Verifying Closed-Loop Contractivity of Learning-Based Controllers via Partitioning

Alexander Davydov

Abstract— We address the problem of verifying closed-loop contraction in nonlinear control systems whose controller and contraction metric are both parameterized by neural networks. By leveraging interval analysis and interval bound propagation, we derive a tractable and scalable sufficient condition closed-loop contractivity that reduces to checking that the dominant eigenvalue of a symmetric Metzler matrix is nonpositive. We combine this sufficient condition with a domain partitioning strategy to integrate this sufficient condition into training. The proposed approach is validated on an inverted pendulum system, demonstrating the ability to learn neural network controllers and contraction metrics that provably satisfy the contraction condition.

I. Introduction

Problem motivation. Designing and verifying controllers for nonlinear systems using first-principles tools is often computationally intractable. As a result of these challenges, it has become increasingly common to employ neural networks as feedback controllers and to train them to promote closed-loop stability, e.g., by penalizing instability at sampled states [14]. However, these empirical training techniques typically do not guarantee that the learned controller will generalize closed-loop stability properties to regions of state space away from these data samples.

To address this gap, a growing body of research has begun exploring formal verification of (i) Lyapunov stability [1], [16], (ii) barrier functions for safety guarantees [12], and (iii) differential contraction properties [9]. While these works offer promising approaches toward theoretical guarantees, they generally draw upon computationally expensive primitives including satisfiability modulo theories (SMT) or branch and bound (BnB) algorithms, both of which are known to face significant scalability challenges in higher-dimensional or more complex systems.

In this work, we focus on learning both a feedback controller and a neural contraction metric that jointly guarantee closed-loop contraction in a prescribed region of interest. Our approach builds on scalable sufficient conditions for contraction which avoid the computational burden of using SMT or BnB-based verification. Contraction analysis is particularly appealing because establishing contraction yields strong corollaries including input-to-state stability, robustness to unmodeled dynamics, and robustness to stochastic disturbances [15], [3].

Related work. One related line of research is focused on learning contracting dynamical systems from data [2], [8]. In these works, the task is to learn a dynamical system

Department of Mechanical Engineering and Ken Kennedy Institute, Rice University, Houston, TX, USA. davydov@rice.edu

$\dot{x} = f(x)$ which is guaranteed to be contracting everywhere in state space. In comparison, in this work, we are given a known control system $\dot{x} = f(x) + Bu$ and instead are trying to learn a feedback controller and contraction metric that ensures contraction in a subset of the state space.

The work that is most closely related to ours is [17] where the authors use the theory of control contraction metrics [11] to ensure closed-loop contraction with a neural network-based controller and neural contraction metric. The authors use a Gershgorin-type sufficient condition in Theorem 2 to provide a finite-dimensional scalar condition as a sufficient condition for contraction. Moreover, in Proposition 1, they parametrize the neural contraction metric to enforce certain Killing vector field conditions which appear in the strong control contraction metric conditions [11, Section III.A].

Contributions. We develop a scalable approach for verifying contraction of nonlinear control systems with neural network controllers and neural contraction metrics. First, we introduce a Metzler majorant sufficient condition that reduces the infinite-dimensional contraction inequality to checking the dominant eigenvalue of a single symmetric Metzler matrix. This sharpens the Gershgorin-type bound of [17] and circumvents expensive SMT or BnB tools.

Second, we combine interval bound propagation, interval arithmetic, and interval Jacobian bounds to construct over-approximations of all terms in the contraction inequality.

We additionally propose a domain-partitioning strategy that reduces conservatism and yields verifiable certificates on large regions of the state space. Finally, we integrate our verification procedure into training, enabling direct learning of closed-loop contracting controllers. A numerical experiment on the control of an inverted pendulum illustrates the effectiveness of the proposed approach.

In this extended technical report, we present proofs of all results and include more remarks throughout.

II. Preliminaries

We collect the necessary background on contraction theory and interval methods used throughout the paper.

Contraction theory. We consider the dynamical system

$$\dot{x} = f(x) \qquad (1)$$

where $x(t) \in \mathbb{R}^n$ for all t and $f : \mathbb{R}^n \to \mathbb{R}^n$ is continuously differentiable. Contraction aims to address the question of when all solution trajectories of (1) converge to one another exponentially quickly in some metric. For arbitrary metrics, this question is intractable. As a result, it is common to restrict oneself to metrics induced by Riemannian metrics

on $\mathbb{R}^n$. To be specific, a matrix-valued mapping $M : \mathbb{R}^n \to \mathbb{R}^{n \times n}$ satisfying $M(x) = M(x)^\top \succ 0$ for all x induces a Riemannian metric on $\mathbb{R}^n$ and the distance between two points $x, y \in \mathbb{R}^n$ is defined by the length of the geodesic connecting x and y, see [13] for more details.

One of the key results in contraction theory is that contraction with respect to the Riemannian metric can be established via a matrix inequality in terms of M, Df, and $\dot{M}_f$, which we highlight below.

Lemma 1 (Contraction inequality [10]). *The dynamical system* (1) *is* contracting *with rate* $c > 0$ *if there exists a continuously-differentiable matrix-valued map* $M : \mathbb{R}^n \to \mathbb{R}^{n \times n}$ *and two constants* $a_0, a_1 > 0$ *such that for all* $x \in \mathbb{R}^n$, $M(x) = M(x)^\top$ *and* $a_0 I_n \preceq M(x) \preceq a_1 I_n$ *and additionally satisfies for all* x

$$M(x)Df(x) + Df(x)^\top M(x) + \dot{M}_f(x) \preceq -2cM(x). \quad (2)$$

$\dot{M}_f$ denotes the Lie derivative of M along the vector field f and it is defined componentwise by $[\dot{M}_f(x)]_{ij} = \nabla M_{ij}(x)^\top f(x)$. In essence, Lemma 1 ensures that for any pair of initial conditions, $x_0, y_0 \in \mathbb{R}^n$, the solutions $x(t), y(t)$ of (1) initialized at these initial conditions satisfy $d(x(t), y(t)) \leq e^{-ct} d(x_0, y_0)$, where d is the geodesic distance induced by M.

Interval arithmetic. For two vectors $\underline{x}, \overline{x} \in \mathbb{R}^n$, we will use the partial order $\underline{x} \leq \overline{x}$ if and only if $\underline{x}_i \leq \overline{x}_i$ for all $i \in \{1, \ldots, n\}$. We use an analogous partial order on the set of matrices $\mathbb{R}^{n \times m}$ (not to be confused with the partial order $\preceq$ on the set of symmetric matrices). For two vectors (or matrices) $\underline{x} \leq \overline{x}$, we define the hyperrectangle $[\underline{x}, \overline{x}] := \{x \in \mathbb{R}^n \mid \underline{x} \leq x \leq \overline{x}\}$. We recall the following useful facts from interval arithmetic:

Lemma 2. *Suppose* $a \in [\underline{a}, \overline{a}] \subseteq \mathbb{R}, b \in [\underline{b}, \overline{b}] \subseteq \mathbb{R}, A \in [\underline{A}, \overline{A}] \subseteq \mathbb{R}^{n \times k}$, *and* $B \in [\underline{B}, \overline{B}] \subseteq \mathbb{R}^{k \times m}$. *Then*

(i) $a + b \in [\underline{a} + \underline{b}, \overline{a} + \overline{b}]$;

(ii) $ab \in [\min\{\underline{a}\underline{b}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b}\}, \max\{\underline{a}\underline{b}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b}\}]$;

(iii) $[AB]_{ij} \in \big[\sum_{p=1}^{k} \min\{\underline{a}_{ip}\underline{b}_{pj}, \underline{a}_{ip}\overline{b}_{pj}, \overline{a}_{ip}\underline{b}_{pj}, \overline{a}_{ip}\overline{b}_{pj}\}, \\ \sum_{p=1}^{k} \max\{\underline{a}_{ip}\underline{b}_{pj}, \underline{a}_{ip}\overline{b}_{pj}, \overline{a}_{ip}\underline{b}_{pj}, \overline{a}_{ip}\overline{b}_{pj}\} \big]$.

We will frequently use the result in Lemma 2(iii). For succinctness, we will write $\mathrm{IMM}([\underline{A}, \overline{A}], [\underline{B}, \overline{B}])$ to denote the mappings which outputs the intervals from Lemma 2(iii).

We also define the *Metzler majorant* of a matrix $A \in \mathbb{R}^{n \times n}$ by $[\lceil A \rceil_{\mathrm{Mzr}}]_{ij} = |a_{ij}|$ if $i \neq j$ and $[\lceil A \rceil_{\mathrm{Mzr}}]_{ii} = a_{ii}$. The name Metzler majorant alludes to the fact that $\lceil A \rceil_{\mathrm{Mzr}}$ is Metzler even if A is not (off-diagonal elements are nonnegative) and that $A \leq \lceil A \rceil_{\mathrm{Mzr}}$.

Interval bound propagation. To construct overapproximations of neural network outputs, we will leverage interval bound propagation (IBP) [6] which provides constant lower and upper bounds on the intermediate and output layers of a feedforward neural network. To be precise, given a neural network $N : \mathbb{R}^n \to \mathbb{R}^m$ parametrized by

$$N(x) = \xi_{L+1} \quad (3a)$$
$$\xi_{k+1} = W_k z_k + b_k, \quad z_0 = x, \quad (3b)$$
$$z_{k+1} = \sigma(\xi_{k+1}) \quad (3c)$$

where W_i are weight matrices, b_i are bias vectors, the ξ_i are pre-activation variables, and z_i are post-activation variables.

Given a set of possible inputs in a hyperrectangle $\mathcal{X} = [\underline{x}, \overline{x}]$, IBP generates a hyperrectangular overapproximation of the set $N(\mathcal{X}) := \{N(x) \mid x \in \mathcal{X}\}$. To be specific, letting $[W]^+ = \mathrm{ReLU}(W)$ and $[W]^- = W - [W]^+$ and assuming that σ acts entrywise and is monotone increasing, IBP provides the outputs

$$\underline{\xi}_{k+1} = [W_k]^+ \underline{z}_k + [W_k]^- \overline{z}_k + b_k \quad (4a)$$
$$\overline{\xi}_{k+1} = [W_k]^+ \overline{z}_k + [W_k]^- \underline{z}_k + b_k \quad (4b)$$
$$\underline{z}_{k+1} = \sigma(\underline{\xi}_{k+1}) \quad (4c)$$
$$\overline{z}_{k+1} = \sigma(\overline{\xi}_{k+1}) \quad (4d)$$

with the initial bounds $\underline{z}_0 = \underline{x}$ and $\overline{z}_0 = \overline{x}$. Then IBP provides the estimate $N(\mathcal{X}) \subseteq [\underline{\xi}_{L+1}, \overline{\xi}_{L+1}]$. Notably, when $\underline{x} = \overline{x} = x$, then $\underline{\xi}_{L+1} = \overline{\xi}_{L+1} = \xi_{L+1}$, so we can see that as the set of possible inputs shrinks, the IBP overapproximation of the output set shrinks as well.

III. TECHNICAL RESULTS

Verification of infinite-dimensional matrix inequalities. The main challenge in verification of contraction of a nonlinear system is that the condition (2) must be verified at infinitely many x. Namely, the condition (2) is an infinite-dimensional linear matrix inequality in M. To abstract away the details of the specific form of the contraction inequality (2), we will investigate arbitrary matrix inequalities of the form $\mathcal{A}(x) \preceq 0$ where $\mathcal{A} : \mathcal{X} \to \mathbb{R}^{n \times n}$ is a matrix-valued mapping satisfying $\mathcal{A}(x) = \mathcal{A}(x)^\top$ for all $x \in \mathcal{X}$. Our main theoretical result is the following one.

Theorem 3. *Consider a matrix-valued mapping* $\mathcal{A} : \mathcal{X} \to \mathbb{R}^{n \times n}$ *satisfying* $\mathcal{A}(x) = \mathcal{A}(x)^\top$ *for all* x. *Suppose there exists a symmetric* $G \in \mathbb{R}^{n \times n}$ *such that* $G \geq \lceil \mathcal{A}(x) \rceil_{\mathrm{Mzr}}$ *for all* $x \in \mathcal{X}$. *If* $\lambda_{\max}(G) \leq 0$, *then* $\mathcal{A}(x) \preceq 0$ *for all* $x \in \mathcal{X}$.

Proof. Let $v \in \mathbb{R}^n$ and let $x \in \mathcal{X}$. Then

$$
\begin{aligned}
v^\top \mathcal{A}(x) v &= \sum_{i=1}^{n} v_i^2 \mathcal{A}(x)_{ii} + \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} v_i v_j \mathcal{A}(x)_{ij} \\
&\leq \sum_{i=1}^{n} v_i^2 \mathcal{A}(x)_{ii} + \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} |v_i||v_j||\mathcal{A}(x)_{ij}| \\
&\leq \sum_{i=1}^{n} v_i^2 G_{ii} + \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} |v_i||v_j|G_{ij} \\
&= |v|^\top G |v| \leq 0,
\end{aligned}
$$

where we denote by $|v|$ the entrywise absolute value of v. We have used that $G \geq \lceil \mathcal{A}(x) \rceil_{\mathrm{Mzr}}$ and that $\lambda_{\max}(G) \leq 0$. $\quad \square$

Theorem 3 demonstrates that a sufficient condition for the infinite-dimensional matrix inequality $\mathcal{A}(x) \preceq 0$ can be verified by finding the dominant eigenvalue of a constant Metzler matrix, which is much more tractable. Of course, there are even constant matrices $\mathcal{A}$ which are negative semidefinite but whose Metzler majorant is not. Therefore, the sufficient condition in Theorem 3 trades off computational efficiency for tightness.

Interval bounds on Jacobians of neural networks. Consider a feedforward neural network $N : \mathbb{R}^n \to \mathbb{R}^m$ given by (3a). Since the contraction condition (2) involves Jacobians of mappings, it will be useful to have bounds on $DN(x)$ for $x \in \mathcal{X}$. First note that because of the feedforward structure of N, $DN(x)$ admits the form

$$DN(x) = W_L J_{L-1}(x) W_{L-1} J_{L-2}(x) \dots J_1(x) W_1 \quad (5)$$

where $J_i(x) = D\sigma(\xi_i)$ where $D\sigma(x)$ is the diagonal matrix with j-th diagonal entry equal to $\sigma'(x_j)$ and ξ_i is given in (3b). Notably, to attain a bound on $DN(x)$, we simply need to bound the product of constant matrices (the W_i) and diagonal matrices (the $J_i(x)$).

To find upper and lower bounds on each $J_i(x)$, we will make the following simplifying assumption:

Assumption 1. *Suppose σ is twice continuously differentiable and $\sigma''(x) \geq 0$ for all x.*

In addition to monotonicity of σ, Assumption 1 imposes convexity on σ as well. Common activation functions which satisfy this assumption include the softplus activation, $\sigma(x) = \log(1 + \exp(x))$ and the smooth leaky ReLU, $\sigma(x) = \alpha x + (1 - \alpha) \log(1 + \exp(x))$ with $\alpha \in (0,1)$. Under Assumption 1, we get the following bounds on each $J_i(x)$:

Lemma 4. *Suppose Assumption 1 holds. Then $\underline{J}_i \leq J_i(x) \leq \overline{J}_i$ where*

$$\underline{J}_i = D\sigma(\underline{\xi}_i), \qquad \overline{J}_i = D\sigma(\overline{\xi}_i) \quad (6)$$

and $\underline{\xi}_i, \overline{\xi}_i$ are given in the IBP algorithm (4a), (4b).

Lemma 4 highlights how the IBP bounds for pre-activation variables immediately provide us with bounds on the $J_i(x)$. These bounds improve on those in [17] where they instead use $\underline{J}_i = aI, \overline{J}_i = bI$ where $0 < a \leq \sigma'(x) \leq b$ for all x.

With these bounds in hand and the fact that the W_i are constant, we can recursively apply Lemma 2(iii) to get interval bounds on the Jacobian of the neural network. The full algorithm is presented in Algorithm 1.

Remark 5. *In the case that Assumption 1 does not hold, then Lemma 4 and Algorithm 1 can be modified as follows. Denote by $\underline{\xi}_{i,j}, \overline{\xi}_{i,j}$ the j-th components of $\underline{\xi}_i$ and $\overline{\xi}_i$, respectively. Then we can define*

$$[\underline{J}_i]_{jj} = \min_{y \in [\underline{\xi}_{i,j}, \overline{\xi}_{i,j}]} \sigma'(y), \qquad [\overline{J}_i]_{jj} = \max_{y \in [\underline{\xi}_{i,j}, \overline{\xi}_{i,j}]} \sigma'(y), \quad (7)$$

and zero on the off-diagonal entries. It is easy to see that these are lower and upper bounds on each $J_i(x)$. Under Assumption 1, these minimization and maximization

---

Algorithm 1 Output bounds on $DN(x)$ for $x \in [\underline{x}, \overline{x}]$

1: Require $x \in [\underline{x}, \overline{x}]$, N feedforward neural network
2: Call IBP$(N; \underline{x}, \overline{x})$ to get bounds $\xi_i \in [\underline{\xi}_i, \overline{\xi}_i]$ for $x \in \mathcal{X}$ and $i \in \{1, \dots, L+1\}$
3: *# Create running variable that tracks matrix bounds as we multiply right to left*
4: Initialize matrix interval $[\underline{P}, \overline{P}]$ where $\underline{P} = \overline{P} = I_n$
5: **for** $i = 1$ to $L - 1$ **do**
6:      *# Multiply on the left by constant matrix W_i*
7:      $[\underline{P}, \overline{P}] \leftarrow \text{IMM}([W_i, W_i], [\underline{P}, \overline{P}])$
8:      *# Get diagonal bounds $\underline{J}_i, \overline{J}_i$*
9:      $\underline{J}_i \leftarrow D\sigma(\underline{\xi}_i), \overline{J}_i \leftarrow D\sigma(\overline{\xi}_i)$
10:      *# Multiply on the left by interval diagonal matrix J_i*
11:      $[\underline{P}, \overline{P}] \leftarrow \text{IMM}([\underline{J}_i, \overline{J}_i], [\underline{P}, \overline{P}])$
12: **end for**
13: *# Multiply on the left by output matrix W_L*
14: $[\underline{P}, \overline{P}] \leftarrow \text{IMM}([W_L, W_L], [\underline{P}, \overline{P}])$
15: **return** $[\underline{P}, \overline{P}]$; optionally **return** $\{[\underline{\xi}_i, \overline{\xi}_i]\}_{i=1}^{L+1}$

---

problems have explicit closed-form expressions which are presented in Lemma 4.

Suppose we want to provide bounds on $BDN(x)$ given that N is a feedforward neural network, B is a matrix and $x \in [\underline{x}, \overline{x}]$. Then we could get the bounds by calling Algorithm 1 to get bounds on $DN(x)$ and then use Lemma 2(iii) to get bounds on the product between B and $DN(x)$. Alternatively, we could define a new feedforward neural network $\tilde{N}$ which shares all the same weights and biases except for output weight BW_{N-1} and output bias Bb_{N-1}. It is clear that $\tilde{N}(x) = BN(x)$ and that $D\tilde{N}(x) = BDN(x)$. This means that we can run Algorithm 1 on the neural network $\tilde{N}$ to get bounds on $BDN(x)$. In the IBP literature this approach is called *eliding the last layer* and it has been shown to produce tighter estimates than using Lemma 2(iii) [6]. Note that this procedure is only possible if the last layer is affine.

IV. CONTRACTION VERIFICATION

Consider the control system

$$\dot{x} = f(x) + Bu, \quad (8)$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $B \in \mathbb{R}^{n \times m}$. Suppose we train a feedforward neural network feedback controller $u : \mathbb{R}^n \to \mathbb{R}^m$ and a *neural contraction metric* $M : \mathbb{R}^n \to \mathbb{R}^{n \times n}$ for (8). We aim to verify whether the contraction condition (2) is satisfied for the closed-loop dynamics $f_{\text{CL}}(x) = f(x) + Bu(x)$ in a hyperrectangular region of interest in the state space, $\mathcal{X} := [\underline{x}, \overline{x}]$.

Define the matrix-valued mapping $\mathcal{A} : \mathbb{R}^n \to \mathbb{R}^{n \times n}$ by $\mathcal{A}(x) = M(x)Df_{\text{CL}}(x) + Df_{\text{CL}}(x)^\top M(x) + \dot{M}_{f_{CL}}(x) + 2cM(x)$. In view of Definition 1, contraction with rate $c > 0$ with respect to metric induced by M is guaranteed provided $\mathcal{A}(x) \preceq 0$. For simplicity, going forward, we will take $c = 0$, but the following analysis will work just as well with $c > 0$.

We can decompose $\mathcal{A}$ into the component coming from the dynamics and the one from the controller, i.e.,

$$\mathcal{A}(x) = \mathcal{A}_f(x) + \mathcal{A}_u(x), \qquad \text{where}$$
$$\mathcal{A}_f(x) = M(x)Df(x) + Df(x)^\top M(x) + \dot{M}_f(x)$$
$$\mathcal{A}_u(x) = M(x)BDu(x) + Du^\top B^\top M(x) + \dot{M}_{Bu}(x)$$

In the end, our goal will be to construct a constant matrix G for which $G \geq \lceil \mathcal{A}(x) \rceil_{\text{Mzr}}$ for all $x \in \mathcal{X}$. To this end, we make the following standing assumption on the dynamics

Assumption 2. *For all $\underline{x} \leq \overline{x}$, there exist $\underline{f}(\underline{x},\overline{x}), \overline{f}(\underline{x},\overline{x}) \in \mathbb{R}^n$ and $\underline{Df}(\underline{x},\overline{x}), \overline{Df}(\underline{x},\overline{x}) \in \mathbb{R}^{n \times n}$ such that for all $x \in \mathcal{X}$*

$$\underline{f}(\underline{x},\overline{x}) \leq f(x) \leq \overline{f}(\underline{x},\overline{x}) \qquad (9)$$
$$\underline{Df}(\underline{x},\overline{x}) \leq Df(x) \leq \overline{Df}(\underline{x},\overline{x}). \qquad (10)$$

For simple dynamical systems, these bounds may be analytically computable in terms of $\underline{x},\overline{x}$, e.g., as we will do in Section VI. Otherwise, for some more complex dynamical systems, dedicated software for interval arithmetic may be used, e.g., npinterval [7]. When the bounds $\underline{x}$ and $\overline{x}$ are clear from context, we will simply write $\underline{f},\overline{f},\underline{Df},\overline{Df}$.

Since u is a feedforward neural network, we can directly apply IBP and Algorithm 1 to attain bounds on $u(x)$ and $Du(x)$ for $x \in [\underline{x},\overline{x}]$. On the other hand, since M must be constrained to be positive definite for all inputs, we need to tailor IBP and its variation to effectively bound $M(x)$ and $\dot{M}_f(x)$, and $\dot{M}_{Bu}(x)$ in terms of the bounds $x \in [\underline{x},\overline{x}]$.

Neural contraction metric bounds. To enforce positive definiteness of $M(x)$ for all x, we use the same parametrization for the neural contraction metric as used in [14]. Namely, the contraction metric is parametrized via

$$M(x) = N(x)^\top N(x) + \varepsilon I_n, \qquad (11)$$

where $\varepsilon > 0$ is a small constant enforcing strict positive definiteness of M and $N : \mathbb{R}^n \to \mathbb{R}^{n \times n}$ is parametrized by a feedforward neural network where the output has been reshaped to be an $n \times n$ matrix.

In addition to bounds on $M(x)$ for $x \in \mathcal{X}$, for the computation of the terms of $\dot{M}_f$ and $\dot{M}_{Bu}$, we need estimates on each of the $\nabla M_{ij}(x)$ for $i,j \in \{1,\ldots,n\}$. Since $M_{ij}(x) = \varepsilon \delta_{ij} + \sum_{k=1}^{n} N_{ki}(x)N_{kj}(x)$, where δ_{ij} is the Kronecker delta, we can compute

$$\nabla M_{ij}(x) = DN_i(x)^\top N_j(x) + DN_j(x)^\top N_i(x), \qquad (12)$$

where $N_i(x)$ denotes the i-th column of the matrix $N(x)$ and $DN_i(x)$ is the Jacobian of the map N_i evaluated at x.

To get bounds on $M(x)$ and each of the $\nabla M_{ij}(x)$, we apply IBP to the network N, reshape the last-layer bounds to get matrix-valued bounds and then use the interval Jacobian procedure in Algorithm 1 to each column, $N_i(x)$. Combining these bounds using (12) and Lemma 2(iii) provides interval bounds on $\nabla M_{ij}(x)$.

Remark 6. *In certain cases, it may be sufficient to use a constant M instead of a state-varying one. In this case, bounds on M and ∇M_{ij} are immediate since M is constant and $\nabla M_{ij} = 0$ for all $i,j \in \{1,\ldots,n\}$.*

Putting all the bounds together. We are now prepared to combine all the bounds together to construct the matrix G.

First, given the bounds on $\nabla M_{ij}(x)$, we can get bounds on $\dot{M}_f(x)$ and $\dot{M}_{Bu}(x)$ by applying Lemma 2(iii) to the inner products between the interval matrices $[\underline{\nabla M}_{ij}, \overline{\nabla M}_{ij}]$ and either $[\underline{f},\overline{f}]$ or $[\underline{Bu},\overline{Bu}]$. These products give us bounds $\dot{M}_f(x) \in [\underline{\dot{M}_f}, \overline{\dot{M}_f}]$ and $\dot{M}_{Bu}(x) \in [\underline{\dot{M}_{Bu}}, \overline{\dot{M}_{Bu}}]$.

Finally by two more applications of Lemma 2(iii), we can construct bounds $M(x)Df(x) \in [L_f, U_f]$ and $M(x)BDu(x) \in [L_u, U_u]$. Recalling the definitions then of $\mathcal{A}_f$ and $\mathcal{A}_u$, we attain the bounds

$$\mathcal{A}_f(x) \in [L_f + L_f^\top + \underline{\dot{M}_f}, U_f + U_f^\top + \overline{\dot{M}_f}] =: [\underline{\mathcal{A}_f}, \overline{\mathcal{A}_f}]$$
$$\mathcal{A}_u(x) \in [L_u + L_u^\top + \underline{\dot{M}_{Bu}}, U_u + U_u^\top + \overline{\dot{M}_{Bu}}] =: [\underline{\mathcal{A}_u}, \overline{\mathcal{A}_u}]$$

Finally, $\mathcal{A}(x) \in [\underline{\mathcal{A}_f} + \underline{\mathcal{A}_u}, \overline{\mathcal{A}_f} + \overline{\mathcal{A}_u}] =: [\underline{\mathcal{A}}, \overline{\mathcal{A}}]$.

To apply Theorem 3, we need to construct a matrix G such that $\lceil \mathcal{A}(x) \rceil_{\text{Mzr}} \leq G$. We can do this using the bounds on $\mathcal{A}(x)$. We define the matrix one element at a time:

$$G_{ij} = \begin{cases} \max\{\overline{\mathcal{A}}_{ij}, -\underline{\mathcal{A}}_{ij}\}, & \text{if } i \neq j \\ \overline{\mathcal{A}}_{ij}, & \text{if } i = j \end{cases}. \qquad (13)$$

By construction, $\lceil \mathcal{A}(x) \rceil_{\text{Mzr}} \leq G$ for all $x \in \mathcal{X}$ by definition of $\underline{\mathcal{A}}, \overline{\mathcal{A}}$. Then we arrive at our main sufficient condition for verifying contraction on a hyperrectangular domain.

Corollary 7. *If $\lambda_{\max}(G) \leq 0$, then $\mathcal{A}(x) \preceq 0$ for all $x \in \mathcal{X}$. In particular, the closed-loop dynamics are contracting with respect to the neural contraction metric M in* (11).

Proof. First we note that by construction of M, we have that $M(x) \succeq \varepsilon I_n$. Moreover, by continuity of M and compactness of the domain $\mathcal{X}$, we also have that $M(x) \preceq R I_n$ for some $R > 0$. Moreover, since $\lambda_{\max}(G) \leq 0$ and $G \geq \lceil \mathcal{A}(x) \rceil_{\text{Mzr}}$ for all $x \in \mathcal{X}$, we also have that $\mathcal{A}(x) \preceq 0$ by Theorem 3. As a result, we verify contraction with respect to the neural contraction metric M. $\square$

We summarize the full algorithm from going from input bounds $x \in [\underline{x},\overline{x}]$ to the construction of G in Algorithm 2.

V. DOMAIN PARTITIONING AND TRAINING ALGORITHM

Domain partitioning. For the time being, we have assumed that $\mathcal{X} = [\underline{x},\overline{x}]$ and that we run IBP over this domain. However, it is known that for deep networks, IBP can be conservative, especially when the input domain $\mathcal{X}$ is large. Similarly, the interval matrix multiplication bound in Lemma 2(iii) can also be conservative when the sets $[\underline{A}, \overline{A}]$ and $[\underline{B}, \overline{B}]$ are large. However, as the sets become smaller, the estimates also become tighter. This observation motivates the idea of *partitioning the domain of interest*.

Suppose we can partition our domain of interest $\mathcal{X}$ into the union of hyperrectangles, i.e., $\mathcal{X} = \cup_{i=1}^{m} [\underline{x}^i, \overline{x}^i]$. Then if we can verify that (2) is satisfied for each $x \in [\underline{x}^i, \overline{x}^i]$ and for each $i \in \{1,\ldots,m\}$, then the closed-loop dynamics are contracting on all of $\mathcal{X}$. Then for each $i \in \{1,\ldots,m\}$, we can apply Algorithm 2 on the domain $[\underline{x}^i, \overline{x}^i]$ to construct a constant G^i. Moreover, if each G^i satisfies $\lambda_{\max}(G^i) \leq 0$,

Algorithm 2 Construction of G matrix from $x \in [\underline{x}, \overline{x}]$

1: Require $x \in [\underline{x}, \overline{x}]$, N and u are feedforward neural networks
2: Get bounds $f(x) \in [\underline{f}, \overline{f}]$ and $Df(x) \in [\underline{Df}, \overline{Df}]$
3: Call Algorithm 1 to get bounds $Bu(x) \in [\underline{Bu}, \overline{Bu}]$, $BDu(x) \in [\underline{BDu}, \overline{BDu}]$
4: Use Algorithm 1 and Lemma 2(iii) to get bounds $M(x) \in [\underline{M}, \overline{M}]$, $\nabla M_{ij}(x) \in [\underline{\nabla M_{ij}}, \overline{\nabla M_{ij}}]$ for $i, j \in \{1, \dots, n\}$
5: **for** $(i, j) \in \{1, \dots, n\}^2$ **do**
6: $\quad [[\underline{\dot{M}_f}]_{ij}, [\overline{\dot{M}_f}]_{ij}] \leftarrow \text{IMM}([\underline{\nabla M_{ij}^\top}, \overline{\nabla M_{ij}^\top}], [\underline{f}, \overline{f}])$
7: $\quad [[\underline{\dot{M}_{Bu}}]_{ij}, [\overline{\dot{M}_{Bu}}]_{ij}] \leftarrow$
$\quad \text{IMM}([\underline{\nabla M_{ij}^\top}, \overline{\nabla M_{ij}^\top}], [\underline{Bu}, \overline{Bu}])$
8: **end for**
9: *# Get bounds on $M(x)Df(x)$ and $M(x)BDu(x)$*
10: $[L_f, U_f] \leftarrow \text{IMM}([\underline{M}, \overline{M}], [\underline{Df}, \overline{Df}])$
11: $[L_u, U_u] \leftarrow \text{IMM}([\underline{M}, \overline{M}], [\underline{BDu}, \overline{BDu}])$
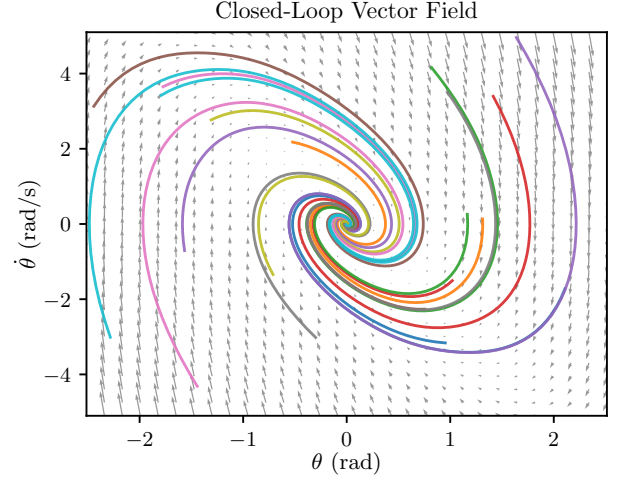12: Construct G via (13)
13: **return** G

Fig. 1. Closed-loop vector field of the inverted pendulum (15) with a neural network controller trained using the loss function (14).

then we have verified contraction on the entirety of $\mathcal{X}$. Intuitively, rather than checking that the eigenvalues of a single symmetric matrix G are nonpositive, we now check that a family of matrices, G^i all have nonpositive eigenvalues, where each matrix G^i is used to verify the contraction inequality (2) on the subdomain $[\underline{x}^i, \overline{x}^i]$.

For verification, this insight can be used to design an algorithm to adaptively partition the domain when contraction cannot be verified on the entirety of the $\mathcal{X}$. Suppose that a single application of Algorithm 2 yields a matrix G with $\lambda_{\max}(G) > 0$. Then one can partition the domain $\mathcal{X}$ into 2^n hyperrectangular subdomains and run Algorithm 2 on each of these subdomains. If $\lambda_{\max}(G^i) \leq 0$ on any of these subdomains, then we have verified that (2) holds on that subdomain. For any subdomain where $\lambda_{\max}(G^i) > 0$, we can again partition it into 2^n more subdomains.

Intuitively, in the limit as the number of partitions goes to infinity, the sets $[\underline{x}^i, \overline{x}^i]$ begin converging to singletons at which point IBP, Lemma 2(iii), and the extension of IBP to Jacobians of neural networks become exact. Then G^i will be arbitrarily close to $\lceil \mathcal{A}(x) \rceil_{\text{Mzr}}$ and verifying $\lambda_{\max}(G^i) \leq 0$ will be equivalent to verifying $\lambda_{\max}(\lceil \mathcal{A}(x) \rceil_{\text{Mzr}}) \leq 0$.

Training algorithm. The sufficient condition based on Theorem 3 and domain partitioning motivate the following training algorithm for formal guarantees of contraction on a domain of interest $\mathcal{X}$. First, partition $\mathcal{X}$ into a collection of hyperrectangles $\{[\underline{x}^i, \overline{x}^i]\}_{i=1}^m$. Then initialize the weights and biases of the neural networks u and N for the controller and the neural contraction metric. Then define the loss function

$$\text{Loss}(\theta) = \sum_{i=1}^m \sum_{j=1}^d \text{ReLU}(\lambda_j(G^i)), \quad (14)$$

where $\lambda_j(G^i)$ denotes the j-th largest eigenvalue of the symmetric matrix G^i generated by Algorithm 2 and θ denotes the collection of trainable parameters in u and N. It is important

to note that $\text{Loss}(\theta) = 0$ only if each G^i has nonpositive eigenvalues and thus Corollary 7 implies that the closed-loop dynamics are contracting on $\mathcal{X}$.

VI. NUMERICAL EXPERIMENT

We consider the classical inverted pendulum

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ g \sin(x_1)/\ell \end{pmatrix} + \begin{pmatrix} 0 \\ 1/(m\ell^2) \end{pmatrix} u, \quad (15)$$

where $x_1 = \theta$ denotes the angular position, $x_2 = \dot{\theta}$ denotes the angular velocity, g is the gravitational constant, m is the mass, and ℓ is the length of the pendulum.

We train a controller with two hidden layers of width 16 and train our NCM using (11) where N has two hidden layers of width 32 and $\varepsilon = 0.1$. For both networks, we use the softplus activation and for the controller network, we constrain inputs to satisfy $|u(x)| \leq 4mg\ell$ by adding a scaled $\tanh$ at the output. We train using the Adam optimizer with a learning rate of 0.01 and we partitioned the domain into 16^2 equal-sized hyperrectangles and used the loss function (14). Code to reproduce our results is available.[1]

We initialize training with a small domain $\overline{x} = (\pi/100, 0.05)$, $\underline{x} = -\overline{x}$ and every time the loss hits zero, we increase the size of the domain $\overline{x} \leftarrow \overline{x} + (\pi/100, 0.06)$. After 20000 epochs, we certify closed-loop contraction for $\overline{x} = (89\pi/100, 5.33)$, $\underline{x} = -\overline{x}$. Figure 1 shows the closed-loop phase portrait and 20 trajectories in phase space.

We can pictorially see that even for this open-loop unstable system, we are able to certify contraction of the closed-loop system with a neural network controller and a neural contraction metric in a rectangular subset of the state space.

VII. CONCLUSIONS

In this work we studied the problem of contraction verification of nonlinear systems with neural network controllers

[1] https://github.com/davydovalexander/interval-bound-contraction

and neural contraction metrics. By leveraging interval analysis and IBP, we provide a scalable sufficient condition for contraction which corresponds to checking that all eigenvalues of a symmetric Metzler matrix are nonpositive. We integrate this sufficient condition and a domain partitioning strategy to enable learning of neural network controllers and neural contraction metrics with certificates of closed-loop contraction. We demonstrate our methodology in the control of an inverted pendulum.

Future work entails more empirical validation in larger-scale nonlinear systems. Additionally, less conservative sufficient conditions than Theorem 3 or sharper neural network bounding methodologies than IBP could yield new training algorithms that provide sharper theoretical guarantees. Moreover, it could be of interest to verify contraction with respect to non-Euclidean norms [4] or Finsler metrics [5] where Metzler matrices often appear in the analysis.

APPENDIX I
ADDITIONAL TRAINING DETAILS

In this section, we elaborate a bit more on some of the details for training.

For the neural network controller, we enforce that $u(0) = 0$ over the duration of training and that $|u(x)| \leq 4mg\ell$ as well. To ensure these constraints are satisfied, we let $\tilde{N}$ be an unconstrained neural network of the form (3a). Then the controller network is parametrized by

$$u(x) = \text{scaledtanh}(\tilde{N}(x) - \tilde{N}(0)),$$

where $\text{scaledtanh}(x) = s \tanh(x/s)$, where s denotes the scale factor and enforces that the output is in the interval $[-s, s]$. In the case of the inverted pendulum, since it is a single-input system, the output u is scalar and we let $s = 4mg\ell$.

We can routinely apply interval bound propagation to this parametrization of the controller because $N(0)$ is a constant shift at the last layer. To apply Algorithm 1, we apply the same procedure to acquire bounds on each J_i. The only modification is in the last layer for interval bounds on the Jacobian of scaledtanh. We get interval bounds using Remark 5. In particular, the expressions in Remark 5 can be written more concretely: for scalar $\underline{x}, \overline{x}$, we have that for all $x \in [\underline{x}, \overline{x}]$, $l \leq \text{scaledtanh}'(x) \leq u$ where $l = \min\{\text{scaledtanh}'(\underline{x}), \text{scaledtanh}'(\overline{x})\}$ and $u = 1$ if $0 \in [\underline{x}, \overline{x}]$ and $\max\{\text{scaledtanh}'(\underline{x}), \text{scaledtanh}'(\overline{x})\}$ otherwise.

In training, we start with r^2 equal-sized partitions (with $r = 16$). We then refine the partition further by resetting $r \leftarrow r + 2$ if the loss function does not converge to 0 within 2000 epochs. The code is written to handle batches so we are able to compute all the eigenvalues of all the G^i in parallel.

REFERENCES

[1] A. Abate, D. Ahmed, M. Giacobbe, and A. Peruffo. Formal synthesis of Lyapunov neural networks. *IEEE Control Systems Letters*, 5(3):773–778, 2020. doi:10.1109/LCSYS.2020.3005328.

[2] H. Beik-Mohammadi, S. Hauberg, G. Arvanitidis, N. Figueroa, G. Neumann, and L. Rozo. Neural contractive dynamical systems. In *International Conference on Learning Representations*, 2023. doi:10.48550/arXiv.2401.09352.

[3] A. Davydov and F. Bullo. Perspectives on contractivity in control, optimization and learning. *IEEE Control Systems Letters*, 8:2087–2098, 2024. doi:10.1109/LCSYS.2024.3436127.

[4] A. Davydov, S. Jafarpour, and F. Bullo. Non-Euclidean contraction theory for robust nonlinear stability. *IEEE Transactions on Automatic Control*, 67(12):6667–6681, 2022. doi:10.1109/TAC.2022.3183966.

[5] F. Forni and R. Sepulchre. A differential Lyapunov framework for contraction analysis. *IEEE Transactions on Automatic Control*, 59(3):614–628, 2014. doi:10.1109/TAC.2013.2285771.

[6] S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, R. Arandjelovic, T. Mann, and P. Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint: arXiv 1810.12715*, 2018. doi:10.48550/arXiv.1810.12715.

[7] A. Harapanahalli, S. Jafarpour, and S. Coogan. A toolbox for fast interval arithmetic in numpy with an application to formal verification of neural network controlled systems. In *ICML Workshop on Formal Verification of Machine Learning*, 2023. doi:10.48550/arXiv.2306.15340.

[8] S. Jaffe, A. Davydov, D. Lapsekili, A. K. Singh, and F. Bullo. Learning neural contracting dynamics: Extended linearization and global guarantees. In *Advances in Neural Information Processing Systems*, 2024. doi:10.48550/arXiv.2402.08090.

[9] H. Li, X. Zhong, B. Hu, and H. Zhang. Neural contraction metrics with formal guarantees for discrete-time nonlinear dynamical systems. In *Learning for Dynamics & Control Conference*, pages 1447–1459, 2025. URL: https://proceedings.mlr.press/v283/li25e.html.

[10] W. Lohmiller and J.-J. E. Slotine. On contraction analysis for nonlinear systems. *Automatica*, 34(6):683–696, 1998. doi:10.1016/S0005-1098(98)00019-3.

[11] I. R. Manchester and J.-J. E. Slotine. Control contraction metrics: Convex and intrinsic criteria for nonlinear feedback design. *IEEE Transactions on Automatic Control*, 62(6):3046–3053, 2017. doi:10.1109/TAC.2017.2668380.

[12] A. Peruffo, D. Ahmed, and A. Abate. Automated and formal synthesis of neural barrier certificates for dynamical models. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 370–388, 2021. doi:10.1007/978-3-030-72016-2_20.

[13] J. W. Simpson-Porco and F. Bullo. Contraction theory on Riemannian manifolds. *Systems & Control Letters*, 65:74–80, 2014. doi:10.1016/j.sysconle.2013.12.016.

[14] D. Sun, S. Jha, and C. Fan. Learning certified control using contraction metric. In *Conference on Robot Learning*, volume 155, pages 1519–1539, 2021. URL: https://proceedings.mlr.press/v155/sun21b.html.

[15] H. Tsukamoto, S.-J. Chung, and J.-J. E Slotine. Contraction theory for nonlinear stability analysis and learning-based control: A tutorial overview. *Annual Reviews in Control*, 52:135–169, 2021. doi:10.1016/j.arcontrol.2021.10.001.

[16] L. Yang, H. Dai, Z. Shi, C.-J. Hsieh, R. Tedrake, and H. Zhang. Lyapunov-stable neural control for state and output feedback: A novel formulation. In *International Conference on Machine Learning*, pages 56033–56046, 2024. URL: https://proceedings.mlr.press/v235/yang24f.html.

[17] M. Zakwan, L. Xu, and G. Ferrari-Trecate. Neural exponential stabilization of control-affine nonlinear systems. In *IEEE Conference on Decision and Control*, page 8602–8607, 2024. doi:10.1109/CDC56724.2024.10886677.