

# Reachability Analysis of Neural Feedback Loops using Sparse Polynomial Optimisation

Matthew Newton and Antonis Papachristodoulou

**Abstract**—Neural networks have seen a recent increased use in control feedback systems. However, providing robustness guarantees on these feedback systems has proven challenging and to combat these issues, there is a significant amount of current research. One of the biggest shortcoming of neural networks is how sensitive they are to adversarial inputs. Given that feedback systems are usually subject to external perturbations, this issue surrounding neural networks must be overcome before they can be used in safety-critical applications. One method to tackle this problem is to compute outer-approximations of the reachable sets, through bounding the activation functions in the neural network controller. Our approach is to use these bounds in a sparse polynomial optimisation framework in conjunction with the Positivstellensatz. The sparsity property is able to exploit the natural cascading structure of the neural network to allow for tractable solve times. The Positivstellensatz is able to provide more accurate bounds over similar methods by asserting the emptiness of a semi-algebraic set. We show through examples that our method can provide tighter bounds over similar methods, with reasonable computational time. Our approach is also able to deal with non-linear polynomial dynamics due to the polynomial optimisation framework we use, while other methods need to use alternative work-around solutions to incorporate the non-linearities.

## I. INTRODUCTION

Due to the increased use of neural networks (NNs) in many applications over the past decade, research in NNs has never been more important. The step increase provided by Alexnet [1] and Resnet [2], demonstrated the capacity for NNs to perform highly complex tasks. In addition, the increase in the availability of ‘big data’ and computational power has led to the increased use of NNs in industrial applications. The success of NNs in general applications has prompted their use in feedback control systems, which are referred to as neural feedback loops (NFLs). Systems of this type have huge potential in safety-critical applications, such as autonomous vehicle technology. However, before they can be implemented, one of the biggest limiting factors of this technology must be overcome. This is a NN’s sensitivity to adversarial inputs, where small changes in the input set can lead to undesirable behaviour in the output set.

Despite the recent resurgence of interest, there is work on NNs in the field of control theory dating back to the 1990s [3]. Amplifying the interest in this field is the emergence of the parallel area of reinforcement learning, which takes a data-based approach to optimal control. By harnessing

the power of deep reinforcement learning, decision-making agents have been able to greatly outperform humans in complex tasks such as the video game Dota 2 [4] and the board game Go [5]. New advances in robust control methods, combined with the success of the reinforcement learning community, motivates work at their intersection.

This paper focuses on the NFL reachability problem in discrete time, where an NN controller is tasked to take the states from an initial to a goal set, whilst avoiding certain regions. Many recent works such as [6] and [7] address this problem by placing bounds on the non-linear activation functions in the NN controller. These bounds can be too loose, meaning that it is not possible to verify the initial set if it becomes too large. Another issue is that these methods often assume that a linear plant is being controlled, which is restrictive when the plant has non-linear dynamics. Work from [8] introduces an efficient method to partition the input sets, that scales well to large NN. They are able to incorporate the non-linear polynomial dynamics by introducing new variables into the optimisation framework, although this does not account for the non-linearities directly.

To overcome these issues we adapt a framework from [9] and incorporate it within the NFL reachability problem. This approach uses sparse polynomial optimisation methods from [10] along with a theory called the Positivstellensatz (Psatz) to assert the emptiness of a carefully constructed semi-algebraic set. We then search for the emptiness of this set using the algebraic formulation to find bounds on the reachable set, using sum of squares (SOS) and semidefinite programming (SDP). One advantage of this framework is that we can easily trade off solution accuracy with computational complexity. Due to the polynomial optimisation construction, we can include any non-linear polynomial dynamics in the plant model, which has not been possible in previous frameworks. This method uses an observation in [11] and [12], which exploits the natural cascading structure of the NN that is inherited in the algorithm formulation. We implement this method on example NFLs and show through numerical experiments the advantages of our method.

In Section II we provide a summary of the preliminary information surrounding the NFL reachability problem, then in Section III we describe how the problem is formulated. Section IV provides a summary of sparse polynomial optimisation and how it links with SOS programming. Section V shows how the NFL reachability problem can be posed as a sparse polynomial optimisation problem. The experimental results are shown in Section VI and the paper is concluded in Section VII.

This work was supported by EPSRC grants EP/L015897/1 (to M. Newton) and EP/M002454/1 (to A. Papachristodoulou).

M. Newton and A. Papachristodoulou are with the Department of Engineering Science, University of Oxford, Parks Road, Oxford, OX1 3PJ, U.K. {matthew.newton, antonis}@eng.ox.ac.uk

## II. PRELIMINARIES

### A. Neural Feedback Loops

Consider a discrete-time time-invariant system

$$z_{t+1} = f(z_t, u_t), \quad (1)$$

where  $f$  is a known Lipschitz function and  $z_t \in \mathbb{R}^{n_z}$ ,  $u_t \in \mathbb{R}^{n_u}$  are the system states and control inputs respectively. Here  $\mathbb{R}^n$  denotes the set of real  $n$ -dimensional vectors. We consider a state feedback controller  $u_t = \pi(z_t) : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_u}$  as a feed-forward fully-connected NN, which is governed by the set of equations:

$$\begin{aligned} x^0 &= z_t, \\ v^k &= W^k x^k + b^k, \text{ for } k = 0, \dots, \ell - 1, \\ x^{k+1} &= \phi(v^k), \text{ for } k = 0, \dots, \ell - 1, \\ \pi(z_t) &= W^\ell x^\ell + b^\ell, \end{aligned} \quad (2)$$

where  $W^k \in \mathbb{R}^{n_{k+1} \times n_k}$ ,  $b^k \in \mathbb{R}^{n_{k+1}}$  are the weights matrix and bias of the  $(k+1)^{th}$  layer respectively and  $z_t = x^0 \in \mathbb{R}^{n_z}$  is the input into the NN. Note that while each  $x^k$  and  $v^k$  should be indexed by  $t$ , we drop this notation for convenience. The number of neurons in the  $k^{th}$  layer is denoted by  $n_k$ . The activation function  $\phi$  is applied element-wise to the  $v^k$  terms such that

$$\phi(v^k) := [\phi(v_1^k), \dots, \phi(v_{n_k}^k)]^T, \quad v^k \in \mathbb{R}^{n_k},$$

where  $v_j^k$  is the pre-activation value.

### B. Reachability Problem

For the closed loop system in (1), the forward reachable set at time-step  $t$  from a set of given initial conditions  $\mathcal{Z}_0 \subseteq \mathbb{R}^{n_z}$  is denoted as  $\mathcal{R}_t(\mathcal{Z}_0)$ . This recursion is defined by

$$\mathcal{R}_{t+1}(\mathcal{Z}_0) = f(\mathcal{R}_t(\mathcal{Z}_0)), \quad \mathcal{R}_0(\mathcal{Z}_0) = \mathcal{Z}_0.$$

This can be used to formulate the finite-time reach-avoid verification problem. This states that given a goal set  $\mathcal{G} \subseteq \mathbb{R}^{n_z}$ , an initial state  $z_0 \in \mathcal{Z}_0$  and a sequence of avoid sets  $\mathcal{A} \subseteq \mathbb{R}^{n_z}$ , we determine if  $z_0$  can reach  $\mathcal{G}$  in finite time while avoiding  $\mathcal{A}_t$  for all  $t$  in the finite time period. This problem is equivalent to testing if the following are true:

$$\begin{aligned} \mathcal{R}_N(\mathcal{Z}_0) &\subseteq \mathcal{G}, \\ \mathcal{R}_t(\mathcal{Z}_0) \cap \mathcal{A}_t &= \emptyset, \forall t = 0, \dots, N, \end{aligned} \quad (3)$$

where  $N$  is the finite time horizon. Since NFLs contain nonlinearities in the activation functions, computing the exact reachable set can be computationally intractable. We can therefore instead compute a convex outer-approximation of the reachable set denoted by  $\bar{\mathcal{R}}_t(\mathcal{Z}_0) \supseteq \mathcal{R}_t(\mathcal{Z}_0)$ . This is useful since if we can verify that  $\bar{\mathcal{R}}_t(\mathcal{Z}_0)$  satisfies (3) then we know that  $\mathcal{R}_t(\mathcal{Z}_0)$  does as well and hence the non-linear NFL is verified. It is desirable that the size of  $\bar{\mathcal{R}}_t(\mathcal{Z}_0)$  is as small as possible as it increases the chance of the system satisfying (3). Therefore, we can pose this as an optimisation problem such that:

$$\begin{aligned} &\text{minimize} \quad \text{Volume}(\bar{\mathcal{R}}_t(\mathcal{Z}_0)), \\ &\text{subject to} \quad \mathcal{R}_t(\mathcal{Z}_0) \subseteq \bar{\mathcal{R}}_t(\mathcal{Z}_0). \end{aligned}$$

This can be solved using  $N+1$  optimisation problems by first optimising  $\bar{\mathcal{R}}_0(\mathcal{Z}_0)$  given  $\mathcal{Z}_0$  and then recursively computing the next time step until all  $N$  time steps are solved.

### C. Neural Network Constraints

To solve each optimisation problem, we form a set of constraints based on the input set and the NN structure and then minimize the output set through defining an appropriate objective function. We constrain the input set to be bounded by a polytope such that

$$\mathcal{Z}_t = \bigcap_{m=1}^{M_{in}} \{z_t \in \mathbb{R}^{n_z} \mid d_m - c_m^T z_t \geq 0\},$$

where  $c_m \in \mathbb{R}^{n_z}$ ,  $d_m \in \mathbb{R}$  and  $M_{in}$  are the parameters and the number of faces of the polytope respectively. The reachable set  $\bar{\mathcal{R}}_t(\mathcal{Z}_0)$  that represents the state at the next time step  $z_{t+1}$  can also be bounded by a polytope such that

$$\mathcal{Z}_{t+1} = \bigcap_{m=1}^{M_{out}} \{z_{t+1} \in \mathbb{R}^{n_z} \mid b_m - a_m^T z_{t+1} \geq 0\},$$

where  $a_m \in \mathbb{R}^{n_z}$  are given parameters of the polytope and  $b_m$  can be set as an optimisation variable that is to be optimised. Each reachability problem at time-step  $t$  can be split in to a further  $M_{out}$  optimisation problems, where each problem aims to push one side of the polytope as close as possible to the exact reachable set.

Recall from (1) that  $z_{t+1} = f(z_t, \pi(z_t))$ . We therefore need to create a set of constraints that represent the NN in (2) to bound the controller input  $\pi(z_t)$ . The constraints on the hidden layers can be represented by relationships between  $\phi(v_j^k)$  and  $v_j^k$ , which depend on the activation function used in the NN. We denote the output of the activation function  $\phi(v_j^k)$  as  $\phi$ , and the input to the activation function  $v_j^k$  as  $x$  to simplify the notation. It is possible to compute a conservative approximation of the bounds using an efficient pre-processing step to find the minimum and maximum bounds on the activation function,  $(\underline{\phi}, \bar{\phi})$  such that

$$\phi - \underline{\phi} \geq 0, \quad -\phi + \bar{\phi} \geq 0. \quad (4)$$

These box constraints are used in the optimisation problem and the preprocessing values can also be combined to make other constraints tighter. The method we use to do this is interval bound propagation (IBP) [13].

The ReLU function is commonly used in NNs and is given by  $\text{ReLU}(x) = \phi(x) = \max(0, x)$ . Tight bounds on the ReLU function can be formed using two inequalities and one equality constraint [14] such that

$$\phi \geq 0, \quad \phi - x \geq 0, \quad \phi(\phi - x) = 0.$$

We can further tighten these constraints by using the approximations from the IBP step. If  $\bar{\phi} \leq 0$ , then the first inequality constraint is replaced with an equality constraint such that  $\phi = 0$ . Equivalently if  $\underline{\phi} > 0$ , the second inequality constraint is replaced with  $\phi - x = 0$ .

For the sigmoid and tanh activation functions it is possible to create two overlapping sector constraints using the IBP values. [15] provides more details about these constraints.

### III. PROBLEM FORMULATION

To use the constraints described in Section II-C, we propose a sparse polynomial optimisation framework in conjunction with the Positivstellensatz (Psatz), which provides an algebraic condition to certify the emptiness of a semi-algebraic set. We summarise here the key concepts that are used in this framework; for more details please refer to [9].

#### A. Positivstellensatz

A semi-algebraic set takes the form

$$S = \{x \in \mathbb{R}^n \mid g_i(x) \geq 0, h_j(x) = 0, \forall i = 1, \dots, p, j = 1, \dots, q\}. \quad (5)$$

where  $g_i$  and  $h_j$  are polynomial functions. We define  $\mathbb{R}[x]$  to be the set of polynomials in  $x$  with real coefficients.

**Definition 1:** A polynomial  $p(x) \in \mathbb{R}[x]$  is said to be a SOS polynomial if there exist  $r_i(x) \in \mathbb{R}[x]$ , such that

$$p(x) = \sum_i r_i^2(x) \equiv p(x) \text{ is SOS.}$$

We denote the set of polynomials that admit this decomposition by  $\Sigma[x]$ .

**Definition 2:** The cone of a set of polynomials is defined

$$\text{cone}\{g_1, \dots, g_p\} = \left\{ \sum_{i=1}^p s_i g_i \mid s_i \in \Sigma[x], g_i \in \mathbb{R}[x] \right\}.$$

**Definition 3:** The ideal of a set of polynomials is defined

$$\text{ideal}\{h_1, \dots, h_q\} = \left\{ \sum_{j=1}^q t_j h_j \mid t_j, h_j \in \mathbb{R}[x] \right\}.$$

**Theorem 1:** (Positivstellensatz [16]) Given the semi-algebraic set  $S$  defined in (5), the following are equivalent:

- 1) The set  $S$  is empty.
- 2) There exist  $s_i \in \Sigma[x]$  and  $t_j \in \mathbb{R}[x]$  such that  $-1 \in \text{cone}\{g_1, \dots, g_p\} + \text{ideal}\{h_1, \dots, h_q\}$  (see Definitions 2 and 3).

Theorem 1 links the emptiness of a semi-algebraic set with an algebraic test, which can be tested in conjunction with SOS and semidefinite programming (SDP).

#### B. Neural Feedback Loop Reachability Emptiness Condition

To construct the NFL reachability problem we modify the Psatz condition to use it in the optimisation framework. Instead of optimising the output polytope faces in (II-C) such that  $b_m - a_m^T z_{t+1} \geq 0$  is feasible, we can show that  $-b_m + a_m^T z_{t+1} \geq 0$  is infeasible using the Psatz. We introduce the optimisation decision variable  $\gamma_m$  to approximate  $b_m$  and find the limiting values as to when this Psatz condition is violated. The optimisation problem for a given time step  $t$  and face  $m$  of the output polytope is given by (6). Here  $h_j$  and  $g_i$  are the equality and inequality constraints respectively. These constraints are generated from the input set and hidden layers as described in Section II-C. Solving this problem can be achieved computationally using parsers such as SOSTOOLS [17] and a semidefinite programming solver. By choosing higher degree multiplier  $s_i, t_j$  etc., we can create a series of nested tests of increasing complexity and non-decreasing accuracy.

### IV. SPARSE POLYNOMIAL OPTIMISATION AND SUM OF SQUARES

#### A. Sum of Squares

SOS conditions are useful since they can be expressed as linear matrix inequality constraints and consequently solved using SDP [18]. Checking the nonnegativity of a polynomial is known to be an NP-hard problem [19], so instead we can check if a polynomial is SOS by solving a semidefinite program in polynomial time. To do this, consider a monomial vector containing a selection of the variables  $x = [x_1, \dots, x_n]$ . A monomial defined by all  $n$  variables is written as  $x^\beta = x_1^{\beta_1} x_2^{\beta_2} \dots x_n^{\beta_n}$ , where the degree and exponent are denoted as  $|\beta| = \beta_1 + \dots + \beta_n$  and  $\beta = (\beta_1, \dots, \beta_n) \in \mathbb{N}^n$  respectively.  $\mathbb{N}^n$  denotes the set of  $n$ -dimensional integers. The column vector of monomials with only certain exponents is expressed as  $x^\mathbb{B} = (x^\beta)_{\beta \in \mathbb{B}}$ , where  $\mathbb{B} \subset \mathbb{N}^n$  is the set of exponents that are used in the monomials. Note that any polynomial  $f$  can be written as  $f = \sum_{\beta \in \mathbb{N}_d^n} f_\beta x^\beta$  for a set of coefficients  $f_\beta \in \mathbb{R}$ , where  $\mathbb{N}_d^n = \{\beta \in \mathbb{N}^n : |\beta| \leq d\}$  is the set of all  $n$ -variate exponents of degree  $d$  or less. We also define the summation operation on  $\mathbb{B}$  as

$$\mathbb{B} + \mathbb{B} := \{\beta + \gamma : \beta, \gamma \in \mathbb{B}\}.$$

The polynomial  $f$  is SOS if and only if it can be written as  $f = (x^\mathbb{B})^T Q x^\mathbb{B}$ , where  $Q \in \mathbb{S}_+^{|\mathbb{B}|}$  is a positive semidefinite matrix; this is referred to as the Gram matrix representation. Note that in general if a polynomial is found to be SOS then it is only a sufficient condition for global non-negativity. The Gram matrix representation can be converted into SDP constraints by first defining the symmetric binary matrix  $A_\alpha \in \mathbb{S}^{|\mathbb{B}|}$  for each exponent  $\alpha \in \mathbb{B} + \mathbb{B}$  as

$$[A_\alpha]_{\beta, \gamma} := \begin{cases} 1, & \beta + \gamma = \alpha, \\ 0, & \text{otherwise.} \end{cases}$$

We can rewrite the Gram representation as

$$(x^\mathbb{B})^T Q x^\mathbb{B} = \langle Q, x^\mathbb{B} (x^\mathbb{B})^T \rangle = \sum_{\alpha \in \mathbb{B} + \mathbb{B}} \langle Q, A_\alpha \rangle x^\alpha.$$

Therefore, the following is true

$$f \in \Sigma[x] \Leftrightarrow \exists Q \in \mathbb{S}_+^{|\mathbb{B}|} \text{ such that } \langle Q, A_\alpha \rangle = f_\alpha, \forall \alpha \in \mathbb{B} + \mathbb{B}.$$

Without using sparsity the vector of monomials  $x^\mathbb{B}$  contains all monomials up to degree  $d$  and the matrix  $Q$  is a fully dense matrix of size  $\binom{n+d}{d} \times \binom{n+d}{d}$ . However, in a system that possesses a significant level of sparsity then it is expected that not all of the monomial terms are needed. Hence if we only include a subset of them, the size of the SDP is reduced; this is where theory from chordal graphs becomes useful.

#### B. Sparse Matrix Decomposition and Chordal Graphs

A graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$  is defined as a set of vertices  $\mathcal{V} = \{1, 2, \dots, n\}$  and a set of edges  $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ .

**Definition 4:** [20] A connected undirected graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$  is a chordal graph if every cycle of length four or greater has at least one chord.

minimize  $\gamma_m$ ,

$$\begin{aligned} \text{subject to} \quad & -a_m^T z_{t+1} + \gamma_m - \sum_j^q t_j h_j - \sum_i^p s_i g_i - \sum_{i \neq j}^p r_{ij} g_i g_j - \dots \text{ is SOS,} \\ & s_i \text{ is SOS, } \forall i = 1, \dots, p, \quad r_{ij} \text{ is SOS, } \forall i, j = 1, \dots, p, \quad t_j \in \mathbb{R}[x], \forall j = 1, \dots, q. \end{aligned} \quad (6)$$

A clique  $\mathcal{C} \subseteq \mathcal{V}$  is a subgraph such that all the vertices in the subgraph  $\mathcal{C}$  form a complete graph. A maximal clique is a clique that is not a subset of any other clique. Chordal graphs are useful since they can be decomposed into their maximal cliques [21]. Consider now a symmetric matrix  $X \in \mathbb{S}^n$  with a sparsity pattern represented by an undirected graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$ , such that  $X_{ij} = X_{ji} = 0$ ,  $\forall i \neq j$  if  $(i, j) \notin \mathcal{E}$ . Just as a chordal graph can be decomposed into its maximal cliques, a matrix  $X$  with a chordal sparsity pattern can be split up into smaller sub-matrices, with the sub-matrices corresponding to the maximal cliques of the chordal graph.

**Theorem 2:** [22] Consider the chordal graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$  that is made up of maximal cliques  $\{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_t\}$ . Then  $Z \in \mathbb{S}_+^n(\mathcal{E}, 0)$  if and only if there exist  $Z_k \in \mathbb{S}_+^{|\mathcal{C}_k|}$  for  $k = 1, \dots, t$  such that

$$Z = \sum_{k=1}^t E_{\mathcal{C}_k}^T Z_k E_{\mathcal{C}_k},$$

where  $\mathbb{S}_+^n(\mathcal{E}, 0) := \{X \in \mathbb{S}^n \mid X \succeq 0 \mid X_{ij} = X_{ji} = 0, \text{ if } i \neq j \wedge (i, j) \notin \mathcal{E}\}$ ,  $|\mathcal{C}_i|$  is the number of vertices in clique  $i$  and

$$(E_{\mathcal{C}_k})_{ij} = \begin{cases} 1, & \text{if } \mathcal{C}_k(i) = j \\ 0, & \text{otherwise.} \end{cases}$$

There are many techniques that this sparsity property can be used with in SOS programming, such as term and correlative sparsity. The details of this will be omitted from this paper for conciseness; for a comprehensive overview the authors recommend referring to [23]. In this paper, we use the Correlative Sparsity - Term Sparse Sum of Squares (CS-TSSOS) hierarchy [10] to construct the sparsity pattern.

## V. SPARSE REACHABILITY CONSTRAINTS

As alluded to previously, one issue with using an SDP framework is that the solve time becomes intractable as the size of the NN controller increases. Note that the structure of the NN in (2) is such that each layer is only connected directly to the adjacent layers, therefore the NN possesses a natural cascading structure. Consequently constraints used to bound the activation functions only contain variables in the current layer and the previous layer. Therefore the constraint matrices have a well-defined structure and the cascading is inherited in the algorithm formulation. In [9] this idea is explored in depth, where the sparsity pattern is shown to match that of the constraints in the SOS hierarchy.

We can use this result for the NFL reachability problem since the sparsity pattern generated by the constraints from the NN is the same. However, instead of bounding the output of the NN as in the NN verification problem, we are bounding the  $z_{t+1}$  state given a bound on  $z_t$ . As in (1) the  $z_{t+1}$  state is a function of both the input and output of the

NN controller. Therefore, the sparse structure in the graph remains and it is possible to create an extension of the graph that is chordal. This allows the use of sparsity techniques to decompose the resulting optimisation problem. The sparsity is shown in Figure 1.

In this paper, we consider the Psatz condition for the NFL reachability problem where no inequality constraints are multiplied together, hence the sparse SOS condition is:

$$-a_m^T z_{t+1} + \gamma_m - \sum_j^q t_j h_j - \sum_i^p s_i g_i \text{ is CS - TSSOS,}$$

$$s_i \text{ is SOS, } \forall i = 1, \dots, p, \quad t_j \in \mathbb{R}[x], \forall j = 1, \dots, q,$$

where  $h_j$  and  $g_i$  are the equality and inequality constraints respectively.

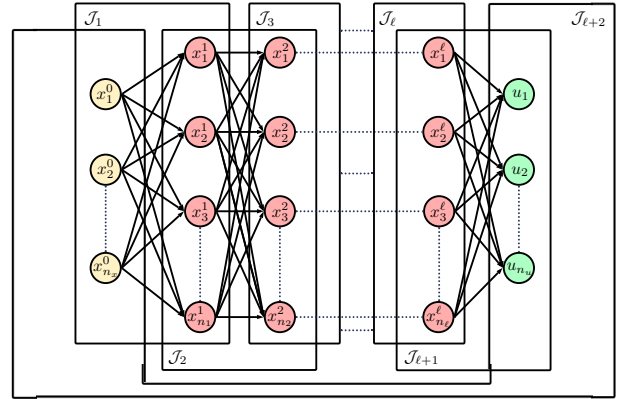


Fig. 1: Diagram of an NN controller showing how the cliques  $\mathcal{J}_k$  are formed. The yellow nodes are the states  $z_t$ , the red nodes are the hidden layers and the green nodes are the control inputs  $u_t$ . The clique  $\mathcal{J}_{l+2}$  represents the connections between the  $z_t$  states and  $u_t$ , however it is not illustrated on the diagram for simplicity.

## VI. EXPERIMENTAL RESULTS

We demonstrate our method on various systems to show its advantages and capabilities. Our experiments were run on a 4-core Intel Xeon processor @3.50GHz with 16GB of RAM. Our method is referred to as ‘ReachSparsePatz’, which is implemented in Julia where the semi-algebraic constraint set is constructed. The optimisation problem is implemented with the CS-TSSOS hierarchy [10] using the TSSOS Julia package [24], which constructs the SDP constraints and parses them to the SDP solver MOSEK [25]. Note that without using sparsity the SDP for these problems is intractable. The true reachable sets are computed using an exhaustive search on each input set.



### A. Duffing Oscillator

One advantage of this polynomial optimisation framework is that we can use any system (1) where  $f$  is a polynomial. This is not possible on similar methods such as [7] and [8], which would require one to either linearise the system or create an alternative solution to bound the non-linearity. To demonstrate this we take the Duffing oscillator example proposed in [8] written as:

$$\begin{aligned}\dot{z}_1 &= z_2, \\ \dot{z}_2 &= -z_1 - 2\zeta z_2 - z_1^3 + u,\end{aligned}$$

where  $[z_1, z_2]^T \in \mathbb{R}^2$  and  $u \in \mathbb{R}$  are the states and control inputs respectively. The damping coefficient is set to  $\zeta = 0.3$  and we discretized the system with sampling time  $\Delta t = 0.3$ . We fit a  $5 \times 5$  NN with ReLU activation functions as the controller of the system. Figure 2 shows that our method is able to obtain tight bounds on this non-linear system. This system is stable without a controller, however we have been able to show that we can effectively compute the reachable sets on a NFL with a non-linear plant model.

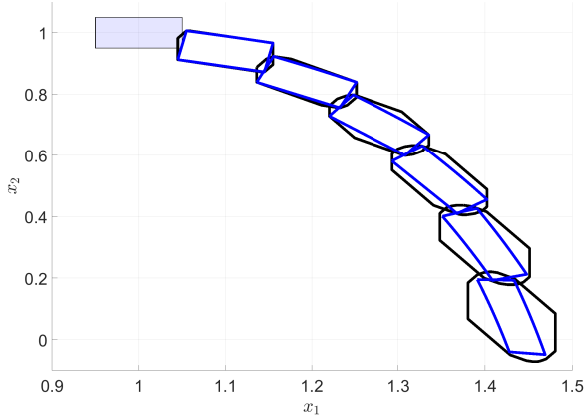


Fig. 2: Reachable sets of the Duffing oscillator (Section VI-A) using ReachSparsePsatz (black) and the true values (blue).

### B. Unstable Polynomial System

We now consider a unstable system with polynomial dynamics given by

$$\begin{aligned}\dot{z}_1 &= -z_1(0.1 + (z_1 + z_2)^2), \\ \dot{z}_2 &= (u + z_1)(0.1 + (z_1 + z_2)^2).\end{aligned}$$

We discretise the system with time step  $\Delta t = 0.5$  and train a large NN controller with 100 layers and three nodes in each layer. Figure 3 shows the reachable steps that are computed at each time step.

### C. Inverted Pendulum

We take the inverted pendulum example from [26] and analyse reachability of this system. The inverted pendulum's dynamics can be written in the form:

$$\begin{aligned}\ddot{\theta}(t) &= \frac{-mglq(t) + mgl\theta(t) - \mu\dot{\theta}(t) + \text{sat}(u(t))}{ml^2}, \\ q(t) &= \theta(t) - \sin(\theta(t)).\end{aligned}$$

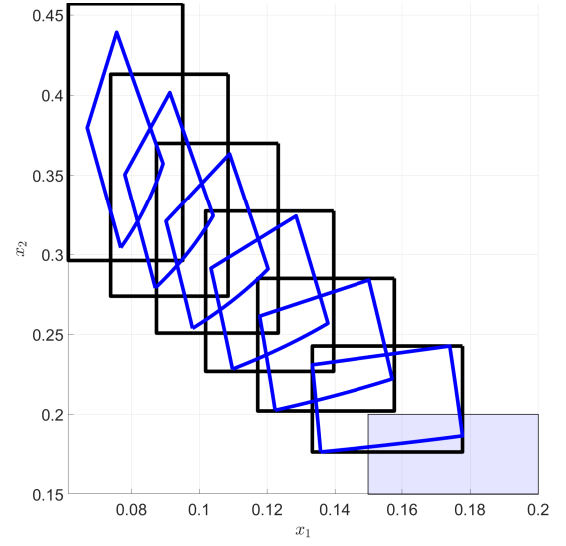


Fig. 3: Reachable sets of the Unstable polynomial system (Section VI-B) using ReachSparsePsatz (black) and the true values (blue).

We discretise the system with time step  $\Delta t = 0.2$ . The system is parameterised by  $m = 0.15\text{kg}$ ,  $l = 0.5\text{m}$ ,  $\mu = 0.5\text{Nm/s/rad}$ ,  $g = 10\text{m/s}^2$  and the saturation limit in the saturation function for the controller is set to  $u_{max} = 1$ . We bound the non-linearity  $q(t)$  by a sector constraint that is determined by the region of each reachable set. We use the NN controller generated in [26], which contains two hidden layers and five nodes in each layer with tanh activation functions. The benefit of our method is the flexibility of being able to add the sector constraint along with the hidden layer constraints. Figure 4, shows the reachable sets when the inverted pendulum is released from the initial condition.

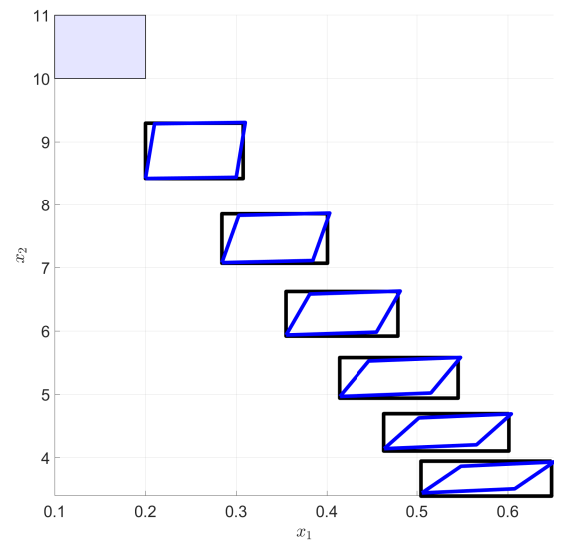


Fig. 4: Reachable sets of the inverted pendulum (Section VI-C) using ReachSparsePsatz (black) compared with the true values of a NFL (blue) with tanh activation functions.

## VII. CONCLUSION

This paper proposes a method for computing the reachable sets for NFLs using sparse polynomial optimisation in conjunction with the Positivstellensatz. Our approach uses tight constraints on the ReLU, sigmoid and tanh activation functions in the NN controller to obtain tight polytope over-approximations of the reachable sets. The key advantages of this method are the ability to compute these sets in a scalable and accurate way, whilst also being able to incorporate polynomial dynamics. The benefits of this method are demonstrated through various examples. Opportunities for future work in this area include using this method to analyse other robustness properties of NFLs, such as input-output properties and backward reachable sets.

## REFERENCES

- [1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Image-net classification with deep convolutional neural networks,” *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.
- [2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [3] W. T. Miller, P. J. Werbos, and R. S. Sutton, *Neural networks for control*. MIT press, 1995.
- [4] C. Berner *et al.*, “Dota 2 with large scale deep reinforcement learning,” *arXiv preprint arXiv:1912.06680*, 2019.
- [5] D. Silver *et al.*, “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, pp. 484–503, 2016.
- [6] C. Huang, J. Fan, W. Li, X. Chen, and Q. Zhu, “Reachnn: Reachability analysis of neural-network controlled systems,” *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, pp. 1–22, 2019.
- [7] H. Hu, M. Fazlyab, M. Morari, and G. J. Pappas, “Reach-sdp: Reachability analysis of closed-loop systems with neural network controllers via semidefinite programming,” in *59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 5929–5934.
- [8] M. Everett, G. Habibi, C. Sun, and J. P. How, “Reachability analysis of neural feedback loops,” *IEEE Access*, vol. 9, pp. 163 938–163 953, 2021.
- [9] M. Newton and A. Papachristodoulou, “Sparse polynomial optimisation for neural network verification,” *arXiv preprint arXiv:2202.02241*, 2022.
- [10] J. Wang, V. Magron, J. B. Lasserre, and N. H. A. Mai, “CS-TSSOS: Correlative and term sparsity for large-scale polynomial optimization,” *arXiv preprint arXiv:2005.02828*, 2020.
- [11] M. Newton and A. Papachristodoulou, “Exploiting sparsity for neural network verification,” in *Learning for Dynamics and Control*, PMLR, 2021, pp. 715–727.
- [12] B. Battern, P. Kouvaros, A. Lomuscio, and Y. Zheng, “Efficient neural network verification via layer-based semidefinite relaxations and linear cuts,” in *International Joint Conference on Artificial Intelligence (IJCAI21)*, 2021, pp. 2184–2190.
- [13] S. Gawal, K. Dvijotham, R. Stanforth, *et al.*, “On the effectiveness of interval bound propagation for training verifiably robust models,” *arXiv preprint arXiv:1810.12715*, 2018.
- [14] A. Raghunathan, J. Steinhardt, and P. Liang, “Semidefinite relaxations for certifying robustness to adversarial examples,” *arXiv preprint arXiv:1811.01057*, 2018.
- [15] M. Newton and A. Papachristodoulou, “Neural network verification using polynomial optimisation,” in *Proceedings of the 60th Conference on Decision and Control*, 2021.
- [16] G. Stengle, “A nullstellensatz and a positivstellensatz in semialgebraic geometry,” *Mathematische Annalen*, vol. 207, no. 2, pp. 87–97, 1974.
- [17] A. Papachristodoulou, J. Anderson, G. Valmorbida, S. Prajna, P. Seiler, and P. Parrilo, “Sostools version 3.00 sum of squares optimization toolbox for matlab,” *arXiv preprint arXiv:1310.4716*, 2013.
- [18] P. A. Parrilo, *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. California Institute of Technology, 2000.
- [19] K. G. Murty and S. N. Kabadi, “Some np-complete problems in quadratic and nonlinear programming,” *Tech. Rep.*, 1985.
- [20] N. Kakimura, “A direct proof for the matrix decomposition of chordal-structured positive semidefinite matrices,” *Linear Algebra and its Applications*, vol. 433, no. 4, pp. 819–823, 2010.
- [21] L. Vandenberghe and M. S. Andersen, *Chordal Graphs and Semidefinite Optimization*. 2015.
- [22] J. Agler, W. Helton, S. McCullough, and L. Rodman, “Positive semidefinite matrices with a given sparsity pattern,” *Linear Algebra and its Applications*, vol. 107, pp. 101–149, 1988.
- [23] Y. Zheng, G. Fantuzzi, and A. Papachristodoulou, “Chordal and factor-width decompositions for scalable semidefinite and polynomial optimization,” *Annual Reviews in Control*, 2021.
- [24] V. Magron and J. Wang, “TSSOS: A julia library to exploit sparsity for large-scale polynomial optimization,” *arXiv preprint arXiv:2103.00915*, 2021.
- [25] A. MOSEK, *MOSEK optimization toolbox for MATLAB. release 9.2. 40*, 2021.
- [26] P. Pauli, D. Gramlich, J. Berberich, and F. Allgöwer, “Linear systems with neural network nonlinearities: Improved stability analysis via acausal Zames-Falb multipliers,” *arXiv preprint arXiv:2103.17106*, 2021.