In particular, we seek to use knowledge of the underlying control problem to guide the design of NN architectures.

Our approach exploits several insights. First, state-of-the-art NN utilizes Rectified Linear Units (ReLU), which in turn restricts the NN controller to implement only Continuous Piecewise Affine (CPWA) functions. As is widely known, CPWA function is compromised of several affine functions (named local linear functions), which are defined over a set of polytypic regions (called local linear regions). In other words, a ReLU NN—by virtue of its CPWA character—partitions its input space into a set of polytypic regions (named activation regions), and applies a linear controller at each of these regions. Therefore, a NN architecture dictates the number of such activation regions in the corresponding CPWA function that is represented by the trainable parameters in the NN. That is, to design a NN architecture, one needs to perform two steps: (i) compute (or upper bound) the number of activation regions required to implement a controller that satisfy the specifications and (ii) transform this number of activation regions into a NN architecture that is guaranteed to give rise to this number of activation regions.

To approach the first step, namely counting the number of the required activation regions, we assume the existence of an unknown robust Lipschitz-continuous, state-feedback controller with some Lipschitz constant upper-bounded by $K_{\text{cont}}$ that is capable of controlling the system while meeting the specifications. Without the knowledge of such controller, other than the upper bound on its Lipschitz constant $K_{\text{cont}}$, we can upper-bound the number of activation regions needed to approximate this controller by a CPWA function while still meeting the same specifications.

Next, we build on recent results obtained by the authors on a novel NN architecture named Two-Level Lattice (TLL) NN architecture [7]. Unlike other NN architecture for which the number of activation regions is unknown a priori, the TLL-NN architecture enjoys the property that it is parametrized directly by the number of its activation regions. That is, once the number of activation regions is computed using the existence of such an unknown robust Lipschitz-continuous controller, a TLL-NN architecture can be directly generated from this knowledge. Such NN is then guaranteed to be sufficiently parametrized to implement a CPWA function that approximates the unknown Lipschitz-continuous controller, providing a systematic approach to design such architecture for NN controllers.
For a set \( V \subseteq \mathbb{R}^n \), let \( \text{int}(V) \) return the interior of \( V \).
For \( x \in \mathbb{R}^n \), we will denote by \( ||x|| \) the infinity norm of \( x \);
for \( x \in \mathbb{R}^n \) and \( \varepsilon > 0 \) we will denote by \( B(x; \varepsilon) \) the ball of
radius \( \varepsilon \) centered at \( x \) as specified by \( ||\cdot|| \).
For \( f : \mathbb{R}^n \rightarrow \mathbb{R}^m \), \( ||f||_{\infty} \) will denote the essential supremum norm of \( f \).
Finally, given two sets \( A \) and \( B \) denote by \( B^A \) the set of all functions
with domain \( A \) and range \( B \).

**B. Dynamical Model**

In this paper, we will consider a continuous-time nonlinear dynamical system
defined by the ordinary differential equation (ODE):

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

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

A control system is said to be (globally) Lipschitz if there

\[
\| f(x, u) - f(x', u') \| \leq K_x \| x - x' \| + K_u \| u - u' \|.
\]

(2)

In the sequel, we will primarily be concerned with solutions
to (1) that result from instantaneous state-feedback controllers,
\( \Psi : X \rightarrow U \). Thus, we use \( \zeta_{x_0, \Psi} \) 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 \( \zeta_{x_0, \Psi} \) as a (closed-loop) trajectory of its
associated control system.

**Definition 2 (Closed-loop Trajectory).** Let \( \Sigma \) be a Lipschitz control system,
and let \( \Psi : \mathbb{R}^n \rightarrow U \) 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 \( \zeta_{x_0, \Psi} : \mathbb{R}^+ \rightarrow X \) that uniquely solves the integral equation:

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

(3)

It is well known that such solutions exist and are unique under these assumptions [8].
We will only consider feedback controllers for which \( X \) is positively invariant under feedback, i.e. range(\( \zeta_{x_0, \Psi} \)) \( \subseteq X \).

For any given feedback controller, \( \Psi \), the open-loop control functions
created by its trajectories may not be elements of \( \mathcal{U} \).
Thus, we make the following additional definition:

**Definition 3 (Feedback Controllable).** A Lipschitz control system \( \Sigma \)

is feedback controllable by a Lipschitz controller \( \Psi : \mathbb{R}^n \rightarrow U \) if the following is satisfied:

\[
\Psi \circ \zeta_{x, \Psi} \in \mathcal{U} \quad \forall x \in X.
\]

(4)

A Lipschitz control system is called feedback controllable if it is
feedback controllable for each globally Lipschitz feedback controller.

**Remark 1.** In this paper, we will henceforth consider only feedback controllable Lipschitz control systems.

We conclude this subsection by defining the (sampled) transition system embedding
of a feedback-controlled system that is inspired by the work in [9].

**Definition 4 (\( \tau \)-Sampled Transition System Embedding).** Let \( \Sigma = (X, U, \mathcal{U}, f) \) be a feedback controllable Lipschitz control system, and let \( \Psi : \mathbb{R}^n \rightarrow U \) be a Lipschitz continuous feedback controller. For any \( \tau > 0 \), the \( \tau \)-sampled transition system embedding of \( \Sigma \) under \( \Psi \) is the tuple \( S_\tau(\Sigma_\Psi) = (X_\tau, \mathcal{U}_\tau, \Psi_{\tau \rightarrow \tau}) \) where:

- \( X_\tau = X \) is the state space;
- \( \mathcal{U}_\tau = \{ (\Psi \circ \zeta_{x_0, \Psi})|_{t \in [0, \tau]} : x_0 \in X \} \) is the set of open
  loop control inputs generated by \( \Psi \)-feedback, each restricted to the domain \( [0, \tau] \); and
- \( \Psi_{\tau \rightarrow \tau} : X_\tau \times \mathcal{U}_\tau \rightarrow X_\tau \) such that \( x \overset{\mathcal{U}}{\rightarrow} x' \) iff
  both \( u = (\Psi \circ \zeta_{x_0, \Psi})|_{t \in [0, \tau]} \) and \( x' = \zeta_{x_0, \Psi}(\tau) \).

\( S_\tau(\Sigma_\Psi) \) is thus a metric transition system [9].

C. Abstract Disturbance Simulation

In this subsection, we propose a new simulation relation, which we call abstract disturbance simulation,
as a formal notion of specification satisfaction for metric transition systems. Abstract disturbance simulation enforces a notion of specification that is robust to perturbation of the state, and this will facilitate solving the main problem in this paper.

Abstract disturbance simulation is inspired by robust bisimulation [10] and especially disturbance bisimulation [11], but it abstracts those notions away from their definitions in terms of control system embeddings and explicit modeling of disturbance inputs. In this way, it is conceptually similar to the technique used in [9] and [12] to define a quantized abstraction, where deliberate non-determinism is introduced in order to account for input errors. As a prerequisite, we introduce the following definition.

**Definition 5 (Perturbed Metric Transition System).** Let \( S = (X, U, \delta) \) be a metric transition system where \( X \subseteq X_M \) for some metric space \((X_M, d)\). Then the \( \delta \)-perturbed metric transition system of \( S \), \( \delta \), is a tuple \( \delta = (X, U, \phi_{\delta \rightarrow \delta}) \) where the (altered) transition relation, \( \phi_{\delta \rightarrow \delta} \), is defined as follows:

\[
x \overset{\phi_{\delta \rightarrow \delta}}{\rightarrow} x' \text{ iff } \exists x'' \in X \text{ s.t. } d(x'', x') \leq \delta \text{ and } x \overset{\mathcal{U}}{\rightarrow} x''.
\]

(5)

Note that \( \delta \) has identical states and input labels to \( S \), and it also subsumes all of the transitions therein, i.e. \( \delta \rightarrow \delta \). However, the transition relation for \( \delta \) explicitly contains new nondeterminism relative to the transition relation of \( S \). This nondeterminism can be thought of as perturbing the targets state of each transition in \( S \); each such perturbation becomes the target of a (nondeterministic) transition with the same input label as the original transition.

\(^1\)For our purposes, a metric transition system is one whose state space is contained in a metric space.
With this definition in hand, we can finally define an abstract disturbance simulation between two metric transition systems.

Definition 6 (Abstract Disturbance Simulation). Let \( S = (X_S, U, \rightarrow_S) \) 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 \( (M, d) \). Then \( T \) abstract-disturbance simulates \( S \) under disturbance \( \delta \), written \( S \preceq_{AD_S} T \) if there is a relation \( R \subseteq X_S \times X_T \) such that:

1. For every \((x, y) \in R\), \( d(x, y) \leq \epsilon\); and
2. For every \( x \in X_S \) there exists a pair \((x, y) \in R\); and
3. For every \((x, y) \in R\), \( x \rightarrow_S u \rightarrow x' \) there exists a \( y \rightarrow_T u \rightarrow y' \) such that \((x', y') \in R\).

Remark 2. \( S \preceq_{AD_S} T \) corresponds with the usual notion of simulation for metric transition systems. Thus,

\[
S \preceq_{AD_S} T \iff \Theta \preceq_{AD_S} T. \tag{6}
\]

D. ReLU Neural Network Architectures

In this paper, our primary focus will be on controlling the nonlinear system defined in (1) with a state-feedback neural network controller \( \mathcal{M} \):

\[
\mathcal{M} : X \rightarrow U \tag{7}
\]

where \( \mathcal{M} \) denotes a Rectified Linear Unit Neural Network (ReLU NN). Such a \((K\)-layer\) ReLU NN is specified by composing \( K \) layer functions (or \( j \) layers). A layer with \( i \) inputs and \( o \) outputs is specified by a \((o \times i)\) real-valued matrix of weights, \( W \), and a \((o \times 1)\) real-valued matrix of biases, \( b \), as follows:

\[
L_\theta : \mathbb{R}^i \rightarrow \mathbb{R}^o \\
z \mapsto \max\{Wz + b, 0\} \tag{8}
\]

where the \( \max \) function is taken element-wise, and \( \theta \triangleq (W, b) \) for brevity. Thus, a \( K \)-layer ReLU NN function as above is specified by \( K \) layer functions \( \{L_{\theta(i)} : i = 1, \ldots, K\} \) whose input and output dimensions are composable; that is they satisfy \( i_i = o_{i-1} \). Specifically:

\[
\mathcal{M}(x) = (L_{\theta(K)} \circ L_{\theta(K-1)} \circ \cdots \circ L_{\theta(1)})(x). \tag{9}
\]

When we wish to make the dependence on parameters explicit, we will index a ReLU function \( \mathcal{M} \) by a list of matrices \( \Theta \triangleq (\theta(1), \ldots, \theta(K)) \).

Specifying the number of layers and the dimensions of the associated matrices \( \theta(i) = (W^{(i)}, b^{(i)}) \) specifies the architecture of the ReLU NN. Therefore, we will use:

\[
\text{Arch}(\Theta) \triangleq ((n, o_1), (i_2, o_2), \ldots, (i_{K-1}, o_{K-1}), (i_K, m)) \tag{10}
\]

to denote the architecture of the ReLU NN \( \mathcal{M}_{\Theta} \).

Since we are interested in designing ReLU architectures, we will also need the following result from [7, Theorem 7], which states that a Continuous, Piecewise Affine (CPWA) function, \( f \), can be implemented exactly using a Two-Level-Lattice (TLL) NN architecture that is parameterized exclusively by the number of local linear functions in \( f \).

Definition 7 (Local Linear Function). Let \( f : \mathbb{R}^n \rightarrow \mathbb{R}^m \) be CPWA. Then a local linear function of \( f \) is a linear function \( \ell : \mathbb{R}^n \rightarrow \mathbb{R}^m \) if there exists an open set \( \mathcal{O} \) such that \( \ell(x) = f(x) \) for all \( x \in \mathcal{O} \).

Definition 8 (Linear Region). Let \( f : \mathbb{R}^n \rightarrow \mathbb{R}^m \) be CPWA. Then a linear region of \( f \) is the largest set \( \mathcal{R} \subseteq \mathbb{R}^n \) such that \( f \) has only one local linear function on the interior of \( \mathcal{R} \).

Theorem 1 (Two-Level-Lattice (TLL) NN Architecture [9, Theorem 7]). Let \( f : \mathbb{R}^n \rightarrow \mathbb{R}^m \) be a CPWA function, and let \( N \) be an upper bound on the number of local linear functions in \( f \). Then there is a Two-Level-Lattice (TLL) NN architecture \( \text{Arch}(\Theta_{TLL}) \) parameterized by \( N \) and values of \( \Theta_{TLL} \) such that:

\[
f(x) = \mathcal{M}_{\Theta_{TLL}}(x). \tag{11}
\]

In particular, the number of linear regions of \( f \) is such an upper bound on the number of local linear functions.

We refer the reader to [7] for more details.

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

III. PROBLEM FORMULATION

We can now state the main problem we will consider in this paper. In brief, we wish to identify the architecture for a ReLU network to be used as an instantaneous feedback controller for the control system \( \Sigma \): this architecture must have parameter weights that allow it to control \( \Sigma \) up to a specification that can be met by some other, non-NN controller.

Despite our choice to consider fundamentally continuous-time models, we formulate our main problem in terms of their \((\tau\text{-sampled})\) transition system embeddings. This choice reflects recent success in verifying specifications for such transition system embeddings by means of techniques adapted from computer science; see e.g. [13], where a variety of specifications are considered in this context, among them LTL formula satisfaction. Thus, our main problem is stated in terms of the simulation relations in the previous section.

Problem 1. Let \( \delta > 0 \) and \( K_{cont} > 0 \) be given. Let \( \Sigma \) be a feedback controllable Lipschitz control system, and let \( S_{spec} = (X_{spec}, U_{spec}, s_{spec}) \) be a transition system encoding for a specification on \( \Sigma \). Finally, let \( \tau = \min(f, K_x, K_u, K_{cont}, \delta) \) be determined by the parameters specified.

Now, suppose that there exists a Lipschitz continuous controller \( \Psi : \mathbb{R}^n \rightarrow U \) with Lipschitz constant \( K_\Psi \leq K_{cont} \) such that:

\[
S_\tau(\Sigma_\Psi) \preceq_{AD_S} S_{spec}. \tag{12}
\]

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

\[
S_\tau(\Sigma_{\mathcal{M}_\Theta}) \preceq_{AD_S} S_{spec}. \tag{13}
\]

One of the primary assumptions in Problem 1 is that there exists a controller \( \Psi \) which satisfies the specification, \( S_{spec} \). We use this assumption largely to help ensure that the
IV. RELU ARCHITECTURES FOR NONLINEAR SYSTEMS

Before we state the main theorem of the paper, we introduce the following notation in the form of two definitions.

Definition 9 (Vector Field Bound, \( K \)). Let:
\[
K \triangleq \max_{x \in X, u \in U} \| f(x, u) \|, \tag{14}
\]
which is well defined because \( X \times U \) is compact and \( f \) is continuous.

Definition 10 (Extent of \( X \)). The extent of the compact set \( X \) is defined as:
\[
\text{ext}(X) \triangleq \max_{k=1, \ldots, n} \left| \max_{x \in X} \pi_k(x) - \min_{x \in X} \pi_k(x) \right|, \tag{15}
\]
where \( \pi_k(x) \) is the projection of \( x \) onto its \( k^{th} \) component.

The main result of the paper is the following theorem, which directly solves Problem 1.

Theorem 2 (ReLU Architecture). Let \( \delta > 0 \) and \( K_{\text{cont}} > 0 \) be given, and let \( \Sigma \) and \( S_{\text{spec}} \) be as in the statement of Problem 1. Finally, choose a \( \mu > 0 \) such that:
\[
K_a \cdot \mu < 6 \cdot K_{\text{cont}} \cdot K \cdot e^{K_{a, \mu} \cdot \frac{\mu}{\mu_{\text{cont}}}} < \delta, \tag{16}
\]
and set:
\[
\tau \leq \frac{\mu}{6 \cdot K_{\text{cont}} \cdot K} \quad \text{and} \quad \eta \leq \frac{\mu}{6 \cdot K_{\text{cont}}}, \tag{17}
\]
(which depend only on \( f, K_a, K_{\text{cont}}, \delta \)).

If there exists a Lipschitz continuous controller \( \Psi : \mathbb{R}^n \to U \) with Lipschitz constant \( K_{\Psi} \leq K_{\text{cont}} \) such that:
\[
S_\tau(\Sigma_{\Psi}) \preceq_{\text{AD}} S_{\text{spec}}. \tag{18}
\]
Then a TLL NN architecture \( \text{Arch}(\Theta_{N_{\mu}}) \) of size:
\[
N \geq m \cdot \left( n! \cdot \sum_{k=1}^{n} \frac{2^{k-1}}{(n-k)!} \right) \frac{\text{ext}(X)}{\eta} \tag{19}
\]
has the property that there exist values for \( \Theta_{N_{\mu}} \) such that:
\[
S_\tau(\Sigma_{\Psi_{\mu}}) \preceq_{\text{AD}} S_{\text{spec}}. \tag{20}
\]

Proof Sketch:

The proof of Theorem 2 consists of 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 – which approximates \( \Psi \) to accuracy \( \mu \) will also satisfy the specification (but under no disturbance). This implication is shown in Lemma 2 of Section V.

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 desired approximation accuracy. This implication is shown in Lemma 4 of Section VI.

We will show these results for any controller \( \Psi \) that satisfies the assumptions of Theorem 2. Thus, these results together show the following implication: if there exists a controller \( \Psi \) that satisfies the assumptions of Theorem 2, then there is a CPWA controller that satisfies the specification. And moreover, this CPWA controller has a at most a number of linear regions that depends only on the parameters of the problem and not the particular controller \( \Psi \).

The conclusion of the theorem will then follow directly from Theorem 1 [7, Theorem 7]: together, they specify that any CPWA with the same number of linear regions (or fewer) can be implemented exactly by a common TLL NN architecture. Since this proof is so short given the lemmas described above, it appears in Appendix E of [14].

V. PROOF OF THEOREM 2, STEP 1: APPROXIMATE CONTROLLERS SATISFY THE SPECIFICATION

The goal of this section is to choose constants \( \mu > 0 \) and \( \tau > 0 \) such that any controller \( \Upsilon \) with \( \| \Upsilon - \Psi \|_\infty \leq \mu / 3 \) satisfies the specification:
\[
S_\tau(\Sigma_{\Upsilon}) \preceq_{\text{AD}} S_{\text{spec}}. \tag{21}
\]

The approach will be as follows. First, we confine ourselves to a region in the state space on which the controller \( \Psi \) doesn’t vary much: the size of this region is determined entirely by the approximation accuracy, \( \mu \), and the bound on the Lipschitz constant, \( K_{\text{cont}} \). Then we confine the trajectories of \( \Sigma_{\Psi} \) to this region by bounding the duration of those trajectories, i.e. \( \tau \). Finally, we feed these results into a Grönwall-type bound to choose \( \mu \). In particular, we choose \( \mu \) small enough such that the error incurred by using \( \Upsilon \) instead of \( \Psi \) is within the disturbance robustness, \( \delta \). From this we will conclude that \( \Upsilon \) satisfies the specification as claimed whenever \( \| \Upsilon - \Psi \| \leq \mu / 3 \). A more detailed road map of these steps is as follows.

- Let \( \mu \) be an approximation error. Then:
  \begin{itemize}
  \item \( i \) Choose \( \eta = \eta(\mu) \) such that a Lipschitz function with constant \( K_{\text{cont}} \) doesn’t vary by more than \( \mu / 3 \) between any two points that are \( 2\eta \) apart.
  \end{itemize}
Lemma 1. Let \( \mu > 0 \) be given, and let \( \Psi \) be as before. Then there exists an \( \eta = \eta(\mu) \) such that:
\[
\|x - x'\| \leq 2\eta \Rightarrow \|\Psi(x) - \Psi(x')\| \leq \mu/3. 
\] (24)

Proposition 2. Let \( \mu > 0 \) be given, and let \( \eta = \eta(\mu) \) as in the previous proposition. Finally, let \( \Sigma \) be as specified in the statement of Theorem 2. Then there exists a \( \tau = \tau(\mu) \) such that for any Lipschitz feedback controller \( \Upsilon \):
\[
\|x - \zeta_x(\tau)\| \leq \eta(\mu) \quad \forall t \in [0, \tau]. 
\] (25)

Proposition 3. Let \( \mu > 0 \) be given. Let \( \Sigma \) and \( \Psi \) be as in the statement of Theorem 2; let \( \eta = \eta(\mu) \) be as in Proposition 1; let \( \tau = \tau(\mu) \) be as in Proposition 2; and let \( \Upsilon : \mathbb{R}^n \to U \) be a Lipschitz continuous function. Then:
\[
\forall t \in [0, \tau] \quad \|\Upsilon(\zeta_x(\tau)) - \Psi(\zeta_x(\tau))\| \leq \|\Upsilon - \Psi\|_\infty + 2\mu/3. 
\] (26)

To prove Step iv) we first need the following two results.

Proposition 4 (Grönwall Bound). Let \( \Sigma \) and \( \Psi \) be as in the statement of Theorem 2, and let \( \Upsilon \) be as in the statement of Proposition 3. If:
\[
\|\Upsilon(\zeta_x(\tau)) - \Psi(\zeta_x(\tau))\| \leq \kappa \quad \forall t \in [0, \tau] 
\] (27)

then:
\[
\|\zeta_x(\tau) - \zeta_x(\tau)\| \leq K_a \cdot \kappa \cdot t \cdot e^{K_a t} \quad \forall t \in [0, \tau]. 
\] (28)

The proof of Proposition 4 appears in Appendix D of [14].

Lemma 1. Let \( \Sigma \), \( \Psi \) and \( \Upsilon \) be as before. Also, suppose that \( \mu' > 0 \) is such that:
\[
K_a \cdot \mu' \cdot \frac{\mu}{6 \cdot K_{cont} \cdot K} \leq e^{K_a \cdot \frac{\mu}{6 \cdot K_{cont} \cdot K}} < \delta. 
\] (29)

If \( \|\Upsilon - \Psi\|_\infty \leq \mu/3 \), then:
\[
\|\zeta_x(\tau(\mu')) - \zeta_x(\tau(\mu))\| \leq \delta. 
\] (30)

Proof. This is a direct consequence of applying Proposition 3 to Proposition 4.

The final result in this section is the following Lemma.
as a CPWA. Moreover, note that this procedure is actually independent of the particular \( \Psi \) chosen, despite appearances: we are basing our construction on a grid size \( \eta \) that depends only on the problem parameters, and the construction will work no matter what particular value \( \Psi(x) \) has within each grid square.

The first step in this procedure will be to show how to extend such a function over the largest-dimensional “gaps” between the smaller inscribed balls; the blue region depicted in Fig. 1 is an example of this large-dimensional gap for \( X \subseteq \mathbb{R}^2 \) (the notation in the figure will be explained later). This result must control the error of the extension so as to preserve our desired approximation bound, as well provide a count of the number of linear regions necessary to do so; this is Lemma 3. This result can then be extended to all of the other gaps between inscribed balls to yield a CPWA function with domain \( X \), approximation error \( \mu/3 \), and a known number of regions; this is Lemma 4.

In order to prove our first lemma of this section, we need a couple of definitions to help with the terminology.

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

\[
\forall x \in F . \quad \bigwedge_{j \in J} \pi_j(x) \in \{0,1\}.
\]

Let \( \mathcal{F}_k(C) \) denote the set of \( k \)-dimensional faces of \( C \), and let \( \mathcal{F}(C) \) denote the set of all faces of \( C \) (of any dimension).

**Remark 3.** A \( k \)-dimensional face of the hypercube \( C = [0,1]^n \) is isomorphic to the hypercube \([0,1]^k\).

**Definition 12 (Corner).** Let \( C = [0,1]^n \). A *corner* of \( C \) is a 0-dimensional face of \( C \).

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

\[
\Gamma_c : \mathcal{F}_0(C) \to \mathbb{R}
\]

is a function defined on the corners of \( C \). Then there is a CPWA function \( \Gamma : C \to \mathbb{R} \) such that:

1. \( \forall x \in \mathcal{F}_0(C), \Gamma(x) = \Gamma_c(x) \), i.e. \( \Gamma \) extends \( \Gamma_c \) to \( C \);
2. \( \Gamma \) has at most \( 2^{n-1} \cdot n! \) linear regions; and
3. for all \( x \in C \),

\[
\min_{x \in \mathcal{F}_0(C)} \Gamma_c(x) \leq \Gamma(x) \leq \max_{x \in \mathcal{F}_0(C)} \Gamma_c(x).
\]

**Proof.** First, we assume without loss of generality that the given function \( \Gamma_c \) takes distinct values on each element of its domain.

This is a proof by induction on dimension. In particular, we will use the following induction hypothesis:

- There is a function \( \Gamma_k : \bigcup_{i=1}^k \mathcal{F}_i(C) \to \mathbb{R} \) such that for all \( F \in \mathcal{F}_k(C), \Gamma_k|_F \) has the following properties:
  - it is CPWA
  - it has at most \( 2^{k-1} \cdot k! \) linear regions; and
  - for all \( x \in F \):

\[
\min_{x \in \mathcal{F}_0(F)} \Gamma_c(x) \leq \Gamma_k(x) \leq \max_{x \in \mathcal{F}_0(F)} \Gamma_c(x).
\]

We start by showing that if the induction hypothesis above holds for \( k \), then it also holds for \( k+1 \).

To show the induction step, first note that for any face \( F \in \mathcal{F}_{k+1}(C) \), all of its faces are already in the domain of \( \Gamma_k \). That is \( \bigcup_{i=1}^k \mathcal{F}_i(F) \subseteq \text{dom}(\Gamma_k) \). Thus, we can define \( \Gamma_{k+1} \) by extending \( \Gamma_k \) to \( \text{int}(F) \) for each \( F \in \mathcal{F}_{k+1}(C) \). Since these interiors are mutually disjoint, we can do this by explicit construction on each individually, in such a way that the desired properties hold.

In particular, let \( F \in \mathcal{F}_{k+1}(C) \), and let \( \nu \) be the midpoint of \( F \), i.e. the \( k \)-cube isomorphism of \( \nu \) is \([1/2,1/2] \). \( \nu \) is clearly in the interior of \( F \), so define:

\[
\Gamma_{k+1}(\nu) = \frac{1}{|\mathcal{F}_0(F)|} \sum_{x \in \mathcal{F}_0(F)} \Gamma_k(x)
\]

and note that the corners of \( F \) are also corners of \( C \), thus \( \Gamma_{k+1}(\nu) \) is the average of all of the corners of the \( k+1 \)-face that contains it. Now, extend \( \Gamma_{k+1} \) to the rest of \( \text{int}(F) \) as follows: let \( b \in \bigcup_{i=1}^k \mathcal{F}_i(F) \) and define:

\[
\Gamma_{k+1}(\lambda \cdot \nu + (1-\lambda) \cdot b) = \lambda \cdot \Gamma_{k+1}(\nu) + (1-\lambda) \cdot \Gamma_k(b) \quad \forall \lambda \in [0,1].
\]

This definition clearly covers \( \text{int}(F) \), and it also satisfies the requirement that:

\[
\min_{x \in \mathcal{F}_0(F)} \Gamma_c(x) \leq \Gamma_{k+1}(x) \leq \max_{x \in \mathcal{F}_0(F)} \Gamma_c(x)
\]

because the induction hypothesis ensures that each \( b \) is on a face of \( F \), and the corners of a face of \( F \) are a subset of the corners of \( F \). Thus, it remains to show the bound on the number of linear regions. But from the construction, \( \Gamma_{k+1}|_F \) has one linear region for linear region of \( \Gamma_k \) on a \( k \)-face of \( F \). Since the \( k+1 \)-face has \( 2 \cdot (k+1) \) faces, we conclude by the induction hypothesis that \( \Gamma_{k+1}|_F \) has at most:

\[
2 \cdot (k+1) \cdot 2^{k-1} \cdot k! = 2^k \cdot (k+1)!
\]

linear regions. This completes the proof of the induction step.

It remains only to show a base case. For this, we select \( k = 1 \), i.e. the line-segment faces of \( C \). Each 1-face of \( C \) has only two corners and no other faces other than itself. Thus, for each \( F \in \mathcal{F}_0(C) \) we can simply define \( \Gamma_1|_F \) to linearly interpolate between those two corners. \( \Gamma_1|_F \) is thus CPWA, and it satisfies the required bounds on its values. Moreover, \( \Gamma_1|_F \) has exactly \( 2^{1-1} \cdot 1! = 1 \) linear region. Thus, the function \( \Gamma_1 \) so defined satisfies the induction hypothesis stated above.

**Definition 13 (\( \eta \)-partition).** Let \( \eta > 0 \) be given. Then an \( \eta \)-partition of \( X \) is a regular, non-overlapping grid of \( \eta/2 \) balls in the sup norm that partitions \( X \). Let \( X_{\text{cent}} \) denote the set of centers of these balls, and let \( X_{\text{part}} = \{B(x; \eta/2) : x \in X_{\text{cent}}\} \) denote the partition.

**Definition 14 (Neighboring Grid Center/Square).** Let \( X_{\text{part}} \) be an \( \eta \)-partition of \( X \), and let \( B(x; \eta/2) \in X_{\text{part}} \). Then a neighboring grid center (resp. square) to \( x \) is an \( x' \in X_c \) (respectively \( B(x'; \eta/2) \in X_{\text{part}} \)) such that \( B(x'; \eta/2) \) shares a face (of any dimension) with \( B(x; \eta/2) \). The set of neighbors of a center, \( x \), will be denoted by \( N(x) \).

**Lemma 4.** Let \( \eta = \eta(\mu) \) be chosen as in Proposition 1, and let \( \Psi \) be as before. Then there is a CPWA function \( \Psi_{\text{cpwa}} : \mathbb{R}^n \to U \) such that:
• \( \| \mathcal{T}_{\text{cpwa}} - \Psi \|_\infty \leq \frac{\eta}{3} \); and
• \( \mathcal{T}_{\text{cpwa}} \) has at most
\[
m \cdot \left( n! \cdot \sum_{k=1}^{n} \frac{2^{2k-1}}{(n-k)!} \cdot \left( \frac{\text{ext}(X)}{\eta} \right)^n \right) \tag{43}
\]
linear regions.

**Proof.** Our proof will assume that \( U \subseteq \mathbb{R} \), since the extension to \( m > 1 \) is straightforward from the \( m = 1 \) case. The basic proof will be to create an \( \eta \)-partition of \( X \), and define \( \mathcal{T}_{\text{cpwa}} \) to be constant on \( \rho \cdot \eta/2 < \eta/2 \) radius balls centered at each of the grid centers in the partition; we will then use Lemma 3 to “extend” this function to the rest of \( X \) as a CPWA function. In particular, for each \( x_c \in X_C \), we start by defining:
\[
\mathcal{T}_{\text{cpwa}}(x) = \Psi(x_c) \quad \forall x \in B(x_c; \rho \cdot \eta/2). \tag{44}
\]
Then we will extend this function to the rest of \( X \), and prove the claims for that extension.

To simplify the proof, we will henceforth focus on a particular \( x_c \), and show how to extend \( \mathcal{T}_{\text{cpwa}} \) from \( B(x_c; \rho \cdot \eta/2) \) to the “gaps” between it and each of the neighboring balls, \( B(x'_c; \rho \cdot \eta/2) \) for \( x'_c \in N(x_c) \). To further simplify the proof, we define here two additional pieces of notation. First, for each \( x_c \in X_C \) and each \( k \in \{1, \ldots, n\} \) define a function \( \omega_k^{(x_c)} \) as follows:
\[
\omega_k^{(x_c)} : \begin{cases} -1,0,1^+ \rightarrow 2^R \\
0 \mapsto [\pi_k(x_c) - \rho/2, \pi_k(x_c) + \rho/2] \\
+1 \mapsto [\pi_k(x_c) + \rho/2, \pi_k(x_c) + \rho/2 + (1 - \rho/2)] \\
-1 \mapsto [\pi_k(x_c) - \rho/2 - (1 - \rho/2), \pi_k(x_c) - \rho/2].
\end{cases}
\]
Then, define the function:
\[
\mathcal{R}(x_c) : (-1,0,1)^n \rightarrow 2^{R^n}
\]
\[
\mathcal{R}(x_c) : t \mapsto \omega_1^{(x_c)}(\pi_1(t)) \times \omega_2^{(x_c)}(\pi_2(t)) \times \cdots \times \omega_n^{(x_c)}(\pi_n(t)),
\]
and let \( 0 \triangleq (0, \ldots, 0) \in \{-1,0,1\}^n \). Also, define \( \dim(i) \) as the number of non-zero elements in \( i \).

Now let \( x_c \in X_C \) be fixed. Using the above notation, the ball \( B(x_c; \rho \cdot \eta/2) \) is given by:
\[
B(x_c; \rho \cdot \eta/2) = \mathcal{R}(x_c)(0). \tag{45}
\]
Similarly each of the “gaps” between \( \mathcal{R}(x_c)(0) \) and its neighbors, \( \mathcal{R}(x'_c) \) for \( x'_c \in N(x_c) \), are the hypercubes:
\[
\mathcal{R}(x'_c)(i) \text{ for } i \in \{-1,0,1\}^n \setminus \{0\}, \tag{46}
\]
and hence:
\[
\bigcup_{i \in \{-1,0,1\}^n \setminus 0} \mathcal{R}(x'_c)(i) = X \setminus \bigcup_{x'_c \in X_c} B(x'_c; \rho \cdot \eta/2). \tag{47}
\]
This notation is illustrated in two dimensions in Fig. 1.

The first step is to show that \( \mathcal{T}_{\text{cpwa}} \) can be extended from the constant-valued region, \( \mathcal{R}(x_c)(0) \), to each of its neighbors, \( \mathcal{R}(x'_c) \), in a consistent way as a CPWA. To do this, first note that \( \mathcal{R}(x'_c)(0) \) has \( 2^n \) neighboring regions with indices \( i' \in \{-1,0,1\}^n \), and of these regions intersects a different \( \mathcal{R}(x'_c)(0) \) for \( x'_c \in N(x_c) \) at each corner, but is otherwise disjoint from them. Thus, Lemma 3 can be used to define a CPWA on each such \( \mathcal{R}(x'_c)(i') \) in a way that is consistent with the definition of \( \mathcal{T}_{\text{cpwa}} \) on the \( \mathcal{R}(x_c)(0) \). These definitions are also consistent with each other, since these regions are disjoint. Moreover, this procedure yields the same extension when started from \( x_c' \in N(x_c) \) instead of \( x_c \) (by the symmetric way that Lemma 3 is proved). Thus, it remains only to define \( \mathcal{T}_{\text{cpwa}} \) on regions with indices of the form \( \iota' \in \{-1,0,1\}^n \setminus \{-1,1\}^n \cup \{0\} \). However, each such \( \mathcal{R}(x'_c)(i') \) intersects \( 2^n \cdot \dim(i') \) regions with indices of the form \( \iota'' \in \{-1,1\}^n \), and each of these intersections is a \( \dim(i'') \) face of the corresponding \( \mathcal{R}(x'_c)(i') \). But on each such \( \dim(i'') \) face, \( \mathcal{T}_{\text{cpwa}} \) is defined and agrees with \( \Gamma_{\dim(i'')} \) from the construction in Lemma 3. Finally, since \( \Gamma_{\dim(i'')} \) (and hence \( \mathcal{T}_{\text{cpwa}} \)) is identical up to isomorphism on each of these \( \dim(i'') \) faces, \( \mathcal{T}_{\text{cpwa}} \) can be extended on to \( \mathcal{R}(x'_c)(i'') \) by isomorphism between the \( \dim(i'') \) nonzero indices, and \( \mathcal{T}_{\text{cpwa}} \) as defined on one of the \( \dim(i'') \) faces of \( \mathcal{R}(x'_c)(i') \). Finally, the symmetry of this procedure and Lemma 3 ensures that this assignment will be consistent when starting from some \( x'_c \in N(x_c) \) instead of \( x_c \).

Next, we show that for this \( \mathcal{T}_{\text{cpwa}} \), \( \| \mathcal{T}_{\text{cpwa}} - \Psi \| \leq \mu/3 \). This largely follows from the interpolation property proven in Lemma 3. In particular, on some \( \mathcal{R}(x_c)(i) \), \( \mathcal{T}_{\text{cpwa}} \) takes exactly the same values as some \( \Gamma_{\dim(i)} \) constructed according to Lemma 3, where the interpolation happens between \( \dim(i) \) points in \( V \triangleq \{ \Psi(x'_c) | x'_c \in N(x_c) \cup \{x_c\} \} \).

\[
\forall x \in \mathcal{R}(x_c)(i) \quad \min_{y \in V} \Psi(y) \leq \mathcal{T}_{\text{cpwa}}(x) \leq \max_{y \in V} \Psi(y). \tag{48}
\]
Let \( x \in \mathcal{R}(x_c)(i) \) be fixed temporarily, and suppose that \( \mathcal{T}_{\text{cpwa}}(x) - \Psi(x) \geq 0 \) and \( \max_{y \in V} \Psi(y) - \Psi(x) \geq 0 \). Then:
\[
| \mathcal{T}_{\text{cpwa}}(x) - \Psi(x) | = \mathcal{T}_{\text{cpwa}}(x) - \Psi(x) \\
\leq \max_{y \in V} \Psi(y) - \Psi(x) = \max_{y \in V} \Psi(y) - \Psi(x) \leq \frac{\mu}{3} \tag{49}
\]
where the last inequality follows from our choice of \( \eta \) from Proposition 1, since \( |y - x| \leq 2\eta \) for all \( y \in V \). The other cases can be considered as necessary, and they lead to the same conclusion. Hence, we conclude \( \| \mathcal{T}_{\text{cpwa}} - \Psi \|_\infty \leq \mu/3 \), since our choice of center \( x_c \) and \( i \) was arbitrary.

Now we just need to (over)-count the number of linear regions needed in the extension \( \mathcal{T}_{\text{cpwa}} \). This too will follow from the construction in Lemma 3. Note that on each \( \mathcal{R}(x_c)(i) \), \( \mathcal{T}_{\text{cpwa}} \) has the same number of linear regions as some \( \Gamma_{\dim(i)} \) that was constructed by Lemma 3, which by the same lemma has \( 2^{\dim(i) - 1} \cdot \dim(i)! \) regions. Thus, we
count at most:
\[ \sum_{k=1}^{n} \frac{n!}{k!} \cdot 2^k \cdot 2^{k-1} \cdot k! = n! \sum_{k=1}^{n} \frac{2^{2k-1}}{(n-k)!} \]  \hspace{1cm} (50)

linear regions. Finally, since we need this many regions for the neighboring regions of a single grid square, we obtain an upper bound for the total number of regions by multiplying (50) by the number of grid squares in the partition, \( \left( \frac{\text{ext}(X)}{\eta} \right)^n \) (then by the \( m \), in the multi-dimensional output case).

VII. NUMERICAL RESULTS

We illustrate the results in this paper on an inverted pendulum described by the following model:

\[ f(x_1, x_2, u) = \left[ \frac{g}{l} \sin(x_1) - \frac{h}{m} x_2 - \frac{1}{m} \cos(x_1) u \right], \]

where \( x_1 \) is the angular position, \( x_2 \) is the angular velocity, and control input \( u \) is the torque applied on the point mass. The parameters are the rod mass, \( m \); the rod length, \( l \); the (dimensionless) coefficient of rotational friction, \( h \); and the acceleration due to gravity, \( g \). For the purposes of our experiments, we considered a subset of the state/control space specified by: \( x_1 \in [-1, 1], \ x_2 \in [-1, 1] \) and \( u \in [-6, 6] \). Furthermore, we considered model parameters: \( m = 0.5 \text{ kg}; \ l = 0.5 \text{ m}; \ h = 2; \) and \( g = 9.8 \text{ N/kg} \). Then for different choices of the design parameters \( \mu \), we obtain the following sizes \( N \) for the corresponding TLL-NN architecture along with the corresponding \( \tau \), \( \eta \) and the \( \delta \) that is required for the specification satisfaction:

| \( \mu \) | \( \delta \)   | \( \tau \) | \( \eta \) | \( N \) |
|---------|-------------|-----------|-----------|--------|
| 0.35    | 0.8694      | 0.0098    | 0.583     | 235    |
| 0.3     | 0.5287      | 0.0083    | 0.5       | 320    |
| 0.25    | 0.3039      | 0.0069    | 0.417     | 460    |
| 0.2     | 0.1610      | 0.0056    | 0.334     | 720    |
| 0.15    | 0.0749      | 0.0042    | 0.25      | 1280   |
| 0.1     | 0.0275      | 0.0028    | 0.167     | 2880   |

In the sequel, we will show the control performance of a TLL-NN architecture with 400 local linear region. While there are a number of techniques that can be used to train the resulting NN, for the sake of simplicity, we utilize Imitation learning where the NN is trained in a supervised fashion from data collected from an expert controller. In particular, we designed an expert controller that stabilizes the inverted pendulum; we chose to use Pessa [15] to design our expert using the parameter values specified above. In particular, we tasked Pessa to design a zero-order-hold controller that stabilizes the inverted pendulum in a subset \( X_{\text{spec}} = [-1, 1] \times [-0.5, 0.5] \); that is the controller should transfer the state of the system to this specified set and keep it there for all time thereafter. From this expert controller, we collected 8400 data points of state-action pairs; this data was obtained by uniformly sampling the state space. We then used Keras [16] to train the TLL NN using this data. Finally, we simulated the motion of the inverted pendulum using this TLL NN controller. Shown in Fig. 2 and Fig. 3 are the state and control trajectories for this controller starting from initial state \([0.7, 0.5]^T \) and \([-0.4, 1.0]^T \), respectively. In both, the TLL NN controller met the same specification that was used to design the expert.

REFERENCES

[1] M. Bojariski, D. Del Testa, D. Dworakowski, B. Birner, 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] F. Pedregosa, “Hyperparameter optimization with approximate gradients,” arXiv:1602.02355, 2016.
[3] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” Journal of Machine Learning Research, vol. 13, no. Feb, pp. 281–305, 2012.
[4] S. Paul, V. Kurin, and S. Whiteson, “Fast efficient hyperparameter tuning for policy gradients,” arXiv:1902.06583, 2019.
[5] B. Baker, O. Gupta, N. Naik, and R. Raskar, “Designing neural network architectures using reinforcement learning,” arXiv:1611.02167, 2016.
[6] Y. Quanming, W. Mengshuo, J. E. Hugo, G. Isabelle, H. Yi-Qi, L. Yu-Feng, T. Wei-Wei, Y. Qiang, and Y. Yang, “Taking human out of learning applications: A survey on automated machine learning,” arXiv:1810.13306, 2018.
[7] 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, NY, USA, 2020.
[8] H. K. Khalil, Nonlinear Systems. Pearson, Third ed., 2001.
[9] M. Zamani, G. Pola, M. Mazo, and P. Tabuada, “Symbolic Models for Nonlinear Control Systems Without Stability Assumptions,” IEEE Transactions on Automatic Control, vol. 57, no. 7, 2012.
[10] V. Kurtz, P. M. Wensing, and H. Lin, “Robust Approximate Simulation for Hierarchical Control of Linear Systems under Disturbances,” 2020.
[11] K. Mallik, A.-K. Schmuck, S. Soudjani, and R. Majumdar, “Compositional Synthesis of Finite-State Abstractions,” IEEE Transactions on Automatic Control, vol. 64, no. 6, pp. 2629–2636, 2019.
[12] G. Pola, A. Girard, and P. Tabuada, “Approximately bisimilar symbolic models for nonlinear control systems,” Automatica, vol. 44, no. 10, pp. 2508–2516, 2008.
[13] P. Tabuada, Verification and Control of Hybrid Systems: A Symbolic Approach. Springer US, 2009.
[14] J. Ferlez, X. Sun, and Y. Shoukry, “Two-level lattice network architectures for control of nonlinear systems,” https://rpsil.eng.uci.edu/files/2020/03/NNArchitectureTechnicalNote.pdf, 2020.
[15] M. Mazo, A. Davitian, and P. Tabuada, “PESSOA: A tool for embedded controller synthesis,” in Proceedings of the 22nd International Conference on Computer Aided Verification, CAV’10, pp. 566–569, Springer-Verlag, 2010.
[16] F. Chollet et al., “Keras.” https://keras.io, 2015.
A. Proof of Proposition 1

Proof. Choose \( \eta = \eta(\mu) \leq \frac{\mu}{2\cdot K_{\text{cont}}} \) and use Lipschitz continuity of \( \Psi \).

B. Proof of Proposition 2

Proof. Since \( f \) is continuous on the compact set \( X \times U \), it is bounded, and let \( K \) be this bound as stated in Definition 9. Then by (3) we have

\[
\|x - \zeta_{xT}(t)\| = \left\| \int_0^t f(\zeta_{xT}(\sigma), \Psi(\zeta_{xT}(\sigma)))d\sigma \right\| \leq \int_0^t \|f(\zeta_{xT}(\sigma), \Psi(\zeta_{xT}(\sigma)))\|d\sigma \leq \int_0^t Kd\sigma = Kt. \tag{51}
\]

Hence, choose \( \tau = \tau(\mu) \leq \frac{\eta(\mu)}{K} \) and the conclusion follows.

C. Proof of Proposition 3

Proof. By the triangle inequality, we have:

\[
\begin{align*}
\|\Psi(\zeta_{xT}(t)) - \zeta_{x\Psi}(t)\| &= \|\Psi(\zeta_{xT}(t)) - \Psi(\zeta_{xT}(t)) + \Psi(\zeta_{xT}(t)) - \zeta_{x\Psi}(t)\|
\leq & \|\Psi(\zeta_{xT}(t)) - \Psi(\zeta_{xT}(t))\| + \|\Psi(\zeta_{xT}(t)) - \zeta_{x\Psi}(t)\|
\leq & \|\Psi(\zeta_{xT}(t)) - \zeta_{x\Psi}(t)\| + \|\Psi(\zeta_{xT}(t)) - \Psi(x)\| + \|\Psi(x) - \zeta_{x\Psi}(t)\|. \tag{54}
\end{align*}
\]

The first term in (54) is bounded by \( \|\Psi - \Psi\|_{\infty} \). Now consider the second term. By Proposition 2, \( \|\zeta_{xT}(t) - x\| \leq \eta \); thus, by Proposition 1 we conclude that \( \|\Psi(\zeta_{xT}(t)) - \Psi(x)\| \leq \mu/3 \). The final term is likewise bounded by \( \mu/3 \) for the same reasons. Thus, the conclusion follows.

D. Proof of Proposition 4

Proof. By definition and the properties of the integral, we have:

\[
\begin{align*}
\|\zeta_{xT}(t) - \zeta_{x\Psi}(t)\| &= \left\| \int_0^t f(\zeta_{xT}(\sigma), \Psi(\zeta_{xT}(\sigma)))d\sigma \right\|
\leq & \left\| \int_0^t f(\zeta_{xT}(\sigma), \Psi(\zeta_{xT}(\sigma)))d\sigma - f(\zeta_{x\Psi}(t), \Psi(\zeta_{x\Psi}(t)))d\sigma \right\|
\leq & \int_0^t K_x \|\zeta_{xT}(\sigma) - \zeta_{x\Psi}(\sigma)\|
\leq & \int_0^t K_x (\|\zeta_{xT}(\sigma) - \zeta_{x\Psi}(\sigma)\| + K_u \|\Psi(\zeta_{xT}(\sigma)) - \Psi(\zeta_{x\Psi}(\sigma))\|d\sigma
\leq & \int_0^t K_x (\|\zeta_{xT}(\sigma) - \zeta_{x\Psi}(\sigma)\| + K_u \cdot \kappa d\sigma. \tag{55}
\end{align*}
\]

The claimed bound now follows directly from the Grönwall Inequality [8].