Reachable Set Estimation and Verification for Neural Network Models of Nonlinear Dynamic Systems

Weiming Xiang, Diego Manzanas Lopez, Patrick Musau, and Taylor T. Johnson

Department of Electrical Engineering and Computer Science,
Vanderbilt University, Nashville, Tennessee 37212, USA
{weiming.xiang,diego.manzanas.lopez,patrick.musau,taylor.johnson}@
vanderbilt.edu

Abstract. Neural networks have been widely used to solve complex real-world problems. Due to the complex, nonlinear, non-convex nature of neural networks, formal safety and robustness guarantees for the behaviors of neural network systems are crucial for their applications in safety-critical systems. In this paper, the reachable set estimation and safety verification problems for Nonlinear Autoregressive-Moving Average (NARMA) models in the forms of neural networks are addressed. The neural networks involved in the model are a class of feed-forward neural networks called Multi-Layer Perceptrons (MLPs). By partitioning the input set of an MLP into a finite number of cells, a layer-by-layer computation algorithm is developed for reachable set estimation of each individual cell. The union of estimated reachable sets of all cells forms an over-approximation of the reachable set of the MLP. Furthermore, an iterative reachable set estimation algorithm based on reachable set estimation for MLPs is developed for NARMA models. The safety verification can be performed by checking the existence of non-empty intersections between unsafe regions and the estimated reachable set. Several numerical examples are provided to illustrate the approach.

Keywords: Neural network; Reachable set estimation; Safety verification; Nonlinear systems; Data-driven models; Robustness; Adversarial machine learning; Nonlinear autoregressive-moving average (NARMA) models; Multi-layer perceptron (MLP); Magnetic levitation systems (Maglev); Feedforward neural networks; Output Over-approximation; Artificial intelligence; Model-free methods; Simulation-based methods; Formal methods; Intelligent systems; Safe autonomous systems

1 Introduction

Artificial neural networks have been widely used in machine learning and artificial intelligence systems. Applications include adaptive control [14, 10], pattern recognition [26, 19], game playing [27], autonomous vehicles [6], and many others. Neural networks are trained over finite amounts of input and output data,

and are expected to be able to generalize to produce desirable outputs for given inputs even including previously unseen inputs. Though neural networks have been showing effectiveness and powerful ability in resolving complex problems, they are confined to systems that comply only to the lowest safety integrity levels since, most of the time, a neural network is viewed as a *black box* without effective methods to assure robustness or safety specifications for its outputs. For nonlinear dynamic systems whose models are difficult or even impossible to establish, using neural network models that are inherently derived from input and output data to approximate the nonlinear dynamics is an efficient and practical way. One standard employment of neural networks is to approximate the Nonlinear Autoregressive-Moving Average (NARMA) model which is a popular model for nonlinear dynamic systems. However, once the NARMA model in the form of neural networks is established, a problem naturally arises: *How to compute the reachable set of an NARMA model that is essentially expressed by neural networks and, based on that, how to verify properties of an NARMA model?* For computing or estimating the reachable set for a nonlinear system starting from an initial set and with an input set, the numbers of inputs and initial state that need to be checked are infinite, which is impossible only by performing experiments. Moreover, it has been observed that neural networks can react in unexpected and incorrect ways to even slight perturbations of their inputs [28], which could result in unsafe systems. Hence, methods that are able to provide formal guarantees are in a great demand for verifying specifications or properties of systems involving neural networks. Verifying neural networks is a hard problem, even simple properties about them have been proven NP-complete problems [17]. The difficulties mainly come from the presence of activation functions and the complex structures, making neural networks large-scale, nonlinear, non-convex and thus incomprehensible to humans. Until now, only few results have been reported for verifying neural networks. The verification for feed-forward multi-layer neural networks is investigated based on *Satisfiability Modulo Theory* (SMT) in [13, 24]. In [23] an abstraction-refinement approach is proposed for verification of specific networks known as *Multi-Layer Perceptrons* (MLPs). In [40, 17], a specific kind of activation functions called *Rectified Linear Unit* (ReLU) is considered for the verification problem of neural networks. A simulation-based approach is developed in [38], which turns the reachable set estimation problem into a neural network maximal sensitivity computation problem that is described in terms of a chain of convex optimization problems. Additionally, some recent reachable set/state estimation results are reported for neural networks [45, 44, 42, 47, 29], these results that are based on Lyapunov functions analogous to stability [34, 36, 35, 43, 41, 32, 33] and reachability analysis of dynamical systems [39, 37], certainly have potentials to be further extended to safety verification.

In this paper, we will use neural networks to represent the forward dynamics of the nonlinear systems that are in the form of NARMA models. Due to the non-convex and nonlinearity existing in the model and inspired by some simulation-based ideas for verification problems [8, 9, 2, 3], a simulation-based approach will be developed to estimate the reachable set of state responses generated from a

NARMA model. The core step of the approach is the reachable set estimation for a class of feed-forward neural networks called Multi-Layer Perceptron (MLP). By discretizing the input space of an MLP into a finite-number of regularized cells, a layer-by-layer computation process is developed to establish an over-approximation of the output set for each individual cell. The union of output set of all cells is the reachable set estimation for the MLP with respect to a given input set. On the basis of the reachable set estimation method for MLPs, the reachable set over any finite-time interval for an NARMA model can be estimated in a recursive manner. Safety verification can be performed if an estimation for the reachable set of an NARMA model is established, by checking the existence of intersections between the estimated reachable set and unsafe regions.

The remainder of this paper is organized as follows. Neural network model of nonlinear systems, that is the NARMA model, is introduced in Section 2. The problem formulation is presented in Section 3. The main results, reachable set estimation for MLPs and NARMA models, are given in Sections 4 and 5, respectively. An example for magnetic levitation systems is presented in Section 6. Conclusions are made in Section 7.

Notations: $\mathbb{R}$ denotes the field of real numbers, $\mathbb{R}^n$ stands for the vector space of all n-tuples of real numbers, $\mathbb{R}^{n\times n}$ is the space of $n \times n$ matrices with real entries. $\|\mathbf{x}\|_\infty$ stands for infinity norm for vector $\mathbf{x} \in \mathbb{R}^n$ defined as $\|\mathbf{x}\|_\infty = \max_{i=1,\ldots,n} |x_i|$. $\mathbf{A}^\top$ denotes the transpose of matrix $\mathbf{A}$. For a set $\mathcal{A}$, $|\mathcal{A}|$ denotes its cardinality.

2 Neural Network Models of Nonlinear Dynamic Systems

Neural networks are commonly used for data-driven modeling for nonlinear systems. One standard model to represent discrete-time nonlinear systems is the Nonlinear Autoregressive-Moving Average (NARMA) model. Given a discrete-time process with past states $\mathbf{x}(k), \mathbf{x}(k-1), \ldots, \mathbf{x}(k-d_x)$ and inputs $\mathbf{u}(k), \mathbf{u}(k-1), \ldots, \mathbf{u}(k-d_u)$, an NARMA model is in the form of

$$\mathbf{x}(k+1) = f\left(\mathbf{x}(k), \mathbf{x}(k-1), \ldots, \mathbf{x}(k-d_x), \mathbf{u}(k), \mathbf{u}(k-1), \ldots, \mathbf{u}(k-d_u)\right), \quad (1)$$

where the nonlinear function $f(\cdot)$ needs to be approximated by training neural networks. The initial state of NARMA model (1) is $\mathbf{x}(0), \ldots, \mathbf{x}(d_x)$, which is assumed to be in set $\mathcal{X}_0 \times \cdots \times \mathcal{X}_{d_x}$, and the input set is $\mathcal{U}$. We assume that the initial state $\{\mathbf{x}(0), \ldots, \mathbf{x}(d_x)\} \in \mathcal{X}_0 \times \cdots \times \mathcal{X}_{d_x}$ and input satisfies $\mathbf{u}(k) \in \mathcal{U}$, $\forall k \in \mathbb{N}$.

A neural network consists of a number of interconnected neurons. Each neuron is a simple processing element that responds to the weighted inputs it received from other neurons. In this paper, we consider the most popular and general feed-forward neural network, MLP. Generally, an MLP consists of three typical classes of layers: An input layer, that serves to pass the input vector to the network, hidden layers of computation neurons, and an output layer composed of at least a computation neuron to produce the output vector.

The action of a neuron depends on its activation function, which is described as

$$y_i = h\left(\sum_{j=1}^{n} \omega_{ij} v_j + \theta_i\right), \tag{2}$$

where v_j is the jth input of the ith neuron, ω_{ij} is the weight from the jth input to the ith neuron, θ_i is called the bias of the ith neuron, y_i is the output of the ith neuron, $h(\cdot)$ is the activation function. The activation function is generally a nonlinear function describing the reaction of ith neuron with inputs v_j, $j = 1, \cdots, n$. Typical activation functions include Rectified Linear Unit (ReLU), logistic, tanh, exponential linear unit, linear functions, etc. In this work, our approach aims at dealing with activation functions regardless of their specific forms, only the following monotonic assumption needs to be satisfied.

Assumption 1 *For any $v_1 \leq v_2$, the activation function satisfies $h(v_1) \leq h(v_2)$.*

Assumption 1 is a common property that can be satisfied by a variety of activation functions. For example, it is easy to verify that the most commonly used such as logistic, tanh, ReLU, all satisfy Assumption 1.

An MLP has multiple layers, each layer ℓ, $1 \leq \ell \leq L$, has $n^{[\ell]}$ neurons. In particular, layer $\ell = 0$ is used to denote the input layer and $n^{[0]}$ stands for the number of inputs in the rest of this paper, and of course, $n^{[L]}$ stands for the last layer, that is the output layer. For a neuron i, $1 \leq i \leq n^{[\ell]}$ in layer ℓ, the corresponding input vector is denoted by $\mathbf{v}^{[\ell]}$ and the weight matrix is

$$\mathbf{W}^{[\ell]} = \left[\boldsymbol{\omega}_1^{[\ell]}, \ldots, \boldsymbol{\omega}_{n^{[\ell]}}^{[\ell]}\right]^\top, \tag{3}$$

where $\boldsymbol{\omega}_i^{[\ell]}$ is the weight vector. The bias vector for layer ℓ is

$$\boldsymbol{\theta}^{[\ell]} = \left[\theta_1^{[\ell]}, \ldots, \theta_{n^{[\ell]}}^{[\ell]}\right]^\top$$

The output vector of layer ℓ can be expressed as

$$\mathbf{y}^{[\ell]} = h_\ell(\mathbf{W}^{[\ell]}\mathbf{v}^{[\ell]} + \boldsymbol{\theta}^{[\ell]}), \tag{4}$$

where $h_\ell(\cdot)$ is the activation function for layer ℓ.

For an MLP, the output of $\ell - 1$ layer is the input of ℓ layer, and the mapping from the input of input layer $\mathbf{v}^{[0]}$ to the output of output layer $\mathbf{y}^{[L]}$ stands for the input-output relation of the MLP, denoted by

$$\mathbf{y}^{[L]} = H(\mathbf{v}^{[0]}), \tag{5}$$

where $H(\cdot) \triangleq h_L \circ h_{L-1} \circ \cdots \circ h_1(\cdot)$.

According to the *Universal Approximation Theorem* [12], it guarantees that, in principle, such an MLP in (5), namely the function $F(\cdot)$, is able to approximate

any nonlinear real-valued function. To use MLP (5) to approximate NARMA model (1), we can let the input of (5) as

$$\mathbf{v}^{[0]} = [\mathbf{x}^\top(k), \mathbf{x}^\top(k-1), \ldots, \mathbf{x}^\top(k-d_x), \mathbf{u}^\top(k), \mathbf{u}^\top(k-1), \ldots, \mathbf{u}^\top(k-d_u)]^\top, \quad (6)$$

and output as

$$\mathbf{y}^{[L]} = \mathbf{x}(k+1). \quad (7)$$

With the input and output data of original nonlinear systems, an approximation of NARMA model (1) can be obtained by standard feed-forward neural network training process. Despite the impressive ability of approximating nonlinear functions, much complexities represent in predicting the output behaviors of MLP (5) as well as NARMA model (1) because of the nonlinearity and nonconvexity of MLPs. In the most of real applications, an MLP is usually viewed as a *black box* to generate a desirable output with respect to a given input. However, regarding property verifications such as the safety verification, it has been observed that even a well-trained neural network can react in unexpected and incorrect ways to even slight perturbations of their inputs, which could result in unsafe systems. Thus, to validate the neural network NARMA model for a nonlinear dynamics, it is necessary to compute the reachable set estimation of the model, which is able to cover all possible values of output, to assure that the state trajectories of the model will not attain unreasonable values that is inadmissible for the original system. It is also necessary to estimate all possible values of state for safety verification of a neural network NARMA model.

3 Problem Formulation

Consider initial set $\mathcal{X}_0 \times \cdots \times \mathcal{X}_{d_x}$ and input set $\mathcal{U}$, the reachable set of NARMA model in the form of (1) is defined as follows.

Definition 1. *Given an NARMA model in the form of (1) with initial set* $\mathcal{X}_0 \times \cdots \times \mathcal{X}_{d_x}$ *and input set* $\mathcal{U}$*, the reachable set at a time instant* k *is:*

$$\mathcal{X}_k \triangleq \{\mathbf{x}(k) \mid \mathbf{x}(k) \text{ satisfies (1) and } \{\mathbf{x}(0), \cdots, \mathbf{x}(d_x)\} \in \mathcal{X}_0 \times \ldots \times \mathcal{X}_{d_x}, \\ \mathbf{u}(k) \in \mathcal{U},\ \forall k \in \mathbb{N}\}, \quad (8)$$

and the reachable set over time interval $[0, k_f]$ *is defined by*

$$\mathcal{X}_{[0,k_f]} = \bigcup_{s=0}^{k_f} \mathcal{X}_s. \quad (9)$$

Since MLPs are often large, nonlinear, and non-convex, it is extremely difficult to compute the exact reachable set $\mathcal{X}_k$ and $\mathcal{X}_{[0,k_f]}$ for an NARMA model with MLPs. Rather than directly computing the exact output reachable set for an NARMA model, a more practical and feasible way is to derive an over-approximation of $\mathcal{X}_k$, which is called reachable set estimation.

Definition 2. *A set $\tilde{\mathcal{X}}_k$ is called a reachable set estimation of NARMA model (1) at time instant k, if $\mathcal{X}_k \subseteq \tilde{\mathcal{X}}_k$ holds and, moreover, $\tilde{\mathcal{X}}_{[0,k_f]} = \bigcup_{s=0}^{k} \tilde{\mathcal{X}}_s$ is a reachable set estimation for NARMA model (1) over time interval $[0, k_f]$.*

Based on Definition 2, the problem of reachable set estimation for an NARMA model is given as below.

Problem 1. How does one find the set $\tilde{\mathcal{X}}_k$ such that $\mathcal{X}_k \subseteq \tilde{\mathcal{X}}_k$, given a bounded initial set $\mathcal{X}_0 \times \ldots \times \mathcal{X}_{d_x}$ and an input set $\mathcal{U}$ and an NARMA model (1)?

In this work, we will focus on the safety verification for NARMA models. The safety specification for output is expressed by a set defined in the state space, describing the safety requirement.

Definition 3. *Safety specification $\mathcal{S}$ formalizes the safety requirements for state $\mathbf{x}(k)$ of NARMA model (1), and is a predicate over state $\mathbf{x}$ of NARMA model (1). The NARMA model (1) is safe over time interval $[0, k_f]$ if and only if the following condition is satisfied:*

$$\mathcal{X}_{[0,k_f]} \cap \neg\mathcal{S} = \emptyset, \tag{10}$$

where $\neg$ is the symbol for logical negation.

Therefore, the safety verification problem for NARMA models is stated as follows.

Problem 2. How can the safety requirement in (10) be verified given an NARMA model (1) with a bounded initial set $\mathcal{X}_0 \times \ldots \times \mathcal{X}_{d_x}$ and an input set $\mathcal{U}$ and a safety specification $\mathcal{S}$?

Before ending this section, a lemma is presented to show that the safety verification of an MLP can be relaxed by checking with the over-approximation of output reachable set.

Lemma 1. *Consider an NARMA model (1) and a safety specification $\mathcal{S}$, the NARMA model is safe in time interval $[0, k_f]$ if the following condition is satisfied*

$$\tilde{\mathcal{X}}_{[0,k_f]} \cap \neg\mathcal{S} = \emptyset, \tag{11}$$

where $\mathcal{X}_{[0,k_f]} \subseteq \tilde{\mathcal{X}}_{[0,k_f]}$.

Proof. Since $\mathcal{X}_{[0,k_f]} \subseteq \tilde{\mathcal{X}}_{[0,k_f]}$, condition (11) directly leads to $\mathcal{X}_{[0,k_f]} \cap \neg\mathcal{S} = \emptyset$. The proof is complete.

Lemma 1 implies that it is sufficient to use the estimated reachable set for the safety verification of an NARMA model, thus the solution of Problem 1 is also the key to solve Problem 2.

4 Reachable Set Estimation for MLPs

As (5)–(7) in previous section, the state of an NARMA model is computed through an MLP recursively. Therefore, the first step for the reachable set estimation for an NARMA model is to estimate the output set of MLP (5).

Given an MLP $\mathbf{y}^{[L]} = H(\mathbf{v}^{[0]})$ with a bounded input set $\mathcal{V}$, the problem is how to compute a set $\mathcal{Y}$ as below:

$$\mathcal{Y} \triangleq \{\mathbf{y}^{[L]} \mid \mathbf{y}^{[L]} = H(\mathbf{v}^{[0]}),\ \mathbf{v}^{[0]} \in \mathcal{V} \subset \mathbb{R}^n\}. \tag{12}$$

Due to the complex structure and nonlinearities in activation functions, the estimation of output reachable set of MLP represents much difficulties if only using analytical methods. One possible way to circumvent those difficulties is to employ the information produced by a finite number of simulations.

Definition 4. *Given a set $\mathcal{V} \subset \mathbb{R}^n$, a finite collection of sets $\mathscr{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_N\}$ is said to be a partition of $\mathcal{V}$ if (1) $\mathcal{P}_i \subseteq \mathcal{V}$; (2) $\mathrm{int}(\mathcal{P}_i) \cup \mathrm{int}(\mathcal{P}_j) = \emptyset$; (3) $\mathcal{V} \subseteq \bigcup_{i=1}^{N} \mathcal{P}_i$, $\forall i \in \{1, \ldots, N\}$. Each elements $\mathcal{P}_i$ of partition $\mathscr{P}$ is called a cell.*

In this paper, we use cells defined by intervals which are given as follows: For any bounded set $\mathcal{V} \subset \mathbb{R}^n$, we have $\mathcal{V} \subseteq \bar{\mathcal{V}}$,where $\bar{\mathcal{V}} = \{\mathbf{v} \in \mathbb{R}^n \mid \underline{\mathbf{v}} \leq \mathbf{v} \leq \bar{\mathbf{v}}\}$, in which $\underline{\mathbf{v}}$ and $\bar{\mathbf{v}}$ are defined as the lower and upper bounds of elements of $\mathbf{v}$ in $\mathcal{V}$ as $\underline{\mathbf{v}} = [\inf_{\mathbf{v}\in\mathcal{V}}(v_1), \ldots, \inf_{\mathbf{v}\in\mathcal{V}}(v_n)]^\top$ and $\bar{\mathbf{v}} = [\sup_{\mathbf{v}\in\mathcal{V}}(v_1), \ldots, \sup_{\mathbf{v}\in\mathcal{V}}(v_n)]^\top$, respectively. Then, we are able to partition interval $\mathcal{I}_i = [\inf_{\mathbf{v}\in\mathcal{V}}(v_i),\ \sup_{\mathbf{v}\in\mathcal{V}}(v_i)]$, $i \in \{1, \ldots, n\}$ into M_i segments as $\mathcal{I}_{i,1} = [v_{i,0}, v_{i,1}]$, $\mathcal{I}_{i,2} = [v_{i,1}, v_{i,2}]$, …, $\mathcal{I}_{i,M_i} = [v_{i,M_i-1}, v_{i,M_i}]$, where $v_{i,0} = \inf_{\mathbf{v}\in\mathcal{V}}(v_i)$, $v_{i,M_i} = \sup_{\mathbf{v}\in\mathcal{V}}(v_i)$ and $v_{i,n} = v_{i,0} + \frac{m(v_{i,M_i} - v_{i,0})}{M_i}$, $m \in \{0, 1, \ldots, M_i\}$. The cells then can be constructed as $\mathcal{P}_i = \mathcal{I}_{1,m_1} \times \cdots \times \mathcal{I}_{n,m_n}$, $i \in \{1, 2, \ldots, \prod_{s=1}^{n} M_s\}$, $\{m_1, \ldots, m_n\} \in \{1, \ldots, M_1\} \times \cdots \times \{1, \ldots, M_n\}$. To remove redundant cells, we have to check if the cell has empty intersection with $\mathcal{V}$. Cell $\mathcal{P}_i$ should be removed if $\mathcal{P}_i \cap \mathcal{V} = \emptyset$. The cell construction process is summarized by `cell` function in Algorithm 1.

With the cells constructed by `cell` function, the next step is to develop a function that is able to estimate the output reachable set for each individual cell as the input to the MLP. A layer-by-layer approach is developed.

Theorem 1. *For a single layer $\mathbf{y} = h(\mathbf{W}\mathbf{v}+\boldsymbol{\theta})$, if the input set is a cell described by $\mathcal{I}_1 \times \cdots \times \mathcal{I}_{n_v}$ where $\mathcal{I}_i = [\underline{v}_i, \bar{v}_i]$, $i \in \{1, \ldots, n_v\}$, the output set can be over-approximated by a cell in the expression of intervals $\mathcal{I}_1 \times \cdots \times \mathcal{I}_{n_y}$, where $\mathcal{I}_i$, $i \in \{1, \ldots, n_y\}$ can be computed by*

$$\mathcal{I}_i = [h(\underline{z}_i + \theta_i),\ h(\bar{z}_i + \theta_i)], \tag{13}$$

where $\underline{z_i} = \sum_{j=1}^{n_v} \underline{g}_{ij}$, $\bar{z}_i = \sum_{j=1}^{n_v} \bar{g}_{ij}$ with $\underline{g}_{ij}$ and $\bar{g}_{ij}$ defined by

$$\underline{g}_{ij} = \begin{cases} \omega_{ij}\underline{v}_j\ \omega_{ij} \geq 0 \\ \omega_{ij}\bar{v}_j\ \omega_{ij} < 0 \end{cases},\ \bar{g}_{ij} = \begin{cases} \omega_{ij}\bar{v}_j\ \omega_{ij} \geq 0 \\ \omega_{ij}\underline{v}_j\ \omega_{ij} < 0 \end{cases}. \tag{14}$$

Algorithm 1 Partition an input set

Require: Set $\mathcal{V}$, partition numbers M_i, $i \in \{1, \ldots, n\}$
Ensure: Partition $\mathscr{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_N\}$

```
function CELL(V, M_i, i ∈ {1,...,n})
    v_{i,0} ← inf_{v∈V}(v_i), v_{i,M_i} ← sup_{v∈V}(v_i)
    for i = 1 : 1 : n do
        for j = 1 : 1 : M_i do
            v_{i,j} ← v_{i,0} + j(v_{i,M_i} − v_{i,0}) / M_i
            I_{i,j} ← [v_{i,j−1}, v_{i,j}]
        end for
    end for
    P_i ← I_{1,m_1} × ··· × I_{n,m_n}, {m_1,...,m_n} ∈ {1,...,M_1} × ··· × {1,...,M_n}
    if P_i ∩ V = ∅ then
        Remove P_i
    end if
    return P = {P_1, P_2, ..., P_N}
end function
```

Proof. By (14), one can obtain that

$$\underline{z}_i = \min_{\mathbf{v} \in \mathcal{I}_1 \times \cdots \times \mathcal{I}_{n_v}} \left(\sum\nolimits_{j=1}^{n_v} \omega_{ij} v_j \right), \tag{15}$$

$$\bar{z}_i = \max_{\mathbf{v} \in \mathcal{I}_1 \times \cdots \times \mathcal{I}_{n_v}} \left(\sum\nolimits_{j=1}^{n_v} \omega_{ij} v_j \right). \tag{16}$$

Consider neuron i, its output is $y_i = h\left(\sum_{j=1}^{n_v} \omega_{ij} v_j + \theta_i \right)$. Under Assumption 1, we can conclude that

$$\min_{\mathbf{v} \in \mathcal{I}_1 \times \cdots \times \mathcal{I}_{n_v}} \left(h\left(\sum\nolimits_{j=1}^{n_v} \omega_{ij} v_j + \theta_i \right) \right) = h(\underline{z}_i + \theta_i), \tag{17}$$

$$\max_{\mathbf{v} \in \mathcal{I}_1 \times \cdots \times \mathcal{I}_{n_v}} \left(h\left(\sum\nolimits_{j=1}^{n_v} \omega_{ij} v_j + \theta_i \right) \right) = h(\bar{z}_i + \theta_i). \tag{18}$$

Thus, it leads to

$$y_i \in [h(\underline{z}_i + \theta_i),\ h(\bar{z}_i + \theta_i)] = \mathcal{I}_i. \tag{19}$$

and therefore, $\mathbf{y} \in \mathcal{I}_1 \times \cdots \times \mathcal{I}_{n_y}$. The proof is complete.

Theorem 1 not only demonstrates the output set of one single layer can be approximated by a cell if the input set is a cell, it also gives out an efficient way to calculate the cell, namely by (13) and (14). For multi-layer neural networks, Theorem 1 plays the key role for the layer-by-layer approach. For an MLP which essentially has $\mathbf{v}^{[\ell]} = \mathbf{y}^{[\ell-1]}$, $\ell = 1, \ldots, L$, if the input set is a set of cells, Theorem 1 assures the input set of every layer can be over-approximated by a set of cells, which can be computed by (13) and (14) layer-by-layer. The output set of layer L is thus an over-approximation of reachable set of the MLP.

Function `reachMLP` given in Algorithm 2 illustrates the layer-by-layer method for reachable set estimation for an MLP.

Algorithm 2 Reachable set estimation for MLP

Require: Weight matrices $\mathbf{W}^{[\ell]}$, bias $\boldsymbol{\theta}^{[\ell]}$, $\ell \in \{1,\dots,L\}$, set $\mathcal{V}$, partition numbers M_i, $i \in \{1,\dots,n\}$
Ensure: Reachable set estimation $\tilde{\mathcal{Y}}$.

function REACHMLP($\mathbf{W}^{[\ell]}$, $\boldsymbol{\theta}^{[\ell]}$, $\ell \in \{1,\dots,L\}$, $\mathcal{V}$, M_i, $i \in \{1,\dots,n\}$)
  $\mathscr{P} \leftarrow \text{cell}(\mathcal{V}, M_i, i \in \{1,\dots,n\})$
  for $p = 1:1:|\mathscr{P}|$ **do**
    $\mathcal{I}_1^{[1]} \times \cdots \times \mathcal{I}_{n^{[1]}}^{[1]} \leftarrow \mathcal{P}_p$
    for $j = 1:1:L$ **do**
      for $i = 1:1:n^{[j]}$ **do**
        $\underline{g}_{ij} \leftarrow \begin{cases} \omega_{ij}\underline{v}_j & \omega_{ij} \geq 0 \\ \omega_{ij}\bar{v}_j & \omega_{ij} < 0 \end{cases}$, $\bar{g}_{ij} \leftarrow \begin{cases} \omega_{ij}\bar{v}_j & \omega_{ij} \geq 0 \\ \omega_{ij}\underline{v}_j & \omega_{ij} < 0 \end{cases}$
        $\underline{z}_i \leftarrow \sum_{j=1}^{n_v} \underline{g}_{ij}$, $\bar{z}_i \leftarrow \sum_{j=1}^{n_v} \bar{g}_{ij}$
        $\mathcal{I}_i^{[j+1]} \leftarrow [h_j(\underline{z}_i + \theta_i),\ h_j(\bar{z}_i + \theta_i)]$
      end for
    end for
    $\tilde{\mathcal{Y}}_p \leftarrow \mathcal{I}_1^{[L]} \times \cdots \times \mathcal{I}_{n^{[L]}}^{[L]}$
  end for
  $\tilde{\mathcal{Y}} \leftarrow \bigcup_{p=1}^{|\mathscr{P}|} \tilde{\mathcal{Y}}_p$
  return $\tilde{\mathcal{Y}}$
end function

Example 1. An MLP with 2 inputs, 2 outputs and 1 hidden layer consisting of 5 neurons is considered. The activation function for the hidden layer is choosen as `tanh` function and `purelin` function is for the output layer. The weight matrices and bias vectors are given as below:

$$\mathbf{W}^{[1]} = \begin{bmatrix} 0.2075 & -0.7128 \\ 0.2569 & 0.7357 \\ -0.6136 & -0.3624 \\ 0.0111 & 0.1393 \\ -1.0872 & -0.2872 \end{bmatrix},\ \boldsymbol{\theta}^{[1]} = \begin{bmatrix} -1.1829 \\ -0.6458 \\ 0.4619 \\ -0.0499 \\ 0.3405 \end{bmatrix},$$

$$\mathbf{W}^{[2]} = \begin{bmatrix} -0.5618 & -0.0851 & -0.4529 & -0.8230 & 0.5651 \\ 0.7861 & -0.0855 & 1.1041 & 1.6385 & -0.3859 \end{bmatrix},\ \boldsymbol{\theta}^{[2]} = \begin{bmatrix} -0.2489 \\ -0.1480 \end{bmatrix}.$$

In this example, the input set is considered as below:

$$\mathcal{V} = \{\mathbf{v} \in \mathbb{R}^2 \mid \|\mathbf{v}\|_\infty \leq 1\}.$$

Then, the partition numbers are chosen to be $M_1 = M_2 = 20$, which means there are in total 400 cells, $\mathcal{P}_i$, $i \in \{1,\dots,400\}$, produced for the reachable set estimation.

Executing function `reachMLP` for input set $\mathcal{V}$, the estimated output reachable set is given in Figure 1, in which it can be seen that 400 reachtubes are obtained and the union of them is the over-approximation of reachable set.

Moreover, we choose a different partition numbers discretizing state space to show how the choice of partitioning input set affects the estimation outcome.

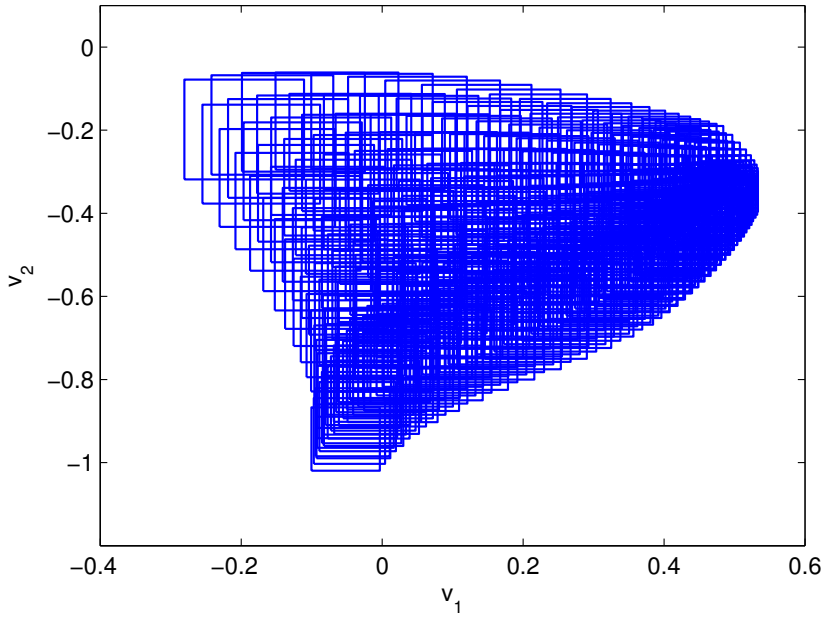

Fig. 1. Output reachable set estimation with input set $\mathcal{V} = \{\mathbf{v} \in \mathbb{R}^2 \mid \|v\|_\infty \leq 1\}$ and partition number $M_1 = M_2 = 20$. 400 reachtubes are computed for the reachable set estimation of the MLP.

Explicitly, larger partition numbers will produce more cells and generate preciser approximations of input sets and are supposed to achieve preciser estimations. Here, we adjust the partition numbers from 10 to 50 for the different estimation results. With this finer discretization, more computation efforts are required for running function `reachMLP`, but a tighter estimation for the reachable set can be obtained. The reachable set estimations are shown in Figure 2. Comparing those results, it can be observed that larger partition numbers can lead to a better estimation result at the expense of more computation efforts. The computation time and number of reachtubes with different partition numbers are listed in Table 1.

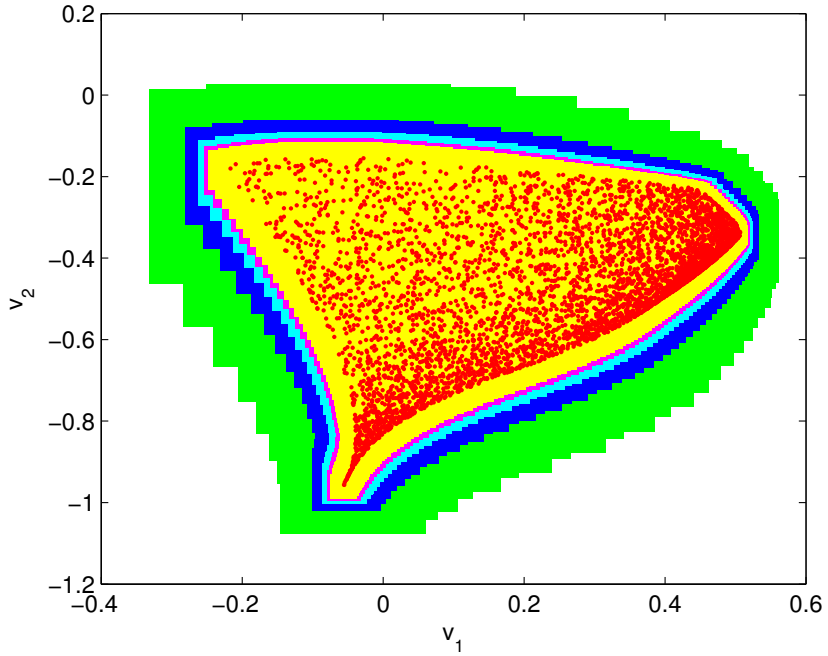

Fig. 2. Output reachable set estimation with input set $\mathcal{V} = \{\mathbf{v} \in \mathbb{R}^2 \mid \|v\|_\infty \leq 1\}$ and partition number $M_1 = M_2 = 10$ (green + blue + cyan + magenta + yellow), $M_1 = M_2 = 20$ (blue + cyan + magenta + yellow), $M_1 = M_2 = 30$ (cyan + magenta + yellow), $M_1 = M_2 = 40$ (magenta + yellow) and $M_1 = M_2 = 50$ (yellow). It can be observed that tighter estimations can be obtained with larger partition numbers. 5000 random outputs (red spots) from input set are all located in the estimated reachable set.

Table 1. Computation time and number of reachtubes with different partition numbers

Partition Number	Computation Time	Number of Reachtubes
$M_1 = M_2 = 10$	0.062304 seconds	100
$M_1 = M_2 = 20$	0.074726 seconds	400
$M_1 = M_2 = 30$	0.142574 seconds	900
$M_1 = M_2 = 40$	0.251087 seconds	1600
$M_1 = M_2 = 50$	0.382729 seconds	2500

To validate the result, 5000 random outputs are generated, it is clear to see in Figure 2 that all the outputs are included in the estimated reachable set, showing the effectiveness of the proposed approach.

5 Reachable Set Estimation for NARMA Models

Based on the developed approach for reachable set estimation for MLP, this section will extend the result to NARMA models. As in previous sections, NARMA models employ MLP to approximate the nonlinear relation between $\mathbf{x}(k), \mathbf{x}(k-1), \ldots, \mathbf{x}(k-d_x), \mathbf{u}(k), \mathbf{u}(k-1), \ldots, \mathbf{u}(k-d_u)$ and state $\mathbf{x}(k+1)$. Without loss of generality, we assume $d_x \geq d_u$, thus the model is valid for any $k \geq d_x$. Thus, with the aid of reachable set estimation results for MLP, the reachable set of NARMA (1) at time instant k can be estimated by recursively using functions `cell` and `reachMLP` for $k - d_x$ times.

Since the reachable sets $\mathcal{X}_k$, $k \in \{0, 1, \ldots, d_x\}$, are given as initial set, let us start with $k = d_x + 1$. In the employment of function `reachMLP` with input of $\mathcal{X}_0 \times \ldots \times \mathcal{X}_{d_x}$ and $\mathcal{U}^{d_u}$, $\tilde{\mathcal{X}}_{d_x+1} = \text{reachMLP}(\mathbf{W}^{[\ell]}, \boldsymbol{\theta}^{[\ell]}, \ell \in \{1, \ldots, L\}, \mathcal{X}_0 \times \ldots \times \mathcal{X}_{d_x}, M_i, i \in \{1, \ldots, n^{[0]}\})$ is an over-approximation of $\mathcal{X}_{d_x+1}$, namely $\mathcal{X}_{d_x+1} \subseteq \tilde{\mathcal{X}}_{d_x+1}$. Then, repeating using function `reachMLP` from d_x+1 to k_f, we can obtain an over-approximation of $\mathcal{X}_k$, $k = d_x + 1, \ldots, k_f$, and $\mathcal{X}_{[0,k_f]}$.

Proposition 1. *Consider NARMA model (1) with initial set $\mathcal{X}_0 \times \ldots \times \mathcal{X}_{d_x}$ and input set $\mathcal{U}$, the reachable set $\mathcal{X}_k$, $k > d_x$ can be recursively over-approximated by*

$$\begin{aligned}\tilde{\mathcal{X}}_k = &\text{reachMLP}(\mathbf{W}^{[\ell]}, \boldsymbol{\theta}^{[\ell]}, \ell \in \{1, \ldots, L\}, \\ &\tilde{\mathcal{X}}_{k-d_x-1} \times \ldots \times \tilde{\mathcal{X}}_{k-1} \times \mathcal{U}^{d_u}, M_i, i \in \{1, \ldots, n^{[0]}\}),\end{aligned} \tag{20}$$

where $\tilde{\mathcal{X}}_k = \mathcal{X}_k$, $k \in \{0, \ldots, d_x\}$. the reachable set over time interval $[0, k_f]$ can be estimated by

$$\tilde{\mathcal{X}}_{[0,k_f]} = \bigcup\nolimits_{s=0}^{k_f} \tilde{\mathcal{X}}_s. \tag{21}$$

The iterative algorithm for estimating reachable set $\mathcal{X}_k$ and $\mathcal{X}_{k_f}$ is summarized as function `reachNARMA` in Algorithm 3.

Function `reachNARMA` is sufficient to solve the reachable set estimation problem for an NARMA model, that is Problem 1. Then, we can move forward to

Algorithm 3 Reachable set estimation for NARMA model

Require: Weight matrices $\mathcal{W}^{[\ell]}$, bias $\boldsymbol{\theta}^{[\ell]}$, $\ell = 1, \ldots, L$, initial set $\mathcal{X}_0 \times \ldots \times \mathcal{X}_{d_x}$, input set $\mathcal{U}$, partition numbers $M_i, i \in \{1, \ldots, n^{[0]}\}$

Ensure: Reachable set estimation $\mathcal{X}_k$, $\mathcal{X}_{[0,k_f]}$.

1: **function** REACHNARMA($\mathcal{W}^{[\ell]}$, $\boldsymbol{\theta}^{[\ell]}$, $\ell = 1, \ldots, L$, $\mathcal{X}_0 \times \ldots \times \mathcal{X}_{d_x}$, $\mathcal{U}$, $M_i, i \in \{1, \ldots, n^{[0]}\}$)
2: **for** $k = d_u + 1 : 1 : k_f$ **do**
3: $\mathcal{V} \leftarrow \mathcal{X}_{k-d_u-1} \times \ldots \times \mathcal{X}_{k-1} \times \mathcal{U}^{d_u}$
4: $\mathcal{X}_k \leftarrow \text{reachMLF}(\mathcal{W}^{[\ell]}, \boldsymbol{\theta}^{[\ell]}, \ell = 1, \ldots, L, \mathcal{V}, M_i, i \in \{1, \ldots, n^{[0]}\}$.
5: **end for**
6: $\mathcal{X}_{[0,k_f]} \leftarrow \bigcup_{s=0}^{k_f} \mathcal{X}_s$
7: **return** $\mathcal{X}_k$, $k = 0, 1 \ldots, k_f$, $\mathcal{X}_{[0,k_f]}$
8: **end function**

Problem 2, the safety verification problem for an NARMA model with a given safety specification $\mathcal{S}$ over a finite interval $[0, k_f]$, with the aid of estimated reachable set $\mathcal{X}_{[0,k_f]}$. Given a safety specification $\mathcal{S}$, the empty intersection between over-approximation $\tilde{\mathcal{X}}_{[0,k_f]}$ and $\neg\mathcal{S}$, namely $\tilde{\mathcal{X}}_{[0,k_f]} \cap \neg\mathcal{S} = \emptyset$, naturally leads to $\mathcal{X}_{[0,k_f]} \cap \neg\mathcal{S} = \emptyset$ due to $\mathcal{X}_{[0,k_f]} \subseteq \tilde{\mathcal{X}}_{[0,k_f]}$. The safety verification result is summarized by the following proposition.

Proposition 2. *Consider NARMA model (1) with initial set $\mathcal{X}_0 \times \ldots \times \mathcal{X}_{d_x}$, input set $\mathcal{U}$, and a safety specification $\mathcal{S}$, the NARMA model (1) is safe in interval $[0, k_f]$, if $\tilde{\mathcal{X}}_{[0,k_f]} \cap \neg\mathcal{S} = \emptyset$, where $\tilde{\mathcal{X}}_{[0,k_f]} = \text{reachNARMA}(\mathcal{W}^{[\ell]}, \boldsymbol{\theta}^{[\ell]}, \ell = 1, \ldots, L, \mathcal{X}_0 \times \ldots \times \mathcal{X}_{d_x}, \mathcal{U}, M_i, i \in \{1, \ldots, n^{[0]}\})$ obtained by Algorithm 3.*

Function `verifyNARMA` is developed based on Proposition 2 for Problem 2, the safety verification problem for NARMA model. If function `verifyNARMA` returns SAFE then the NARMA model is safe. If it returns UNCERTAIN, caused by the fact $\tilde{\mathcal{X}}_{[0,k_f]}$, that means the safety property is unclear for this case.

A numerical example is provided to show the effectiveness of our developed approach.

Example 2. In this example, we consider an NARMA model as below:

$$\mathbf{x}(k+1) = f(\mathbf{x}(k), \mathbf{u}(k)), \tag{22}$$

where $\mathbf{x}(k), \mathbf{u}(k) \in \mathbb{R}$. We use an MLP with 2 inputs, 1 outputs and 1 hidden layer consisting of 5 neurons to approximate f with weight matrices and bias vectors below:

$$\mathbf{W}^{[1]} = \begin{bmatrix} 0.1129 & 0.4944 \\ 2.2371 & 0.4389 \\ -1.1863 & -0.7365 \\ 0.2965 & 0.3055 \\ -0.6697 & 0.5136 \end{bmatrix}, \ \boldsymbol{\theta}^{[1]} = \begin{bmatrix} -13.8871 \\ -8.2629 \\ 5.8137 \\ -3.2035 \\ -0.6697 \end{bmatrix},$$

$$\mathbf{W}^{[2]} = \begin{bmatrix} -3.3067 & 1.3905 & -0.6422 & 2.5221 & 1.8242 \end{bmatrix}, \ \boldsymbol{\theta}^{[2]} = \begin{bmatrix} 5.8230 \end{bmatrix}$$

Algorithm 4 Safety verification for NARMA model

Require: Weight matrices $\mathcal{W}^{[\ell]}$, bias $\boldsymbol{\theta}^{[\ell]}$, $\ell = 1, \ldots, L$, initial set $\mathcal{X}_0 \times \ldots \times \mathcal{X}_{d_x}$, input set $\mathcal{U}$, partition numbers $M_i, i \in \{1, \ldots, n^{[0]}\}$, safety specification $\mathcal{S}$
Ensure: SAFE or UNCERTAIN.
1: **function** VERIFYNARMA($\mathcal{W}^{[\ell]}$, $\boldsymbol{\theta}^{[\ell]}$, $\ell = 1, \ldots, L$, $\mathcal{X}_0 \times \ldots \times \mathcal{X}_{d_x}$, $\mathcal{U}$, $M_i, i \in \{1, \ldots, n^{[0]}\}$, $\mathcal{S}$)
2: $\mathcal{X}_{[0,k_f]} \leftarrow$ reachNARMA($\mathcal{W}^{[\ell]}, \boldsymbol{\theta}^{[\ell]}, \ell = 1, \ldots, L, \mathcal{X}_0 \times \ldots \times \mathcal{X}_{d_x}, \mathcal{U}, M_i, i \in \{1, \ldots, n^{[0]}\}$)
3: **if** $\mathcal{X}_{[0,k_f]} \cap \mathcal{S} = \emptyset$ **then**
4: **return** SAFE
5: **else**
6: **return** UNCERTAIN
7: **end if**
8: **end function**

The activation function for the hidden layer is choose `tanh` function and `purelin` function is for the output layer. The initial set and input set are given by the following set

$$\mathcal{X}_0 = \{\mathbf{x}(0) \in \mathbb{R} \mid -0.2 \leq \mathbf{x}(0) \leq 0.2\}, \tag{23}$$

$$\mathcal{U} = \{\mathbf{u}(k) \in \mathbb{R} \mid 0.8 \leq \mathbf{u}(k) \leq 1.2, \ \forall k \in \mathbb{N}\}. \tag{24}$$

We set the partition numbers to be $M_1 = M_2 = 10$, where M_1 is for input $\mathbf{u}$ and M_2 is for state $\mathbf{x}$. The time horizon for the reachable set estimation is set to be $[0, 50]$. Using function `reachNARMA`, the reachable set can be estimated, which is shown in Figure 3. To show the effectiveness of our proposed approach, we randomly generate 100 state trajectories that are all within the estimated reachable set.

Furthermore, with the estimated reachable set, the safety verification can be easily performed. For example, if the safety region is assumed to be $\mathcal{S} = \{\mathbf{x} \in \mathbb{R} \mid \mathbf{x} \leq 16\}$, it is easy to verify that $\tilde{\mathcal{X}}_{[0,50]} \cap \neg\mathcal{S} = \emptyset$ which means the NARMA model is safe.

6 Magnetic Levitation Systems (Maglev)

6.1 Brief Introduction

Magnetic Levitation Systems, which are called Maglev Systems in short, are systems in which an object is suspended exclusively by the presence of magnetic fields. In such schemes, the force exerted by the presence of magnetic fields is able to counteract gravity and any other forces acting on the object [15]. In order to achieve levitation, there are two principle concerns. The first concern is to exert a sufficient lifting force with which to counteract gravity and the second concern is stability. Once levitation has been achieved, it is critical to ensure that the system does not move into a configuration in which the lifting forces are

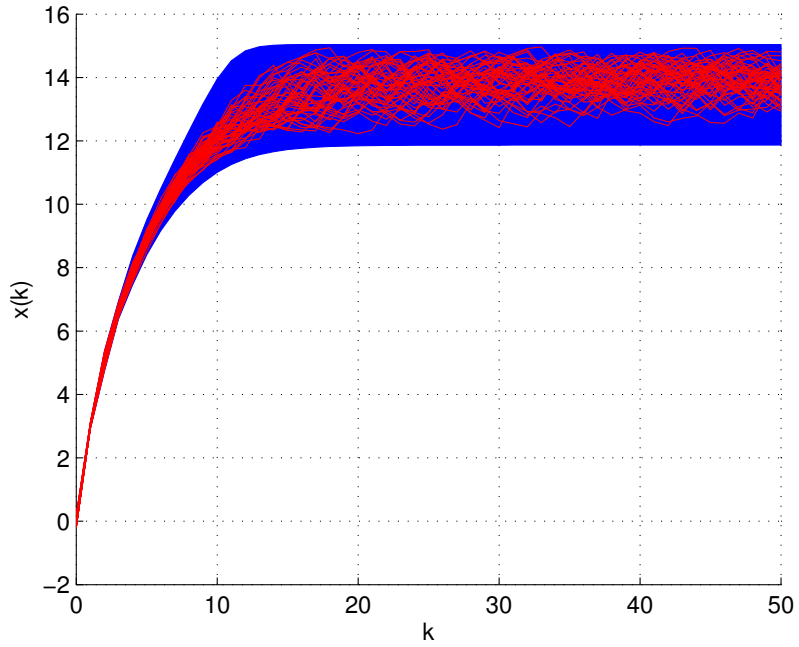

Fig. 3. Reachable set estimation for NARMA model. Blue area is the estimated reachable set and red solid lines are 100 randomly generated state trajectories. All the randomly generated state trajectories are in the reachable set estimation area.

neutralized [25]. However, attaining stable levitation is a considerably complex task, and in his famous theorem, Samuel Earnshaw demonstrated that there is no static configuration of stability for magnetic systems [7]. Intuitively, the instability of magnetic systems lies in the fact that magnetic attraction or repulsion increases or decreases in relation to the square of distance. Thus, most control strategies for Maglev Systems make use of servo-mechanisms [31] and a feedback linearization [30] around a particular operating point of the complex nonlinear differential equations [46] describing the sophisticated mechanical and electrical dynamics. Despite their intrinsic complexity, these systems have exhibited utility in numerous contexts and in particular Maglev System have generated considerable scientific interest in transportation due to their ability to minimize mechanical loss, allow faster travel [18], minimize mechanical vibration, and emit low levels of noise[16]. Other application domains of such systems include wind tunnel levitation [31], contact-less melting, magnetic bearings, vibrator isolation systems, and rocket-guiding designs [11]. Consequently, Maglev Systems have been extensively studied in control literature [15].

Due to their unstable, complex, and nonlinear nature, it is difficult to build a precise feedback control model for the dynamic behavior of complex Maglev System. In most cases, a linearization of the nonlinear dynamics is susceptible to a great deal of inaccuracy and uncertainty. As the system deviates from an assumed operating point, the accuracy of the model deteriorates [1]. Additionally, models based on simplifications are often unable to handle the presence of disturbance forces. Thus, to improve control schemes, a stricter adherence to the complex nonlinear nature of the Maglev Systems is needed. In the last several years, neural network control systems have received significant attention due to their ability to capture complex nonlinear dynamics and model nonlinear unknown parameters [46].

In the control of magnetic levitation systems the nonlinear nature can be modeled by a neural network that is able to describe the input-output nature of the nonlinear dynamics [31]. Neural networks have shown the ability to approximate any nonlinear function to any desired accuracy [20]. Using the neural network model of the plant we wish to control, a controller can be designed to meet system specifications. While neural control schemes have been successful in creating stable controllers for nonlinear systems, it is essential to demonstrate that these systems do not enter undesirable states. As an example, in the requirements for a Maglev train system developed in 1997 by the Japanese Ministry of transportation, the measurements of the 500 km/h train's position and speed could deviate by a maximum of 3 cm and 1 km/h, respectively, in order to prevent derailment and contact with the railway [22]. As magnetic systems become more prevalent in transportation and in other domains, the verification of these systems is essential. Thus, in this example, we perform a reachable set estimation of a NARMA neural network model (1) of a Maglev System.

6.2 Neural Network Model

The Maglev System we consider consists of a magnet suspended above an electromagnet where the magnet is confined to only moving in the vertical direction [15]. Using the results of De Jésus et. al [15], the nonlinear equation of motion for the system is

$$\frac{d^2y(t)}{dt^2} = -g + \frac{\alpha}{M}\frac{i^2(t)}{y(t)} - \frac{\beta}{M}\frac{dy(t)}{dt}, \tag{25}$$

where $y(t)$ is the vertical position of the magnet above the electromagnet in mm, $i(t)$, in Amperes, is the current flowing in the electromagnet, M is the mass of the magnet, g is the gravitational constant, β is the frictional coefficient, and α is the field strength constant. The frictional coefficient β is dictated by the material in which the magnet moves. In our case, the magnet moves through air. The field strength constant α is determined by the number of turns of wire in our electromagnet and by the strength of the magnet being levitated [15].

To capture the nonlinear input-output dynamics of the system, we trained a NARMA neural network (1) to predict the magnet's future position values. In order to predict the magnet's future position values, two inputs are supplied to

the network: the first is the past value of the current flowing in the electromagnet $i(k-1)$ and the second input is the magnet's previous position value $y(k-1)$. The output of the neural network is the current position $y(k)$. The network consists of one hidden layer with eight neurons and an output layer with one neuron. The transfer function of the first layer is `tanh` and `purelin` for the output layer.

The network is trained using a data set consisting of 4001 target position values for the output and 4001 input current values. The Levenberg-Marquard algorithm [21] is used to train the network using batch training. Using batch training, the weights and biases of the NARMA model (1) are updated after all the inputs and targets are supplied to the network and a gradient descent algorithm is used to minimize error [4]. To avoid over-fitting the network, the training data is divided randomly into three sets: the training set, which consists of 2801 values, the validation set, which consists of 600 values, and a test set which is the same size as the validation set. The training set is used to adjust the weight and bias values of the network as well as to compute the gradient, while the validation set is used to measure the network's generalization. Training of the networks ceases when the network's generalization to input data stops improving. The testing data does not take part into the training, but it is used to check the performance of the net during and after training.

In this example, we set the minimum gradient to 10^{-7}, and set the number of validation checks to 6. Thus, training ceases if the error on the validation set increases for 6 consecutive iterations or the minimum gradient achieves a value of 10^{-7}. In our case, the training stopped when the validation checks reached its limit of 6, obtaining a performance of 0.000218. Initially, before the training begins, the values of the weights, biases, and training set are initialized randomly. Thus, the value of the weights and the biases may be different every time that the network is trained. The weights and biases of the hidden layer are

$$W^{[1]} = \begin{bmatrix} -68.9367 & -3.3477 \\ -0.0802 & -2.1460 \\ 0.1067 & -3.7875 \\ 0.1377 & -1.5763 \\ -0.3954 & -1.4477 \\ -0.4481 & -6.9485 \\ 0.0030 & 1.5819 \\ 5.9623 & -5.5775 \end{bmatrix}, \boldsymbol{\theta}^{[1]} = \begin{bmatrix} 47.8492 \\ 2.2129 \\ 1.9962 \\ -0.0091 \\ -0.0727 \\ -3.8435 \\ 1.7081 \\ 7.5619 \end{bmatrix}$$

and in the output layer, the weights and the biases are

$$W^{[2]} = \begin{bmatrix} -0.0054 & -0.3285 & -0.0732 & -0.4019 & -0.1588 & -0.0128 & 0.5397 & -0.0279 \end{bmatrix},$$

$$\boldsymbol{\theta}^{[2]} = \begin{bmatrix} 0.1095 \end{bmatrix}.$$

Once the NARMA network model (1) is trained and the weight and bias values are adjusted to the values shown above, the reachable set estimation of the system can be computed and a safety requirement $\mathcal{S}$ could be verified. This computation is executed following the process described in the previous section.

6.3 Reachable Set Estimation

In order to compute the reachable set and verify if the given specification is satisfied, Algorithm 3 is employed. First, the reachable set estimation using 5 partitions is computed, followed by the reachable set estimation using 20 partitions. After both reachable set estimations are calculated, 200 random trajectories are generated and plotted into Figure 4.

The reachable set estimations and the random trajectories are computed with an initial set and input set that are assumed to be given by

$$\mathcal{X}_0 = \{\mathbf{x}(0) \in \mathbb{R} \mid 4.00 \leq \mathbf{x}(0) \leq 5.00\}, \tag{26}$$

$$\mathcal{U} = \{\mathbf{u}(k) \in \mathbb{R} \mid 0.10 \leq \mathbf{u}(k) \leq 1.10,\ \forall k \in \mathbb{N}\}. \tag{27}$$

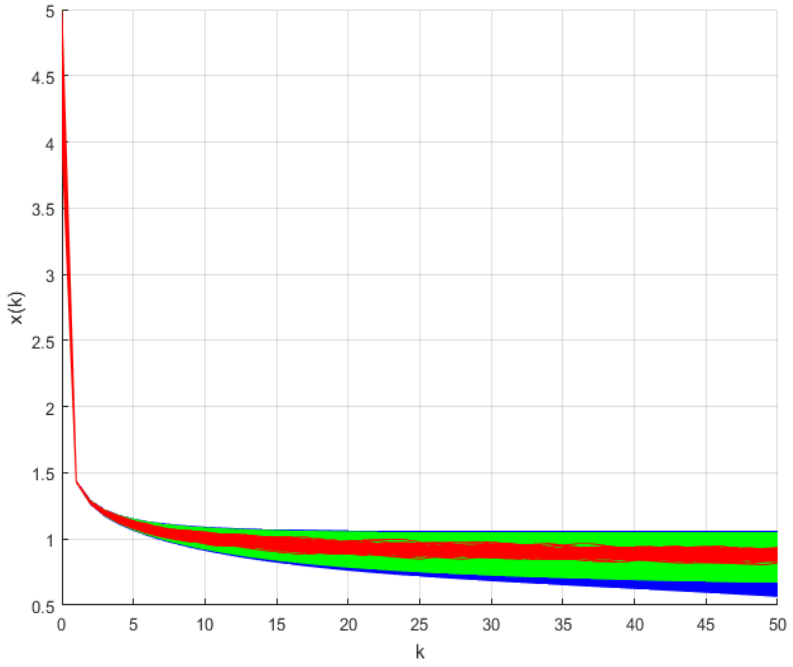

Fig. 4. Reachable set estimation using 5 and 20 partitions. The blue area corresponds to the estimated reachable set using 5 partitions, the tighter green area corresponds to the reachable set estimation using 20 partitions, and the red lines correspond to 200 randomly generated state trajectories, which all of them lie within the estimated reachable set area.

As is observed from Figure 4, all the randomly generated trajectories lie within the estimated reachable set. Also, it can be noted that the area of the reachable set estimation using a larger partition number, that is 20, represented in green, it is smaller than the blue area, which corresponds to the reachable set estimation using a lower partition number ($M_1 = M_2 = 5$) . This is especially noticeable as the time k increases to 40–50, where the difference between the blue region and green region increases as the lower limit of the state $\mathbf{x}(k)$ using 5 partitions keeps decreasing towards 0.6, while the lower limit of the green area maintains a more steady line at 0.7 approximately.

Table 2. Computational time for different partition numbers

Partition Number	Computation Time
$M_1 = M_2 = 5$	0.048700 seconds
$M_1 = M_2 = 20$	0.474227 seconds

In Table 2, the computational time has been recorded for each reachable set estimation. It can be observed that the computational time increases as the partition number increases. For this system, the computational time is approximately 10 times greater when 20 partitions are used. This means that every approach has its different advantages. For the cases when a more precise estimation is needed, we can increase the number of partitions, while for the cases when an larger over-approximation is enough, the number of partitions may be decreased to reduce its computational cost.

The reachable set estimation for the NARMA neural network model (1) of the Maglev Systems shows that all system responses to inputs are contained within the reachable set. Thus, our over-approximation of the reachable states is valid. Given a safety specification $\mathcal{S}$ and the reachable set calculated using Algorithm 3, we are able to determine whether our system model satisfies $\mathcal{S}$. In our example, we did not perform a safety analysis but rather demonstrated the robustness of Algorithm 3 in capturing a large number of possible predictions of the NARMA network model (1). The magnet in our example was confined to moving in one dimension. In magnetic levitation systems that are not physically constrained to a set of axes, there are six degrees of freedom (three rotational and three transnational) [5]. Thus, while we have demonstrated that our algorithm is robust for two-dimensional systems, it will be good to demonstrate its efficacy on higher dimensional systems. However, as the dimensionality and size of the neural networks increases, the computation time needed to compute the reachable set increases significantly as well.

7 Conclusions

This paper studies the reachable set estimation problem for neural network NARMA models of nonlinear dynamic systems. By partitioning the input set

into a finite number of cells, reachable set estimation for MLPs can be done for each individual cells and get the union of output set of cells to form an over-approximation of output set. Then, the reachable set estimation for NARMA models can be performed by iterating the reachable set estimation process for MLP step-by-step to establish an estimation of the state trajectories of a NARMA model. Safety properties of NARMA models can then be verified by checking that the intersection between the estimated reachable set and unsafe regions (sets) is empty. The approach is demonstrated by a Maglev System, for which the reachable set of its NARMA neural network model is estimated. The approach is applicable for a variety of neural network models with different activation functions. However, since the estimation is an over-approximation and error will accumulate at each layer, much finer discretization for the input space is required for deep neural networks that necessarily have large numbers of layers, which will introduce a large computational effort, as otherwise the estimation results will be too conservative to be useful. Reducing the conservativeness caused by the increase of layers and generalizing it to deep neural networks will be a future focus for our approach.

Acknowledgments. The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers CNS 1464311, CNS 1713253, SHF 1527398, and SHF 1736323, the Air Force Research Laboratory (AFRL) under contract numbers FA8750-15-1-0105, as well as FA8650-12-3-7255 via subcontract number WBSC 7255 SOI VU 0001, and the Air Force Office of Scientific Research (AFOSR) under contract numbers FA9550-15-1-0258, FA9550-16-1-0246, and FA9550-18-1-0122. The U.S. government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFRL, AFOSR, or NSF

References

1. J. I. Baig and A. Mahmood. Robust control design of a magnetic levitation system. In *2016 19th International Multi-Topic Conference (INMIC)*, pages 1–5, Dec 2016.
2. Stanley Bak and Parasara Sridhar Duggirala. HyLAA: A tool for computing simulation-equivalent reachability for linear systems. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, pages 173–178. ACM, 2017.
3. Stanley Bak and Parasara Sridhar Duggirala. Rigorous simulation-based analysis of linear hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 555–572. Springer, 2017.
4. Mark Hudson Beale, Martin T. Hagan, and Howard B. Demuth. Neural network toolbox users guide. In *R2016a, The MathWorks, Inc., 3 Apple Hill Drive Natick, MA 01760-2098, , www.mathworks.com*, 2012.
5. Peter J. Berkelman and Ralph L. Hollis. Lorentz magnetic levitation for haptic interaction: Device design, performance, and integration with physical simulations. *The International Journal of Robotics Research*, 19(7):644–667, 2000.

6. Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.
7. R. J. Duffin. Free suspension and earnshaw's theorem. *Archive for Rational Mechanics and Analysis*, 14(1):261–263, Jan 1963.
8. Parasara Sridhar Duggirala, Sayan Mitra, Mahesh Viswanathan, and Matthew Potok. C2E2: a verification tool for stateflow models. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 68–82. Springer, 2015.
9. Chuchu Fan, Bolun Qi, Sayan Mitra, Mahesh Viswanathan, and Parasara Sridhar Duggirala. Automatic reachability analysis for nonlinear hybrid models with C2E2. In *International Conference on Computer Aided Verification*, pages 531–538. Springer, 2016.
10. Shuzhi Sam Ge, Chang Chieh Hang, and Tao Zhang. Adaptive neural network control of nonlinear systems by state and output feedback. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(6):818–828, 1999.
11. A. El Hajjaji and M. Ouladsine. Modeling and nonlinear control of magnetic levitation systems. *IEEE Transactions on Industrial Electronics*, 48(4):831–838, Aug 2001.
12. Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.
13. Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. *arXiv preprint arXiv:1610.06940*, 2016.
14. K Jetal Hunt, D Sbarbaro, R Żbikowski, and Peter J Gawthrop. Neural networks for control systems: a survey. *Automatica*, 28(6):1083–1112, 1992.
15. O. De Jesus, A. Pukrittayakamee, and M. T. Hagan. A comparison of neural network control algorithms. In *Neural Networks, 2001. Proceedings. IJCNN '01. International Joint Conference on*, volume 1, pages 521–526 vol.1, 2001.
16. J. Kaloust, C. Ham, J. Siehling, E. Jongekryg, and Q. Han. Nonlinear robust control design for levitation and propulsion of a maglev system. *IEE Proceedings - Control Theory and Applications*, 151(4):460–464, July 2004.
17. Guy Katz, Clark Barrett, David Dill, Kyle Julian, and Mykel Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. *arXiv preprint arXiv:1702.01135*, 2017.
18. C. H. Kim, J. Lim, J. M. Lee, H. S. Han, and D. Y. Park. Levitation control design of super-speed maglev trains. In *2014 World Automation Congress (WAC)*, pages 729–734, Aug 2014.
19. Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE Transactions on Neural Networks*, 8(1):98–113, 1997.
20. Xiao-Dong Li, J. K. L. Ho, and T. W. S. Chow. Approximation of dynamical time-variant systems by continuous-time recurrent neural networks. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 52(10):656–660, Oct 2005.
21. L. S. H. Ngia and J. Sjoberg. Efficient training of neural nets for nonlinear adaptive filtering using a recursive levenberg-marquardt algorithm. *IEEE Transactions on Signal Processing*, 48(7):1915–1927, Jul 2000.
22. M. Ono, S. Koga, and H. Ohtsuki. Japan's superconducting maglev train. *IEEE Instrumentation Measurement Magazine*, 5(1):9–15, Mar 2002.

23. Luca Pulina and Armando Tacchella. An abstraction-refinement approach to verification of artificial neural networks. In *International Conference on Computer Aided Verification*, pages 243–257. Springer, 2010.
24. Luca Pulina and Armando Tacchella. Challenging SMT solvers to verify neural networks. *AI Communications*, 25(2):117–135, 2012.
25. D. M. Rote and Yigang Cai. Review of dynamic stability of repulsive-force maglev suspension systems. *IEEE Transactions on Magnetics*, 38(2):1383–1390, Mar 2002.
26. Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.
27. David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
28. Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.
29. Mai Viet Thuan, Hieu Manh Tran, and Hieu Trinh. Reachable sets bounding for generalized neural networks with interval time-varying delay and bounded disturbances. *Neural Computing and Applications*, pages 1–12, 2016.
30. R. Uswarman, A. I. Cahyadi, and O. Wahyunggoro. Control of a magnetic levitation system using feedback linearization. In *2013 International Conference on Computer, Control, Informatics and Its Applications (IC3INA)*, pages 95–98, Nov 2013.
31. R. J. Wai and J. D. Lee. Robust levitation control for linear maglev rail system using fuzzy neural network. *IEEE Transactions on Control Systems Technology*, 17(1):4–14, Jan 2009.
32. Weiming Xiang. On equivalence of two stability criteria for continuous-time switched systems with dwell time constraint. *Automatica*, 54:36–40, 2015.
33. Weiming Xiang. Necessary and sufficient condition for stability of switched uncertain linear systems under dwell-time constraint. *IEEE Transactions on Automatic Control*, 61(11):3619–3624, 2016.
34. Weiming Xiang. Parameter-memorized Lyapunov functions for discrete-time systems with time-varying parametric uncertainties. *Automatica*, 87:450–454, 2018.
35. Weiming Xiang, James Lam, and Jun Shen. Stability analysis and $\mathcal{L}_1$-gain characterization for switched positive systems under dwell-time constraint. *Automatica*, 85:1–8, 2017.
36. Weiming Xiang, Hoang-Dung Tran, and T. T. Johnson. Robust exponential stability and disturbance attenuation for discrete-time switched systems under arbitrary switching. *IEEE Transactions on Automatic Control*, 2017, doi: 10.1109/TAC.2017.2748918.
37. Weiming Xiang, Hoang-Dung Tran, and Taylor T Johnson. On reachable set estimation for discrete-time switched linear systems under arbitrary switching. In *American Control Conference (ACC), 2017*, pages 4534–4539. IEEE, 2017.
38. Weiming Xiang, Hoang-Dung Tran, and Taylor T Johnson. Output reachable set estimation and verification for multi-layer neural networks. *arXiv preprint arXiv:1708.03322*, 2017.
39. Weiming Xiang, Hoang-Dung Tran, and Taylor T Johnson. Output reachable set estimation for switched linear systems and its application in safety verification. *IEEE Transactions on Automatic Control*, 62(10):5380–5387, 2017.

40. Weiming Xiang, Hoang-Dung Tran, and Taylor T Johnson. Reachable set computation and safety verification for neural networks with ReLU activations. *arXiv preprint arXiv: 1712.08163*, 2017.
41. Weiming Xiang and Jian Xiao. Stabilization of switched continuous-time systems with all modes unstable via dwell time switching. *Automatica*, 50(3):940–945, 2014.
42. Zhaowen Xu, Hongye Su, Peng Shi, Renquan Lu, and Zheng-Guang Wu. Reachable set estimation for Markovian jump neural networks with time-varying delays. *IEEE Transactions on Cybernetics*, 47(10):3208–3217, 2017.
43. Lixian Zhang and Weiming. Xiang. Mode-identifying time estimation and switching-delay tolerant control for switched systems: An elementary time unit approach. *Automatica*, 64:174–181, 2016.
44. Lixian Zhang, Yanzheng Zhu, and Wei Xing Zheng. Synchronization and state estimation of a class of hierarchical hybrid neural networks with time-varying delays. *IEEE Transactions on Neural Networks and Learning Systems*, 27(2):459–470, 2016.
45. Lixian Zhang, Yanzheng Zhu, and Wei Xing Zheng. State estimation of discrete-time switched neural networks with multiple communication channels. *IEEE Transactions on Cybernetics*, 47(4):1028–1040, 2017.
46. S. T. Zhao and X. W. Gao. Neural network adaptive state feedback control of a magnetic levitation system. In *The 26th Chinese Control and Decision Conference (2014 CCDC)*, pages 1602–1605, May 2014.
47. Zhiqiang Zuo, Zhenqian Wang, Yinping Chen, and Yijing Wang. A non-ellipsoidal reachable set estimation for uncertain neural networks with time-varying delay. *Communications in Nonlinear Science and Numerical Simulation*, 19(4):1097–1106, 2014.