}(\alpha) \) with \( G_{s_i} \) enumerated as \( G_{s_i} = \{a_i\}_{i=1}^{n+1} \). Then \( s_i \) defines at least one hyperplane in \( \mathbb{R}^{n+1} \) specified by

\[
h_i : x \in \mathbb{R}^{n+1} \mapsto \left( w_i^T - b_i \right).
\]

Now that the \( \gamma_{\Delta} \) is achieved on the closest corners of \( C_{xy} \), and \( x \rightarrow x' \) defines a line parallel to the \( i_{\Delta} \)th coordinate.

\[

\Xi^n_D(x) = \min_{x \in F_0(C_n)} \alpha(x) \leq \Xi^n_D(x) \leq \max_{x \in F_0(C_n)} \alpha(x).
\]

Thus, to obtain interpolate between such an \( \alpha \) it suffices to construct a dissection of \( C_n \) with the properties assumed in Lemma 3. However, to prove Lemma 2, we use a particular
dissection in order to get a suitable CPWA extension; specifically, we need to consider how these CPWA extensions interact on adjacent interpolation hypercubes, whence ii) below.

**Lemma 4.** Let \( C = [0,1]^n \), and suppose that:

\[
\alpha : \mathcal{F}_0(C_n) \to \mathbb{R}
\]

is a function defined on the corners of \( C_n \).

Also, define \( n \cdot (n - 1)/2 \) hyperplanes from the equations

\[
b_{ij}(x) = 0, \quad i = 1, \ldots, n; \quad j < i,
\]

where the \( b_{ij} \) are given by:

\[
b_{ij} : \mathbb{R}^n \to \mathbb{R}
\]

\[
b_{ij}(x) = \pi_i(x) - \pi_j(x) \quad \text{for} \quad j < i
\]

so that \( B_n \triangleq \{ b_{ij} \}_{i \in \{1, \ldots, n\}; j < i} \) is the so-called braid arrangement of hyperplanes \([36]\).

Furthermore, let

\[
D_{B_n} \triangleq \{ R \cap \text{int}(C_n) | R \text{ is a region in } B \}
\]

be the dissection of \( C_n \) by \( B_n \).

Then \( D_{B_n} \) is a dissection of \( C_n \) by \( n \)-simplexes, each of which has vertices only in \( \mathcal{F}_0(C_n) \). Thus, Lemma 3 applies, and the resultant interpolation, \( \Xi_{B_n} \), has further properties:

i) \( \Xi_{B_n}^\mathcal{F} \) has at most \( n! \) linear regions in \( C_n \);

ii) for any \( n - 1 \)-dimensional face \( F_{k}^i \in \mathcal{F}_{n-1}(C_n) \) (as in Definition 11) and any other function \( \alpha' : \mathcal{F}_0(C_n) \to \mathbb{R} \), we have that

\[
\forall x \in (F_{k}^i \cap \mathcal{F}_0(C_n)). \quad \alpha(x) = \alpha'(|x - e_i|) \implies \forall x \in F_{k}^i. \quad \Xi_{B_n}^{\mathcal{F},i}(x) = \Xi_{B_n}^{\mathcal{F},i}(|x - e_i|)
\]

i.e. if \( \alpha \) and \( \alpha' \) agree on the corners of opposing \( n - 1 \)-dimensional faces \( F_{k}^i \) and \( F_{k-1}^{i+1} \), then the interpolations \( \Xi_{B_n}^{\mathcal{F},i} \) and \( \Xi_{B_n}^{\mathcal{F},i+1} \) agree on the entirety of those faces (that is agree under translation by \( e_i \)).

We defer the proof of Lemma 4 until Appendix A. However, for context, the dissection obtained from the braid arrangement is shown below in Fig. 2 for \( n = 3 \).

Fig. 2: Dissection of \( C_3 \) by the braid hyperplane arrangement, \( B_3 \), to yield \( D_{B_3} \); an exploded view depicts the individual simplexes more clearly. See Lemma 4.

Fig. 2 also illustrates a property of \( D_{B_n} \) that is crucial to establish ii) in Lemma 4: note that each face is split into two triangles, and each of those triangles is a single-coordinate translation of a triangle on the opposite face. This fact is considered in the proof of Lemma 4 in Appendix A.

**Remark 6.** Property ii) of Lemma 4 can also be interpreted as a consequence of the following observation: the braid dissection of a hypercube results in “symmetric” dissections on opposite \( n - 1 \)-dimensional faces (c.f. Fig. 2). Thus, dissecting each individual interpolation hypercube this way yields a collection of simplexes that “tile” their union; i.e. a full-dimensional face of any simplex is shared exactly by one and only one other simplex – hence the desired continuity.

**Remark 7.** The braid arrangement dissection is a particular dissection by Schäfli orthoscheme. \([37]\)

Now we can finally state the proof of Lemma 2.

**Proof. (Lemma 2)** The proof will proceed as suggested above. First, we will define the function \( \Upsilon_\omega \) on the elements of \( X_\eta \) using the parameter vector \( \omega \). Then we will extend \( \Upsilon_\omega \) to the corners of all the interpolation hypercubes associated with \( X_\eta \); i.e. we will define \( \Upsilon_\omega \) on the extra corners of the aforementioned interpolation hypercubes, again in terms of \( \omega \) only (see Definition 17). Then we will employ Lemma 3 and Lemma 4 to define a CPWA within each interpolation hypercube that interpolates between the values on its corners – those values being derived exclusively from \( \omega \) by the construction above. Finally, Lemma 4 can be used to show that this interpolation scheme is consistent from one interpolation hypercube to the next, thus making it a viable CPWA defined over the entirety of \( X \). Moreover, it will also lead to the claimed bound on the number of linear regions needed.

To this end, first define \( \Upsilon_\omega \) on \( X_\eta \) as follows. For each \( x_i \in X_\eta = \{ x_i \}_{i=0}^{\n-1} \), define: \( \Upsilon_\omega(x_i) \triangleq \pi_i(\omega) \). Now, we have to define \( \Upsilon_\omega \) to the extra corners associated with the interpolation hypercubes of \( X_\eta \), i.e. \( \Delta X_\eta \) (see Definition 17). In particular, we extend \( \Upsilon_\omega \) to \( \Delta X_\eta \) using the following definition:

\[
\forall x \in \Delta X_\eta. \quad \Upsilon_\omega(x) \triangleq \min_{x \in B(x) \cap X_\eta} \Upsilon_\omega(x).
\]

Note that (73) is well defined, since \( x \in \Delta X_\eta \) is the corner of an interpolation hypercube – which has edge length \( \eta \) – and each interpolation hypercube has at least one corner that belongs to \( X_\eta \); see Fig. 3. Moreover, we have just defined \( \Upsilon_\omega \) for any such \( x \in X_\eta \) in terms of \( \omega \) only, so this scheme for defining \( \Upsilon_\omega \) on \( \Delta X_\eta \) likewise depends only on \( \omega \).

The previous construction ensures that \( \Upsilon_\omega \) is defined on the corners of each interpolation hypercube associated with \( X_\eta \), and thus Lemma 3 and Lemma 4 may be applied to obtain a number of CPWAs, each of whose domains is a single interpolation hypercube. Crucially, however, if these CPWAs are continuous from one interpolation hypercube to any adjacent one, then they may together be considered to define a
CPWA whose domain is the union of all such interpolation hypercubes (simply switching between them as appropriate). The result is a CPWA that contains $X$ in its domain. Then, since each individual CPWA requires at most $n!$ linear regions (by Lemma 4), this construction leads to the bound in claim $ii$) of the Lemma simply by using Proposition 3.

Thus, it remains to show that the CPWAs defined on each interpolation hypercube are consistent (continuous) from one to the next. However, this follows more or less directly from property $ii$) of Lemma 4. To see this, let $C_{x'p}$ be an interpolation hypercube, and let $C_{x'p'}$ be any other interpolation hypercube that shares an $n-1$-dimensional face, $F_{x'p}$, with $C_{x'p}$. In particular, both $\Psi \omega |_{C_{x'p}}$ and $\Psi \omega |_{C_{x'p'}}$ can be regarded as inducing separate functions on the unit hypercube $C_n$ (by translation), and because $\Psi \omega$ is defined on the corners of all such interpolation hypercubes, these functions satisfy the condition of property $ii$) of Lemma 4. Thus, their interpolations are identical on opposite faces, which yields the claimed continuity under translation back to domains $C_{x'p}$ and $C_{x'p'}$. □

D. Proof of Theorem 2

Using the consequences of Step 1 (see Section IV-B) and Step 2 (see Section IV-C), we can now prove Theorem 2.

Proof. By Corollary 2, there is a $\Psi \omega \omega$ that:

- approximates any controller $\Psi$ to the desired accuracy via $\| \cdot \|_X$ merely by changing its parameter values;
- requires no more than the claimed number of linear regions on $X$, given this accuracy, independent of $\Psi$; and
- has a Lipschitz constant at most $3 \cdot K_{cont}$ on $X$, also independent of $\Psi$.

By Corollary 1 (a consequence of [2, Theorem 2]) we can find a TLL NN architecture to implement the CPWA $\Psi \omega \omega$. Moreover, if such a TLL is implemented using only linear functions whose Lipschitz constants are bounded by $3 \cdot K_{cont}$, as is the case to implement $\Psi \omega \omega$ exactly on $X$, then that TLL will likewise have a global Lipschitz constant of at most $3 \cdot K_{cont}$. Thus, Lemma 1 applies, and that completes the proof. □

V. EXTENSION: RELU ARCHITECTURES FOR SYSTEM IDENTIFICATION

The methodology of the previous section can be summarized in the following way: determine the amount of (instantaneous) controller error that can be tolerated relative to a robust controller (Step 1), and design an architecture that can achieve that error for an arbitrary robust controller (Step 2). In particular, though, the methodology of Step 1 – chiefly the Grönwall inequality – only recognizes instantaneous controller error as an error in the (controlled) vector field. As a result, essentially the same calculations can be recycled when the source of error comes not from a different controller, but an alternate, erroneous vector field under the same controller.

Indeed, this is almost a complete program to repurpose the results of Section IV to obtain assured architectures for system identification. In this context, we treat the instantaneous error as coming from the fact that the identified vector field – a NN in this case – can’t exactly replicate the vector field on which it is trained. Nevertheless, from Step 2, we can find a sufficiently large NN system architecture that can be trained to a proscribed uniform approximation accuracy (relative to the unknown system). This architecture leads to closed-loop assurances under system identification, since a Grönwall calculation – like Proposition 2 of Step 1 – connects training error to a controller robustness margin (and conversely).

In fact, this proposed architecture will lead to something like the following assurance: assuming the architecture has been trained adequately, then an appropriately robust controller designed for the identified system will also control the original, unknown system. In this way, the proposed system identification architecture comes with both a (uniform) training accuracy requirement and a robustness requirement for controllers designed to meet a specification. If these requirements can be met by the identified architecture and a controller designed for said identified architecture, then the very same controller is assured to control the original, unknown system.

This section begins with Section V-A, in which we formulate an assured NN architecture design problem for system identification. Then, in Section V-B, we present Theorem 3, which provides just such an assured NN architecture.

A. Problem Formulation: NN Architectures for System Identification

As suggested, we will formulate the problem of designing assured architectures for system identification in roughly the reverse way as the controller architecture problem, Problem 2. That is we start from a given accuracy to which the identified dynamics can be learned relative to the original, unknown dynamics; this accuracy will be denoted by the suggestive symbol $\mu > 0$. Then the problem will be to find an identified architecture and robustness requirements $\delta$ and $\tau$ such that: if the identified architecture is learned to a uniform error of $\mu$, then any $\delta$-robust controller designed on the identified system will satisfy the same specification on the original system (non-robustly). Hence, we have the following problem of designing assured ReLU architectures for system identification.

Problem 3 (ReLU Architectures for System Identification). Let $\mu > 0$ and $K_{cont} > 0$ be given. Let $\Sigma$ be a feedback controllable Lipschitz control system, and let $S_{\text{spec}} = (X_{\text{spec}}, U_{\text{spec}}, s_{\text{spec}})$ be a transition system specification on $\Sigma$.

Then the problem is to find $\tau = \tau(K_x, K_u, K_{cont}, \mu)$ and $\delta = \delta(K_x, K_u, K_{cont}, \mu)$ together with a ReLU architecture, $\text{Arch}(\Theta)$, which corresponds to an identified system

$$\Sigma^\Theta = (X, U, U, \mathcal{N}_\Theta),$$

where $X, U$ and $U$ are as in $\Sigma$, and $\mathcal{N}_\Theta : X \times U \rightarrow X$. The identified system, $\Sigma^\Theta$, must further have the properties that:

- there exists values of $\Theta$ s.t.

$$\| f - \mathcal{N}_\Theta \| \leq \mu \quad (75)$$

i.e. the architecture is able to learn the vector field $f$ to the proscribed accuracy; and

- for $\Theta$ as in (75), if $\Psi : X \rightarrow U$ is a $\delta$-$\tau$ invariant Lipschitz-continuous controller with $K_{\Psi} \leq K_{cont}$ s.t.

$$S_\tau(\Sigma^\Theta) \preceq_{\mathcal{A}D_3} S_{\text{spec}},$$

then

$$S_\tau(\Sigma^\Theta) \preceq_{\mathcal{A}D_0} S_{\text{spec}} \quad (77)$$

i.e. if $\Psi$ controls the identified system to meet the specification robustly, then $\Psi$ also controls the original system, $S(\Sigma^\Theta)$, to the same specification (albeit non-robustly).
B. ReLU Architectures for Nonlinear-System Identification

Now we have the following theorem, which solves Problem 3 with the claimed guarantees for system identification.

Theorem 3. Let $\mu > 0$ and $K_{\text{cont}} > 0$ be given, and let $\Sigma$ and $S_{\text{spec}}$ be as in Problem 3. Furthermore, let $\delta, \tau > 0$ be s.t.

$$K_{\text{a}} \cdot \mu \cdot \tau \cdot e^{(K_{\text{u}} + K_{\text{u}}) \tau} < \delta.$$  \hfill (78)

If $\eta \leq \frac{\mu}{3K_{\text{cont}}}$ is such that there exists an $\eta$-grid of $X \times U$, then an $n$-fold TLL NN architecture $\text{Arch}(\Theta_{n \times n \times N})$ of size:

$$N \geq (n + m)! \left[ \frac{\text{ext}(X \times U)}{\eta} + 2 \right]^{n+m}$$  \hfill (79)

defines an identified system as in (74) of Problem 3, which has the following properties:

i) there exist values for $\Theta_{n \times n \times N}$ such that

$$\|f - \mathcal{M}\Theta_{n \times n \times N}\|_{X \times U} < \mu; \text{ and }$$  \hfill (80)

ii) for this $\Theta_{n \times n \times N}$ and $\Psi : X \rightarrow U$ $\delta$-$\tau$ positive invariant w.r.t to $X$ and Lipschitz continuous with $K_{\Psi} \leq K_{\text{cont}}$.

$$S_{\tau}(\Sigma_{\Psi}^{n \times n \times N}) \preceq_{\text{AD}} S_{\text{spec}} \implies S_{\tau}(\Sigma_{\Psi}) \preceq_{\text{AD}} S_{\text{spec}}.$$  \hfill (81)

For convenience, denote a trajectory of $\Sigma_{\Psi}^{n \times n \times N}$ by $\zeta_{x_{x_{x_{x}}}}$, and the corresponding vector field by $f^{-\mathcal{M}}$.

Proof. From the results in Section IV-C, particularly Corollary 2, we know that a parallel TLL architecture of size determined by (79) can indeed approximate the unknown dynamics as required by conclusion i) above. Thus, it remains to show that if this NN is trained to that accuracy with respect to (unknown) $f$, then conclusion ii) holds for a controller, $\Psi$.

To this end, let $\Psi$ be such a controller that is $\delta$-$\tau$ positive invariant, and where $\delta$ and $\tau$ are selected according to the assumptions of the Theorem. Moreover, suppose that $f^{-\mathcal{M}}$ has been trained so that $\|f - f^{-\mathcal{M}}\| \leq \mu$ for $\mu$ as specified.

Thus, we can proceed as in the proof of Lemma 1. In particular, we seek to bound the difference

$$\|\zeta_{x_{x_{x_{x}}}}(\tau) - \zeta_{x_{x_{x_{x}}}}^{x_{x_{x_{x}}}}(\tau)\| < \delta$$  \hfill (82)

so that we can construct the appropriate simulation relations.

To prove this bound, observe that:

$$\|\zeta_{x_{x_{x}}}(t) - \zeta_{x_{x_{x}}}(t)\|

\leq \int_{0}^{t} \|f(\zeta_{x_{x}}(\sigma), \Psi(\zeta_{x_{x}}(\sigma))) - f(\zeta_{x_{x}}(\sigma), \Psi(\zeta_{x_{x}}(\sigma)))\| \, d\tau + \|f(\zeta_{x_{x}}(\sigma), \Psi(\zeta_{x_{x}}(\sigma))) - f(\zeta_{x_{x}}(\sigma), \Psi(\zeta_{x_{x}}(\sigma)))\| \, d\sigma

\leq \int_{0}^{t} (K_{x} + K_{u}K_{\text{cont}})\|\zeta_{x_{x}}(\sigma) - \zeta_{x_{x}}(\sigma)\| + \|f - f^{-\mathcal{M}}\|_{x_{x_{x}}} \, d\sigma

\leq \int_{0}^{t} (K_{x} + K_{u}K_{\text{cont}})\|\zeta_{x_{x}}(\sigma) - \zeta_{x_{x}}(\sigma)\| + \|f - f^{-\mathcal{M}}\|_{x_{x_{x}}} \, d\sigma

$$  \hfill (83)

where the last inequality follows from the fact that $\Psi$ was assumed to be designed such that it is $\delta$-$\tau$ positive invariant with respect to $X$ (and by construction $\Psi$ has codomain $U$). Of course the Grönewall inequality applies to (83) when we use the assumed training error bound $\|f - f^{-\mathcal{M}}\| \leq \mu$. Since $\delta$ and $\tau$ were chosen according to (78) – which has exactly the form of the Grönewall bound obtained in (83) – we conclude that (82) holds. Hence, the conclusion of the Theorem follows by way of arguments already discussed in Section IV.

VI. CONCLUSION

In this paper, we considered the problem of automatically designing assured ReLU NN architectures for both control and system identification of nonlinear control systems. To the best of our knowledge, our approach uniquely leverages the flexible TLL architecture and a generic specification paradigm via ADS in order to do automatic, assured NN architecture design for incompletely known nonlinear systems. We thus expect this work to serve as a starting point for further work on the problem of automatic NN architecture design, particularly with closed-loop dynamical specifications in mind. In conclusion, we offer two observations that present opportunities for generalization.

First, we note that in this paper, we have constructed CPWA functions in terms of their linear regions (see Step 2 of the proof of Theorem 2: Section IV-C). This construction by linear regions is ideally suited for conversion to TLL NNs using Theorem 1. However, in this paper, we have obtained these linear regions by a specific construction: first by setting down a regular grid of interpolation hypercubes and then subdividing those hypercubes into simplexes using the braid hyperplane arrangement. This scheme works because grid of hypercubes controls the size of the resulting simplexes, and the braid arrangement ensures that any full-dimensional face of one simplex is exactly shared by one and only one other simplex – i.e. it constitutes a “tiling”, which leads to a continuous piecewise function. However, this specific construction is not needed to employ Theorem 1 to obtain a TLL network: any other “tiling” by sufficiently small simplexes could be converted to a TLL approximate controller just as easily. Thus, a direction of future work would be to find a more efficient (or even minimal) “tiling” by simplexes, which would lead to smaller architectures.

Second, we make a similar observation but in the other direction. In particular, we have chosen to convert the above-described approximate CPWA controller specifically to a TLL NN, but a TLL may not be the most neuron-efficient NN representation. In other words, there may be other architectures that can represent the same, approximate CPWA controller using considerably fewer neurons. Thus, another compelling direction of future research would be to seek more compact assured architectures by exploring classes of NN architectures other than TLL NNs.

VII. ACKNOWLEDGMENTS

The authors wish to thank Xiaowu Sun (University of California, Irvine) for noting by example that a function defined on the corners of a hypercube can be extended by a CPWA that uses fewer regions than in Lemma 4 of [3].

REFERENCES

[1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al., “End to end learning for self-driving cars,” arXiv:1604.07316, 2016.

[2] J. Ferlez and Y. Shoukry, “AReN: Assured ReLU NN Architecture for Model Predictive Control of LTI Systems,” in Hybrid Systems: Computation and Control 2020 (HSCC’20), ACM, New York, 2020.

[3] J. Ferlez, X. Sun, and Y. Shoukry, “Two-Level Lattice Neural Network Architectures for Control of Nonlinear Systems,” in 2020 59th IEEE Conference on Decision and Control (CDC), pp. 2198–2203, 2020.

[4] T. Elsken, J. H. Metzen, and F. Hutter, “Neural Architectures Search: A Survey,” Journal of Machine Learning Research, vol. 20, no. 55, pp. 1–21, 2019.

[5] H. Mendoza, A. Klein, M. Feurer, J. T. Springenberg, and F. Hutter, “Towards Automatically-Tuned Neural Networks,” in Workshop on Automatic Machine Learning, pp. 58–65, PMLR, 2016.
A. Proof of Lemma 4

To prove Lemma 4, we need a few preliminary results.

**Proposition 4** (Coordinate Orderings and Braid Arrangements Regions [36]). Let $B_n$ be the braid arrangement defined in Lemma 4. Then for each (open) region $R$ of $B_n$ there is a unique permutation $\sigma_R : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$ such that
\[ \forall x \in R. \quad \sigma_R(i) < \sigma_R(j) \implies \pi_{\sigma_R(i)}(x) < \pi_{\sigma_R(j)}(x). \]  

In other words, the coordinates of any point in $R$ can be sorted into ascending order using a single, common permutation, and this permutation is unique to $R$.

**Proof.** See [36].

**Lemma 5.** Let $B_n$ be the braid arrangement, and let $D_{B_n}$ be the braid dissection of $C_n$ (both as defined in Lemma 4). Then any region $R$ of $B_n$ intersects exactly one $n-1$-dimensional face not containing the origin, i.e.
\[ F_R^R \triangleq R \cap F_1^1 \]  

is nonempty for exactly one $i = i_R \in \{1, \ldots, n\}$. Moreover, for any two distinct regions $R$ and $R'$ of $B_n$,
\[ F_R \cap F_{R'} = \emptyset, \]  

where the notation $F_{1^1}^R$ is simplified to $F_R$ by the above claim.

Finally, for each $R$, the closure $\overline{F}_R$ is isomorphic to a region $P$ of the braid dissection $D_{B_{n-1}}$ by means of the projection and embedding ($e_i$ is a column of the $(n \times n)$ identity matrix):
\[ \pi_{\sigma_R} : \mathbb{R}^n \rightarrow \mathbb{R}^{n-1}, \quad \pi_{\sigma_R} : x \mapsto (e_i \cdots e_{i+1} \cdots e_{n-1})' x \]  

(87) \[ (\overline{F}_R^1)^{1^1} \triangleq P : \overline{F}_R \]  

i.e. $\pi_{\sigma_R} : F_R \rightarrow P$ is a bijection that maps corners of $C_n$ to corners of $C_{n-1}^1$; its inverse is $F_{1^1}^P : P \rightarrow \overline{F}_R$.

**Proof.** First, we claim that the any region $R$ of $B_n$ intersects the exact one $F_1^1$. This follows because each point in (the open set) $R$ necessarily has the same unique maximal coordinate by Proposition 4; identify this maximal coordinate (the open set)
\[ \mathbb{R}^{n-1} \]  

because its coordinates respect the sorting associated with $R$. To show the second claim, merely note that the regions of $B_n$ are disjoint by definition (they are regions of a hyperplane arrangement), so $R \neq R'$ implies that $\overline{F}_R \cap \overline{F}_{R'} = \emptyset$.

Finally, consider $\overline{F}_R$ for $R$ in $B_n$. We claim $\overline{F}_R$ is isomorphic to the region in $D_{B_{n-1}}$ given by $P = \overline{F}_R \cap \overline{F}_{C_{n-1}^1}$, where $R'$ is the region of $B_{n-1}$ associated with:
\[ \sigma_R : i \mapsto \begin{cases} \sigma_R(i) & i < i_R \\ \sigma_R(i+1) & i \geq i_R \end{cases} \]  

(90)
which is in effect the ordering \( \sigma_R \) with its maximal element excluded. Thus, any point \( x \in R \) maps to a point in \( P \) so defined, since it trivially maps to a point in \( R' \) via \( \pi_{-i} \). Likewise, any point in \( P \cap R' \) maps to a point in \( R \cap F_{1i}^1 \) by the inclusion of the additional coordinate specified by \( \mathcal{F}_{1i} \). Furthermore, these mappings extend to the closed sets specified by continuity. It follows that corners of one set are mapped to corners of the other and vice versa, since \( \pi_{-i} \) and \( \mathcal{F}_{1i} \) either remove or include a coordinate with value 1. 

Now we can state formally the proof of Lemma 4.

**Proof.** We claim that the dissection \( D_{B_n} \) (see (71)) consists of exactly \( n! \) simplexes, each of whose vertices are in \( \mathcal{F}_0(C) \). Thus, \( \Xi_{\alpha_n}^n \) has at most \( n! \) linear regions, one for each simplex.

First, note that the interior of each region of \( B \) intersects \( \text{int}(C_n) \); this is because all possible coordinate orderings of the point \((\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2})\) are contained in \( \text{int}(C_n) \). Hence, each region of \( B_n \) contributes one region to \( D_{B_n} \), so \( |D_{B_n}| = n! \).

Thus, it remains to show that the intersection of \( C_n \) with each region in \( B \) is in fact an \( n \)-simplex, and we show this by induction on dimension. As the base case, consider \( n = 2 \).

In particular, \( B_2 \) has only one hyperplane, and it dissects \( R \) into two triangles (i.e. 2-simplexes), each of whose vertices are vertices of the square \( C_2 \). Hence, suppose that \( B_n \) divides \( C_n \) into \( n! \)-simplexes with vertices from \( \mathcal{F}_0(C_n) \); we will show that this implies that \( B_{n+1} \) divides \( C_{n+1} \) into \( n+1 \)-simplexes, each of whose vertices are from \( \mathcal{F}_0(C_{n+1}) \).

To complete the proof of the induction step, let \( R \) be a region from \( B_{n+1} \), and observe that for any \( x \in R \cap \text{int}(C_{n+1}) \) and any \( t > 0 \), \( tx \) remains in \( R \), since multiplication by a non-negative scalar doesn’t change the ordering of the coordinates of \( x \). For a similar reason, there is a \( t_{\max}(x) \in \mathbb{R} \) such that

\[
t_{\max}(x) \triangleq \sup_{t \in \text{int}(C_{n+1})} t = \sup_{t \in \text{int}(C_{n+1})} t_{\max}, \quad \alpha(x) = 1. \tag{91}
\]

Thus, we conclude that \( R \cap \text{int}(C_{n+1}) \) is the interior of a cone that extends from the origin, and it has a (convex) base contained entirely in some \( F_{1i}^1 \) by Lemma 5; indeed, that base is isomorphic to \( C_n \cap R' \) where \( R' \) is a region from \( B_n \). Of course, by the induction hypothesis, \( C_n \cap R' \) is an \( n \)-simplex whose vertices are contained in \( \mathcal{F}_0(C_n) \), so the base of the cone \( R \cap \text{int}(C_{n+1}) \) is necessarily an \( n \)-simplex whose vertices lie in \( F_{1i}^1 \cap \mathcal{F}_0(C_{n+1}) \), again by Lemma 5. Thus, those vertices, together with the origin, constitute the \( n + 2 \) vertices of \( R \cap \text{int}(C_{n+1}) \), and each vertex is a corner of \( C_{n+1} \) by construction. Hence, \( R \cap \text{int}(C_{n+1}) \) is an \( n + 1 \) simplex whose vertices are in \( \mathcal{F}_0(C_{n+1}) \) as claimed.

It remains to show that this \( \Xi_{\alpha_n}^n \) satisfies iii). To wit, let \( \alpha, \alpha' : \mathcal{F}_0(C) \to \mathbb{R} \), and assume without loss of generality that \( k = 1 \) so that

\[
\forall x \in F_1^1 \cap \mathcal{F}_0(C), \quad \alpha(x) = \alpha'(|x - e_1|). \tag{92}
\]

Now let \( R \) be a region of \( B_n \), with \( R \cap F_{1i}^1 \neq \emptyset \). It suffices to show that there is a region \( R' \) with \( R' \cap F_0^0 \neq \emptyset \) such that \( \overline{R \cap F_{1i}^1} \) and \( \overline{R' \cap F_0^0} \) are isomorphic under the (self-inverse) translation map \( T_i : x \mapsto |x - e_1| \). This is because \( \Xi_{\alpha_n}^n \) and \( \Xi_{\alpha'} \) linearly interplate between the values of \( \alpha \) and \( \alpha' \) on \( \overline{R \cap F_{1i}^1} \) and \( \overline{R' \cap F_0^0} \), respectively, and \( \alpha \) and \( \alpha' \) agree on translates of these points in their domains. For example, consider the illustration of \( D_{B_2} \) shown in Fig. 2: note that any triangle (2-simplex) created by \( D_{B_2} \) on one face is the single-coordinate translate of a triangle created by \( D_{B_2} \) on the opposite face.

However, this follows as a more or less direct consequence of Lemma 5. Indeed, we note that \( R \cap F_{1i}^1 \) is isomorphic to some \( R_{n-1} \in B_{n-1} \). But it is easy to see that this \( R \) is isomorphic to a set of the claimed form \( R \cap F_{1i}^1 \), by using the inclusion map \( \mathcal{F}_{1i} \) instead of \( \mathcal{F}_{1i}^1 \) and with the usual inverse \( \pi_{-i} \).