ANALYSIS AND CONTROL OF NEURAL NETWORK DYNAMICAL SYSTEMS

Shaoru Chen

A DISSERTATION

in

Electrical and Systems Engineering

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2022

Supervisor of Dissertation

Victor M. Preciado, Associate Professor of Electrical and Systems Engineering

Graduate Group Chairperson

Troy Olsson, Associate Professor of Electrical and Systems Engineering

Dissertation Committee

George J. Pappas, UPS Foundation Professor of Electrical and Systems Engineering
Manfred Morari, Practice Professor, Peter and Susanne Armstrong Faculty Fellow of Electrical and Systems Engineering
Nikolai Matni, Assistant Professor of Electrical and Systems Engineering

ANALYSIS AND CONTROL OF NEURAL NETWORK DYNAMICAL SYSTEMS

To my family.

ACKNOWLEDGEMENT

First and foremost, I would like to thank my advisor, Victor, for his support and guidance during my Ph.D. study. Thank you for giving me the freedom to explore research topics and making this journey enjoyable.

I would like to thank all my committee members, Nik, Manfred and George. Nik, I am so grateful that you joined Penn in my third year of Ph.D. study. Under your guidance, I was able to figure out some problems on MPC that gave me confidence to continue my research in control. Talking with you has always been inspiring. Manfred, I have benefited so much from the discussions with you on all kinds of problems. Your experience, insights, and high standards of research have greatly motivated me. George, discussing with you has always been a pleasure for me. Through the discussions with you, I learned how to think about the big picture and find interesting problems.

I would also like to thank my labmates in Victor's group. Mahyar, we have been close collaborators and I cannot thank you too much for your guidance of my research. Without you I may not have touched upon the verification problem which consists the main portion of this dissertation. Ximing, you taught me a lot in the first two years of my Ph.D. and showed me how to conduct research independently for which I am truly grateful. I really enjoyed the time spent with Cassiano, Mikhail, Paco, Jacob, Alp, Jingqi, and Alex. Life is so fun with you!

This dissertation wouldn't be possible without the support from my colleages and friends from and outside Penn. I would like to thank George's group and Nik's group from whom I learned about many exciting research problems. I am also grateful to Prof. James Anderson and his group for the useful discussions. I want to thank Lekan Molu for his precious advice and help on my career planning. I want to thank my friends at Penn Mingyuan Zhang, Zhiyang Wang, Xingran Chen, Zebang Shen, Harshat Kumar, Luana Ruiz, Vinicius Lima, Juan Cervino, Luiz Chamon, Santiago Paternain, Zhiqi Bu, Kan Chen, and many others

for the wonderful time spent together.

Last but not least, I want to thank my parents for their unconditional love and support throughout my life. I am forever indebted to you! And my wife, Han Wang, thank you for your love and company. Whenever I feel stuck, you can always provide me with new perspectives. I have questioned a lot of things during my Ph.D. study, but you are undoubtedly the most beautiful person in the world!

ABSTRACT

ANALYSIS AND CONTROL OF NEURAL NETWORK DYNAMICAL SYSTEMS

Shaoru Chen

Victor M. Preciado

Learning for control has achieved remarkable success in controlling complex dynamical systems such as autonomous vehicles and quadrupedal robots. The resulting controlled system often has a neural network (NN) in the loop which represents the system dynamics, control policy, or perception. While enjoying high empirical performance, NN dynamical systems suffer from the lack of formal guarantees which significantly limits the deployment of NNs or learning algorithms in safety critical applications. This thesis aims to address this challenge by combining control theoretical analysis of dynamical systems and specialized optimization algorithm design for NNs.

In the first part of the thesis, we focus on verifying the safety and stability of a NN dynamical system. A hierarchy of verification problems are considered: (i) isolated output range analysis of a NN, (ii) closed-loop reachability analysis, and (iii) closed-loop stability analysis of a NN dynamical system. Output range analysis of a NN concerns over-approximating the output of a NN given a bounded input set. This problem can be formulated as a global optimization problem which is NP-hard to solve, and even its convex relaxations are numerically challenging to solve when large NNs are considered. Verification of closed-loop properties such as reachability and stability of NN dynamical systems becomes even more challenging since the objectives are more complex and the system dynamics must be taken into account. For Problem (i), by utilizing the structure of NNs, we propose a salable operator splitting method for solving a linear program relaxation of the verification problem which achieves a preferable complexity-conservatism trade-off. We solve Problem (ii) and (iii) by combining NN output range analysis tools and specialized reachability (i.e., verifying unrolled NN dynamics in one-shot) or stability (i.e., synthesizing a Lyapunov function using

cutting-plane methods) analysis frameworks.

In hope of combining robust model predictive control (MPC) and NN verification tools for safe control of NN dynamical systems, in the second part of the thesis, we propose a novel robust MPC method for uncertain linear dynamical systems with polytopic model uncertainty and bounded additive disturbances. This formulation allows abstracting nonlinear NN dynamics by polytopic uncertainties. Drawing tools from System Level Synthesis which transforms state feedback controller design into closed-loop system responses design, our proposed method can simultaneously search over robust linear time-varying state feedback controllers and bounds on the effects of model uncertainty. Extensive simulation shows that our proposed method achieves significantly reduced conservatism compared with existing robust MPC baselines.

TABLE OF CONTENTS

CHAPTER 1

Introduction

Learning-based methods such as reinforcement learning [7] have achieved great success in controlling complex systems with high empirical performance and little expert knowledge even when the system dynamics is nonlinear and unknown, the environment is dynamic and unstructured, and only raw, high-dimensional sensory data is available [8, 9]. After training, the resulting closed-loop system often has neural networks (NN) in the control loop representing the system dynamics [10], control policy [11] or perception [12, 13]. We denote such systems with NNs in the control loop as NN dynamical systems.

Neural networks are computational networks consisting of connected nodes called neurons. They are universal function approximators capable of approximating any Borel measurable function to any desired degree of accuracy [14]. More importantly, the boom of inexpensive computational resources such as GPUs and TPUs and the availability of massive data from the real world have enabled sufficient training of NNs and allowed machine learning to solve many long-standing problems across various engineering fields in the last decade. A rich collection of NN architectures such as multilayer perceptron (MLP) [15], convolutional neural network (CNN) [16], residual neural network (ResNet) [17], recurrent neural network (RNN) [18], long short-term memory (LSTM) [19], generative adversarial network (GAN) [20], transformers [21], etc., has been proposed and applied with great success in computer vision [22], robotic control [8], natural language processing [23], and scientific discovery [24]. With the ever-evolving machine learning ecosystem, we can expect more powerful and accessible learning-based methods for control and an increasing number of NN dynamical systems in diverse engineering applications in the future.

However, a fundamental problem of learning for control remains: learning-based methods do not provide formal guarantees on the system's behavior despite their high empirical performance. For example, it is hard to know whether an autonomous car will make a

dangerous move, a drone will run into an obstacle, or a robot manipulator will hurt its collaborating human worker when they are deployed in the wild after training. While we benefit from the expressivity of NNs, their black-box nature significantly limits the deployment of learning methods and NN dynamical systems in safety critical applications where the failure of the system may endanger human lives or induce excessive economic cost. Indeed, NNs have been shown vulnerable to adversarial attacks [25], which indicates that small perturbation to the inputs of NNs can drastically change the outputs of NNs. This is problematic when NNs are deployed in the real world control systems where the sensor/process noises are pervasive.

To close this practical gap, in this dissertation, we focus on providing formal guarantees for NN dynamical systems through control theoretical analysis and numerical algorithm design. Our approaches are motivated by the following observation: while there is a rich body of well established tools for safety/stability analysis of dynamical systems and controller design with guarantees from control, applying these tools on NN dynamical systems is numerically challenging for general-purpose solvers. Therefore, specialized numerical solvers that utilize structures of NNs for efficient verification and control of NN dynamical systems are desired.

In Part I of the dissertation, we solve a hierarchy of analysis problems regarding NN dynamical systems: verifying the property of a NN in isolation, over-approximating the reachable set of a NN dynamical systems over multiple time steps, and estimating the region of attraction of a NN dynamical system from a Lyapunov function. In Part II, we consider robust model predictive control (MPC) of linear uncertain systems which hold promises for safe control of NN dynamical systems using the NN verification bounds. The background of each of these problems and our contributions in this dissertation are summarized in the following section.

1.1. Main Contributions

1.1.1. Scalable Neural Network Verification

Neural network verification refers to the problem of verifying whether the output of a neural network satisfies certain properties for a bounded set of input perturbations. This problem can be formulated as a global optimization problem which is NP-hard to solve [26]. For example, a piecewise linear NN can be encoded by mixed-integer constraints and verified by mixed-integer linear programming (MILP) [27]; sigmoid-based NNs can be transformed into an equivalent hybrid system for which off-the-shelf verification tools can be applied [28]. However, these complete verifiers scale poorly with the size of the NN which motivates the development of convex relaxation-based NN verification methods that trade-off conservatism and computational complexity.

A wide range of convex relaxations of the verification problem have been proposed with distinct conservatism-complexity features. For example, the linear programming (LP) relaxation can be found in [29, 30] with single neuron relaxation and in [31, 32] with multiple neuron relaxation. Linear bounds propagation methods [33, 34, 4] which adopt looser relaxation for each neuron are more scalable than LP-based methods at the cost of conservatism. Semidefinite programming (SDP)-based relaxation [35, 36] takes the coupling between neurons into account and achieves tighter bounds than both the LP and propagation-based methods, but suffer from poor scalability. The aforementioned methods can be used as bounding subroutine in Branch-and-Bound (BaB) [37, 38, 39] or combined with input domain partitioning [40, 41] to further reduce the conservatism of verification.

Despite the availability of the diverse NN verification methods, the operating regime of the MILP, LP, and SDP-based methods are rather limited. They become costly to solve for NNs of 10k neurons already. On the other hand, the most scalable linear bounds propagation methods use the weakest relaxation with a high risk of generating vacuous bounds on the NN outputs. In this dissertation, we propose to scale up the relatively tight LP relaxation using operator splitting. The proposed method, DeepSplit [42], is based on the Alternating

Direction Method of Multipliers (ADMM) [43] and enjoys preferable convergence rates and parallelism. DeepSplit achieves comparable scalability as the linear bounds propagation methods and demonstrates improved tightness compared with Lagrangian-based and BaB-based baselines. We provide a detailed discussion on DeepSplit and the comparison in Chapter 2.

1.1.2. Reachability analysis of NN dynamical systems

Reachability analysis of NN dynamical systems concerns verifying whether the system will evolve into unsafe regions from a set of initial conditions. The fundamental problem is bounding the reachable sets of the NN dynamical systems since we can certify the safety of the system by showing that the reachable set over-approximations do not intersect any unsafe regions. Compared with NN verification, reachability analysis requires taking the system dynamics into account.

Although NN verification methods only consider NNs in isolation, they can be conveniently combined with existing reachability analysis tools to certify properties of NN dynamical systems [44, 45, 46] over a finite horizon. For a discrete-time NNDS, NN verification methods can readily compute an over-approximation of the one-step reachable set given a bounded input set. Then, applying such one-step over-approximation method recursively leads to a bounding tube of the NN dynamical system trajectories.

We denote the above methodology as the recursive reachability analysis framework [47, 48, 49], and compare it with an alternative framework that computes over-approximations of the reachable sets in one shot [50, 51]. In one-shot reachability analysis, the reachable set of NN dynamical system at time t is bounded by directly applying NN verification methods on the unrolled NN dynamics for t steps. While one may observe that the one-shot analysis generates tighter bounds than the recursive counterpart due to the use of iterated dynamics [50], this property does not hold for general NN verification tools. In this dissertation, we provide a counter-example where one-shot analysis results in worse bounds than the recursive framework, highlighting that the one-shot analysis does not

always lead to tighter bounds. In addition, we formally prove conditions under which the one-shot framework provides tighter bounds compared with the recursive framework for a general class of NN verification methods. Our analysis applies to NNs with general architectures and allows us to consider disturbances in the computation of the reachable set over-approximations. The details of the analysis and numerical examples are included in Chapter 3.

1.1.3. Stability analysis of NN dynamical systems

Analyzing the stability of the NN dynamical system requires finding a Lyapunov function for the system. Unlike in NN verification or reachability analysis where the objective function is normally linear, searching Lyapunov functions does not have a straight-forward optimization problem formulation and requires a significant amount of algorithm design. Importantly, the layer-by-layer representation of NNs impose constraints on the algorithm design since most of existing Lyapunov function synthesis methods cannot readily handle NNs. For example, in theory we can transform a piecewise linear NN into a piecewise affine (PWA) function [52] and apply established stability analysis methods [53, 54, 55] for PWA systems, doing such transformation requires non-trivial effort and may result in a PWA function with a huge number of partitions [56].

In this dissertation, we propose a sampling-based method for Lyapunov function synthesis of NN dynamical systems with piecewise linear NNs. More generally, our method can handle linear hybrid systems such as piecewise affine systems, mixed-logical dynamical systems [57], and linear complementarity systems [58] which allow a mixed-integer representation. The proposed method comprises an alternation between a learner and a verifier to find a valid Lyapunov function inside a convex set of Lyapunov function candidates. By designing the learner and the verifier according to the analytic center cutting-plane method from convex optimization, we show that when the set of Lyapunov functions is full-dimensional in the parameter space, our method finds a Lyapunov function in a finite number of steps. The details of the proposed approach are shown in Chapter 4.

1.1.4. Robust model predictive control

Robust MPC is an optimization-based method for feedback controller synthesis with robust constraint satisfaction guarantees in the face of uncertainties. While applying MPC on NN dynamical systems is intractable due to the nonlinearity of NNs, NN verification tools such as CROWN [34] and LiRPA [4] provide simple local linear bounds on the NN dynamics which can be integrated into robust MPC. However, before we combine NN verification and robust MPC for safe controller synthesis of NN dynamical systems, several challenges remain regarding the robust MPC tools.

First, robust MPC of linear systems with model uncertainty is a challenge itself. Model uncertainty complicates the analysis of the effects of both controller and uncertainty parameters on the system trajectory. To allow tractable computation, existing methods [59, 60, 61, 62, 63, 64] often use controller parameterization with limited expressivity or conservative bounds on the uncertainty effects. Since the NN dynamics is over-approximated by conservative linear bounds, it is desirable to reduce the conservatism of robust MPC methods as much as possible for model predictive control of NN dynamical systems to be practical. Second, the linear bounds on the NN dynamical systems cannot be readily handled by existing robust MPC methods which require representing the model uncertainty as a polytopic set. Transforming the linear bounds into a polytopic set will give rise to a large number of vertices and significantly increase the computational cost for methods that require a vertex enumeration of the uncertainty set [59, 60, 61, 62].

To address the above challenges, we propose a novel method for robust model predictive control of uncertain linear systems with both model uncertainty and additive disturbances. Our method optimizes over a linear time-varying (LTV) state feedback controller and can conveniently handle both the standard polytopic model uncertainty and uncertainty constrained by linear bounds. In our method, we over-approximate the actual uncertainty by a surrogate additive disturbance which simplifies constraint tightening of the robust optimal control problem. Using System Level Synthesis [65], we optimize over a robust linear state

feedback control policy and the uncertainty over-approximation parameters jointly and in a convex manner. The proposed method is demonstrated to achieve significant improvement in conservatism compared with a wide range of robust MPC baselines through numerical examples. We present the details of the proposed method and numerical examples in Chapter 5.

Finally, Chapter 6 concludes the dissertation and discusses a few open problems for future research.

CHAPTER 2

Salable Neural Network Verification

2.1. Background on Neural Network Verification

Neural network verification refers to the problem of verifying whether the output of a neural network satisfies certain properties for a bounded set of input perturbations. This problem can be framed as optimization problems of the form

$$J^\star \leftarrow \text{minimize} \quad J(f(x)) \quad \text{subject to} \quad x \in \mathcal{X}, \tag{2.1}$$

where f is given by a deep neural network, J is a real-valued function representing a performance measure (or a specification), and $\mathcal{X}$ is a set of inputs to be verified. In this formulation, verifying the neural network amounts to certifying whether the optimal value of (2.1) is bounded below by a certain threshold.

As an example, consider a classification problem with n_f classes, in which for a data point $x \in \mathbb{R}^{n_x}$, $f(x) \in \mathbb{R}^{n_f}$ denotes the vector of scores for all the classes. The classification rule is $C(x) = \arg\max_{1 \le i \le n_f} e_i^\top f(x)$ where e_i denotes the i-th standard basis. Given a correctly classified $x^\star \in \mathbb{R}^{n_x}$ and a perturbation set $\mathcal{X} \subset \mathbb{R}^{n_x}$ that contains x, we say that f is locally robust at $x^\star$ with respect to $\mathcal{X}$ if $C(x) = C(x^\star)$ for all $x \in \mathcal{X}$. Verifying the local robustness at $x^\star$ then amounts to verifying that the optimal values of the following $n_f - 1$ optimization problems

$$\underset{x \in \mathcal{X}}{\text{minimize}} \quad (e_{i^\star} - e_i)^\top f(x), \quad i \neq i^\star, \tag{2.2}$$

are positive, where $i^\star = C(x^\star)$ is the class index of $x^\star$. Problem (2.2) is an instance of (2.1) where J is a linear function.

Problem (2.1) is large-scale and non-convex, making it extremely difficult to solve efficiently–both in terms of time and memory. For ReLU activation functions and linear objectives, the

problem in (2.1) can be cast as a Mixed-Integer Linear Program (MILP) [66, 67, 68, 69], which can be solved for the global solution via, for example, Branch-and-Bound (BaB) methods. While we do not expect these approaches to scale to large problems, for small neural networks they can still be practical.

Instead of solving (2.1) for its global minimum, one can instead find *guaranteed lower bounds* on the optimal value via convex relaxations, such as Linear Programming (LP) [29] and Semidefinite Programming (SDP) [35, 70, 36]. Verification methods based on convex relaxations are sound but incomplete, i.e., they are guaranteed to detect all false negatives but also produce false positives, whose rate depends on the tightness of the relaxation. Although convex relaxations are polynomial-time solvable (in terms of number of decision variables), in practice they are not computationally tractable for large-scale neural networks. To improve scalability, these relaxations must typically be further relaxed [29, 33, 34, 1, 71].

In this work, we propose an algorithm to solve LP relaxations of (2.1) for their global solution and for large-scale feed-forward neural networks. Our starting point is to express (2.1) as a constrained optimization problem whose constraints are imposed by the forward passes in the network. We then introduce additional decision variables and consensus constraints that naturally split the corresponding problem into independent subproblems, which often have closed-form solutions. Finally, we employ an operator splitting technique based on the Alternating Direction Method of Multipliers (ADMM) [43], to solve the corresponding Lagrangian relaxation of the problem. This approach has several favorable properties. First, the method requires minimal parameter tuning and relies on simple operations, which scale to very large problems and can achieve a good trade-off between runtime and solution accuracy. Second, all the solver operations are amenable to fast parallelization with GPU acceleration. Third, our method is fully modular and applies to standard network architectures.

We employ our method to compute exact solutions to LP relaxations on the worst-case performance of adversarially trained deep networks, with a focus on networks whose con-

vex relaxations are difficult to solve due to their size. Specifically, we perform extensive experiments in the ℓ_∞ perturbation setting, where we verify robustness properties of image classifiers for CIFAR10 and deep Q-networks (DQNs) in Atari games [72]. Our method is able to solve LP relaxations at scales that are too large for exact MILP verifiers, SDP relaxations, or commercial LP solvers such as Gurobi. We also demonstrate the use of our method in computing reachable set over-approximations of a neural network dynamical system over a long horizon, where our method is compared with the state-of-the-art BaB method [38].

Notation We denote the set of real numbers by $\mathbb{R}$, the set of nonnegative real numbers by $\mathbb{R}_+$, the set of real n-dimensional vectors by $\mathbb{R}^n$, the set of $m \times n$-dimensional real-valued matrices by $\mathbb{R}^{m\times n}$, and the n-dimensional identity matrix by I_n. The p-norm ($p \geq 1$) is denoted by $\|\cdot\|_p \colon \mathbb{R}^n \to \mathbb{R}_+$. For a set $\mathcal{S}$, we define the indicator function $\mathbb{I}_{\mathcal{S}}(x)$ of $\mathcal{S}$ as $\mathbb{I}_{\mathcal{S}}(x) = 0$ if $x \in \mathcal{S}$ and $\mathbb{I}_{\mathcal{S}}(x) = +\infty$ otherwise. Given a function $f \colon \mathcal{X} \to \mathcal{Y}$, the graph of f is the set $\mathcal{G}_f = \{(x, f(x)) \mid x \in \mathcal{X}\}$.

2.2. Neural network verification via operator splitting

We consider an ℓ-layer feed-forward neural network $f(x) \colon \mathbb{R}^{n_0} \to \mathbb{R}^{n_\ell}$ described by the following recursive equations,

$$\begin{aligned} x_0 &= x, \\ x_{k+1} &= \phi_k(x_k), \quad k = 0, \cdots, \ell - 1, \\ f(x) &= x_\ell \end{aligned} \tag{2.3}$$

where $x_0 \in \mathbb{R}^{n_0}$ is the input to the neural network, $x_k \in \mathbb{R}^{n_k}$ is the input to the k-th layer, and $\phi_k \colon \mathbb{R}^{n_k} \to \mathbb{R}^{n_{k+1}}$ is the operator of the k-th layer, which can represent any commonly-used operator in deep networks, such as linear (convolutional) layers [1], MaxPooling units, and activation functions.

Given the neural network f, a specification function $J \colon \mathbb{R}^{n_\ell} \mapsto \mathbb{R}$, and an input set $\mathcal{X} \subset \mathbb{R}^{n_0}$,

[1] Convolution is a linear operator and conceptually it can be analyzed in the same way as for linear layers.

we say that f satisfies the specification J if $J(f(x)) \geq 0$ for all $x \in \mathcal{X}$. This is equivalent to verifying that the optimal value of (2.1) is non-negative. We assume $\mathcal{X} \subset \mathbb{R}^{n_0}$ is a closed convex set and $J\colon \mathbb{R}^{n_\ell} \to \mathbb{R} \cup \{+\infty\}$ is a convex function.

Using the sequential representation of the neural network in (2.3), we may rewrite the optimization problem in (2.1) as the following constrained optimization problem,

$$\begin{aligned} J^\star \leftarrow \text{minimize} \quad & J(x_\ell) \\ \text{subject to} \quad & x_{k+1} = \phi_k(x_k),\ k = 0, \cdots, \ell - 1, \\ & x_0 \in \mathcal{X}, \end{aligned} \tag{2.4}$$

with $n := \sum_{k=0}^{\ell} n_k$ decision variables $x_0, \cdots, x_\ell$. We can rewrite (2.4) equivalently as

$$\begin{aligned} J^\star \leftarrow \text{minimize} \quad & J(x_\ell) \\ \text{subject to} \quad & (x_k, x_{k+1}) \in \mathcal{G}_{\phi_k},\ k = 0, \cdots, \ell - 1, \\ & x_0 \in \mathcal{X}, \end{aligned} \tag{2.5}$$

where

$$\mathcal{G}_{\phi_k} = \{(x_k, x_{k+1}) \mid x_{k+1} = \phi_k(x_k),\ \underline{x}_k \leq x_k \leq \bar{x}_k\},$$

is the graph of ϕ_k. Here $\underline{x}_k$ and $\bar{x}_k$ are *a priori* known bounds on x_k when $x_0 \in \mathcal{X}$ [2].

The problem in (2.5) is non-convex due to presence of nonlinear operators in the network, such as activation layers, which render the set $\mathcal{G}_{\phi_k}$ nonconvex. By over-approximating $\mathcal{G}_{\phi_k}$ by a convex set (or ideally by its convex hull), we arrive at a direct layer-wise convex relaxation of the problem, in which each two consecutive variables (x_k, x_{k+1}) are sequentially coupled together. Solving this relaxation directly cannot exploit this structure and is hence unable to scale to even medium-sized neural networks [73]. In this section, by exploiting the sequential structure of the constraints, and introducing auxiliary decision variables (variable splitting), we propose a reformulation of (2.4) whose convex relaxation can be decomposed

[2] See Section 2.2.4 for comments on finding the bounds $\underline{x}_k$ and $\bar{x}_k$.

into smaller sub-problems that can be solved efficiently and in a scalable manner.

2.2.1. Variable splitting

By introducing the intermediate variables y_k and z_k, we can rewrite (2.4) as

$$\begin{aligned} J^\star \leftarrow \text{minimize} \quad & J(x_\ell) \\ \text{subject to} \quad & y_k = x_k, && k = 0, \cdots, \ell - 1 \\ & z_k = \phi_k(y_k), && k = 0, \cdots, \ell - 1 \\ & x_{k+1} = z_k, && k = 0, \cdots, \ell - 1 \\ & x_0 \in \mathcal{X}, \end{aligned} \tag{2.6}$$

which has now $3n - n_0 - n_\ell$ decision variables. Intuitively, we have introduced additional "identity layers" between consecutive layers (see Figure 2.1). By overapproximating $\mathcal{G}_{\phi_k}$ by a convex set $\mathcal{S}_{\phi_k}$, we obtain the convex relaxation

$$\begin{aligned} J^\star_{\text{relaxed}} \leftarrow \text{minimize} \quad & J(x_\ell) \\ \text{subject to} \quad & y_k = x_k, && k = 0, \cdots, \ell - 1 \\ & (y_k, z_k) \in \mathcal{S}_{\phi_k}, && k = 0, \cdots, \ell - 1 \\ & x_{k+1} = z_k, && k = 0, \cdots, \ell - 1 \\ & x_0 \in \mathcal{X}, \end{aligned} \tag{2.7}$$

for which $J^\star_{\text{relaxed}} \leq J^\star$. This form is known as consensus as y_k and z_{k-1} are just copies of the variable x_k. As shown below, this "overparameterization" allows us to split the optimization problem into smaller sub-problems that can be solved in parallel and often in closed form.

2.2.2. Lagrangian relaxation and operator splitting

We use $\mathbf{x} = (x_0, \cdots, x_\ell)$, $\mathbf{y} = (y_0, \cdots, y_{\ell-1})$ and $\mathbf{z} = (z_0, \cdots, z_{\ell-1})$ to denote the concatenated variables. By relaxing the equality constraints with Lagrangian multipliers, we define

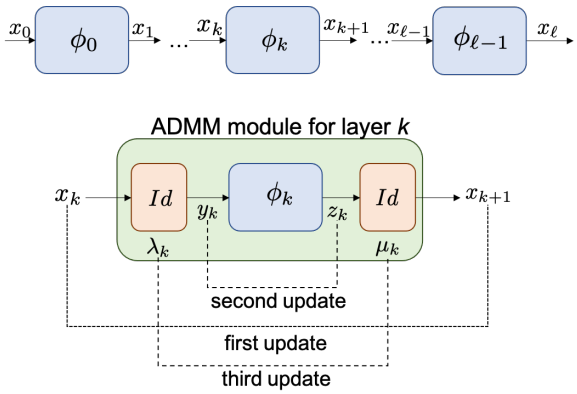


Figure 2.1: Illustration of the network structure (top) and DeepSplit computation module for a generic layer (bottom). Adding identity layers in between the neural network layers decouples the variables x_k and allows processing them independently.

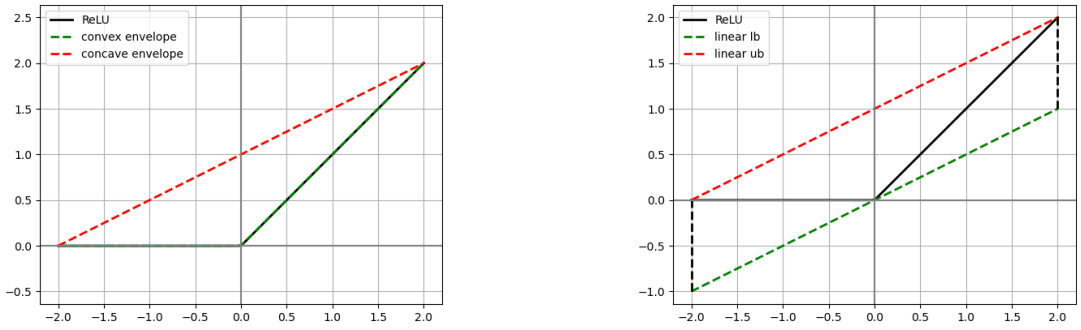


Figure 2.2: Over-approximation of the graph of ReLU function by convex hull (left) and linear bounds (right).

the augmented Lagrangian for (2.7) as follows,

$$\mathcal{L}(\mathbf{x},\mathbf{y},\mathbf{z},\boldsymbol{\lambda},\boldsymbol{\mu}) = J(x_\ell)+\sum_{k=0}^{\ell-1}\mathbb{I}_{\mathcal{S}_{\phi_k}}(y_k,z_k)+\mathbb{I}_{\mathcal{X}}(x_0)+(\rho/2)\sum_{k=0}^{\ell-1}\left(\|x_k-y_k+\lambda_k\|_2^2-\|\lambda_k\|_2^2\right) + (\rho/2)\sum_{k=0}^{\ell-1}\left(\|x_{k+1}-z_k+\mu_k\|_2^2-\|\mu_k\|_2^2\right). \tag{2.8}$$

where $\boldsymbol{\lambda} = (\lambda_0,\cdots,\lambda_{\ell-1})$ and $\boldsymbol{\mu} = (\mu_0,\cdots,\mu_{\ell-1})$ are the scaled dual variables (by $1/\rho$) and $\rho > 0$ is the augmentation constant. Note that we have only relaxed the equality constraints in (2.7), and the constraints describing the sets $\mathcal{S}_{\phi_k}$ as well as the input set $\mathcal{X}$ are kept intact. Furthermore, the inclusion of augmentation will render the dual function differentiable, and hence, easier to optimize.

For the Lagrangian in (2.8), the dual function, which provides a lower bound to $J^\star_{\text{relaxed}}$, is given by $g(\boldsymbol{\lambda},\boldsymbol{\mu}) = \inf_{(\mathbf{x},\mathbf{y},\mathbf{z})}\ \mathcal{L}(\mathbf{x},\mathbf{y},\mathbf{z},\boldsymbol{\lambda},\boldsymbol{\mu})$. The best lower bound can then be found by maximizing the dual function. [3] However, solving the inner problem jointly over $(\mathbf{x},\mathbf{y},\mathbf{z})$ to find the dual function is as difficult as solving a direct convex relaxation of (2.4). Instead, we split the primal variables $(\mathbf{x},\mathbf{y},\mathbf{z})$ into $\mathbf{x}$ and $(\mathbf{y},\mathbf{z})$ and apply the classical ADMM algorithm to obtain the following iterations (shown in Figure 2.1) for updating the primal and dual variables,

$$\mathbf{x}^+ \in \text{argmin}_{\mathbf{x}}\ \mathcal{L}(\mathbf{x},\mathbf{y},\mathbf{z},\boldsymbol{\lambda},\boldsymbol{\mu}) \tag{2.9a}$$

$$(\mathbf{y}^+,\mathbf{z}^+) \in \text{argmin}_{(\mathbf{y},\mathbf{z})}\ \mathcal{L}(\mathbf{x}^+,\mathbf{y},\mathbf{z},\boldsymbol{\lambda},\boldsymbol{\mu}) \tag{2.9b}$$

$$(\boldsymbol{\lambda}^+,\boldsymbol{\mu}^+) = (\boldsymbol{\lambda},\boldsymbol{\mu}) + \nabla_{(\boldsymbol{\lambda},\boldsymbol{\mu})}\ \mathcal{L}(\mathbf{x}^+,\mathbf{y}^+,\mathbf{z}^+,\boldsymbol{\lambda},\boldsymbol{\mu}). \tag{2.9c}$$

As we show below, the Lagrangian has a separable structure by construction that can be exploited in order to efficiently implement each step of (2.9).

[3] If strong duality holds, then this best lower bound would match $J^\star_{\text{relaxed}}$. As shown in [73], strong duality holds under mild conditions.

2.2.3. The x-update

The Lagrangian in (2.8) is separable across the x_k variables; hence, the minimization in (2.9a) can be done independently for each x_k. Specifically, for $k = 0$, we obtain the following update rule for x_0,

$$x_0^+ = \text{Proj}_{\mathcal{X}}(y_0 - \lambda_0). \tag{2.10a}$$

Projections onto the ℓ_∞ and ℓ_2 balls can be done in closed-form. For the ℓ_1 ball, we can use the efficient projection scheme proposed in [74], which has $\mathcal{O}(n_0)$ complexity in expectation. For subsequent layers, we obtain the updates

$$x_k^+ = \frac{1}{2}(y_k - \lambda_k + z_{k-1} - \mu_{k-1}),\ k = 1, \cdots, \ell - 1, \tag{2.10b}$$

$$x_\ell^+ = \arg\min_{x_\ell}\ J(x_\ell) + \frac{\rho}{2}\|x_\ell - z_{\ell-1} + \mu_{\ell-1}\|_2^2. \tag{2.10c}$$

For convex J and $\rho > 0$, the optimization problem for updating x_ℓ is strongly convex with a unique optimal solution. Indeed, its solution is the proximal operator of J/ρ evaluated at $z_{\ell-1} - \mu_{\ell-1}$. For the special case of linear objectives, $J(x_\ell) = c^\top x_\ell$, we obtain the closed-form solution

$$x_\ell^+ = -\frac{1}{\rho}c + (z_{\ell-1} - \mu_{\ell-1}).$$

2.2.4. The $(\mathbf{y}, \mathbf{z})$-update

Similarly, the Lagrangian is also separable across the (y_k, z_k) variables. Updating these variables in (2.9b) corresponds to the following projection operations per layer,

$$(y_k^+, z_k^+) = \text{Proj}_{\mathcal{S}_{\phi_k}}(x_k^+ + \lambda_k, x_{k+1}^+ + \mu_k), \tag{2.11}$$

for $k = 0, \cdots, \ell - 1$. Depending on the type of the layer (linear, activation, convolution, etc.), we obtain different projections which we describe below.

Affine layers

Suppose $\phi_k(y_k) = W_k y_k + b_k$ is an affine layer representing a fully-connected, convolutional, or an average pooling layer. Then the graph of ϕ_k is already a convex set given by $\mathcal{G}_{\phi_k} = \{(y_k, z_k) \mid z_k = W_k y_k + b_k\}$. Choosing $\mathcal{S}_{\phi_k} = \mathcal{G}_{\phi_k}$, the projection in (2.11) takes the form

$$\begin{aligned} y_k^+ &= (I_{n_k} + W_k^\top W_k)^{-1}(x_k^+ + \lambda_k + W_k^\top (x_{k+1}^+ + \mu_k - b_k)), \\ z_k^+ &= W_k y_k^+ + b_k. \end{aligned} \tag{2.12}$$

The matrix $(I_{n_k} + W_k^\top W_k)^{-1}$ can be pre-computed and cached for subsequent iterations. We can do this efficiently for convolutional layers using the fast Fourier transform (FFT).

Activation layers

For an activation layer of the form $\phi(x) := [\varphi_1(x_1) \ \cdots \ \varphi_n(x_n)]^\top$, the convex relaxation of $\mathcal{G}_\phi$ is given by the Cartesian product of individual convex relaxations i.e., $\mathcal{S}_\phi = \mathcal{S}_{\varphi_1} \times \cdots \times \mathcal{S}_{\varphi_n}$. For a generic activation function $\varphi \colon \mathbb{R} \to \mathbb{R}$, suppose we have a concave upper bound $\bar{\varphi}$ and a convex lower bound $\underline{\varphi}$ on φ over an interval $I = [\underline{x}, \bar{x}]$, i.e., $\underline{\varphi}(x) \leq \varphi(x) \leq \bar{\varphi}(x) \ \forall \ x \in [\underline{x}, \bar{x}]$. A convex overapproximation of $\mathcal{G}_\varphi$ is

$$\mathcal{S}_\varphi = \{(x, y) \mid \underline{\varphi}(x) \leq y \leq \bar{\varphi}(x), \ \underline{x} \leq x \leq \bar{x}\}, \tag{2.13}$$

which turns out to be the convex hull of $\mathcal{G}(\varphi)$ when $\bar{\varphi}$ and $\underline{\varphi}$ are concave and convex envelopes of φ, respectively. The assumed pre-activation bounds $\underline{x}$ and $\bar{x}$ used to relax the activation functions can be obtained a priori via a variety of existing techniques such as linear bounds [29, 34, 75] which propagate linear lower and upper bounds on each activation function (see Figure 2.2) throughout the network in a fast manner.

Example 1 (ReLU activation function.)**.** Consider the ReLU activation function $\varphi(x) = \max(x, 0)$ over the interval $[\underline{x}, \bar{x}]$. When $\underline{x} < 0 < \bar{x}$, the ReLU function admits the envelopes

$\underline{\varphi}(x) = \max(0, x)$, $\bar{\varphi}(x) = \underline{y} + \frac{\bar{y}-\underline{y}}{\bar{x}-\underline{x}}(x - \underline{x})$ on $[\underline{x}, \bar{x}]$, where $\underline{y} = \max(0, \underline{x})$ and $\bar{y} = \max(0, \bar{x})$ [76, 29]. In this case, the projection of a point $(x^{(0)}, y^{(0)})$ onto the convex hull of G_φ, which is a triangle shown in Figure 2.2, has a closed-form solution. Letting $s = \frac{\bar{y}-\underline{y}}{\bar{x}-\underline{x}}$, we first project $(x^{(0)}, y^{(0)})$ onto each facet of the triangle and then select the point with the minimal distance:

$$
\begin{aligned}
x^{(1)} &= \min(\max(\frac{x^{(0)} + y^{(0)}}{2}, 0), \bar{x}), \quad y^{(1)} = x^{(1)}, \\
x^{(2)} &= \min(\max(\frac{x^{(0)} + sy^{(0)} + s(s\underline{x} - \underline{y})}{s^2 + 1}, \underline{x}), \bar{x}), \\
y^{(2)} &= \frac{s(x^{(0)} - \underline{x}) + s^2 y^{(0)} + \underline{y}}{s^2 + 1}, \\
x^{(3)} &= \min(\max(0, x^{(0)}), \underline{x}), \quad y^{(3)} = 0.
\end{aligned}
$$

The projected point is $(x', y') = (x^{(i^\star)}, y^{(i^\star)})$, where $i^\star = \arg\min_{1 \le i \le 3} \sqrt{(x^{(0)} - x^{(i)})^2 + (y^{(0)} - y^{(i)})^2}$. See Figure 2.3 for illustration.

When either $\underline{x} \ge 0$ or $\bar{x} \le 0$, the graph G_φ of the ReLU function becomes a line segment which allows closed-form projection of a point.

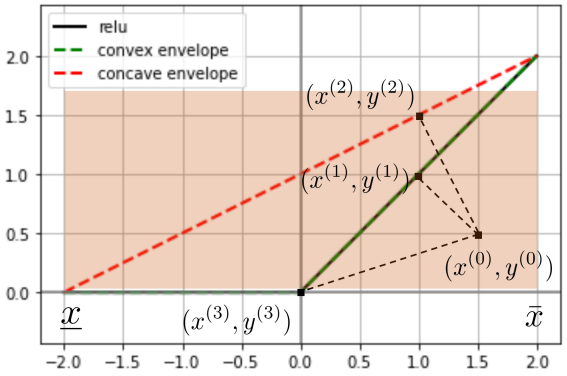


Figure 2.3: Projection of a point (x_0, y_0) onto the convex hull of the ReLU function $y = \max(0, x)$ over the interval $[\underline{x}, \bar{x}]$. We first project (x_0, y_0) onto all facets of the convex hull and then select the point with minimal distance to (x_0, y_0).

2.2.5. The $(\boldsymbol{\lambda}, \boldsymbol{\mu})$-update

Finally, we update the scaled dual variables as follows,

$$\begin{aligned} \lambda_k^+ &= \lambda_k + (x_k^+ - y_k^+), \qquad k = 0, \cdots, \ell - 1, \\ \mu_k^+ &= \mu_k + (x_{k+1}^+ - z_k^+), \quad k = 0, \cdots, \ell - 1. \end{aligned} \tag{2.14}$$

The DeepSplit Algorithm is summarized in Algorithm 1.

Algorithm 1 DeepSplit Algorithm

Input: neural network f (Eq. (2.3)), bounded convex input set $\mathcal{X}$, convex function J.
Output: lower bound $J^\star_{\text{relaxed}}$ on Problem (2.1).

1: **Initialization:** $x_0 \in \mathcal{X}$, $x_{k+1} = \phi_k(x_k)$, $y_k = x_k$, $z_k = x_{k+1}$, $\lambda_k = 0, \mu_k = 0$, $k = 0, \cdots, \ell - 1$, augmentation constant $\rho > 0$.
2: **while** stopping criterion is not met (see Section 2.2.6) **do**
3: **Step 1:** $\mathbf{x}$-update (2.10)
4: **Step 2:** $(\mathbf{y}, \mathbf{z})$-update (2.11)
5: **Step 3:** dual update (2.14)
6: **Output:** $J(x_\ell)$

2.2.6. Convergence and stopping criterion

The DeepSplit algorithm converges to the optimal solution of the convex problem (2.7) under mild conditions. Specifically, when J is closed, proper and convex, and when the sets $\mathcal{S}_k$ (convex outer approximations of the graph of the layers) along with $\mathcal{X}$ are closed and convex, we can resort to the convergence results of ADMM [43].

Following from [43], for the LP relaxation (2.7) of a feed-forward neural network, the primal and dual residuals are defined as

$$\begin{aligned} r_p &= \sum_{k=0}^{\ell-1} \{\|y_k^+ - x_k^+\|_2^2 + \|x_{k+1}^+ - z_k^+\|_2^2\}, \\ r_d &= \rho \sum_{k=1}^{\ell-1} \|(y_k^+ - y_k) + (z_{k-1}^+ - z_{k-1})\|_2^2 + \rho \left(\|y_0^+ - y_0\|_2^2 + \|z_{\ell-1}^+ - z_{\ell-1}\|_2^2\right). \end{aligned}$$

These are the residuals of the optimality conditions for (2.7) and converge to zero as the algorithm proceeds. A reasonable termination criterion is that the primal and dual residuals

must be small, i.e. $r_p \leq \epsilon_p$ and $r_d \leq \epsilon_d$, where $\epsilon_p > 0$ and $\epsilon_d > 0$ are tolerance levels [43, Chapter 3]. These tolerances can be chosen using an absolute and relative criterion, such as

$$\epsilon_p = \sqrt{p}\ \epsilon_{abs} + \epsilon_{rel} \max\{(\|x_0\|_2^2 + 2\sum_{i=1}^{\ell-1} \|x_i\|_2^2 + \|x_\ell\|_2^2)^{\frac{1}{2}} + (\sum_{i=0}^{\ell-1}(\|y_i\|_2^2 + \|z_i\|_2^2)^{1/2}\},$$

$$\epsilon_d = \sqrt{n}\ \epsilon_{abs} + \epsilon_{rel}(\|\lambda_0\|_2^2 + \sum_{i=1}^{\ell-1} \|\lambda_i + \mu_{i-1}\|_2^2 + \|\mu_{\ell-1}\|_2^2)^{\frac{1}{2}},$$

where $p = n_0 + 2\sum_{i=1}^{\ell-1} n_i + n_\ell$, $n = \sum_{i=0}^{\ell} n_i$, $\epsilon_{abs} > 0$ and $\epsilon_{rel} > 0$ are absolute and relative tolerances. Here n is the dimension of $\mathbf{x}$, the vector of primal variables that are updated in the first step of the algorithm, and p is the total number of consensus constraints.

2.2.7. Residual balancing for convergence acceleration

A proper selection of the augmentation constant ρ has a dramatic effect on the convergence of the ADMM algorithm. Large values of ρ enforce consensus more quickly, yielding smaller primal residuals but larger dual ones. Conversely, smaller values of ρ lead to larger primal and smaller dual residuals. Since ADMM terminates when both the primal and dual residuals are small enough, in practice we prefer to choose the augmentation parameter ρ not too large or too small in order to balance the reduction in the primal and dual residuals. A commonly used heuristic to make this trade-off is residual balancing [77], in which the penalty parameter varies adaptively based on the following rule:

$$\rho^+ = \begin{cases} \tau\rho & \text{if } r_p > \mu r_d \\ \rho/\tau & \text{if } r_d > \mu r_p \\ \rho & \text{otherwise,} \end{cases}$$

where $\mu, \tau > 1$ are given parameters. In our experiments, we set $\tau = 2, \mu = 10$ and found this rule to be effective in speeding up the practical convergence which is demonstrated numerically in Section 2.4.5.

2.3. Connection to Lagrangian-based methods

In this section, we consider verification of the feed-forward neural network (2.3) and draw connections to two related methods relying on Lagrangian relaxation. Specifically, we relate our approach to an earlier dual method [1] (Section 2.3.1), as well as a recent Lagrangian decomposition method [71] (Section 2.3.2). Overall, these approaches use a similar Lagrangian formulation, but the specific choices in splitting and augmentation of the Lagrangian result in slower theoretical convergence guarantees when solving the convex relaxation to optimality.

2.3.1. Dual method via Lagrangian relaxation of the nonconvex problem

Instead of splitting the neural network equations (2.4) with auxiliary variables, an alternative strategy is to directly relax (2.4) with Lagrangian multipliers [1]:

$$\mathcal{L}(\mathbf{x},\boldsymbol{\lambda}) = J(x_\ell) + \sum_{k=0}^{\ell-1} \lambda_k^\top (x_{k+1} - \phi_k(x_k)) + \mathbb{I}_{\mathcal{X}}(x_0). \tag{2.15}$$

where $\mathbf{x} = (x_0, \cdots, x_\ell)$ and $\boldsymbol{\lambda} = (\lambda_0, \cdots, \lambda_{\ell-1})$. This results in the dual problem $g^\star \leftarrow$ maximize $g(\boldsymbol{\lambda})$, where the dual function is

$$g(\boldsymbol{\lambda}) = \inf_{\underline{x}_\ell \le x_\ell \le \bar{x}_\ell} \{J(x_\ell) + \lambda_{\ell-1}^\top x_\ell\} + \inf_{x_0 \in \mathcal{X}_0} \{-\lambda_0^\top \phi_0(x_0)\} + \sum_{k=1}^{\ell-1} \inf_{\underline{x}_k \le x_k \le \bar{x}_k} \{\lambda_{k-1}^\top x_k - \lambda_k^\top \phi_k(x_k)\}.$$

By weak duality, $g^\star \le J^\star$. The inner minimization problems to compute the dual function $g(\boldsymbol{\lambda})$ for a given $\boldsymbol{\lambda}$ can often be solved efficiently or even in closed-form [1]. The resulting dual problem is unconstrained but non-differentiable; hence it is solved using dual subgradient method [1]. However, subgradient methods are known to be very slow with convergence rate $O(1/\sqrt{N})$ where N is the number of updates [78], making it inefficient to find exact solutions to the convex relaxation. On the other hand, this method can be stopped at any time to obtain a valid lower bound.

2.3.2. Lagrangian method via a non-separable splitting

Another related approach is the Lagrangian decomposition method from [71]. To decouple the constraints for the convex relaxation of (2.4), this approach can be viewed as introducing *one* set of intermediate variables y_k as copies of x_k to obtain

$$\begin{aligned} J^\star_{\text{relaxed}} \leftarrow \text{minimize} \quad & J(y_\ell) && \\ \text{subject to} \quad & (y_k, x_{k+1}) \in \mathcal{S}_{\phi_k} && k = 0, \cdots, \ell-1 \\ & y_k = x_k && k = 0, \cdots, \ell \\ & x_0 \in \mathcal{X} && \end{aligned} \tag{2.16}$$

This splitting is in the spirit of the splitting introduced in [71, 2],[4] and differs from our splitting which uses *two* sets of variables in (2.6). By relaxing the consensus constraints $y_k = x_k$, the Lagrangian is

$$\mathcal{L}(\mathbf{x}, \mathbf{y}, \boldsymbol{\mu}) = J(y_\ell) + \sum_{k=0}^{\ell} \mu_k^\top (y_k - x_k) + \sum_{k=0}^{\ell-1} \mathbb{I}_{\mathcal{S}_{\phi_k}}(y_k, x_{k+1}) + \mathbb{I}_{\mathcal{X}}(x_0). \tag{2.17}$$

Again the Lagrangian is separable and its minimization results in the following dual function

$$g(\boldsymbol{\mu}) = \inf_{\underline{x}_\ell \le y_\ell \le \bar{x}_\ell} \{J(y_\ell) + \mu_\ell^\top y_\ell\} + \inf_{x_0 \in \mathcal{X}} \{-\mu_0^\top x_0\} + \sum_{k=0}^{\ell-1} \inf_{(y_k, x_{k+1}) \in \mathcal{S}_{\phi_k}} \{\mu_k^\top y_k - \mu_{k+1}^\top x_{k+1}\}.$$

Since the dual function is not differentiable, it must be maximized by a subgradient method, which again has an $O(1/\sqrt{N})$ rate. To induce differentiability in the dual function and improve speed, [71] uses augmented Lagrangian. Since only one set of variables was introduced in (2.16), the augmented Lagrangian is no longer separable across the primal variables. Therefore, for each update of the dual variable, the augmented Lagrangian must be minimized iteratively. To this end, [71] uses the Frank-Wolfe Algorithm in a block-coordinate as an iterative subroutine. However, this slows down overall convergence and suffers from

[4] If we define $\phi_k(x_k) = W_{k+1}\sigma(x_k) + b_{k+1}$, where W_{k+1}, b_{k+1} are the parameters of the affine layer and σ is a layer of activation functions, this splitting coincides with the one proposed in [71, 2]

compounding errors when the sub-problems are not fully solved. When stopping early, the primal minimization must be solved to convergence in order to compute the dual function and produce a valid bound.

In contrast to the approach described above, in this paper we used a different variable splitting scheme in (2.6) that allows us to *fully* separate layers in a neural network. This subtle difference has a significant impact: we can efficiently minimize the corresponding augmented Lagrangian in *closed form*, without resorting to any iterative subroutine. Specifically, we use the ADMM algorithm, which is known to converge at an $O(1/N)$ rate [79]. In summary, our method enjoys an order of magnitude faster theoretical convergence, is more robust to numerical errors, and has minimal requirements for parameter tuning. We remark that during the updates of ADMM, the objective value is not necessarily a lower bound on $J^\star$, and hence, one must run the algorithm until convergence to produce such a bound. To stop early, we can use a similar strategy as the Frank-Wolfe approach from [71] and run the primal iteration to convergence with fixed dual variables in order to compute the dual function, which is a lower bound on $J^\star$.

2.4. Experiments

The strengths of our method are (a) its ability to exactly solve LP relaxations and (b) do so at scales. To evaluate this, we first demonstrate how solving the LP to optimality leads to tighter certified robustness guarantees in image classification and reinforcement learning tasks (Section 2.4.1) compared with the scalable fast linear bounds methods. We then stress test our method in both speed and scalability against a commercial LP solver (Section 2.4.2) and in the large network setting, e.g., solving the LP relaxation for a standard ResNet18 (Section 2.4.3). In Section 2.4.4, we apply our method on reachability analysis of neural network dynamical systems and compare with the state-of-the-art complete verification method α, β-CROWN [38]. Section 2.4.5 demonstrates the effectiveness of residual balancing in our numerical examples.

Setup In all the experiments, we focus on the setting of *verification-agnostic* networks which are networks trained without promoting verification performances, similar to [80]. All the networks have been trained adversarially with the projected gradient descent (PGD) attack [81].

In all of the test accuracy certification results reported in this paper, we initialize $\rho = 1.0$ and apply residual balancing when running ADMM. In different experiments, the stopping criterion parameters $\epsilon_{\text{abs}}, \epsilon_{\text{rel}}$ of ADMM are chosen by trial-and-error to achieve a balance between the accuracy and the runtime of the algorithm. In all these experiments the objective functions are linear [29].

2.4.1. Improved bounds from exact LP solutions

We first demonstrate how solving the LP exactly with our method results in tighter bounds than prior work that solve convex relaxations in a scalable manner. We consider two main settings: certifying the robustness of classifiers for CIFAR10 and deep Q-networks (DQNs) in Atari games. In both experiments, the pre-activation bounds for formulating the LP verification problem are obtained by the fast linear bounds developed in [29].

CIFAR10 We consider a convolutional neural network (CNN) of around 60k hidden units whose convex relaxations cannot be feasibly solved by Gurobi (for LP relaxation) or SDP solvers (for the SDP relaxation). Up until this point, the only solutions for large networks were linear bounds [3, 4] and Lagrangian-based specialized solvers [1, 71] to the LP relaxation.

In Table 2.1, we report the certified accuracy of solving the LP exactly with ADMM in comparison to a range of baselines. Verification of the CNN with ℓ_∞ perturbation at the input image with different radii ϵ is conducted on the $10,000$ test images from CIFAR10, and certified test accuracy is reported as the percentage of verified robust test images by each method. We compare with methods that have previously demonstrated the ability to bound networks of this size: fast bounds of the LP (Linear) [3, 4], and interval bounds (IBP) [5]. We additionally compare to a suite of Lagrangian-based baselines, whose effectiveness at this

scale was previously unknown. These methods [5] include supergradient ascent (Adam) [2], dual supergradient ascent (Dual Adam) [1] and a variant thereof (Dual Decomp Adam) [2], and a proximal method (Prox) [71]. As mentioned in Section 2.3.2, these baselines require solving an inner optimization problem through the iterations of the algorithms. In the experiments of Table 2.1, the number of iterations of different algorithms are bounded separately such that each Lagrangian-based method has an average runtime of 9 seconds to finish verifying one example. Our ADMM solver averages 9 seconds runtime per example in this verification task, which is the same as the average runtime of the Lagrangian-based methods, with the stopping criterion of $\epsilon_{abs} = 10^{-4}, \epsilon_{rel} = 10^{-3}$ and ρ initialized as 1.0.

In Table 2.1, we find that solving the LP exactly leads to consistent gains in certified robustness for large networks, with up to 2% additional certified robustness over the best-performing alternative. All the methods in Table 2.1 are given the same time budget. Indeed, the better theoretical convergence guarantees of ADMM translate to better results in practice: when given a similar budget, the Lagrangian baselines have worse convergence and cannot verify as many examples.

Remark 1. *By Table 2.1, our goal is to compare the performances of ADMM and other Lagrangian-based methods in solving the same LP-based neural network verification problem. The certified test accuracy can be further improved if tighter convex relaxations or the BaB techniques are applied. In fact, for the verification problem considered in Table 2.1, the state-of-the-art neural network verification method α, β-CROWN [38], which integrates fast linear bounds methods into BaB, is able to achieve certified test accuracies of $65.4\%, 56.0\%, 37.9\%, 19.9\%$ for $\epsilon = 1/255, 1.5/255, 2/255, 2.5/255$, respectively, given the same $9s$ time budget per example. However, as will be shown in Section 2.4.4, our method outperforms α, β-CROWN in reachability analysis of a neural network dynamical system, which highlights the importance of adapting neural network verification tools for tasks of different structures and scales.*

[5] These Lagrangian-based baselines were implemented using the codes at https://github.com/oval-group/decomposition-plnn-bounds

Table 2.1: Certified test accuracy (%) of PGD-trained models on CIFAR10 through ADMM, the Lagrangian decomposition methods [1, 2], and fast dual/linear [3, 4] or interval bounds [5]. All the LP-based methods (ADMM and Lagrangian) are given the same verification time budget of 9s per example. The fast linear and IBP bounds can be obtained almost instantly in this experiment, and their achievable certified test accuracy is used for reference.

	Exact	Lagrangian methods				Fast bounds	
ϵ	ADMM	Adam	Prox	Dual Adam	Dual Decomp Adam	Linear	IBP
1/255	**64.0**	60.5	62.4	59.8	60.3	59.8	42.8
1.5/255	**45.7**	41.2	43.5	40.5	41.1	36.8	16.8
2/255	**19.5**	17.3	18.2	16.9	17.1	13.2	3.6
2.5/255	**5.5**	4.6	4.9	4.5	4.6	3.3	0.7

State-robust RL We demonstrate our approach on a non-classification benchmark from reinforcement learning: verifying the robustness of deep Q-networks (DQNs) to adversarial state perturbations [72]. Specifically, we verify whether a learned DQN policy outputs stable actions in the discrete space when given perturbed states. Similar to the large network considered in the CIFAR10 setting, this benchmark has only been demonstrably verified with fast but loose linear bounds-based methods[4].

We consider three Atari game benchmarks: BankHeist, Roadrunner, and Pong [6] and verify pretrained DQNs which were trained with PGD-adversarial training [72]. For each benchmark, we verify the robustness of the DQN over $10,000$ randomly sampled frames as our test dataset.

Similar to the CIFAR10 setting, we observe consistent improvement in certified robustness of the DQN when solving the LP exactly with ADMM across multiple RL settings. We summarize the results using our method and the linear bounds on LP relaxations [3, 4] in Table 2.2.

2.4.2. Speed

We compare the solving speeds of our method with state-of-the-art solvers for convex relaxations: a commercial-grade LP solver, Gurobi. Since Gurobi cannot handle large networks, we benchmark the approaches on a fully connected network that Gurobi can handle which is

[6] [72] considers one additional RL setting (Freeway). However, the released PGD-trained DQN is completely unverifiable for nearly all epsilons that we considered.

Table 2.2: The percentage of actions from a deep Q-network that are certifiably robust to changes in the state space for three RL tasks: Bankheist, Roadrunner, and Pong. We compare fast linear bounds (Linear) [3, 4] and ADMM.

Bankheist			Roadrunner			Pong		
ϵ	Linear	ADMM	ϵ	Linear	ADMM	ϵ	Linear	ADMM
0.0016	67.0	**71.4**	0.0012	32.6	**36.6**	0.0004	96.1	**97.4**
0.0020	39.7	**49.5**	0.0016	26.3	**27.5**	0.0008	93.4	**95.6**
0.0024	12.7	**25.9**	0.0020	19.6	**22.8**	0.0012	92.1	**94.3**
0.0027	1.4	**7.3**	0.0024	1.1	**3.7**	0.0016	82.1	**84.0**

an MNIST network with architecture $784-600-400-200-100-10$ and ReLU activations.

To demonstrate the effectiveness of GPU-acceleration in the DeepSplit algorithm, we compare the runtime of DeepSplit and Gurobi in solving LP relaxations that bound the output range of the MNIST network with ℓ_∞ perturbations in the input. Specifically, for a given example in the MNIST test data set, we apply DeepSplit/Gurobi layer-by-layer to find the tightest pre-activation bounds under the LP-relaxation.

With the Gurobi solver, we need to solve 2×600 LPs sequentially to obtain the lower and upper bounds for the first activation layer, 2×400 LPs for the second activation layer, and so forth. With DeepSplit, the pre-activation bounds can be computed in batch and allows GPU-acceleration.

In our experiment, we fix the radius of the ℓ_∞ perturbation at the input image as $\epsilon = 0.02$. For the Gurobi solver, we randomly choose 10 samples from the test data set and compute the pre-activation bounds layer-by-layer. The LP relaxation is formulated in CVXPY and solved by Gurobi v9.1 on an Intel Core i7-6700K CPU, which has 4 cores and 8 threads. For each example, the total solver time of Gurobi is recorded with the average solver time being 275.9 seconds. For the DeepSplit method, we compute the pre-activation bounds layer-by-layer on 19 randomly chosen examples. The algorithm applies residual balancing with the initial $\rho = 1.0$ and the stopping criterion is given by $\epsilon_{\text{abs}} = 10^{-4}, \epsilon_{\text{rel}} = 10^{-3}$. The total running time of DeepSplit is 717.9 seconds, with 37.8 seconds per example on average. With the GPU-acceleration, our method achieves 7x speedup in verifying NN properties

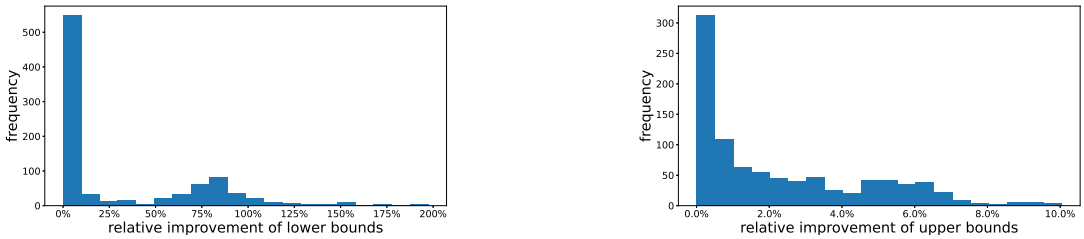


Figure 2.4: A total of 1000 ResNet18 output lower and upper bounds are computed from ADMM and LiRPA for comparison in CIFAR10. Histograms of the relative improvement percentage of ADMM over LiRPA are shown for the lower (top) and upper (bottom) bounds, which have an average relative improvement of 31.61% and 2.32%, respectively.

compared with the commercial-grade Gurobi solver.

2.4.3. Scalability

To test the scalability and generality of our approach, we consider solving the LP relaxation for a ResNet18, which up to this point has simply not been possible due to its size. The only applicable method here is LiRPA [4]—a highly scalable implementation of the linear bounds that works for arbitrary networks but can be quite loose in practice. For this experiment, we measure the improvement in the bound from solving the LP exactly in comparison to LiRPA.

The ResNet18 network is trained on CIFAR10 whose max pooling layer is replaced by a down-sampling convolutional layer for comparison with LiRPA [4] [7] which is capable of computing provable linear bounds for the outputs of general neural networks and is the only method available so far that can handle ResNet18. The ResNet18 is adversarially trained using the fast adversarial training code from [82].

In our experiments, for the first 100 test examples in CIFAR10, we use LiRPA to compute the preactivation bounds for each ReLU layer in ResNet18 and then apply ADMM to compute the lower and upper bounds of ResNet18 outputs (there are 10 outputs corresponding to the 10 classes of the dataset). The ADMM is run with stopping criterion

[7] The max pooling layer has not been considered in the implementation of LiRPA by the submission of this paper. Codes of LiRPA are available at https://github.com/KaidiXu/auto_LiRPA under the BSD 3-Clause "New" or "Revised" license.

$\epsilon_{abs} = 10^{-5}, \epsilon_{rel} = 10^{-4}$. With ℓ_∞ ball input perturbation of radius $\epsilon = 1/255$, we find that exact LP solutions with our ADMM solver can produce substantial improvements in the bound at ResNet18 scales, as shown in Figure 2.4. For a substantial number of examples, we find that ADMM can find significantly tighter bounds (especially for lower bounds).

2.4.4. Reachability analysis of dynamical systems

We consider over-approximating the reachable sets of a discrete-time neural network dynamical system $x(t+1) = f_{NN}(x(t))$ over a finite horizon where $x(t) \in \mathbb{R}^{n_x}$ denotes the state at time $t = 0, 1, \cdots$, f_{NN} is a feed-forward neural network, and n_x is the dimension of the system. Specifically, we consider a cart-pole system with 4 states under nonlinear model predictive control [83], and train a $4 - 100 - 100 - 4$ neural network $f_{NN}(x)$ with ReLU activations to approximate the closed-loop dynamics with sampling time $0.05s$. The 4 states $x = [x_1\ x_2\ x_3\ x_4]^\top$ of the cart-pole system represent the position and velocity of the cart, and the angle and angular speed of the pendulum, respectively. Given an initial set $\mathcal{X}_0 \subset \mathbb{R}^4$ such that $x(0) \in \mathcal{X}_0$, we want to over-approximate the reachable set of $x(t) = f_{NN}^{(t)}(x(0))$ where $f_{NN}^{(t)}$ denotes the t-th order composition of f_{NN} and is a feed-forward neural network itself.

Over-approximating $x(t)$ can be formulated as a neural network verification problem (2.1) where $f = f_{NN}^{(t)}$ is given by the sequential concatenation of t copies of f_{NN} and the input set $\mathcal{X}$ is chosen as the initial set $\mathcal{X}_0$. A box approximation of $x(t)$ can be obtained by minimizing/maximizing the i-th output of the neural network $f_{NN}^{(t)}$ for $1 \leq i \leq n_x = 4$.

With a randomly chosen initial set $x(0) \in \mathcal{X}_0 = \{x \mid x_1 \in [-1.8\ -1.6]$ m, $x_2 \in [0.5\ 0.7]$ m/s, $x_3 \in [0.035\ 0.105]$ rad, $x_4 \in [0.2\ 0.4]$ rad/s$\}$, we consider the horizon $t = 20$ and compute a box over-approximation of the reachable set of $x(20)$ through both DeepSplit and the state-of-the-art BaB-based verification method α, β-CROWN [38] [8]. From 1000 simulated trajectories with uniformly sampled initial states from $\mathcal{X}_0$, the emperically estimated lower and upper bounds for $x(20)$ are given by $[-1.74\ 0.14\ -0.10\ 0.022]^\top \leq x(20) \leq [-0.91\ 0.92\ -$

[8] Codes of α, β-CROWN are available at https://github.com/huanzhang12/alpha-beta-CROWN under the BSD 3-Clause "New" or "Revised" license.

$0.036\ 0.11]^\top$. With the time budget 1470s per bound, the box over-approximations of $x(20)$ given by DeepSplit and α, β-CROWN are given as follows:

$$\text{DeepSplit: } \begin{bmatrix} -4.19 \\ -4.39 \\ -2.14 \\ -3.88 \end{bmatrix} \leq x(20) \leq \begin{bmatrix} 3.00 \\ 4.81 \\ 1.99 \\ 3.91 \end{bmatrix},$$

$$\alpha, \beta\text{-CROWN : } \begin{bmatrix} -38.38 \\ -46.52 \\ -23.85 \\ -55.97 \end{bmatrix} \leq x(20) \leq \begin{bmatrix} 37.08 \\ 48.58 \\ 22.86 \\ 48.21 \end{bmatrix}.$$

We observe that DeepSplit gives a reasonable over-approximation of the reachable set of $x(20)$, while the bounds obtained by α, β-CROWN in this task are an order of magnitude more conservative. Such comparison result holds for other randomly chosen initial sets $\mathcal{X}_0$ too.

2.4.5. Effects of residual balancing

We demonstrate the effects of residual balancing on the convergence of ADMM through the MNIST network. We conduct our experiment on the 1938-th example which is randomly chosen from the MNIST test dataset. For this example, the MNIST network predicts its class (number 4) correctly. We add an ℓ_∞ perturbation of radius $\epsilon = 0.02$ to the input image and verify if the network outputs are robust with respect to class number 3. This corresponds to setting $i^\star = 4, i = 3$ and $\mathcal{X} = \{x \mid \|x - x^*\|_\infty \leq 0.02\}$ in problem (2.2) where x^* denotes the chosen test image.

The maximum number of iterations is restricted to 3000. The objective values, primal and dual residuals of ADMM for the network under different fixed augmentation parameters ρ are plotted in Figure 2.5. The residual balancing in this experiment is applied with $\tau = 2$, $\mu = 10$, and ρ initialized as 10.0.

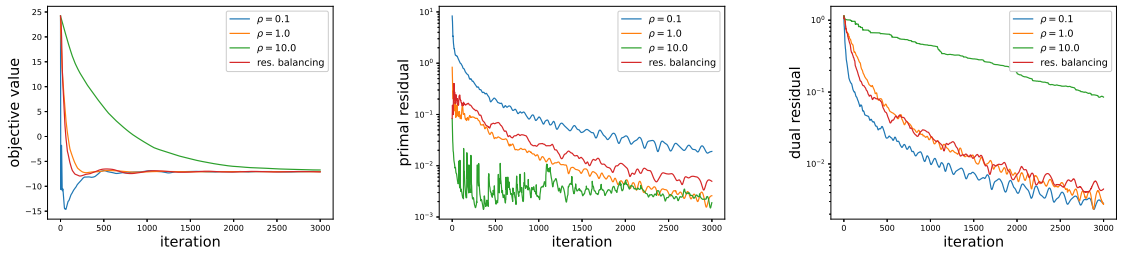


Figure 2.5: The objective values (left), primal residuals (middle), and dual residuals (right) of ADMM under different augmentation parameters ρ on the fully connected MNIST network.

The effects of ρ on the convergence rates of the primal and dual residuals are illustrated empirically in Figure 2.5. Despite initialized at a large value 10.0, residual balancing is able to adapt the value of ρ and achieves significant improvement in convergence rate compared with the case of constant $\rho = 10.0$. As observed in our other experiments, with residual balancing, ADMM becomes insensitive to the initialization of ρ and usually achieves a good convergence rate.

2.5. Chapter Summary

In this paper, we proposed DeepSplit, a scalable and modular operator splitting technique for solving convex relaxation-based verification problems for neural networks. The method can exactly solve large-scale LP relaxations with GPU acceleration with favorable convergence rates. Our approach leads to tighter bounds across a range of classification and reinforcement learning benchmarks, and can scale to a standard ResNet18. We leave as future work a further investigation of variations of ADMM that can improve convergence rates in deep learning-sized problem instances, as well as extensions beyond the LP setting. Furthermore, it would be interesting to extend the proposed method to verification of recurrent neural neworks (RNNs) such as vanilla RNNs, LSTMs[9], and GRUs[10] [84, 85, 86].

[9]Long Short-Term Memory.
[10]Gated Recurrent Unit.

CHAPTER 3

Reachability Analysis of Neural Network Dynamical Systems

3.1. Background

Although NN verification methods only consider NNs in isolation, they can be conveniently combined with existing reachability analysis tools to certify properties of NNDS [44, 45, 46] over a finite horizon. For a discrete-time NNDS, NN verification methods can readily compute an over-approximation of the one-step reachable set given a bounded input set. Then, applying such one-step over-approximation method recursively for $t = 0, 1, \cdots, T$ leads to a bounding tube of the NNDS trajectories (blue boxes in Fig. 3.1).

We denote the above methodology as the *recursive reachability analysis* framework [47, 48, 49], and compare it with an alternative framework that computes over-approximations of the reachable sets in one shot [50, 51]. In *one-shot reachability analysis*, the reachable set of NNDS at time t is bounded by directly applying NN verification methods on the unrolled NN dynamics for t steps (red boxes in Fig. 3.1). While one may observe that the one-shot analysis generates tighter bounds than the recursive counterpart due to the use of iterated dynamics [50], this property does not hold for general NN verification tools. Correspondingly, our contributions are:

- We provide a counter-example where one-shot analysis results in worse bounds than the recursive framework, highlighting that the one-shot analysis does not always lead to tighter bounds.
- We formally prove conditions under which the one-shot framework provides tighter bounds compared with the recursive framework for a general class of NN verification methods.
- Our analysis applies to NNs with general architectures and allows us to consider disturbances in the computation of the reachable set over-approximations. Numerical

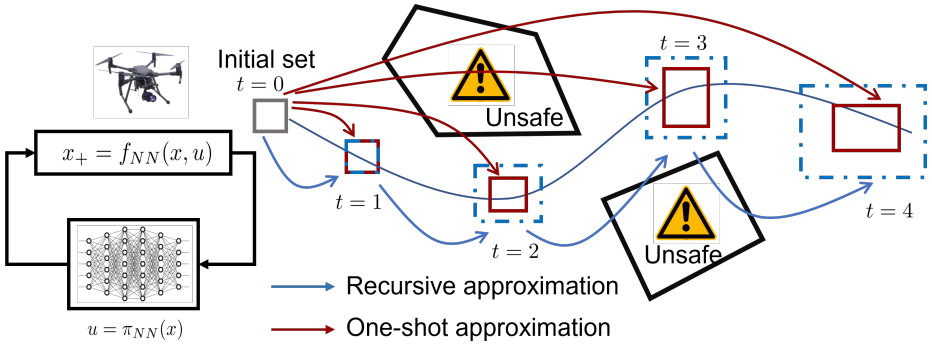


Figure 3.1: Reachable set over-approximations of a NN dynamical system can be obtained by applying NN verification methods recursively (blue arrows) or in one-shot (red arrows). In the one-shot analysis, unrolled NN dynamics over multiple time steps is considered. In this work, we investigate in what cases the one-shot analysis generates tighter bounds than the recursive counterpart.

examples are provided to demonstrate the applicability of the one-shot framework.

3.1.1. Related works

NN verification: NN verification methods analyze NNs in isolation and can certify the worst-case performance of NNs under bounded input perturbations. A rich set of tools with distinct tightness/complexity trade-offs are developed, including those based on mixed-integer programming (MIP) [27], semidefinite programming (SDP) [35, 36], linear programming (LP) [29], and linear bounds propagation [33, 34, 4]. Particularly, [4] considers NNs with general architectures. Tightness comparison of different convex relaxation-based NN verifiers is discussed in [73, 39] from the dual perspective.

Reachability analysis of NNDS: Verifying a closed-loop dynamics with a NN controller often involves first bounding the set of control inputs (e.g., by polytopes [87], linear bounds [88], star sets [89], polynomial zonotopes [46], hybrid automaton [28], Taylor models [90]) in each control cycle, and then estimating the reachable set of the states. Several works [45, 91, 49] process the overall closed-loop dynamics directly. The application of the recursive analysis framework to find an over-approximation of the reachable set in a finite horizon can be found in [89, 92, 88, 49]. The effectiveness of the one-shot analysis in reduc-

ing compounding errors is demonstrated and analyzed in detail in OVERT [50] [11], a safety verification method for nonlinear systems with NN controllers. In OVERT, the nonlinear dynamics is over-approximated by piecewise linear relations such that the safety of the closed-loop system can be verified through mixed-integer programming (MIP). Instead of focusing on one particular NN verification method, in this work we aim to provide guarantees on the performance of the one-shot framework for a range of NN verification methods including MIP [27], linear programming (LP) [29], and linear bounds propagation-based ones [34, 4]. Our analysis allows straightforward integration of these NN methods into the one-shot framework to further boost their performances in safety verification.

3.2. Preliminaries and Problem Formulation

In this section, we formalize the reachability analysis problem for NNDS and two frameworks to approach it: the recursive and the one-shot analysis. Although, intuitively, the one-shot method should outperform the recursive method in tightness at the cost of computational overhead, we provide a motivating example that falsifies this claim, which calls for a formal comparison between these two frameworks.

3.2.1. Neural network dynamical systems

We denote a discrete-time NNDS with disturbances as

$$x_{t+1} = f(x_t, w_t) \tag{3.1}$$

where $x_t \in \mathbb{R}^{n_x}$ is the state, $w_t \in \mathbb{R}^{n_w}$ denotes the disturbances at time t, and f is a neural network with an arbitrary architecture taking (x_t, w_t) as its input. For example, Eq. (3.1) can represent the closed-loop dynamics of a system consisting of several NN modules:

$$\begin{cases} x_{t+1} = f_{NN}(x_t, u_t) + w_t^x \\ y_t = p_{NN}(x_t) + w_t^y \\ u_t = \pi_{NN}(y_t) \end{cases} \tag{3.2}$$

[11] The one-shot framework is denoted as the symbolic approach in [50].

where f_{NN}, p_{NN}, π_{NN} denote, respectively, the NN dynamics, measurement function and control policies. The process and measurement disturbances w_t^x and w_t^y can be considered as components of a compound disturbance $w_t = [w_t^{x\top}\ w_t^{y\top}]^\top$. Then, $f(x_t, w_t)$ denotes a NN with the connection diagram shown in Fig. 3.2.

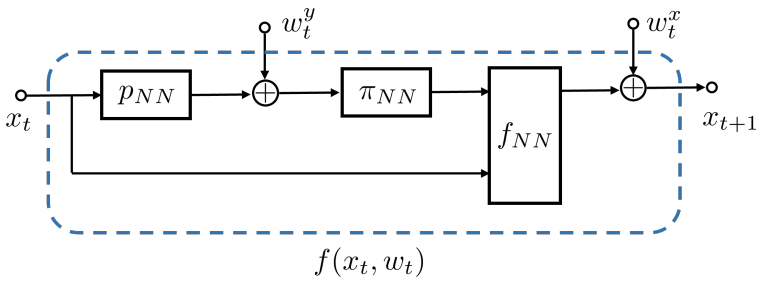


Figure 3.2: Illustration of a neural network dynamical system with several interconnected NNs.

3.2.2. Finite-step reachability analysis

For the NNDS (3.1), define the one-step forward reachable set of $f(\cdot)$ from a given set $\mathcal{X} \times \mathcal{W} \subset \mathbb{R}^{n_x} \times \mathbb{R}^{n_w}$ as

$$f(\mathcal{X} \times \mathcal{W}) := \{y \in \mathbb{R}^{n_x} \mid y = f(x, w),\ (x, w) \in \mathcal{X} \times \mathcal{W}\}. \tag{3.3}$$

For a given set of initial conditions $\mathcal{X}_0 \subset \mathbb{R}^{n_x}$, the forward reachable set $\mathcal{R}_t(\mathcal{X}_0)$ of the NNDS (3.1) at time t is defined by the recursion

$$\mathcal{R}_{t+1}(\mathcal{X}_0) = f(\mathcal{R}_t(\mathcal{X}_0) \times \mathcal{W}),\ \mathcal{R}_0(\mathcal{X}_0) = \mathcal{X}_0. \tag{3.4}$$

Due to the complex structure of the function $f(\cdot)$, exactly identifying $\mathcal{R}_t(\mathcal{X}_0)$ is computationally challenging. Instead, we aim to find over-approximations $\bar{\mathcal{R}}_t(\mathcal{X}_0) \supseteq \mathcal{R}_t(\mathcal{X}_0)$ of the reachable sets at time t. Once these reachable set over-approximations are found, we can verify that the NNDS is safe, i.e., its trajectories starting from an initial set $\mathcal{X}_0$ never reach predefined unsafe regions in the state space over a finite horizon T under disturbances. When the initial set is clear from context, we drop the $\mathcal{X}_0$ argument from $\overline{\mathcal{R}}_t$ for notational simplicity.

3.2.3. Reachability analysis frameworks

We first define the concept of *propagator* as a basic reachability analysis module that abstracts different bounding methods of the output range of NNs.

Definition 1 (Propagator)**.** *Any method P that can bound the output of the NNDS $f(x, w)$ given a bounded input set $\mathcal{X} \times \mathcal{W}$ is defined as a* propagator*, i.e., $P(\mathcal{X} \times \mathcal{W}; f) \subset \mathbb{R}^{n_x}$ and $P(\mathcal{X} \times \mathcal{W}; f) \supseteq f(\mathcal{X} \times \mathcal{W})$.*

The first argument of the propagator. $\mathcal{X} \times \mathcal{W}$, denotes the input set while the second argument denotes the NN it acts on. The propagator can be obtained by running a NN verification algorithm, and its output set $P(\mathcal{X} \times \mathcal{W})$ often follows a pre-fixed geometric template specific to each NN verification algorithm, e.g., polytope [93], ellipsoid [49], zonotope [94], or polynomial zonotope [46]. In this work, we compare two frameworks that employ propagators to compute the reachable set over-approximations $\bar{\mathcal{R}}_t$ of the NNDS over a finite horizon:

a) Recursive reachability analysis which applies the propagator P for one-step reachable set over-approximation recursively, i.e., by using the reachable set over-approximation $\bar{\mathcal{R}}_t$ as the input set for computing $\bar{\mathcal{R}}_{t+1}$, as follows:

$$\begin{aligned} \textbf{Recursive:} \quad & \bar{\mathcal{R}}_0 = \mathcal{X}_0, \\ & \bar{\mathcal{R}}_{t+1} = P(\bar{\mathcal{R}}_t \times \mathcal{W}; f),\ t \geq 0. \end{aligned} \tag{3.5}$$

b) One-shot reachability analysis which applies the propagator on the t-th order composition of the NNDS (3.1) to compute $\bar{\mathcal{R}}_t$ directly for $t > 1$, as follows:

$$\begin{aligned} \textbf{One-shot:} \quad & \bar{\mathcal{R}}_0 = \mathcal{X}_0, \\ & \bar{\mathcal{R}}_t = P(\mathcal{X}_0 \times \mathcal{W}^t; f^{(t-1)}), t \geq 1, \end{aligned} \tag{3.6}$$

where $f^{(t)}(x_0, w_{0:t})$ denotes the t-th order composition of the NNDS, i.e., $x_{t+1} = f^{(t)}(x_0, w_{0:t})$,

and $\mathcal{W}^t$ denotes the Cartesian product of t disturbance sets. The notation $w_{0:t}$ is shorthand for the set $\{w_0, \cdots, w_t\}$. Due to the cascading structure of NNs, one can easily represent the composition of the NNDS (3.1) over multiple time steps as a concatenated network.

Intuitively, the one-shot method should generate tighter $\bar{\mathcal{R}}_t$ than the recursive method, since it utilizes the exact description $f^{(t)}(x_0, w_{0:t})$ of x_{t+1} for reachable set over-approximation and incurs more computational cost due to the use of larger NNs (t times the size of the NN considered in the recursive framework). However, we find a counterexample where the opposite holds, which shows that the one-shot verification method does not always result in bounds tighter than those obtained from the recursive framework.

Example 2 (Counterexample)**.** *We approximate the discretized dynamics of the 2D Rayleigh-Duffing oscillator [6] as a feed-forward ReLU NN. No disturbances are considered. In Fig. 3.3, the box reachable set over-approximations obtained from the one-shot and recursive frameworks are compared using the backward linear bounds propagation (left figure) and forward linear bounds propagation (right figure), respectively (see [4] for details). While the one-shot framework gives tighter bounds than the recursive one in the former case, the opposite holds in the latter.*

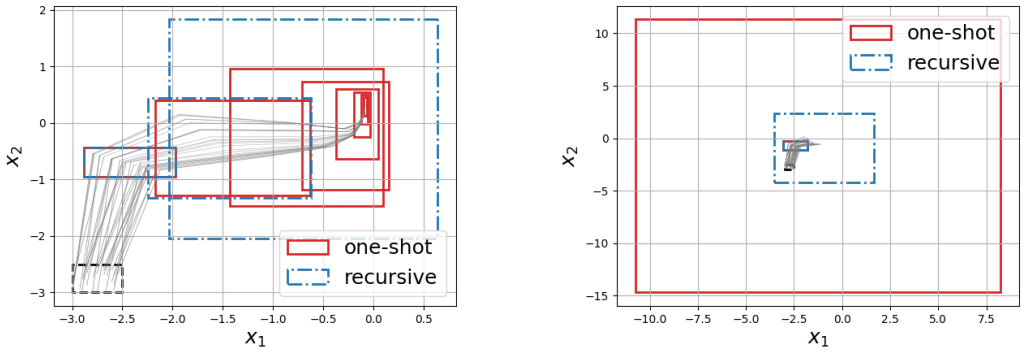
3.2.4. Problem formulation

Motivated by Example 2, in this paper we investigate the following question: In what cases is it beneficial to verify or optimize over the unrolled NN dynamics in one-shot?

In Section 3.3, we characterize a general class of NN verification methods that are guaranteed to benefit from the one-shot analysis. Our proof applies to NNDS with arbitrary architecture. The implication of our analysis for verification algorithm design is discussed in Section 3.3.4, with numerical examples validating our analysis in Section 3.4.

3.3. Tightness Improvement Guarantees

Neural networks implement mathematical functions that can be represented by a computational graph. In this section, we first describe the graph representation of NNs with



(a) Backward bounds propagation.

(b) Forward bounds propagation.

Figure 3.3: Comparison of one-shot and recursive analysis of a NNDS approximating a Rayleigh-Duffing oscillator [6]. Box reachable set over-approximations obtained by the backward (left)/forward (right) linear bounds propagation are plotted, as well as randomly sampled trajectories of the NNDS. In the left figure, the recursively synthesized bounds are shown for only 3 steps. In the right figure, bounds over 2 steps are plotted for both one-shot and recursive frameworks.

arbitrary architecture, based on which the class of separable NN verification methods and propagators are defined. Then, a formal tightness improvement guarantee of the one-shot framework for separable propagators is provided in Theorem 1.

3.3.1. NN representation

Similar to [4], we can represent a general NN through its computational graph, which is defined as a directed acyclic graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V} = \{z_0, \cdots, z_q, z_{q+1}, \cdots, z_L\}$. We denote $\mathcal{I} = \{z_0, \cdots, z_q\}$ as the input nodes, or independent nodes, of the graph and $\mathcal{D} = \{z_{q+1}, \cdots, z_L\}$ as the dependent nodes. The edge set $\mathbf{E}$ is a set of pairs (i, j) denoting that node z_i is an input argument of node z_j. Define $\text{Pre}(z_j) = \{z_{j,1}, \cdots, z_{j,m(j)}\}$ as the set of input nodes to z_j with cardinality $|\text{Pre}(z_j)| = m(j)$. With this notation, we can describe the operator generating z_j as $z_j = \phi_j(\text{Pre}(z_j))$ where ϕ_j can represent any commonly used operators in deep NNs such as linear, convolutional, MaxPooling, and activation layers, etc. We have $|\text{Pre}(z_j)| = 0$ for all input nodes $z_j \in \mathcal{I}$, and $|\text{Pre}(z_j)| > 0$ for the rest. All maps from the input nodes $\mathcal{I}$ to any dependent node $z_j \in \mathcal{D}$ are defined by the computational graph $\mathbf{G}$ and the operators $\phi_j(\cdot)$ associated to each node. We assume z_L is the output of

the NN whose property is of our concern.

Connection to NNDS: Given $T > 0$, the unrolled NNDS $x_T = f^{(T)}(x_0, w_{0:T-1})$ for T steps can be considered as a single NN with the computational graph $\mathbf{G}$. In this graph, the input nodes represent $\{x_0, w_0, \cdots, w_{T-1}\}$ and the dependent nodes represent all intermediate variables including states $x_{1:T-1}$. The state x_T is the output node of $\mathbf{G}$, and our goal is to bound the reachable set of x_T for safety verification.

3.3.2. Separable propagator

In this paper, we consider NN verification methods that solve an optimization problem based on a layer-wise abstraction of $\phi_j(\cdot)$ in the NN computational graph. Specifically, these methods verify the properties of a NN by solving problems of the following form:

$$\begin{aligned} \underset{z_{0:L}}{\text{minimize}} \quad & J(z_L) \\ \text{subject to} \quad & (\text{Pre}(z_k), z_k) \in \mathcal{S}_k, k = q+1, \cdots, L, \\ & z_k \in \mathcal{X}_k, k = 0, \cdots, q, \end{aligned} \tag{3.7}$$

where J is a real-valued function encoding the specification to be verified, $\mathcal{X}_k$ is the bounded input set of each input node z_k for $0 \leq k \leq q$, and $\mathcal{S}_k$ denotes the constraints describing the relationship between node z_k and its input nodes. We can view $\mathcal{S}_k$ as a set defined on the joint space of $(\text{Pre}(z_k), z_k)$, possibly described by expressions with intermediate variables, as shown in the following example.

Example 3 (Optimization constraints)**.** *Consider a scalar ReLU function* $y = \max(x, 0)$. *Given lower and upper bounds on the scalar input* $x \in [\ell, u]$ [12]*, the constraint* $(x, y) \in \mathcal{S}$ *in different NN verification methods is formulated as:*

[12]We consider the non-trivial case where $\ell < 0, u > 0$. Otherwise, the ReLU layer is reduced to a linear one.

a) Mixed-integer linear constraints [27]:

$$y \geq 0,\ y \geq x,\ y \leq x - (1-\mu)\ell,\ y \leq \mu u,\ \mu \in \{0,1\}. \tag{3.8}$$

b) Linear constraints [29]:

$$y \geq 0,\ y \geq x,\ y \leq \frac{u}{u-\ell}x - \frac{u\ell}{u-\ell}. \tag{3.9}$$

c) Simplified linear constraints [34] that are amenable to linear bounds propagation:

$$y \leq \frac{u}{u-\ell}x - \frac{u\ell}{u-\ell},\ y \geq \alpha x. \tag{3.10}$$

where $\alpha \in [0,1]$ is a user-defined parameter.

In (3.8)*, an intermediate binary variable μ is introduced to formulate $\mathcal{S}$ as the graph of the ReLU function. In* (3.9)*,* (3.10)*, the set $\mathcal{S}$ is a polyhedron in the space of (x, y). The above abstractions of a ReLU function can be easily extended to all neurons in a NN, leading to the verification problem* (3.7)*.*

If a NN verifier does not consider the coupling between nodes from different layers, the constraints $\mathcal{S}_k$ in problem (3.7) are separable as defined below.

Definition 2 (Separable constraints)**.** *Constraints $\mathcal{S}_i$ and $\mathcal{S}_j$ such that $(x_i, y_i) \in \mathcal{S}_i$, $(x_j, y_j) \in \mathcal{S}_j$ are separable if the intermediate variables defining $\mathcal{S}_i$ and $\mathcal{S}_j$ are independent.*

Definition 3 (Separable NN verifier)**.** *We denote any NN verification method that solves an optimization problem of the form* (3.7) *with separable constraints $\mathcal{S}_k$ a separable NN verifier.*

Generality of separable NN verifiers: Separable NN verifiers include verification methods that either explicitly or *implicitly* solve a primal optimization problem of the

form (3.7) with separable constraints. For example, for a ReLU network, all complete verifiers [26, 76, 37] are considered separable since they are equivalent to the MIP-based method [27]. Similarly, scalable Lagrangian-based methods [71, 42] that solve the LP-based verification problem using the dual formulation are separable due to the strong duality of LP. Importantly, we observe that backward linear bounds propagation methods [34, 4], which are scalable and can handle general activation functions, are also separable since they exactly solve an LP verification problem built on linear bounds abstractions of non-linear activation functions with constraints similar to (3.10). On the contrary, we cannot identify any implicit separable optimization problem for the forward linear bounds propagation method [4]. In fact, the forward method is used to construct the counterexample in Example 2.

Construction of the propagator: A propagator can be constructed by applying a NN verifier to solve problem (3.7) multiple times with different linear objective functions $J_i(z_L) = c_i^\top z_L, i = 1, \cdots, N$ in order to obtain a polytopic over-approximation of the output set of the NN. In what follows, we assume the normal vectors $\{c_i\}_{i=1}^N$ defining the polytopic over-approximation are fixed. A propagator is called separable if it applies a separable NN verifier to bound the NN outputs.

3.3.3. Tightness comparison

Recall that we can represent the T-step unrolled NNDS by a computational graph $\mathbf{G}$ with x_T being the output node. In the one-shot framework, a propagator is directly applied to the computational graph $\mathbf{G}$. In the recursive framework, sub-networks of $\mathbf{G}$ are extracted such that the ranges of the intermediate nodes $x_{1:T-1}$ are over-approximated by the propagator as $\bar{\mathcal{R}}_t$ sequentially. Once the bound $\bar{\mathcal{R}}_t$ on the range of the intermediate state x_t for $1 \leq t \leq T-1$ is computed, x_t is treated as an input node to the downstream dependent nodes with input set $\bar{\mathcal{R}}_t$. We denote this process as *concretization* of a dependent node in the computational graph $\mathbf{G}$.

We now show that for a separable propagator, concretizing the range of any dependent

nodes in the NN computational graph $\mathbf{G}$ increases conservatism in bounding the output node. This immediately indicates that in reachability analysis with a separable propagator, the recursive framework is more conservative than the one-shot framework. Our analysis is conducted in two steps:

Recursive framework In the recursive framework, we first bound the range of a dependent node z_k for some $q < k < L$. This can be achieved by applying a NN verifier solving problem (3.7) with objectives $J_i(z_k) = c_i^\top z_k$, $i = 1, \cdots, N$ and the truncated constraints corresponding to the sub-network with z_k as the output. To extract relevant constraints of this sub-network, we apply Breath First Search (BFS) on the graph $\mathbf{G}$ as shown in Algorithm 2. The set of extracted constraints is denoted as $\mathcal{S}_{input\to k} = \textsc{ExtractConstraints}(\text{InputNodes} = z_{0:q}, \text{OutputNodes} = z_k)$ [13], and the NN verifier solves

$$\begin{aligned} \text{minimize} \quad & J_i(z_k) \\ \text{subject to} \quad & (z_{0:q}, z_k) \in \mathcal{S}_{input\to k}, \\ & z_{0:q} \in \mathcal{X}, \end{aligned} \tag{3.11}$$

for $i = 1, \cdots, N$ where the input set is given by $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_q$. Denote J_i^* the optimal values of (3.11). Then, the over-approximation of the range of z_k is given by [14]

$$\bar{\mathcal{X}}_k = \{z \mid J_i(z) \geq J_i^*, i = 1, \cdots, N\}. \tag{3.12}$$

Next, we treat z_k as an input node with domain $\bar{\mathcal{R}}_k$ to bound z_L. We consider the sub-network with $z_{0:1} \cup \{z_k\}$ as inputs and z_L as output. We extract the associated constraints as $\mathcal{S}_{input,k\to L} = \textsc{ExtractConstraints}(\text{InputNodes} = z_{0:q} \cup \{z_k\}, \text{OutputNodes} = z_L)$.

[13] We omit the arguments of the NN computational graph $\mathbf{G}$ and the constraint set $\{\mathcal{S}_i\}$ from (3.7) since they are clear from context.

[14] Here we consider arbitrary dependent nodes z_k in the computational graph of the unrolled NNDS. If the chosen node corresponds to state x_t, the over-approximation set $\bar{\mathcal{X}}_k$ from (3.12) coincides with $\bar{\mathcal{R}}_t$.

Bounding the range of z_L is now achieved by solving

$$\begin{aligned} \text{minimize} \quad & J_i(z_L) \\ \text{subject to} \quad & (z_{0:q} \cup z_k, z_L) \in \mathcal{S}_{input,k\to L}, \\ & z_{0:q} \in \mathcal{X}, z_k \in \bar{\mathcal{X}}_k, \end{aligned} \tag{3.13}$$

for $i = 1, \cdots, N$. Denote the optimal values of (3.13) as $J^*_{i,re}$. The over-approximation of z_L obtained by the recursive framework is given by $\bar{\mathcal{X}}_{L,re} = \{z \mid J_i(z) \geq J^*_{i,re}, 1 \leq i \leq N\}$.

One-shot framework In the one-shot analysis framework, we directly solve problem (3.7) with objectives $J_i(z_L)$ for $i = 1, \cdots, N$. Denote the optimal values of these problems as $J^*_{i,os}$. Then, the over-approximation of z_L obtained by the one-shot framework is given by $\bar{\mathcal{X}}_{L,os} = \{z \mid J_i(z) \geq J^*_{i,os}, 1 \leq i \leq N\}$.

Algorithm 2 Extract concretization constraints

Input: InputNodes $\subseteq \{z_i\}_{i=0}^{L}$, OutputNodes z_k, computational graph $\mathbf{G}$, constraints $\{\mathcal{S}_i\}_{i=q+1}^{L}$ from problem (3.7).
Output: Constraint set $\mathcal{S}$.

```
procedure ExtractConstraints
    Let Q be a queue and S = ∅.
    Q.enqueue(z_k). Mark node z_k as explored.
    while Q is not empty do
        z_i = Q.dequeue().        ▷ Subscript i denotes the label of the node in graph G.
        S = S ∪ S_i.
        for z_j in Pre(z_i) do
            if z_j ∉ InputNodes and is not explored then
                Q.enqueue(z_j) and label z_j as explored.
```

Lemma 1. *With a separable propagator, the one-shot framework generates a tighter over-approximation of the NN output than the recursive framework, i.e.,* $\bar{\mathcal{X}}_{L,os} \subseteq \bar{\mathcal{X}}_{L,re}$.

Proof. To allow a straightforward comparison, we combine the two-step computation (3.11) and (3.13) of the recursive framework into one optimization problem by introducing auxiliary variables $\tilde{z}_{0:L}$ and $\tilde{z}^j_{0:L}$ for $j = 1, \cdots, N$. We can equivalently rewrite (3.13) as

$$
\begin{aligned}
\text{minimize} \quad & J_i(\tilde{z}_L) \\
\text{subject to} \quad & (\tilde{z}^j_{0:q}, \tilde{z}^j_k) \in \mathcal{S}_{input \to k},\ \tilde{z}^j_{0:q} \in \mathcal{X}, \\
& J_i(\tilde{z}_k) \geq J_i(\tilde{z}^j_k), j = 1, \cdots, N, \\
& (\tilde{z}_{0:q} \cup \tilde{z}_k, \tilde{z}_L) \in \mathcal{S}_{input, k \to L}, \\
& \tilde{z}_{0:q} \in \mathcal{X}.
\end{aligned} \tag{3.14}
$$

By the separability of constraints $\mathcal{S}_k$ in (3.7), we have that any feasible solution $z_{0:L}$ of (3.7) constructs a feasible one of (3.14) by setting $\tilde{z}_\ell = \tilde{z}^j_\ell = z_\ell$ for all $0 \leq \ell \leq L$, $1 \leq j \leq N$ with the same objective value. Then we have $J^*_{i,os} \geq J^*_{i,re}$ for all $1 \leq i \leq N$ and $\bar{\mathcal{X}}_{L,os} \subseteq \bar{\mathcal{X}}_{L,re}$. □

The proof of Lemma 1 shows that the performance gap between the one-shot and recursive frameworks arises from the concretization errors, i.e., the over-approximation error of applying a finite number of hyperplanes to bound the reachable set of a dependent node in the NN. Such errors may compound when we concretize the ranges of a sequence of dependent nodes one-by-one as in recursive reachability analysis of the NNDS (3.1).

Theorem 1. *The one-shot framework is guaranteed to generate tighter over-approximations of reachable sets than the recursive framework for the NNDS* (3.1) *when a separable propagator is applied.*

Proof. We can treat the T-step unrolled NNDS as the NN with input nodes $(x_0, w_0, \cdots, w_{T-1})$ and x_T as the output node. By Lemma 1, concretizing any intermediate dependent node x_t leads to more conservative over-approximation of x_T. □

3.3.4. Practical Implications

We now summarize the comparison between the one-shot and recursive frameworks: (a) The one-shot framework solves the NN verification problem on NNs $f^{(t)}(\cdot)$ with growing sizes as t increases, while in the recursive framework a NN $f(\cdot)$ of constant size is considered. Therefore, the one-shot framework is always more computationally costly to run than the

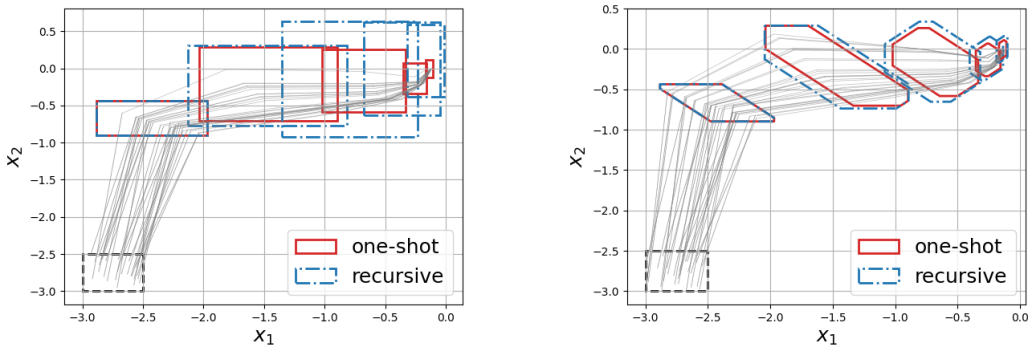


Figure 3.4: The LP-based propagator using relaxation (3.9) is applied to bound the reachable sets of the Rayleigh-Duffing oscillator NNDS using a box template (left, 4 hyperplanes) and a more complex polyhedral template (right, 8 hyperplanes). Randomly sampled trajectories of the NNDS are shown in gray. The tightness gap between the one-shot and recursive frameworks decreases as the complexity of the bounding templates increases.

recursive framework. (b) The recursive framework generates looser bounds than the one-shot framework due to the concretization error as shown in Lemma 1 and the induced conservatism could explode as time t increases.

One-shot analysis principle: The above two points indicate that we can treat the size of NNs, or the order of composition of the NNDS $f(\cdot)$, as a tuning parameter which makes a trade-off between tightness and computational complexity. Within a given computational resource budget (either in time or in memory), our analysis suggests that we should run a separable propagator on the composition $f^{(t)}$ of the NNDS with t as large as possible.

Concretization error reduction: Problem (3.14) indicates that increasing N tightens the search space of $\tilde{z}_k$ and consequently the output variable $\tilde{z}_L$. Therefore, the concretization error from the recursive framework can be decreased by considering a richer set of objective functions $\{c_i^\top z_L\}$ in constructing the propagator. This phenomenon is demonstrated in Fig. 3.4. However, the overall complexity of the recursive framework increases with N, and it remains a challenge to identify a set of effective bounding hyperplanes $\{c_i\}_{i=1}^N$ for systems with high dimensions.

3.4. Numerical Examples

We demonstrate the improvement of tightness of the one-shot analysis on a NNDS that approximates the closed-loop dynamics of a cart-pole system. Separable propagators relying on LP and backward linear bounds propagation are considered with different NNDS structures. All experiments are run on an Intel Core i7-6700K CPU.

3.4.1. LP-based propagator

We consider the cart-pole dynamics without friction from [95] where the cart has a mass of 0.25 kg, and the pole has a mass of 0.1 kg and length of 0.4 m. The entries of the 4-dimensional state $x = [x_1\ x_2\ x_3\ x_4]^\top$ represent the position, the velocity of the cart, the angle, and the angular speed of the pole, respectively. We train a feedforward ReLU NN $f_{cl,NN}(x)$ with the structure $4 - 100 - 100 - 4$ to approximate the discretized closed-loop dynamics of the cart-pole system under a nonlinear model predictive controller (MPC) [83] with $dt = 0.05$s. An LP-based verifier that bounds ReLU by (3.9) is applied to compute reachable sets of the NNDS

$$x_{t+1} = f_{cl,NN}(x_t). \tag{3.15}$$

The pre-activation bounds (ℓ, u) in (3.9) are computed in an inductive manner by the verifier. The initial set is given by $\mathcal{X}_0 = \{x \mid x_1 \in [2.0, 2.2]\ \text{m}, x_2 \in [1.0, 1.2]\ \text{m/s}, x_3 \in [-0.174, -0.104]\ \text{rad}, x_4 \in [-1.0, -0.8]\ \text{rad/s}\}$. In Fig. 3.5, the over-approximating boxes obtained by the recursive method (dashed blue) for 8 steps are plotted which quickly become overly conservative, while the one-shot method is able to provide informative bounds (solid red) on the system states for 30 steps. The LP propagators are solved by Gurobi [96], and the total solver time for running the one-shot and recursive methods for 8 steps are 161.3 and 3.1 seconds, respectively.

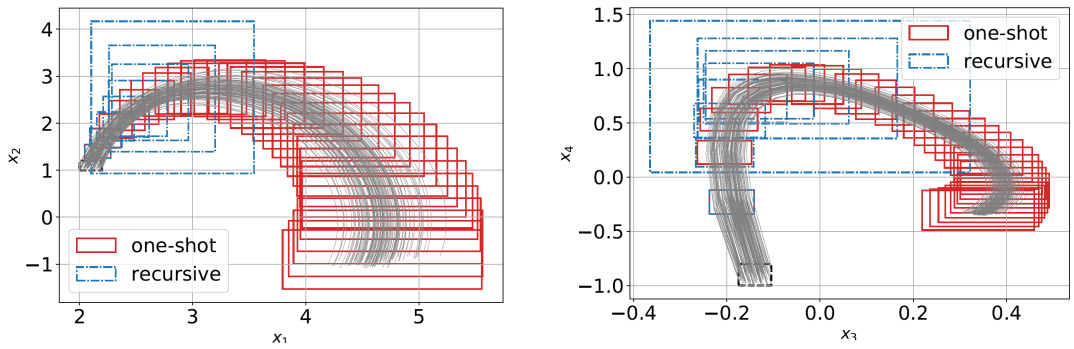


Figure 3.5: Box reachable set over-approximations of the feedforward NNDS (3.15) obtained by the one-shot (solid red boxes, 30 steps) and recursive (dashed blue boxes, 8 steps) frameworks using the LP-based propagator. The boxes are projected onto the (x_1, x_2) and (x_3, x_4) planes for plotting. Randomly sampled trajectories of the NNDS are shown in gray.

3.4.2. Backward propagation-based propagator

Next, we adopt a different approximation of the closed-loop dynamics of the cart-pole system which has a more complex structure:

$$x_{t+1} = Ax_t + Bu_t + f(x_t, u_t) + w_t \tag{3.16a}$$

$$u_t = \pi_{NN}(x_t) \tag{3.16b}$$

where the linear dynamics (A, B) are derived from the Jacobian matrices of the cart-pole dynamics at the origin. A feedforward NN $f(x, u) : \mathbb{R}^5 \mapsto \mathbb{R}^4$ is trained to learn the residual nonlinear dynamics with training data randomly sampled from the discretized cart-pole dynamics with $dt = 0.05$s. We implement a nonlinear MPC for the open-loop NNDS (3.16a) in MATLAB to collect samples of (x_t, u_t) and train the NN controller $\pi(\cdot) : \mathbb{R}^4 \mapsto \mathbb{R}$ in PyTorch to approximate the MPC policy. The residual NN dynamics $f(x, u)$ in (3.16a) is a feedforward tanh network of structure $5 - 50 - 4$, and the controller $\pi(x)$ in (3.16b) is a feedforward ReLU network of structure $4 - 100 - 1$. The disturbance w_t is assumed bounded by $\|w_t\|_\infty \leq 0.05$. We compare the one-shot and recursive frameworks on system (3.16) using backward linear bounds propagation [4] since it can handle general activation functions. The results are shown in Fig. 3.6 where the initial set is chosen as $\mathcal{X}_0 = \{x \mid x_1 \in$

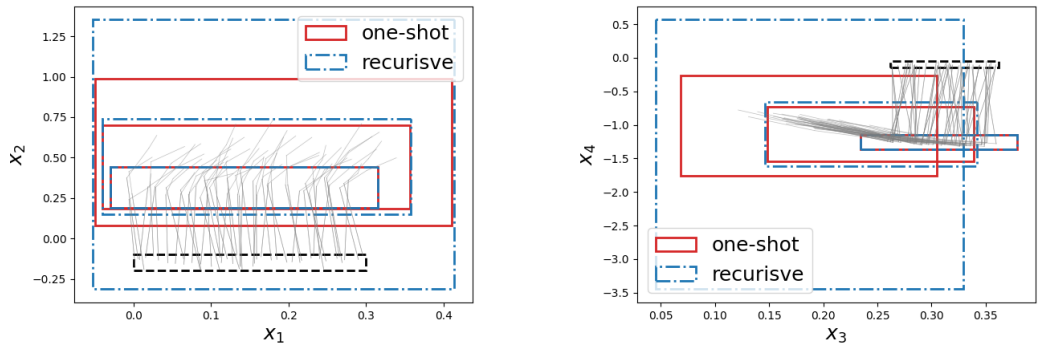


Figure 3.6: Box reachable set over-approximations of the closed-loop NNDS (3.16) obtained by backward linear bounds propagation are plotted for 3 steps. Bounds produced by the one-shot framework (red boxes) are tighter than those from the recursive framework (blue boxes). Randomly sampled trajectories are plotted in gray.

$[0.0, 0.3]$ m, $x_2 \in [-0.2, -0.1]$ m/s, $x_3 \in [0.262, 0.312]$ rad, $x_4 \in [-0.15, -0.05]$ rad/s} and the horizon is set as $T = 3$ [15]. We observe that the one-shot framework generates tighter bounds than the recursive framework, validating Theorem 1. The total solver time is 2.22s for the one-shot framework and 1.01s for the recursive framework.

3.5. Chapter Summary

We proved that for a wide range of NN verification methods, analyzing the unrolled dynamics of a NN dynamical system generates tighter bounds than a recursive, step-by-step analysis. Our proof holds for NN dynamical systems with general computational graphs. The applicability of one-shot reachability analysis was demonstrated numerically. Future work includes designing adaptive bounding polytopes such that the performance gap between the recursive and one-shot frameworks is minimized.

[15] Backward linear bounds propagation solves a weaker relaxation of the NN verification problem than LP-based methods. Therefore, in this example, we only consider $T = 3$ since both the one-shot and recursive frameworks become too conservative afterward.

CHAPTER 4

Stability Analysis of Neural Network Dynamical Systems

4.1. Background

Hybrid systems have become widespread within the systems and control community in the last decades thanks to their flexibility in modeling the interaction of continuous and discrete dynamical systems that frequently arise in cyber-physical systems (CPS) [57, 97]. As today's cyber-physical systems are getting more complex, developing new methods for design, analysis, and control of hybrid systems is increasingly important.

Analysis and control design for general hybrid systems is challenging and therefore, various methods have been proposed to tackle special classes of hybrid systems such as linear complementarity (LC) systems [98, 58], mixed logical dynamical (MLD) systems [57], and piecewise affine (PWA) systems [52]. While these classes are mathematically equivalent [99], their representation could have a high impact on their numerical tractability. When it comes to stability analysis, various methods have been proposed for PWA systems [53, 54, 55] while methods that directly deal with MLD and LC systems are relatively scarce. Nevertheless, stability analysis tools for PWA systems are applicable to MLD and LC systems as they can be transformed into PWA systems [99].

Transforming various types of hybrid systems into a PWA representation for stability analysis may not always be efficient. For example, for a PWA system in feedback with a ReLU neural network controller, although the closed-loop dynamics is PWA, identifying the PWA representation may be tedious and the stability analysis task may become very challenging since the number of partitions generated by the ReLU network can be very large [56].

In this paper, we propose a learning-based approach to stability analysis of hybrid systems that admit a mixed-integer formulation. These systems include PWA, MLD, LC systems and ReLU networks. Our method comprises a learner and a verifier, which iteratively search

for a Lyapunov function from a target class $\mathcal{F}$ of Lyapunov functions (e.g., quadratic or piecewise quadratic). In each iteration, the learner uses a set of samples of the hybrid system to localize $\mathcal{F}$ by a convex set $\tilde{\mathcal{F}} \supseteq \mathcal{F}$ and then solves a semidefinite program (SDP) to select a Lyapunov function candidate from $\tilde{\mathcal{F}}$. The verifier then solves a mixed-integer program in the state space to either validate the Lyapunov function candidate or reject it with a counterexample, i.e., a state where the Lyapunov condition fails. This counterexample is then added to the sample set of the learner to refine the set $\tilde{\mathcal{F}}$ of Lyapunov function candidates. By designing the alternation between the learner and the verifier according to the analytic center cutting-plane method (ACCPM), we show that when the set $\mathcal{F}$ of Lyapunov functions is full-dimensional and contains a norm ball with radius $\epsilon > 0$ in the parameter space, our method is guaranteed to find a Lyapunov function in $\mathcal{O}(n^3/\epsilon^2)$ steps, where n is the ambient dimension.

4.1.1. Related work

LMI-based stability analysis of PWA systems: Among various stability analysis methods for PWA dynamical systems [54, 55], linear matrix inequality (LMI)-based approaches are relatively prominent. These methods construct an SDP whose solution gives a valid Lyapunov function. For continuous-time PWA systems, LMI-based approaches to synthesize piecewise affine [100], piecewise quadratic (PWQ) [101] and piecewise polynomial Lyapunov functions [102] have been proposed. The adaptation of these Lyapunov function synthesis methods to handle discrete-time PWA systems is summarized in [53]. For discrete-time PWA systems, a common feature of the LMI-based methods is computing the transition map between all pairs of modes. This step may become time-consuming when the number of modes is large.

Sampling-based Synthesis Methods: The iterative approach of alternating between a learning module and a verification module to synthesize a certificate for control systems is known as the Counter-Example Guided Inductive Synthesis (CEGIS) framework proposed by [103, 104] in the verification community. The application of CEGIS to Lyapunov function

synthesis for continuous-time nonlinear autonomous systems can be found in [105, 106, 107] using Satisfiability Modulo Theory (SMT) solvers for verification. In general, the termination of the iterative procedures in these works is not guaranteed. Notably, Ravanbakhsh et al. [108] apply the CEGIS framework to synthesize control Lyapunov functions for nonlinear continuous-time systems and provide finite-step termination guarantees for the iterative algorithm through careful design of the learner which essentially implements the maximum volume ellipsoid cutting-plane method [109]. Our work differs from [108] in the algorithm design and the application on hybrid systems. Other than the CEGIS framework, a learning-based approach to synthesize control barrier functions for hybrid systems is proposed in [110].

4.1.2. Notations

We denote the set of real numbers by $\mathbb{R}$, the set of integers by $\mathbb{Z}$, the n-dimensional real vector space by $\mathbb{R}^n$, and the set of $n \times m$-dimensional real matrices by $\mathbb{R}^{n\times m}$. The standard inner product between two matrices $A, B \in \mathbb{R}^{n\times m}$ is given by $\langle A, B\rangle = \text{tr}(A^\top B)$ and the Frobenius norm of a matrix $A \in \mathbb{R}^{n\times m}$ is given by $\|A\|_F = (\text{tr}(A^\top A))^{1/2}$. Denote $\mathbb{S}^n$ the set of $n\times n$-dimensional symmetric matrices, and $\mathbb{S}^n_+$ ($\mathbb{S}^n_{++}$) the set of $n\times n$-dimensional positive semidefinite (definite) matrices. Given a set $\mathcal{S} \subseteq \mathbb{R}^{n_x+n_y}, \text{Proj}_x(\mathcal{S}) = \{x \in \mathbb{R}^{n_x} | \exists y \in \mathbb{R}^{n_y} \text{ s.t. } (x, y) \in \mathcal{S}\}$ denotes the orthogonal projection of $\mathcal{S}$ onto the subspace $\mathbb{R}^{n_x}$. We denote $\text{int}(\mathcal{S})$ the set of all interior points in $\mathcal{S}$.

4.2. Mixed-integer Formulation of Hybrid Systems

Consider a discrete-time autonomous hybrid system

$$x_+ = f(x) \tag{4.1}$$

where $x \in \mathbb{R}^{n_x}$ is the state and $f : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_x}$ is a continuous function. Without loss of generality, assume system (4.1) has an equilibrium at the origin, i.e., $0 = f(0)$. Let $\mathcal{R} \subset \mathbb{R}^{n_x}$ be the domain of the system and $\mathcal{R}$ is a compact set which contains the origin in its interior. Denote x_k the state of system (4.1) at time k and x_0 the initial state. The

nonlinear dynamics $x_+ = f(x)$ with domain $\mathcal{R}$ can be equivalently described through its graph defined as

$$\text{gr}(f) = \{(x, y) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} | x \in \mathcal{R}, y = f(x)\}. \tag{4.2}$$

In this paper, we study the Lyapunov stability of the origin of (4.1) for a class of hybrid systems that admit a mixed-integer formulation.

Definition 4 (Mixed-integer formulation of a set [111])**.** *For a set $\mathcal{Q} \subset \mathbb{R}^{n_z}$, consider the set $\mathcal{L}_\mathcal{Q} \subseteq \mathbb{R}^{n_z} \times \mathbb{R}^{n_\lambda} \times \mathbb{Z}^{n_\mu}$ in a lifted space given by*

$$\mathcal{L}_\mathcal{Q} = \{(z \in \mathbb{R}^{n_z}, \lambda \in \mathbb{R}^{n_\lambda}, \mu \in \mathbb{Z}^{n_\mu}) | \ell(z, \lambda, \mu) \leq v\}, \tag{4.3}$$

with a function $\ell : \mathbb{R}^{n_z} \times \mathbb{R}^{n_\lambda} \times \mathbb{Z}^{n_\mu} \to \mathbb{R}^{n_\ell}$ and a vector $v \in \mathbb{R}^{n_\ell}$. The set $\mathcal{L}_\mathcal{Q}$ is a mixed-integer formulation of $\mathcal{Q}$ if $Proj_z(\mathcal{L}_\mathcal{Q}) = \mathcal{Q}$. If the function ℓ is linear, we call the related formulation mixed-integer linear (MIL).

In the next subsections, we show how to find mixed-integer formulations for PWA, MLD, LC systems and ReLU networks.

4.2.1. Piecewise affine systems

Consider a discrete-time piecewise affine system with control inputs

$$x_+ = \psi_i(x, u) = A_i x + B_i u + c_i, \forall (x, u) \in \mathcal{R}_i, \tag{4.4}$$

where $\mathcal{R}_i = \{(x, u) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} | F_i x + G_i u \leq h_i\}$ for $i \in \mathcal{I} = \{1, 2, \cdots, N_{mode}\}$ are polyhedral partitions of the state-input space $\mathcal{R} = \bigcup_{i \in \mathcal{I}} \mathcal{R}_i$. We assume that the partitions $\mathcal{R}_i$ are bounded for all $i \in \mathcal{I}$. To make the PWA system (4.4) well-posed, we assume that $\text{int}(\mathcal{R}_i) \cap \text{int}(\mathcal{R}_j) = \emptyset, \forall i \neq j$ and $f_i(x, u) = f_j(x, u), \forall (x, u) \in \mathcal{R}_i \cap \mathcal{R}_j$ if the intersection is not an empty set.

We denote the PWA dynamics (4.4) collectively as $x_+ = \psi(x, u)$. The graph of $\psi(x, u)$ is

given by $\text{gr}(\psi) = \bigcup_{i\in\mathcal{I}} \text{gr}(\psi_i)$, where each graph $\text{gr}(\psi_i)$ is defined as

$$\text{gr}(\psi_i) = \{(x, u, x_+)|Q_i[x^\top\ u^\top\ x_+^\top]^\top \leq q_i\}, \tag{4.5}$$

with

$$Q_i = \begin{bmatrix} A_i & B_i & -I \\ -A_i & -B_i & I \\ F_i & G_i & 0 \end{bmatrix}, \quad q_i = \begin{bmatrix} -c_i \\ c_i \\ h_i \end{bmatrix}. \tag{4.6}$$

In this paper, we apply a disjunctive programming-based formulation [111] which states that $x_+ = \psi(x, u)$ is equivalent to the following set of constraints [111]

$$\begin{aligned} &F_i x_i + G_i u_i \leq \mu_i h_i,\ \mu_i \in \{0, 1\},\ \forall i \in \mathcal{I}, \\ &1 = \sum_{i\in\mathcal{I}} \mu_i,\ x = \sum_{i\in\mathcal{I}} x_i,\ u = \sum_{i\in\mathcal{I}} u_i \\ &x_+ = \sum_{i\in\mathcal{I}} (A_i x_i + B_i u_i + \mu_i c_i). \end{aligned} \tag{4.7}$$

In the disjunctive programming formulation (4.7), we have N_{mode} binary variables μ_i and $2N_{mode}$ auxiliary continuous variables $\{x_i, u_i\}$. The binary variable μ_i can be interpreted as the indicator of the mode where the state-input pair (x, u) lives in. Note that $\mu_i = 1$ imposes $\mu_j = 0, \forall j \neq i$. By the boundedness of the partitions $\mathcal{R}_j$, we have $F_j x_j + G_j u_j \leq \mu_j h_j = 0 \Rightarrow x_j = 0, u_j = 0$, and correspondingly $x = \sum_{k\in\mathcal{I}} x_k = x_i, u = \sum_{k\in\mathcal{I}} u_k = u_i, x_+ = \sum_{k\in\mathcal{I}} (A_k x_k + B_k u_k + \mu_k c_k) = A_i x_i + B_i u_i + c_i$. We can also obtain a mixed-integer formulation of the PWA dynamics (4.7) through the big-M method [112].

When the PWA system (4.4) is interconnected with a controller which also has a mixed-integer formulation, we can describe the closed-loop dynamics through mixed-integer constraints. When an autonomous PWA system is considered, we obtain its mixed-integer formulation by removing the control input related variables in (4.7).

4.2.2. Linear complementarity systems

Consider a discrete-time linear complementarity system [99, 98]

$$x_+ = Ax + B_1u + B_2w \tag{4.8a}$$

$$v = E_1x + E_2u + E_3w + g_4 \tag{4.8b}$$

$$0 \leq v \perp w \geq 0 \tag{4.8c}$$

where $v, w \in \mathbb{R}^s$ and $\perp$ denotes that $v^\top w = 0$. The complementarity constraint (4.8c) can be equivalently formulated as a set of mixed-integer linear constraints through the big-M method [112] as

$$v_i w_i = 0 \Leftrightarrow 0 \leq v_i \leq \mu_i M_{1,i},\ 0 \leq w_i \leq (1-\mu_i)M_{2,i}, \tag{4.9}$$

where $\mu_i \in \{0, 1\}$ and the subscript i denotes the i-th entry of the variable v and w. The binary variable μ_i either forces v_i to be zero ($\mu_i = 0$) or forces w_i to be zero ($\mu_i = 1$). In general, selecting the big-M values $M_{1,i}$ and $M_{2,i}$ is no simple task [113]. In practice, when the big-M values are hard to obtain, we can alternatively use the special-ordered set constraint in Gurobi [96] which forces constraint (4.8c) through a branching rule instead of specifying the big-M's explicitly.

One interesting application of the formulations in (4.8) and (4.9) is in the description of the closed-loop MPC systems. We refer the readers to [114] for the details of this formulation.

4.2.3. Mixed-logical dynamical systems

The mixed-logical dynamical system [57] can be written as

$$x_+ = Ax + B_1u + B_2\delta + B_3z \tag{4.10}$$

$$E_1x + E_2u + E_3\delta + E_4z \leq g_5 \tag{4.11}$$

where $x = [x_r^\top \; x_b^\top]^\top$ with $x_r \in \mathbb{R}^{n_r}$ and $x_b \in \{0,1\}^{n_b}$ (u has a similar structure), $z \in \mathbb{R}^{n_z}$ and $\delta \in \{0,1\}^{n_\delta}$ are auxiliary variables. The MLD system is explicitly constructed through mixed-integer linear constraints.

4.2.4. ReLU neural networks

Consider a PWA system in feedback with an L-layer ReLU neural network controller $u = \pi(x)$, where $\pi : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$ is given by

$$\begin{aligned} z_0 &= x \\ z_{\ell+1} &= \max(W_\ell z_\ell + b_\ell, 0), \quad \ell = 0, \cdots, L-1 \\ \pi(x) &= W_L z_L + b_L. \end{aligned} \tag{4.12}$$

Here $z_0 = x \in \mathbb{R}^{n_0}$ ($n_0 = n_x$) is the input to the neural network, $z_{\ell+1} \in \mathbb{R}^{n_{\ell+1}}$ is the vector representing the output of the $(\ell+1)$-th hidden layer with $n_{\ell+1}$ neurons, $\pi(x) \in \mathbb{R}^{n_{L+1}}$ ($n_{L+1} = n_u$) is the output of the neural network, and $W_\ell \in \mathbb{R}^{n_{\ell+1} \times n_\ell}, b_\ell \in \mathbb{R}^{n_{\ell+1}}$ are the weight matrix and the bias vector of the $(\ell+1)$-th hidden layer, respectively.

Consider a scalar ReLU function $y = \max(0, x)$ where $\underline{x} \le x \le \bar{x}$. Then it can be shown that the ReLU function admits the following mixed-integer representation [27],

$$\begin{aligned} &y = \max(0, x), \; \underline{x} \le x \le \bar{x} \iff \\ &y \ge 0, \; y \ge x, y \le x - \underline{x}(1-t), \; y \le \bar{x}t, \; t \in \{0,1\}, \end{aligned} \tag{4.13}$$

where the binary variable $t \in \{0,1\}$ is an indicator of the activation function being active ($y = x$) or inactive ($y = 0$). For a ReLU network described by the equations in (4.12), let $\underline{m}^\ell$ and $\bar{m}^\ell$ be the element-wise lower and upper bounds on the input to the $(\ell+1)$-th activation layer, i.e., $\underline{m}_\ell \le W_\ell z_\ell + b_\ell \le \bar{m}_\ell$. Then the neural network equations are equivalent to a set

of mixed-integer constraints:

$$
\begin{aligned}
&z_{\ell+1} = \max(W_\ell z_\ell + b_\ell, 0) \iff \\
&\begin{cases}
z_{\ell+1} \geq W_\ell z_\ell + b_\ell \\
z_{\ell+1} \leq W_\ell z_\ell + b_\ell - \mathrm{diag}(\underline{m}_\ell)(\mathbf{1} - t_\ell) \\
z_{\ell+1} \geq 0 \\
z_{\ell+1} \leq \mathrm{diag}(\bar{m}_\ell) t_\ell,
\end{cases}
\end{aligned} \tag{4.14}
$$

where $t_\ell \in \{0,1\}^{n_{\ell+1}}$ is a vector of binary variables for the $(\ell+1)$-th activation layer. We note that the element-wise pre-activation bounds $\{\underline{m}_\ell, \bar{m}_\ell\}$ can be found by, for example, interval bound propagation or linear programming assuming known bounds on the input of the neural network [33, 115, 29]. Combined with the MIL formulation of PWA systems shown in Section 4.2.1, the closed-loop dynamics of a ReLU neural network controlled PWA system admits a mixed-integer formulation.

4.3. Stability Analysis via Lyapunov Functions

The convergence behavior of system (4.1) around its equilibrium points can be studied by Lyapunov stability.

Definition 5. (Lyapunov stability [116, Chapter 13]) The equilibrium point $x = 0$ of the autonomous system (4.1) is

- **Lyapunov stable** if for each $\epsilon > 0$, there exists $\delta = \delta(\epsilon)$ such that if $\|x_0\| < \delta$, then $\|x_k\| < \epsilon, \forall k \geq 0$.
- **asymptotically stable** if it is Lyapunov stable and there exists $\delta > 0$ such that if $\|x_0\| < \delta$, then $\lim_{k\to\infty} \|x_k\| = 0$.

Since the hybrid dynamics (4.1) is nonlinear, Lyapunov stability is often a local property and it is of interest to estimate its region of attraction defined as

Definition 6 (Region of attraction)**.** *The region of attraction $\mathcal{O}$ of the nonlinear system* (4.1) *is the set of states from which the trajectory of system* (4.1) *converges to the origin, i.e.,* $\mathcal{O} = \{x_0 \in \mathcal{R} | \lim_{t\to\infty} x_t = 0\}$.

In this work, we do not assume the domain $\mathcal{R}$ to be positive invariant. Instead, we introduce the region of interest (ROI) $\mathcal{X} \subseteq \mathcal{R}$ which is a polytopic set given by

$$\mathcal{X} := \{x \in \mathbb{R}^{n_x} | F_{\mathcal{X}} x \leq h_{\mathcal{X}}\}, \tag{4.15}$$

to guide our search for an inner approximation $\tilde{\mathcal{O}}$ of the ROA $\mathcal{O}$. We can certify the asymptotic stability of system (4.1) and find an $\tilde{\mathcal{O}}$ by constructing Lyapunov functions defined in the following theorem:

Theorem 2. [116, Chapter 13] Consider the discrete-time nonlinear hybrid system (4.1). If there is a continuous function $V(x) : \mathcal{R} \mapsto \mathbb{R}$ with domain $\mathcal{R}$ such that

$$V(0) = 0 \text{ and } V(x) > 0, \forall x \in \mathcal{X} \setminus \{0\} \tag{4.16a}$$

$$V(f(x)) - V(x) \leq 0, \forall x \in \mathcal{X}, \tag{4.16b}$$

where the set $\mathcal{X}$ is the region of interest (ROI) satisfying $\mathcal{X} \subseteq \mathcal{R}$ and $0 \in \text{int}(\mathcal{X})$, then the origin is Lyapunov stable. If, in addition,

$$V(f(x)) - V(x) < 0, \forall x \in \mathcal{X} \setminus \{0\}, \tag{4.17}$$

then the origin is asymptotically stable.

We call any $V(\cdot)$ satisfying (4.16a) a Lyapunov function *candidate*. If additionally, $V(\cdot)$ satisfies the condition (4.16b) or (4.17), then $V(\cdot)$ is called a *valid* Lyapunov function candidate, or simply, a Lyapunov function. Since asymptotic stability is our primary focus in this paper, Lyapunov functions refer to any $V(\cdot)$ satisfying constraints (4.16a) and (4.17) unless specified otherwise. Once a Lyapunov function $V(x)$ is obtained, an inner estimate

of the ROA is given by $\tilde{\mathcal{O}} = \{x|V(x) \leq \tau\}$, where $\tau = \inf_{x\in\mathcal{R}\setminus\mathcal{X}} V(x)$. In other words, $\tilde{\mathcal{O}} \subset \mathcal{X}$ is the largest sublevel set of $V(x)$ that is contained in $\mathcal{X}$.

Remark 2. *The ROI $\mathcal{X}$ can be set according to prior knowledge or from simulation of the nonlinear hybrid dynamics* (4.1)*. See Section 3.4 for examples of choosing the ROI.*

4.3.1. Lyapunov function parameterization

Searching for a Lyapunov function in the function space is intractable since the problem is infinite-dimensional in this space. Instead, we reduce our search space to the class of Lyapunov functions defined by

$$V^{(k)}(x; P) = {z^{(k)}}^\top P z^{(k)}, \tag{4.18}$$

where $P \in \mathbb{S}_{++}^{(k+1)n_x}$, and $z^{(k)}$ is given by

$$z^{(k)} = [x^\top \quad f(x)^{(1),\top} \quad f^{(2),\top}(x) \quad \cdots \quad f^{(k),\top}(x)]^\top \tag{4.19}$$

Here we use the notation $f^{(0)}(x) = x, f^{(k+1)} = f(f^{(k)}(x)), \forall k \geq 0$. We call $V^{(k)}(x; P)$ the Lyapunov function candidate of order k, which is a quadratic function in composition with the system dynamics $f(x)$ evolved for k steps. Indeed, we can increase the complexity of the function class monotonically by increasing the order k.

Since P is positive definite, searching for a valid Lyapunov function of the form (4.18) reduces to find a matrix P that satisfies the Lyapunov difference condition (4.17). Explicitly, we can characterize the space of matrices P that admit a valid Lyapunov function candidate as

$$\mathcal{F} = \{P \in \mathbb{S}^{(k+1)n_x} | \alpha I \preceq P \preceq \beta I, \Delta V^{(k)}(x, P) < 0, \forall x \in \mathcal{X} \setminus \{0\}\}. \tag{4.20}$$

where $0 \leq \alpha < \beta$, and $\Delta V^{(k)}(x, P)$ is the Lyapunov difference given by

$$\Delta V^{(k)}(x, P) := V^{(k)}(f(x); P) - V^{(k)}(x; P). \tag{4.21}$$

The constraint $\alpha I \preceq P$ guarantees the condition (4.16a), while the constraint $P \preceq \beta I$ is imposed to make $\mathcal{F}$ bounded without loss of generality since we can always scale P while satisfying (4.17).

We call $\mathcal{F}$ the target set. It follows that $\mathcal{F}$ is convex since it is defined by semidefinite as well as linear constraints on P. As $\alpha I \preceq P \preceq \beta I$ imposes an additional constraint on the condition number of P, in practice we choose β/α large or simply set $\alpha = 0$ [16]. Then finding a Lyapunov function in a given function class $V^{(k)}(x; P)$ is stated as the following problem:

Problem 1. *For each parameterized function class $V^{(k)}(x; P)$ and the target set $\mathcal{F}$ defined in* (4.20)*, find a feasible point in $\mathcal{F}$ or certify that $\mathcal{F}$ is empty.*

Although the target set $\mathcal{F}$ is convex, the fact that it is characterized by infinitely many linear constraints (i.e., the constraints $\Delta V^{(k)}(x, P) < 0, \forall x \in \mathcal{X} \setminus \{0\}$) poses computational challenges to solving Problem 1. In this work, we propose a learning-based approach to address this challenge by iteratively drawing state samples to refine our over-approximation of the target set $\mathcal{F}$. By designing the learning strategy based on ACCPM from convex optimization, we show that when the target set $\mathcal{F}$ is full-dimensional in the parameter space of P, our method is guaranteed to find a feasible point in $\mathcal{F}$ in a finite number of steps.

Remark 3. *The parameterization of $V^{(k)}(x; P)$ is inspired by the finite-step Lyapunov function [117, 118] and the non-monotonic Lyapunov functions [119] which also construct Lyapunov function candidates using the system states several steps ahead. $V^{(k)}(x; P)$ allows us to parameterize complex function classes with a relatively small number of parameters. For example, when $f(x)$ is a PWA function with N_{mode} (possibly large) partitions, $V^{(1)}(x; P)$ is a PWQ function with the same partitions in the state space.*

[16] As will be shown next, the proposed method always finds a feasible solution in the interior of $\mathcal{F}$. Therefore, choosing $\alpha = 0$ does not affect the positive definiteness of the solution.

4.4. Learning Lyapunov Functions from Counterexamples

We recall from the previous section that finding a feasible point of the convex set $\mathcal{F}$ is computationally intractable since the condition $\Delta V^{(k)}(x;P) < 0$ must hold for all $x \in \mathcal{X} \setminus \{0\}$. To overcome this intractability, we adopt a learning-based approach, in which we first select a set of finite samples $\mathcal{S} = \{x^1, x^2, \cdots, x^N\} \subset \mathcal{X}$ and then enforce the linear constraint $\Delta V^{(k)}(x;P) < 0$ to hold only for $x \in \mathcal{S}$. This results in an over-approximation of $\mathcal{F}$ given by

$$\tilde{\mathcal{F}} = \{P \in \mathbb{S}^{(k+1)n_x} | \alpha I \preceq P \preceq \beta I, \Delta V^{(k)}(x,P) \leq 0, \forall x \in \mathcal{S}\}. \tag{4.22}$$

We call $\tilde{\mathcal{F}}$ the localization set. Finding a feasible point in $\tilde{\mathcal{F}}$ now becomes a tractable convex feasibility problem. However, there is no guarantee that a $P \in \tilde{\mathcal{F}}$ would correspond to a valid Lyapunov function, even if the number of samples approaches infinity. As one of our contributions, we propose an efficient learning strategy based on the analytic center cutting-plane method (ACCPM) to iteratively grow the sample set and refine the set $\tilde{\mathcal{F}}$ until we find a feasible point in $\mathcal{F}$. In the next subsections, we describe the proposed approach and provide finite-step termination guarantees when $\mathcal{F}$ is non-empty and satisfies the following assumption:

Assumption 1. *The target set $\mathcal{F}$ defined in* (4.20) *is full-dimensional and there exists $P_{center} \in \mathbb{S}^{(k+1)n_x}$ such that $\{P \in \mathbb{S}^{(k+1)n_x} | \|P - P_{center}\|_F \leq \epsilon\} \subset \mathcal{F}$ where $\|\cdot\|_F$ is the Frobenius norm.*

4.4.1. Analytic center cutting-plane method

Cutting-plane methods [120, 121, 122] are iterative algorithms to find a point in a target convex set $\mathcal{F}$ or to determine whether $\mathcal{F}$ is empty. In these methods, we have no information on $\mathcal{F}$ except for a "cutting-plane oracle", which can verify whether $P \in \mathcal{F}$ for a given P. Let $\tilde{\mathcal{F}}$ be a localization set defined by a finite set of inequalities that over approximates the target set, i.e., $\mathcal{F} \subseteq \tilde{\mathcal{F}}$. If $\tilde{\mathcal{F}}$ is empty, then we have proof that the target set $\mathcal{F}$ is also empty.

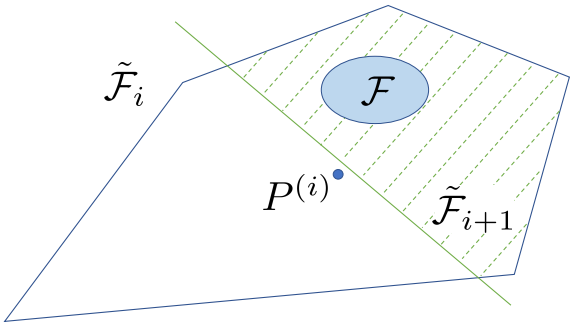


Figure 4.1: In the i-th iteration, the query point $P^{(i)}$ is chosen as the "center" of the localization set $\tilde{\mathcal{F}}_i$ to guarantee the removal of a portion of $\tilde{\mathcal{F}}_i$ from the search space whenever a separating hyperplane (green solid line) is given. The localization set is then updated to $\tilde{\mathcal{F}}_{i+1}$ (shaded set) which is a finer over-approximation of the target set $\mathcal{F}$.

Otherwise, we query the oracle at a point $P \in \tilde{\mathcal{F}}$. If $P \in \mathcal{F}$, the oracle returns 'yes' and the algorithm terminates; if $P \notin \mathcal{F}$, it returns 'no' together with a separating hyperplane that separates P and $\mathcal{F}$. In the latter case, the cutting-plane method updates the localization set by $\tilde{\mathcal{F}} \leftarrow \tilde{\mathcal{F}} \cap \{\text{half space defined by the separating hyperplane}\}$ as shown in Fig. 4.1. This process continues until either a point in the target set is found or the target set is certified to be empty.

Based on how the query point is chosen, different cutting-plane methods have been proposed including the center of gravity method [123], the maximum volume ellipsoid (MVE) cutting-plane method [109], the Chebyshev center cutting-plane method [121], the ellipsoid method [124, 125], and the analytic center cutting-plane method [126, 127, 120]. In this paper, we use the analytic center cutting-plane method since it allows the localization set $\tilde{\mathcal{F}}$ to be described by linear matrix inequalities. In the ACCPM, the query point P is chosen as the analytic center of the localization set $\tilde{\mathcal{F}}$.

Definition 7 (Analytic center [128])**.** *The analytic center x_{ac} of a set of convex inequalities and linear equalities $h_i(x) \leq 0, i = 1,\cdots,m,\ Fx = g$, is defined as the solution of the convex*

problem

$$\underset{x}{\text{minimize}} \quad -\sum_{i=1}^{m} \log(-h_i(x)) \quad \textit{subject to} \quad Fx = g. \tag{4.23}$$

We design the learning strategy according to the ACCPM by constructing a learner, which proposes Lyapunov function candidates based on a set of samples, and a verifier, which serves as a cutting-plane oracle and updates the sample set with counterexamples.

4.4.2. The learner

Let $\mathcal{S} = \{x^1, x^2, \cdots, x^N\} \subset \mathcal{X}$ be a collection of samples from $\mathcal{X}$. The localization set $\tilde{\mathcal{F}}$ in the space of P is given by (4.22), which represents the learner's knowledge about $\mathcal{F}$ by observing the k-step trajectories of the dynamical system (4.1) starting from $\mathcal{S}$. According to the ACCPM, the learner proposes a Lyapunov function candidate $V^{(k)}(x; P_{ac})$ with P_{ac} as the analytic center of $\tilde{\mathcal{F}}$:

$$\begin{aligned} P_{ac} := \underset{P}{\text{argmin}} \quad & -\sum_{x \in \mathcal{S}} \log(-\Delta V^{(k)}(x, P)) \\ & - \log\det(\beta I - P) - \log\det(P - \alpha I). \end{aligned} \tag{4.24}$$

This is a convex program that can be solved efficiently through off-the-shell convex optimization solvers. We denote the objective function in (4.24) as $\phi(\tilde{\mathcal{F}})$ and call it the potential function on the set $\tilde{\mathcal{F}}$. If (4.24) is infeasible, then we have a proof that no Lyapunov function exists in the function class of order k. Otherwise, the learner proposes $V^{(k)}(x; P_{ac}) = {z^{(k)}}^\top P_{ac} z^{(k)}$ as a Lyapunov function candidate. Due to the log-barrier function in (4.24), P_{ac} is in the interior of $\tilde{\mathcal{F}}$. When the sample set $\mathcal{S}$ is empty, the potential function simply becomes $\phi(\tilde{\mathcal{F}}) = -\log\det(\beta I - P) - \log\det(P - \alpha I)$ with $P_{ac} = \frac{\alpha+\beta}{2} I$ as the optimal solution.

4.4.3. The verifier

Suppose the learner proposes the Lyapunov function candidate $V^{(k)}(x; P)$ by solving (4.24). Given $V^{(k)}(x; P)$, the verifier either ensures that this function satisfies the constraints in (4.16a) and (4.17), or returns a state where constraint (4.16a) or (4.17) is violated

as a counterexample. Since the log-barrier function in (4.24) guarantees $P \succ 0$, constraint (4.16a) is readily satisfied and the verifier must check the violation of constraint (4.17). This can be done by solving the optimization problem

$$\underset{x \in \mathcal{X} \setminus \{0\}}{\text{maximize}} \quad \Delta V^{(k)}(x, P). \tag{4.25}$$

When $f(x)$ has a mixed-integer formulation

$$\mathcal{L}_{\text{gr}(f)} = \{(x, x_+, \lambda, \mu) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_\lambda} \times \mathbb{Z}^{n_\mu} | \ell(x, x_+, \lambda, \mu) \leq v\},$$

for Lyapunov function candidates $V^{(k)}(x; P)$ of order k, we write problem (4.25) explicitly as a mixed-integer quadratic program (MIQP):

$$\underset{\{x_i\},\{\lambda_i\},\{\mu_i\}}{\text{maximize}} \quad \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{k+1} \end{bmatrix}^\top P \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{k+1} \end{bmatrix} - \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_k \end{bmatrix}^\top P \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_k \end{bmatrix} \tag{4.26a}$$

$$\text{subject to} \quad x_0 \in \mathcal{X} \tag{4.26b}$$

$$\|x_0\|_\infty \geq \epsilon \tag{4.26c}$$

$$\ell(x_i, x_{i+1}, \lambda_i, \mu_i) \leq v_i, i = 0, 1, \cdots, k \tag{4.26d}$$

where the objective function (4.26a) is equal to the Lyapunov difference $\Delta V^{(k)}(x_0, P)$, constraint (4.26b) restricts the search of counterexamples inside the ROI $\mathcal{X}$, constraint (4.26c) excludes a small ℓ_∞ norm ball centered at the origin $B_\epsilon = \{x | \|x\|_\infty < \epsilon\}$ from the search space, and constraints (4.26d) enforce the dynamical constraint $(x_i, x_{i+1}) \in \text{gr}(f)$, or equiv-

alently $x_{i+1} = f(x_i)$ for $i = 0, \cdots, k$.

Denote p^* the optimal value and x_0^* the x_0-component of the optimal solution of (4.26). When $p^* \geq 0$, x_0^* is a counterexample for the Lyapunov function candidate $V^{(k)}(x; P)$ since $\Delta V^{(k)}(x_0^*, P) \geq 0$. Then, the separating hyperplane induced by the counterexample x_0^* is given as $\Delta V^{(k)}(x_0^*, P) = 0$ where $\Delta V^{(k)}(x_0^*, P)$ is interpreted as a linear function in the matrix variable P. When $p^* < 0$, we certify the convergence of system trajectories to a neighborhood of the origin as shown in the following corollary.

Corollary 1. *Let $\Omega = \{x | V^{(k)}(x; P) \leq \gamma\}$ be the largest sublevel set of $V^{(k)}(x; P)$ inside $\mathcal{X}$ with $\gamma = \inf_{x \in \mathcal{R} \setminus \mathcal{X}} V^{(k)}(x; P)$. Define the successor set of B_ϵ as $suc(B_\epsilon) := \{y | y = f(x), x \in B_\epsilon\}$. Assume $B_\epsilon \subset \Omega$, $suc(B_\epsilon) \subset \Omega$, and Ω_{loc} is the smallest sublevel set of $V^{(k)}(x; P)$ such that $suc(B_\epsilon) \subseteq \Omega_{loc}$. If $p^* < 0$, then we have $\lim_{t \to \infty} x_t \in \Omega_{loc}$ for all $x_0 \in \Omega$.*

Proof. First, note that all trajectories starting from $x_0 \in \Omega$ and $x_0 \notin B_\epsilon$ reach B_ϵ in a finite number of steps since $V^{(k)}(x_{t+1}; P) - V^{(k)}(x_t; P) \leq p^* < 0$ as long as $x_t \notin B_\epsilon$ and $V^{(k)}(x_t; P)$ is lower-bounded by 0 for all x_t. If x_t never reaches B_ϵ, we will have a contradiction that $V^{(k)}(x_N; P) < 0$ for some N. After the trajectories reach B_ϵ, the subsequent states will remain inside Ω_{loc} by construction. Therefore, we have $\lim_{t \to \infty} x_t \in \Omega_{loc}$ for all $x_0 \in \Omega$. □

The alternation between the learner and the verifier is summarized in Algorithm 3. In the next subsection, we show that when the target set $\mathcal{F}$ satisfies Assumption 1, our proposed algorithm is guaranteed to find a feasible point in $\mathcal{F}$ in a finite number of steps.

Remark 4. *In this paper, the verifier is constructed as a global optimization problem* (4.25)*. Alternative constructions of the verifier do exist, e.g., by using SMT solvers as shown in [105, 106]. The termination results in the next subsection will always hold no matter how the verifier is formulated.*

Algorithm 3 Learning-based Lyapunov function synthesis

procedure LEARNINGLYAPUNOV
  $\mathcal{S}_1 = \emptyset$ ▷ initialize the sample set
  $i = 1$
  while True **do**
    Generate $\tilde{\mathcal{F}}_i$ from sample set $\mathcal{S}_i$ by (4.22).
    if $\tilde{\mathcal{F}}_i = \emptyset$ **then**
      Return: Status = Infeasible, $P_* = \emptyset$.
    $P^{(i)} = \arg\min$ (4.24) with $\mathcal{S}_i$ ▷ find analytic center
    solve (4.26) with $P^{(i)}$ ▷ query the verifier
    if max (4.26) < 0 **then**
      Return: Status = Feasible, $P_* = P^{(i)}$.
    else
      $x_*^{(i)} = \arg\min_{x_0}$ (4.26) ▷ find a counterexample
      $\mathcal{S}_{i+1} = \mathcal{S}_i \cup \{x_*^{(i)}\}$
    $i = i + 1$

4.4.4. Convergence Analysis

The convergence and complexity of the ACCPM have been studied in [120, 127, 129, 130, 131, 132] under various assumptions on the localization set, the form of the separating hyperplane, whether multiple cuts are applied, etc. Directly related to Algorithm 3 and the search for $V^{(k)}(x; P)$ is [132] which analyzes the complexity of the ACCPM with a matrix variable and semidefiniteness constraints. Notably, it provides an upper bound on the number of iterations that Algorithm 3 can run before termination when the target set is non-empty. In [132], it is assumed that

- A1: $\mathcal{F}$ is a convex subset of $\mathbb{S}^n$.
- A2: See Assumption 1.
- A3: $\mathcal{F} \subset \{P \in \mathbb{S}^n | 0 \preceq P \preceq I\}$.

For the Lyapunov function candidate class of order k, we have $n = (k+1)n_x$. Let the ACCPM start with the localization set $\tilde{\mathcal{F}}_1 = \{P | 0 \preceq P \preceq I\}$ and initialize the first query point $P^{(1)} = \frac{1}{2}I$ correspondingly. If at iteration i a query point $P^{(i)}$ is rejected by the oracle, a separating hyperplane of the form $\langle D_i, P - P^{(i)} \rangle = 0$ with $\|D_i\|_F = 1$ is given by

the verifier. By induction, we have that at iteration $i > 1$, the localization set $\tilde{\mathcal{F}}_i$ is

$$\tilde{\mathcal{F}}_i = \{P | 0 \preceq P \preceq I, \langle D_j, P \rangle \leq c_j, j = 1, \cdots, i-1\}.$$

with D_j defining the separating hyperplane and $c_j = \langle D_j, P^{(j)} \rangle$. The query point at iteration i is given by $P^{(i)} = \arg\min \phi(\tilde{\mathcal{F}}_i)$ with the potential function

$$\phi(\tilde{\mathcal{F}}_i) = -\sum_{j=1}^{i-1} \log(c_j - \langle D_j, P \rangle) - \log\det(I - P) - \log\det(P).$$

Theorem 3. *Under Assumption 1 on the target set $\mathcal{F}$, Algorithm 3 finds a feasible point in $\mathcal{F}$ in at most $O(((k+1)n_x)^3/\epsilon^2)$ iterations.*

Proof. The proof follows from [132] which states for a general target set $\mathcal{F} \subset \mathbb{S}^n$ under assumptions A1 to A3, the analytic center cutting-plane method with separating hyperplanes of the form $\langle D_i, P \rangle \leq c_i$ is shown to find a feasible point in at most $O(n^3/\epsilon^2)$ iterations. For the sequence of localization sets $\tilde{\mathcal{F}}_i$, [132] computes an upper bound on the potential function $\phi(\tilde{\mathcal{F}}_i)$ which is approximately $i\log(\frac{1}{\epsilon})$, and a lower bound on $\phi(\tilde{\mathcal{F}}_i)$ which is proportional to $\frac{i}{2}\log(\frac{i}{n^3})$. Since the ACCPM must terminate before the lower bound exceeds the upper bound of $\phi(\tilde{\mathcal{F}}_i)$, we obtain the $O(n^3/\epsilon^2)$ upper bound on the number of iterations.

To show that the result in [132] applies to Algorithm 3, note that for each counterexample $x_*^{(i)}$ found by the verifier in iteration i, the separating hyperplane can be constructed as $\Delta V^{(k)}(x_*^{(i)}, P) \leq \Delta V^{(k)}(x_*^{(i)}, P^{(i)})$ since $\Delta V^{(k)}(x_*^{(i)}, P^{(i)}) \geq 0$. We can rewrite this cutting plane in the form $\langle D_i, P \rangle \leq \langle D_i, P^{(i)} \rangle$ by setting $\hat{D}_i = z_{k,*}^{+,(i)} z_{k,*}^{+,(i),\top} - z_{k,*}^{(i)} z_{k,*}^{(i),\top}, D_i = \hat{D}_i / \|\hat{D}_i\|_F$, where $z_{k,*}^{(i)}$ denotes the basis (4.19) with $x = x_*^{(i)}$, and $z_{k,*}^{+,(i)}$ denotes the basis (4.19) after setting $x = f(x_*^{(i)})$. Then with $\alpha = 0, \beta = 1$ in (4.20), Algorithm 3 satisfies assumptions A1 to A3 and it terminates in at most $O(((k+1)n_x)^3/\epsilon^2)$ iterations according to [132]. □

Theorem 3 provides a finite-step termination guarantee for Algorithm 3 when the target

set satisfies Assumption 1. However, when $\mathcal{F}$ is empty, we do not have such a guarantee. Certification of the non-existence of Lyapunov functions in $V^{(k)}(x;P)$ relies on detecting that an over-approximation $\tilde{\mathcal{F}}$ is empty. Hence, a quick expansion of the sample set $\mathcal{S}$ as shown in Section 4.4.5 is preferred.

4.4.5. Implementation

Complexity of the learner

For a sample set $\mathcal{S}$ with N samples, the localization set $\tilde{\mathcal{F}}$ in (4.22) is described by N linear as well as two semidefinite constraints on the variable $P \in \mathbb{S}^{(k+1)n_x}$. We first decide if $\tilde{\mathcal{F}}$ is empty by solving an SDP feasibility problem. If the SDP is infeasible, then $\tilde{\mathcal{F}} = \emptyset$ and so is $\mathcal{F}$. If $\mathcal{F}$ is non-empty, we move on to the analytic center problem (4.24) which can be solved, e.g., through an infeasible start Newton's method [128].

Complexity of the MIQP

Since MIQP is well-known to be NP-hard, we use the number of binary variables, which we denote by N_μ in (4.26), as a rough measure of the complexity of (4.26). When $f(x)$ is a PWA function with N_{mode} modes, we have N_{mode} binary variables in the MIL formulation of $f(x)$. For the LC system (4.8), N_μ equals the dimension of the orthogonal variables v and w. N_μ is explicitly given in the MLD system and is equal to the number of neurons in the mixed-integer formulation of the ReLU networks. It follows that for the LC systems and ReLU networks, it is possible to use a small number of integer variables to encode a PWA system with many more modes. Since we need to evaluate the hybrid dynamics $f(x)$ for k times when $V^{(k)}(x;P)$ is applied, the number of binary variables N_μ in (4.26) is largely linear in the order k.

The actual solving time of the MIQP has a complex dependence not only on the number of variables and constraints but also on how the constraints are formulated. The exploration of the numerical performance of the proposed algorithm is left for future research.

Solvability of the MIQP

The optimization problem (4.26) is a nonconvex MIQP since the quadratic objective function is indefinite. Therefore, the relaxation of the problem after removing the integrality constraints would result in a nonconvex quadratic program. Nonconvex MIQP can be solved to global optimality through Gurobi v9.0 [96] by transforming the nonconvex quadratic expression into a bilinear form and applying spatial branching [133]. More information on solving nonconvex mixed-integer nonlinear programming can be found in [134, 135, 136]. In this paper, we rely on Gurobi to solve the nonconvex MIQP (4.26) automatically.

Exclusion of the origin

In constraint (4.26c), we add a guard B_ϵ at the origin to approximate the exact constraint $x \in \mathcal{X} \setminus \{0\}$. Since constraint (4.26c) allows a mixed-integer linear formulation through the big-M method, problem (4.26) is an MIQP. Adding B_ϵ bounds p^* off from 0 if $V_k(x; P)$ is in fact a Lyapunov function and we can decide the negativity of p^* by checking if $p^* < -\epsilon_{tol}$ for some tolerance $\epsilon_{tol} > 0$ to handle round-off errors in computation.

When the dynamics $x_+ = f(x)$ is linear inside B_ϵ, i.e., $x_+ = Ax$ for $x \in B_\epsilon$, we can show convergence to the origin of the system trajectories starting inside B_ϵ by checking the magnitude of the eigenvalues of A. Combined with Corollary 1, asymptotic convergence inside the sublevel set Ω can be established.

Early termination of the MIQP

We can terminate the Branch & Bound [137] algorithm in solving (4.26) once a feasible solution is found with non-negative objective function since in this case we already have a counterexample.

4.5. Numerical Examples

We demonstrate our method through two examples: closed-loop MPC systems and ReLU neural network controlled PWA systems. In particular, we compare the performances of our method with the LMI-based Lyapunov function synthesis methods [53] on the MPC

example. Algorithm 3 is implemented in Python 3.7 with Gurobi v9.0 [96] and the LMI-based method is implemented in the MPT3 toolbox [138] with Mosek [139] in Matlab. All the simulation is implemented on an Intel i7-6700K CPU with 32 GB of RAM.

Throughout the numerical experiments in this section, Algorithm 3 is run with $\alpha = 0.0, \beta = 1.0$ and $\epsilon_{tol} = 10^{-8}$ to decide negativity of the optimal value of the MIQP. The Gurobi solver is set with feasibility tolerance 10^{-9}, integer tolerance 10^{-9}, and optimality tolerance 10^{-9}. In addition, we terminate the MIQP once it finds a feasible solution that generates an objective $\geq 10^{-4}$ which means a counterexample is already found.

4.5.1. A 2-dimensional closed-loop MPC system

For an unstable linear system $x_+ = Ax + Bu$ with

$$A = \begin{bmatrix} 1.2 & 1.2 \\ 0 & 1.2 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}, \tag{4.27}$$

we design an MPC controller with horizon $T = 10$, state constraint $[-5 \ -5]^\top \leq x \leq [5 \ 5]^\top$, control input constraint $-1 \leq u \leq 1$, stage cost $p(x, u) = x^\top Q x + u^\top R u$ with $Q = 10I, R = 1$, and terminal cost $q(x) = x^\top P_\infty x$ where $P_\infty = \text{DARE}(A, B, Q, R)$ is the solution to the discrete algebraic Riccati equation defined by (A, B, Q, R). The terminal set is chosen as the maximum positive invariant set [97, Chapter 10] of the closed-loop system $x_+ = (A + BK_\infty)x$ where $K_\infty = -(B^\top P_\infty B + R)^{-1} B^\top P_\infty A$.

With the above setup, we obtain the explicit MPC controller, which is a PWA function of the state x with 211 partitions, through the MPT3 toolbox in Matlab. The domain $\mathcal{R}$ of the explicit MPC is a polytope shown in Fig. 4.2 together with its partitions. After obtaining $\mathcal{R}$, we verify that it is positive invariant for the closed-loop dynamics. Through the LMI-based method, we are able to synthesize a discontinuous PWQ Lyapunov function in MPT3 which verifies $\mathcal{R}$ is the ROA for the closed-loop MPC system. The total running time is 38.890 seconds, with 0.25 spent in solving the constructed SDP and the rest in computing the transition map.

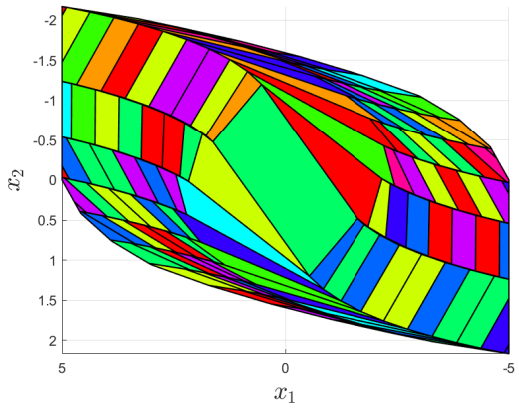


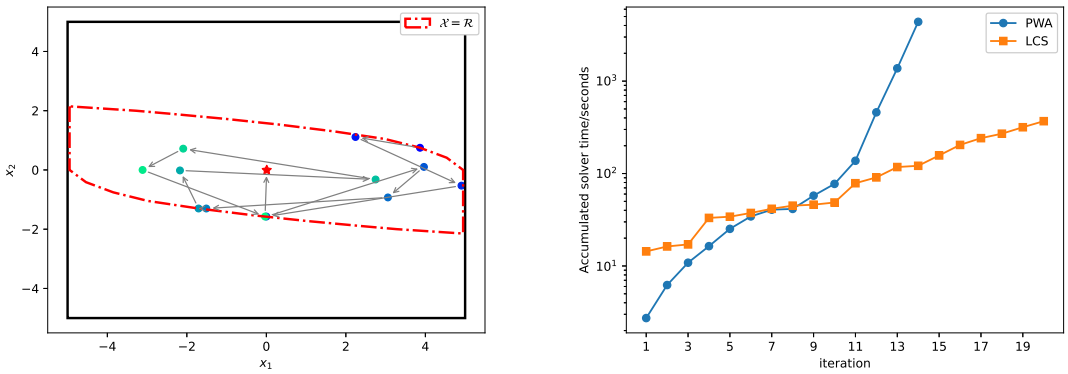
Figure 4.2: Domain $\mathcal{R}$ and the 211 modes of the closed-loop MPC system of (4.27).

Algorithm 3 with PWA representation

We import the PWA representation of the closed-loop MPC dynamics which we denote as $x_+ = f_{cl}(x)$ from MPT3 and run Algorithm 3 with the mixed-integer formulation of $f_{cl}(x)$ as shown in Section 4.2.1. Since the domain $\mathcal{R}$ is positively invariant, we set $\mathcal{X} = \mathcal{R}$. For Lyapunov function candidates of order $k = 0, 1$, Algorithm 3 certifies the non-existence of Lyapunov functions after 1.709 seconds with 4 iterations and 28.333 seconds with 7 iterations, respectively. With $V^{(k)}(x; P)$ of order $k = 2$, Algorithm 3 terminates in 14 iterations with a valid Lyapunov function candidate and the total running time is 4379.716 seconds. We plot the counterexamples found in each iteration in Fig. 4.3a. The accumulated running time of Algorithm 3 in each iteration is shown in Fig. 4.3b.

Algorithm 3 with LCS representation

As shown in Section 4.2.2 and [114], we can obtain a mixed-integer formulation of the closed-loop MPC dynamics through its LC system representation instead of the PWA one. Based on this mixed-integer formulation, we run Algorithm 3 with Lyapunov function candidates $V^{(k)}(x; P)$ of order $k = 2$. The algorithm terminates in 368.618 seconds with 20 iterations and returns a valid Lyapunov function which certifies that the domain $\mathcal{R}$ is the ROA. The accumulated running time at each iteration is summarized in Fig. 4.3b.



(a) Sequence of counterexamples found in Algorithm 3.

(b) Accumulated running time of Algorithm 3.

Figure 4.3: (a) Sequence of counterexample states found by the verifier in Algorithm 3 with the PWA representation of the closed-loop MPC system. (b) Accumulated running time of Algorithm 3 in each iteration with mixed-integer formulations induced by the PWA (blue) and LC (orange) representations of the closed-loop MPC system.

Discussion

Fig. 4.3b shows that Algorithm 3 depends on the mixed-integer representation of the system. In this example, we need 211 binary variables to describe the map $x_+ = f_{cl}(x)$ using the PWA representation, while with the LCS representation, we only need to use 64 binary variables to describe the map $u = \pi_{MPC}(x)$, and hence the closed-loop dynamics. However, when compared with the LMI-based method which synthesizes a PWQ discontinuous Lyapunov functions in 38.890 seconds, Algorithm 3 is rather inefficient. This is partly due to that the continuous Lyapunov function candidate class $V^{(k)}(x; P)$ is more conservative than the discontinuous one [53]. To compensate for the conservatism, we need to apply high order $V^{(k)}(x; P)$ which increases the complexity of the MIQP. In the next subsection, we show that Algorithm 3 can synthesize a Lyapunov function when the LMI-based method fails.

4.5.2. A 4-dimensional closed-loop MPC system

Consider model predictive control of a randomly generated 4-dimensional open-loop unstable linear system given by

$$A = \begin{bmatrix} 0.4346 & -0.2313 & -0.6404 & 0.3405 \\ -0.6731 & 0.1045 & -0.0613 & 0.3400 \\ -0.0568 & 0.7065 & -0.0861 & 0.0159 \\ 0.3511 & 0.1404 & 0.2980 & 1.0416 \end{bmatrix}, \; B = \begin{bmatrix} 0 \\ -0.0065 \\ -0.5238 \\ 0.4605 \end{bmatrix}.$$

With the state constraint $\|x\|_\infty \leq 5$ and the input constraint $-1 \leq u \leq 1$, we design the MPC controller by choosing horizon $T = 10$, stage cost with $Q = 10I, R = 1$, terminal cost with P_∞ and the terminal set as in Section 4.5.1. The explicit MPC controller is constructed through MPT3 and has 193 partitions in its domain $\mathcal{R}$, which is validated to be positive invariant under the closed-loop dynamics. Although it took only 29.9 seconds to compute the transition map, the constructed SDP is ill-posed and Mosek failed to find a feasible solution.

Then we apply Algorithm 3 to synthesize a Lyapunov function $V^{(k)}(x; P)$ with $k = 1$ and the LCS representation of the closed-loop MPC system. The ROI is chosen as $\mathcal{X} = \mathcal{R}$. Algorithm 3 terminates in 9 iterations with a valid Lyapunov function candidate and therefore proves that the closed-loop system is asymptotically stable in the domain $\mathcal{R}$. The total running time is 680.515 seconds and the accumulated running time in each iteration is plotted in Fig. 4.4.

4.5.3. Neural network controlled PWA system

We use a hybrid system example from [140] and consider the inverted pendulum shown in Fig. 4.5 with parameters $m = 1, \ell = 1, g = 10, k = 100, d = 0.1$. We denote the angle and the angular velocity of the pendulum by q and $\dot{q}$, respectively, and define the system state as $x = (q, \dot{q})$. By linearizing the dynamics of the inverted pendulum around $x = 0$, we obtain a hybrid system which has two modes: not in contact with the elastic wall (mode 1) and

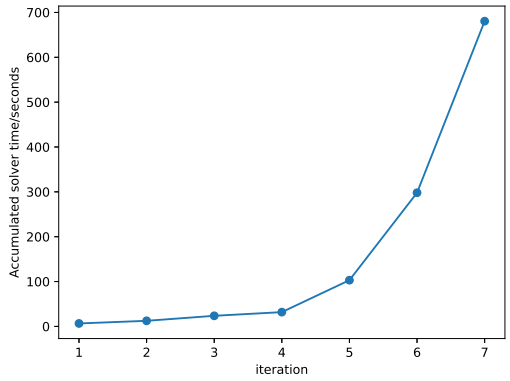


Figure 4.4: Accumulated solver time of Algorithm 3 in each iteration for the 4-dimensional closed-loop MPC system.

in contact with the elastic wall (mode 2). After discretizing the model using the explicit Euler scheme with a sampling time $h = 0.01$, a PWA model $x_+ = \psi(x, u)$ of the form (4.4) is obtained with the following parameters

$$
\begin{aligned}
&A_1 = \begin{bmatrix} 1 & 0.01 \\ 0.1 & 1 \end{bmatrix}, B_1 = \begin{bmatrix} 0 \\ 0.01 \end{bmatrix}, c_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \\
&\mathcal{R}_1 = \{x | [-0.2 \ -1.5]^\top \leq x \leq [0.1 \ 1.5]^\top\}. \\
&A_2 = \begin{bmatrix} 1 & 0.01 \\ -0.9 & 1 \end{bmatrix}, B_2 = \begin{bmatrix} 0 \\ 0.01 \end{bmatrix}, c_2 = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix}, \\
&\mathcal{R}_2 = \{x | [0.1 \ -1.5]^\top \leq x \leq [0.2 \ 1.5]^\top\}.
\end{aligned}
\tag{4.28}
$$

We then synthesize a hybrid MPC controller $\pi_{MPC}(x)$ for the PWA system [111] where the control input constraints are given by $-4 \leq u \leq 4$ and the horizon of MPC is set as $T = 10$. The stage and terminal costs are given by $Q = I, R = 1$, and $P_\infty = \text{DARE}(A_1, B_1, Q, R)$. We evaluate $\pi_{MPC}(x)$ on a uniform 40×40 grid samples from the state space and let the reference ROI $\mathcal{X}_0$ be the convex hull of all the feasible state samples. Then the ROI $\mathcal{X} = \gamma \mathcal{X}_0$ with $0 < \gamma \leq 1$ is applied to guide the search for an estimate of ROA.

A total number of 1354 feasible samples of state and control input pairs $(x, \pi_{MPC}(x))$ are

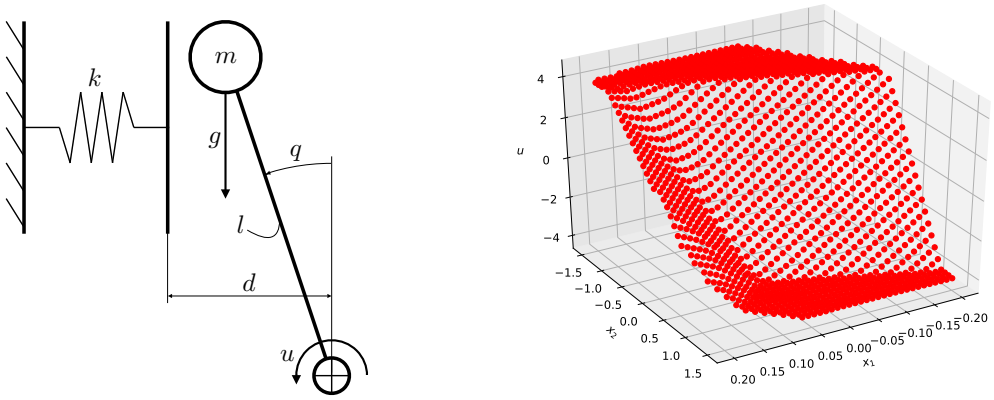
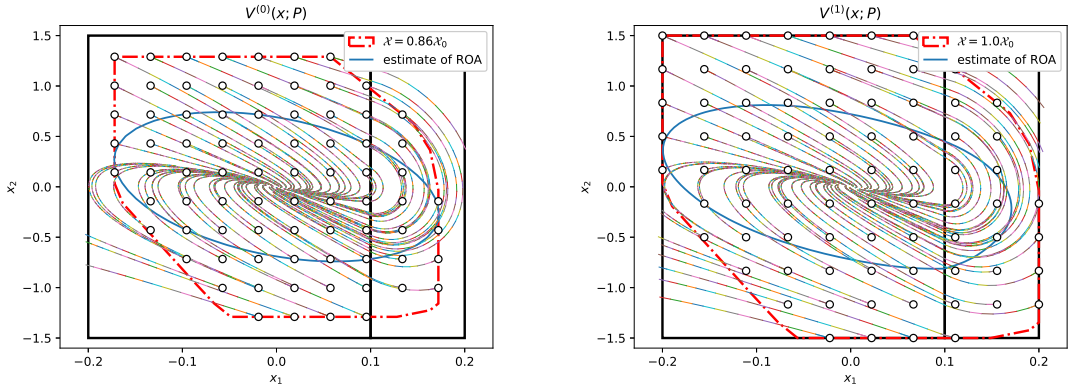


Figure 4.5: A ReLU neural network (right) that approximates a hybrid MPC controller is applied on the inverted pendulum system with an elastic wall (left).

generated to train a ReLU neural network $\pi(x)$ in Keras [141] to approximate the MPC controller. The neural network has 2 hidden layers with 20 neurons in each layer and its output layer bias term is modified after training to guarantee $\pi(0) = 0$. We plot the neural network controller in Fig. 4.5 and set $\epsilon = 0.0158$ in (4.26) since the closed-loop dynamics is linear and asymptotically stable inside B_ϵ.

For Lyapunov function candidates of order $k = 0$ and $k = 1$, we run Algorithm 3 with ROI $\mathcal{X} = \gamma\mathcal{X}_0$ of varying values of γ. For the quadratic function class $V^{(0)}(x; P)$, the largest ROI is given by $\mathcal{X} = 0.86\mathcal{X}_0$ through bisection with which Algorithm 3 terminates in 16 iterations with a total running time of 13.687 seconds. The corresponding estimate of ROA is shown in Fig. 4.6a. For the PWQ function class $V^{(1)}(x; P)$, the largest ROI is given by $\mathcal{X} = 1.0\mathcal{X}_0$ in which case Algorithm 3 terminates in 11 iterations with a total running time of 90.917 seconds. The estimate of ROA obtained by the found PWQ Lyapunov function candidate is shown in Fig. 4.6b. It shows that the synthesized Lyapunov function in $V^{(1)}(x; P)$ is less conservative compared with the one in $V^{(0)}(x; P)$ and Algorithm 3 can obtain non-trivial estimates of ROA for the neural network controlled hybrid systems.



(a) Estimate of ROA of the neural network controlled system by a quadratic Lyapunov function in $V^{(0)}(x;P)$.

(b) Estimate of ROA of the neural network controlled system by a PWQ Lyapunov function in $V^{(1)}(x;P)$.

Figure 4.6: Estimates of ROA found by Algorithm 3 with Lyapunov function candidates in $V^{(0)}(x;P)$ and $V^{(1)}(x;P)$. The partitions of the state space are marked by the black boxes. Simulated closed-loop trajectories with the neural network controller are plotted for a grid of initial conditions.

4.6. Chapter Summary

We have proposed a learning-based method to learn Lyapunov functions for autonomous hybrid systems that have a mixed-integer formulation, including piecewise affine, linear complementarity, mixed logical dynamical systems and ReLU neural networks. By designing the method according to the analytic center cutting-plane method, we show that the proposed algorithm is guaranteed to find a Lyapunov function in a finite number of steps when the set of Lyapunov functions is full-dimensional in the parameter space.

In terms of stability analysis of NN dynamical systems, our work proposes an effective Lyapunov function synthesis method utilizing the exact mixed-integer representation of ReLU networks. Although our method minimizes the conservatism in both network encoding and algorithm design, it can be computationally challenging when the ReLU network becomes complex. For future work, it will be useful to investigate how to combine convex relaxation-based NN verification tools and stability analysis algorithms.

CHAPTER 5

Robust MPC of Uncertain Linear Systems

5.1. Background on Robust MPC

In model predictive control (MPC), a finite time constrained optimal control problem (OCP) is solved at each time step and the first optimal control input is applied. When the system model is uncertain, robust model predictive control explicitly takes the system uncertainty into account by solving a robust OCP at each time step to guarantee that the state and control input constraints are robustly satisfied for the closed-loop system. Although in theory dynamic programming [97, Chapter 15] can exactly solve the resulting robust OCPs, such methods suffer from prohibitive computational complexity, making them impractical. This has motivated the development of alternative solutions to the robust OCPs which aim to reach a reasonable compromise between conservatism as measured by the size of the set of feasible initial states, and computational complexity.

Robust MPC with additive disturbances For uncertain linear time-invariant (LTI) systems with the only uncertainty arising from additive disturbances, a variety of closed-loop methods have been proposed to solve the robust OCPs with feedback policies as decision variables. The use of closed-loop policies does not suffer from the conservatism that open-loop strategies do in the presence of uncertainty [142]. To maintain computational tractability, closed-loop methods only search over control policies that admit a finite dimensional parameterization. Tube MPC is a widely used class of robust MPC methods where a tube, i.e., a sequence of polytopes, is constructed such that it contains all possible system trajectories under uncertainty. This is achieved by first parameterizing a class of tube and the associated control policy, and then enforcing trajectory containment constraints across the tube cross-sections in an inductive manner. The state and input constraints are tightened based on the tube to guarantee robust satisfaction of these constraints under uncertainty. Different designs of tube MPC such as rigid [143], homothetic [144], and elastic [145] tube MPC have been proposed with increasing flexibility in the parameterization

of the tube and control policy.

One limitation of most existing tube MPC methods (see the introduction of [146]) is the offline design of the tube cross-sections and the associated control law, e.g., by pre-fixing the shape of the cross-section and the feedback gain of controller [144], which can be conservative. State feedback robust MPC [147, 148] overcomes this limitation by optimizing the state feedback or the equivalent disturbance feedback gains [148] online, resulting in improved conservatism at a manageable computational cost. In fact, the number of decision variables scales linearly in the horizon for tube MPC using an offline design, and quadratically in the horizon for state/disturbance feedback robust MPC. A conservatism improvement together with the quadratic complexity is also achieved by the parameterized tube MPC [146] which adopts online optimization over both the tube and the control policies. The authors in [149] propose system level tube MPC which searches over state feedback gains online through the framework of System Level Synthesis (SLS) [65] with structural restrictions such that the number of variables is linear in the horizon.

Robust MPC with model uncertainty When model uncertainty is present, robust MPC becomes significantly more challenging since the perturbation to the nominal predicted trajectory depends on the states and control inputs, making it difficult to characterize the effects of uncertainty or the feedback policy to be designed.

A simple approach that globally over-approximates the effects of the state- and input-dependent uncertainty as additive disturbances reduces the problem to the disturbance-only case discussed above, for which a wide range of robust MPC methods are readily available [63]. Although easily implementable, such methods suffer from the inherent conservatism induced by the global uncertainty over-approximation which fails to exploit the structure of uncertainty.

For polytopic model uncertainty, robust MPC methods based on linear matrix inequalities (LMI) are presented in [150, 151, 152, 153] where affine state feedback policies for a min-max cost function over an infinite horizon are considered. The LMI-based methods are

straight-forward to use since they do not require much offline design, but their computational cost tends to scale badly with the system dimension and problem size because of the use of semidefinite programming [154]. In addition, the ellipsoidal inner-approximation of polyhedral state and control input constraints [150] can be conservative.

The application of tube MPC on handling model uncertainty can be found in [59, 155, 62, 61]. The authors in [59] utilize a homothetic tube to bound the system trajectories and parameterize a feedback control policy associated with each vertex of the tube cross-section. In [155, 62, 61], an offline-designed tube and a control policy $u_t = Kx_t + v_t$ with a pre-stabilizing feedback gain K is applied to guarantee robust constraint satisfaction. Similar to tube MPC in the disturbance-only scenario, the offline design of either the tube or the associated control policy can be conservative.

The recent work [64] first bounds the effects of model uncertainty offline as a function of the control inputs, and then optimizes a disturbance feedback controller online. While the online optimization of feedback gains is preferable for reducing conservatism, the tightness of the method is limited by the constraint tightening parameters that are synthesized offline.

Contribution In this work, we propose a state feedback robust MPC method which optimizes over robust feedback gains online for systems subject to polytopic model uncertainty and additive disturbances. Inspired by [63], we describe the uncertain system dynamics as the sum of the nominal dynamics and the lumped uncertainty that captures the deviation from the nominal predicted state at each time instant. Unlike [63] which simply over-approximate the lumped uncertainty uniformly by a sufficiently large additive disturbance signal, our method exactly characterizes how the lumped uncertainty depends on the model uncertainty and the controller parameters by leveraging the framework of System Level Synthesis [65], which establishes the equivalence between the optimization of linear time-varying (LTV) state feedback controllers and closed-loop system responses. Based on the exact characterization of the lumped uncertainty, we propose a novel constraint tightening method that employs a parameterized filtered additive disturbance signal to over-approximate the

lumped uncertainty at each time step. Importantly, the over-approximating additive disturbance signal is optimized *jointly* with the LTV state feedback controllers in a convex manner which leads to a convex quadratic program solved recursively in MPC.

We denote the proposed state feedback robust MPC method as *robust SLS MPC* since our robust OCP formulation is derived in the space of system responses made possible by SLS. By construction, robust SLS MPC considers a more general control policy class than the tube MPC approaches [155, 62, 61] with a fixed feedback gain. Compared with the simple methods [63] relying on a global additive disturbance over-approximation of the lumped uncertainty, our method captures the dependence of the lumped uncertainty on the designed controllers and is therefore less conservative. Robust SLS MPC solves a convex quadratic program at each time step and handles polyhedral constraints directly. Therefore, it is more computationally efficient and less conservative than the LMI-based approaches [150]. Our contributions are summarized as follows.

1. We propose a novel robust MPC method, robust SLS MPC, for uncertain LTI systems with polytopic model uncertainty and additive disturbances. At each time step, robust SLS MPC searches over linear state feedback policies for prediction and solves a convex inner approximation of the robust OCP in the space of system responses using SLS. The convex inner approximation is derived based on the dynamics of the lumped uncertainty and is shown to achieve significant reduction in conservatism compared with existing robust MPC methods.

2. We numerically compare our proposed method with representative robust MPC approaches including tube-MPC [59, 62, 60, 61], the lumped-uncertainty method in [63], the recent work in [64]. Our proposed method is shown to outperform all the methods by a significant margin in conservatism as measured by the sets of feasible initial states. In addition, we show that the computational complexity of our method is comparable to that of existing baselines.

Notation Let $\mathbb{R}_{\geq 0}$ denote the set of non-negative real numbers. For a dynamical system, we denote the system state at time t by $x(t)$ and the t-step prediction of the state in an MPC loop by x_t. For two vectors x and y, $x \leq y$ denotes element-wise comparison. For a symmetric matrix Q, $Q \succeq 0$ denotes that Q is positive semidefinite. The notation $x_{i:j}$ is shorthand for the set $\{x_i, x_{i+1}, \cdots, x_j\}$. The norm $\|\cdot\|_{\infty\to\infty}$ is the ℓ_∞ induced norm. $S = \text{blkdiag}(S_1, \cdots, S_N)$ denotes that S is a block diagonal matrix whose diagonal blocks consist of $S_1, \cdots, S_N$ arranged in the order. We represent a linear, causal operator $\mathbf{R}$ defined over a horizon of T by the block-lower-triangular matrix

$$\mathbf{R} = \begin{bmatrix} R^{0,0} & & & \\ R^{1,1} & R^{1,0} & & \\ \vdots & \ddots & \ddots & \\ R^{T,T} & \cdots & R^{T,1} & R^{T,0} \end{bmatrix} \tag{5.1}$$

where $R^{i,j} \in \mathbb{R}^{p\times q}$ is a matrix of compatible dimension. The set of such matrices is denoted by $\mathcal{L}_{TV}^{T,p\times q}$ and we will drop the superscript $T, p \times q$ when it is clear from the context. Let $\mathbf{R}(i,:)$ denote the i-th block row of $\mathbf{R}$, and $\mathbf{R}(:,j)$ denote the j-th block column of $\mathbf{R}$, both indexing from 0[17].

Recall from [156] that a function $\gamma(\cdot) : \mathbb{R}_{\geq 0} \mapsto \mathbb{R}_{\geq 0}$ is a class-$\mathcal{K}$ function if it is continuous, strictly increasing and $\gamma(0) = 0$. The function $\gamma(\cdot)$ is a class-$\mathcal{K}_\infty$ function if it is class-$\mathcal{K}$ and $\lim_{r\to\infty} \gamma(r) = \infty$. A function $\beta(\cdot,\cdot) : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \mapsto \mathbb{R}_{\geq 0}$ is a class-$\mathcal{KL}$ function if for each fixed $t \geq 0$, $\beta(\cdot, t)$ is a class-$\mathcal{K}$ function, and for each fixed $s \geq 0$, $\beta(s,\cdot)$ is decreasing and $\beta(s,t) \to 0$ as $t \to \infty$.

[17] In this chapter, we refer to a block matrix in a block-lower-triangular matrix $\mathbf{R}$ using its superscripts shown in Eqn. (5.1). If we let $\mathbf{R}(i,j)$ denote the block matrix in the i-th row and j-th column, then we have $\mathbf{R}(i,j) = R^{i,i-j}$ with (i,j) indexing from 0.

5.2. Problem Formulation

Consider a discrete-time linear system with uncertain dynamics

$$x(k+1) = (\hat{A} + \Delta_A)x(k) + (\hat{B} + \Delta_B)u(k) + w(k), \forall k \geq 0 \tag{5.2}$$

where $x(k) \in \mathbb{R}^{n_x}$ is the state, $u(k) \in \mathbb{R}^{n_u}$ is the control input, $w(k) \in \mathbb{R}^{n_x}$ is the additive disturbance, $(\hat{A}, \hat{B})$ are the known nominal dynamics, and the model uncertainty (Δ_A, Δ_B) lies in a polytopic set consisting of M vertices

$$(\Delta_A, \Delta_B) \in \text{Conv}\{(\Delta_{A,1}, \Delta_{B,1}), \cdots, (\Delta_{A,M}, \Delta_{B,M})\} \tag{5.3}$$

with Conv denoting the convex hull. For simplicity, we denote the polytopic uncertainty set in (5.3) as $\mathcal{P}$. The additive disturbances $w(k)$ are norm-bounded, i.e, $\|w(k)\|_\infty \leq \sigma_w$, and we denote this uncertainty set as $\mathcal{W}$.

In robust MPC, at each time instant k, we solve a finite time robust OCP which aims to synthesize a feedback controller that guarantees the robust satisfaction of state and input constraints. The first control input in the optimal solution is applied to drive the system to the next state, and this process is repeated. In this paper, we focus on solving the following robust OCP with polytopic model uncertainty and norm-bounded additive disturbances.

Problem 2. *Solve the following finite time constrained robust OCP with horizon T:*

$$\begin{aligned}
\min_{\pi} \quad & J_T(x(k), \pi) \\
s.t. \quad & x_{t+1} = (\hat{A} + \Delta_A)x_t + (\hat{B} + \Delta_B)u_t + w_t \\
& u_t = \pi_t(x_{0:t}) \\
& x_t \in \mathcal{X}, u_t \in \mathcal{U}, x_T \in \mathcal{X}_T, t = 0, 1, \cdots, T-1 \\
& \forall (\Delta_A, \Delta_B) \in \mathcal{P}, \forall w_t \in \mathcal{W}, t = 0, 1, \cdots, T-1 \\
& x_0 = x(k)
\end{aligned} \tag{5.4}$$

where the search is over causal linear time-varying (LTV) state feedback control policies $\pi = \pi_{0:T-1}$ *and the cost function* $J_T(x(k), \pi)$ *is to be specified. The sets* $\mathcal{X}$,$\mathcal{U}$, *and* $\mathcal{X}_T$ *are polytopic state, input, and terminal constraints, defined as*

$$\mathcal{X} = \{x \in \mathbb{R}^{n_x} \mid F_x x \leq b_x\},\ \mathcal{U} = \{u \in \mathbb{R}^{n_u} \mid F_u u \leq b_u\},$$
$$\mathcal{X}_T = \{x \in \mathbb{R}^{n_x} \mid F_T x \leq b_T\}.$$

We assume that the sets $\mathcal{X}$,$\mathcal{U}$, *and* $\mathcal{X}_T$ *are compact and contain the origin in their interior.*

In this paper, we consider using a nominal cost function given by

$$\begin{aligned} J_T(x(k), \pi) \quad &= \textstyle\sum_{t=0}^{T-1} (\hat{x}_t^\top Q \hat{x}_t + u_t^\top R u_t) + \hat{x}_T^\top Q_T \hat{x}_T \\ \text{s.t.} \quad &\hat{x}_{t+1} = \hat{A}\hat{x}_t + \hat{B}u_t, u_t = \pi_t(\hat{x}_{0:t}) \\ &\hat{x}_0 = x(k), \quad \forall t = 0, 1, \cdots, T-1 \end{aligned} \tag{5.5}$$

with $\hat{x}_{0:T}$ denoting the nominal trajectory. Here $Q \succeq 0, R \succ 0$, and $Q_T \succeq 0$ are the state, input, and terminal weight matrices, respectively.

Searching over arbitrary feedback control policies π_t in the robust OCP (5.4) is intractable due to its infinite dimensionality. Furthermore, even when only a finite-dimensional parameterized class of control policies π_t is considered, exactly solving the robust OCP (5.4) is often intractable due to the complex interaction between π_t and the model uncertainties. As a result, the performance of robust MPC is largely dependent on our ability to efficiently find a feasible and sub-optimal solution to (5.4) that is not overly conservative. The conservatism of solving (5.4) jointly depends on the parameterized feedback policy π_t and the algorithms we apply to find a feasible solution. In this work, we consider searching over causal LTV state feedback controllers represented by $u_t = \sum_{i=0}^{t} K^{t,t-i} x_i$ for $t = 0, \cdots, T-1$ with $K^{t,t-i}$ being our design parameters. Our algorithm to solve (5.4) relies on over-approximating the effects of uncertainties in the space of closed-loop system responses as shown in the following sections.

5.3. Characterization of Uncertainty Effects

In the robust OCP, the system dynamics can be decomposed as the sum of nominal dynamics and uncertainty-related terms:

$$\begin{aligned} x_{t+1} &= \hat{A}x_t + \hat{B}u_t + \underbrace{\Delta_A x_t + \Delta_B u_t + w_t}_{\eta_t} \\ &= \hat{A}x_t + \hat{B}u_t + \eta_t \end{aligned} \tag{5.6}$$

where we define $\eta_t := \Delta_A x_t + \Delta_B u_t + w_t$ as the *lumped uncertainty* which models the perturbation to the nominal dynamics at each time step. To find a feasible solution to (5.4) with minimal conservatism, a proper characterization of the effects of η_t is required. However, this is challenging since the lumped uncertainty is dependent on both the uncertainty parameters and the feedback controller to be designed. In this section, we show that SLS allows us to characterize the dynamics of the lumped uncertainty η_t as a function of both system uncertainties and the applied state feedback controller.

5.3.1. Finite-horizon system level synthesis

To apply SLS, as the first step, we stack all relevant state, control input, and uncertainty variables over horizon T as

$$\begin{aligned} &\mathbf{x} = [x_0^\top \ \cdots \ x_T^\top]^\top, \quad \mathbf{u} = [u_0^\top \ \cdots \ u_T^\top]^\top, \\ &\boldsymbol{\eta} = [x_0^\top \ \eta_0^\top \ \cdots \ \eta_{T-1}^\top]^\top, \quad \mathbf{w} = [x_0^\top \ w_0^\top \ \cdots \ w_{T-1}^\top]^\top. \end{aligned} \tag{5.7}$$

Note that the initial state x_0 is set as the first component in $\boldsymbol{\eta}$ and $\mathbf{w}$, and x_0 can be interpreted as a known disturbance from the origin in this case. The vectors in (5.7) can be interpreted as finite horizon signals. The parameterization of the LTV state feedback controller $\mathbf{K} \in \mathcal{L}_{TV}^{T,n_u \times n_x}$ is represented by the block-lower-triangular matrix (5.1) with entries $K^{t,t-i}$. The controller $\mathbf{K}$ can be interpreted as an LTV linear operator and the state feedback controller is given by $\mathbf{u} = \mathbf{K}\mathbf{x}$. Similarly, we stack the dynamics matrices and

uncertainty matrices as

$$\begin{aligned}&\hat{\mathbf{A}} = \text{blkdiag}(\hat{A}, \cdots, \hat{A}), \quad \hat{\mathbf{B}} = \text{blkdiag}(\hat{B}, \cdots, \hat{B}),\\ &\boldsymbol{\Delta}_A = \text{blkdiag}(\Delta_A, \cdots, \Delta_A), \boldsymbol{\Delta}_B = \text{blkdiag}(\Delta_B, \cdots, \Delta_B).\end{aligned} \tag{5.8}$$

With the above defined compact notations, the open-loop dynamics of the system can be equivalently written as

$$\mathbf{x} = Z\hat{\mathbf{A}}\mathbf{x} + Z\hat{\mathbf{B}}\mathbf{u} + \boldsymbol{\eta} \tag{5.9}$$

and the closed-loop dynamics under $\mathbf{u} = \mathbf{K}\mathbf{x}$ follows as

$$\mathbf{x} = Z(\hat{\mathbf{A}} + \hat{\mathbf{B}}\mathbf{K})\mathbf{x} + \boldsymbol{\eta} \tag{5.10}$$

where Z is a block down-shifting operator with the first block sub-diagonal filled with identity matrices and zeros everywhere else. Note that the lumped uncertainty $\boldsymbol{\eta}$ also depends on $\mathbf{K}$ and $\mathbf{x}$ as will be shown in the next subsection.

From (5.10), the mapping from the lumped uncertainty to the state and control input under the feedback controller is given by

$$\begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix} = \begin{bmatrix}(I - Z(\hat{\mathbf{A}} + \hat{\mathbf{B}}\mathbf{K}))^{-1}\\ \mathbf{K}(I - Z(\hat{\mathbf{A}} + \hat{\mathbf{B}}\mathbf{K}))^{-1}\end{bmatrix}\boldsymbol{\eta}. \tag{5.11}$$

From the block-down-shifting operator Z, we know that the matrix inverse in (5.11) exists and has a block-lower-triangular structure. The maps from $\boldsymbol{\eta}$ to $(\mathbf{x}, \mathbf{u})$ in (5.11) are called *system responses*, and we introduce maps $\boldsymbol{\Phi}_x \in \mathcal{L}_{TV}^{T, n_x \times n_x}$, $\boldsymbol{\Phi}_u \in \mathcal{L}_{TV}^{T, n_u \times n_x}$ following [65] such that

$$\begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix} = \begin{bmatrix}\boldsymbol{\Phi}_x\\ \boldsymbol{\Phi}_u\end{bmatrix}\boldsymbol{\eta}. \tag{5.12}$$

The relationship between the state feedback controller $\mathbf{K}$ and the system responses $(\boldsymbol{\Phi}_x, \boldsymbol{\Phi}_u)$

is given by (5.11). The following theorem allows us to equivalently transform design of the feedback controller $\mathbf{K}$ into design of the system responses $(\mathbf{\Phi}_x, \mathbf{\Phi}_u)$ without explicitly using the nonlinear map (5.11).

Theorem 4. *[65, Theorem 2.1] Over the horizon $t = 0, 1, \cdots, T$, for the system dynamics* (5.6) *with the block-lower-triangular state feedback control law $\mathbf{K} \in \mathcal{L}_{TV}^{T,n_u \times n_x}$ defining the control action as $\mathbf{u} = \mathbf{K}\mathbf{x}$, we have:*

1. *The affine subspace defined by*

$$\begin{bmatrix} I - Z\hat{\mathbf{A}} & -Z\hat{\mathbf{B}} \end{bmatrix} \begin{bmatrix} \mathbf{\Phi}_x \\ \mathbf{\Phi}_u \end{bmatrix} = I, \ \mathbf{\Phi}_x, \mathbf{\Phi}_u \in \mathcal{L}_{TV} \tag{5.13}$$

   parameterizes all possible system responses (5.12)*.*

2. *For any block-lower-triangular matrices $\{\mathbf{\Phi}_x, \mathbf{\Phi}_u\} \in \mathcal{L}_{TV}$ satisfying* (5.13)*, the controller $\mathbf{K} = \mathbf{\Phi}_u \mathbf{\Phi}_x^{-1} \in \mathcal{L}_{TV}$ achieves the desired responses* (5.12)*.*

Theorem 4 shows the equivalence between system responses and linear state feedback controllers through the affine constraint (5.13), with which optimization problems originally on $\mathbf{K}$ can be transformed into one on $(\mathbf{\Phi}_x, \mathbf{\Phi}_u)$. Such transformation may result in a convex problem in $(\mathbf{\Phi}_x, \mathbf{\Phi}_u)$ while the original one is not, and provides a direct description on the effects of the uncertainty $\boldsymbol{\eta}$ on the states and control inputs. More importantly, as shown in this paper, such transformation can reveal additional structured properties of the problem in the space of system responses that can be exploited for high-performance controller design.

5.3.2. Dynamics of lumped uncertainty

By the definition of lumped uncertainty (5.6) and the compact notations from (5.7), (5.8), we have

$$\boldsymbol{\eta} = Z \begin{bmatrix} \mathbf{\Delta}_A & \mathbf{\Delta}_B \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} + \mathbf{w}. \tag{5.14}$$

Then, the system responses (5.12) allows an explicit characterization of the dynamics of η_t under the controller $\mathbf{K}$ as

$$\boldsymbol{\eta} = Z \begin{bmatrix} \boldsymbol{\Delta}_A & \boldsymbol{\Delta}_B \end{bmatrix} \begin{bmatrix} \boldsymbol{\Phi}_x \\ \boldsymbol{\Phi}_u \end{bmatrix} \boldsymbol{\eta} + \mathbf{w} \tag{5.15}$$

which can be equivalently decomposed into the following equations [18]

$$\eta_t = \Delta_A(\Phi_x^{t,t} x_0 + \sum_{i=1}^{t} \Phi_x^{t,t-i} \eta_{i-1}) + \Delta_B(\Phi_u^{t,t} x_0 + \sum_{i=1}^{t} \Phi_u^{t,t-i} \eta_{i-1}) + w_t \tag{5.16}$$

for $t = 0, \cdots, T-1$. The dynamics of the lumped uncertainty (5.15) exactly describes how the values of η_t are jointly and uniquely determined by the uncertainty parameters $(\boldsymbol{\Delta}_A, \boldsymbol{\Delta}_B, \mathbf{w})$ and the closed-loop system responses $(\boldsymbol{\Phi}_x, \boldsymbol{\Phi}_u)$, which implicitly encode a linear state feedback controller $\mathbf{K} = \boldsymbol{\Phi}_u \boldsymbol{\Phi}_x^{-1}$. The uniqueness can be seen from (5.16) that η_t is only dependent on η_i with $i \leq t-1$. Therefore, once the controller $(\boldsymbol{\Phi}_x, \boldsymbol{\Phi}_u)$ and the uncertainty parameters $(\boldsymbol{\Delta}_A, \boldsymbol{\Delta}_B, \mathbf{w})$ are fixed, the values of η_t for $0 \leq t \leq T-1$ are correspondingly decided. When (Δ_A, Δ_B) and w_t are unknown, the values of η_t also become uncertain and we denote by $\mathcal{R}(\boldsymbol{\eta}; \{\boldsymbol{\Phi}_x, \boldsymbol{\Phi}_u\})$ the set of all possible values of $\{\eta_t\}_{t=0}^{T-1}$ under the uncertainty assumption in Section 5.2 for a given controller defined by $\{\boldsymbol{\Phi}_x, \boldsymbol{\Phi}_u\}$.

Despite being exact, the lumped uncertainty value set $\mathcal{R}(\boldsymbol{\eta}; \{\boldsymbol{\Phi}_x, \boldsymbol{\Phi}_u\})$ generated by (5.15) is too complex to use for solving the robust OCP (5.4). In the next section, we over-approximate $\mathcal{R}(\boldsymbol{\eta}; \{\boldsymbol{\Phi}_x, \boldsymbol{\Phi}_u\})$ by an uncertainty set with simpler structure that is amenable for solving the robust OCP (5.4).

5.4. Uncertainty Set Over-approximation

To over-approximate $\mathcal{R}(\boldsymbol{\eta}; \{\boldsymbol{\Phi}_x, \boldsymbol{\Phi}_u\})$, we devise a disturbance signal represented by $\boldsymbol{\Sigma}\tilde{\mathbf{w}}$ where $\boldsymbol{\Sigma} \in \mathcal{L}_{TV}^{T, n_x \times n_x}$ is a filter operating on a unit norm-bounded virtual disturbance signal

$$\tilde{\mathbf{w}} = [x_0^\top \ \tilde{w}_0^\top \ \cdots \ \tilde{w}_{T-1}^\top]^\top \text{ with } \|\tilde{w}_t\|_\infty \leq 1. \tag{5.17}$$

[18] We adopt the convention that when $t = 0$, the summation terms in (5.16) vanish.

We denote the set of $\widetilde{\mathbf{w}}$ satisfying the unit norm bound constraint (5.17) as $\mathcal{W}_{\widetilde{\mathbf{w}}}$, and the set of all possible values of $\mathbf{\Sigma}\widetilde{\mathbf{w}}$ when $\widetilde{\mathbf{w}} \in \mathcal{W}_{\widetilde{\mathbf{w}}}$ as $\mathcal{R}(\mathbf{\Sigma}\widetilde{\mathbf{w}})$. Our goal is to over-approximate the value set of the lumped uncertainty $\mathcal{R}(\boldsymbol{\eta}; \{\mathbf{\Phi}_x, \mathbf{\Phi}_u\})$ by that of the filtered disturbances $\mathcal{R}(\mathbf{\Sigma}\widetilde{\mathbf{w}})$, i.e., $\mathcal{R}(\boldsymbol{\eta}; \{\mathbf{\Phi}_x, \mathbf{\Phi}_u\}) \subseteq \mathcal{R}(\mathbf{\Sigma}\widetilde{\mathbf{w}})$, such that it suffices to consider the dynamical system

$$\mathbf{x} = Z\hat{\mathbf{A}}\mathbf{x} + Z\hat{\mathbf{B}}\mathbf{u} + \mathbf{\Sigma}\widetilde{\mathbf{w}} \tag{5.18}$$

with the surrogate disturbances $\mathbf{\Sigma}\widetilde{\mathbf{w}}$ for the robust OCP. The unit norm-bounded assumption on the virtual disturbances $\tilde{w}_t$ simplifies the constraint tightening of the robust OCP (5.4), while the filter $\mathbf{\Sigma}$, which is our design parameter, controls the complexity of $\mathcal{R}(\mathbf{\Sigma}\widetilde{\mathbf{w}})$ such that it can over-approximate $\mathcal{R}(\boldsymbol{\eta}; \{\mathbf{\Phi}_x, \mathbf{\Phi}_u\})$ with minimal conservatism.

5.4.1. Parameterization of the filter

We can parameterize the filter $\mathbf{\Sigma}$ as a block diagonal matrix $\mathbf{\Sigma} = \text{blkdiag}(I, \sigma_0 I, \cdots, \sigma_{T-1} I)$ with which $\mathcal{R}(\mathbf{\Sigma}\widetilde{\mathbf{w}})$ represents a Cartesian product of ℓ_∞ norm balls with radii $\sigma_t > 0$. Using this parameterization, we bound the lumped uncertainty η_t by an ℓ_∞ ball of varying radius σ_t at each time t. This over-approximation method has demonstrated outstanding performance in robust MPC of systems subject to norm-bounded model uncertainty [157]. To handle the polytopic model uncertainty considered in this paper and further reduce the conservatism of robust MPC, we propose a novel method for over-approximating the uncertainty set $\mathcal{R}(\boldsymbol{\eta}; \{\mathbf{\Phi}_x, \mathbf{\Phi}_u\})$ using a more flexible parameterization of the filter $\mathbf{\Sigma}$.

Specifically, we consider the full block-lower-triangular parameterization of $\mathbf{\Sigma} \in \mathcal{L}_{TV}^{T, n_x \times n_x}$ where the sub-diagonal blocks $\Sigma^{t,t-i}$ are non-zero. The only restriction on $\mathbf{\Sigma}$ is that the matrix blocks $\Sigma^{t,0}$ on the diagonal of $\mathbf{\Sigma}$ are diagonal matrices with positive entries, i.e., $\Sigma^{t,0} = \text{diag}(d_{t-1})$ for $1 \leq t \leq T$ where $d_{t-1} \in \mathbb{R}^{n_x}$ satisfies $d_{t-1} > 0$. Additionally, we require $\Sigma^{0,0} = I$ such that the first component of $\mathbf{\Sigma}\widetilde{\mathbf{w}}$ is still the initial state x_0. By construction, $\mathbf{\Sigma}$ is invertible. Such parameterization contains the block-diagonal parameterization discussed above as a special case, and is therefore less conservative. Indeed, when the sub-diagonal matrix blocks of $\mathbf{\Sigma}$ are enforced zero, $\mathcal{R}(\mathbf{\Sigma}\widetilde{\mathbf{w}})$ represents a sequence of hyperrectangles

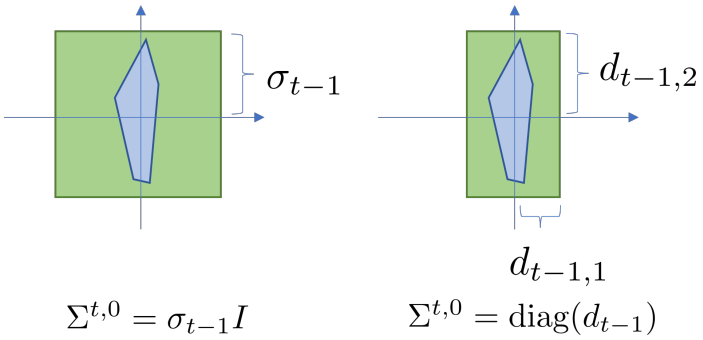


Figure 5.1: Blue: value set of the lumped uncertainty η_t. Green: over-approximation by the filtered disturbance signal $\sigma_t \tilde{w}_t$ under block diagonal parameterization of the filter $\mathbf{\Sigma} = \mathrm{blkdiag}(I, \Sigma^{1,0}, \cdots, \Sigma^{T-1,0})$ with $\Sigma^{t,0} = \sigma_{t-1} I$ (left) or $\Sigma^{t,0} = \mathrm{diag}(d_{t-1})$ (right).

instead of norm-balls, and is able to bound the values of η_t in a tighter way as shown in Figure 5.1.

5.4.2. Formulation of the over-approximation problem

The uncertainty set over-approximation problem can be stated as finding $(\mathbf{\Phi}_x, \mathbf{\Phi}_u, \mathbf{\Sigma})$ such that

$$\mathcal{R}(\boldsymbol{\eta}; \{\mathbf{\Phi}_x, \mathbf{\Phi}_u\}) \subseteq \mathcal{R}(\mathbf{\Sigma}\tilde{\mathbf{w}}) \tag{5.19}$$

for all $(\Delta_A, \Delta_B) \in \mathcal{P}$ and $w_t \in \mathcal{W}$. This is equivalent to the following problem:

Problem 3 (Uncertainty set over-approximation)**.** *Find a controller* $(\mathbf{\Phi}_x, \mathbf{\Phi}_u)$ *and a filter* $\mathbf{\Sigma}$ *such that*

$$Z \begin{bmatrix} \mathbf{\Delta}_A & \mathbf{\Delta}_B \end{bmatrix} \begin{bmatrix} \mathbf{\Phi}_x \\ \mathbf{\Phi}_u \end{bmatrix} \mathbf{\Sigma}\tilde{\mathbf{w}} + \mathbf{w} = \mathbf{\Sigma}\tilde{\mathbf{w}} \tag{5.20}$$

has a solution $\tilde{\mathbf{w}}^* \in \mathcal{W}_{\tilde{\mathbf{w}}}$ *for all possible realizations of the uncertainty* $(\mathbf{\Delta}_A, \mathbf{\Delta}_B, \mathbf{w})$.

To see the equivalence, note that for any given realization of $(\mathbf{\Delta}_A, \mathbf{\Delta}_B, \mathbf{w})$, the value of the lumped uncertainty is uniquely determined by (5.15) which we denote as $\boldsymbol{\eta}^*$. Since the filter $\mathbf{\Sigma}$ is invertible, $\tilde{\mathbf{w}}^* = \mathbf{\Sigma}^{-1}\boldsymbol{\eta}^*$ is the unique solution to the system of equations (5.20). Then (5.19) holds if and only if $\tilde{\mathbf{w}}^* \in \mathcal{W}_{\tilde{\mathbf{w}}}$ for all possible $(\mathbf{\Delta}_A, \mathbf{\Delta}_B, \mathbf{w})$.

Change of variables

The bilinear terms $\mathbf{\Phi}_x\mathbf{\Sigma}$ and $\mathbf{\Phi}_u\mathbf{\Sigma}$ in (5.20) make it challenging to solve Problem 3 since they lead to a non-convex optimization problem. To resolve this issue, we apply the change of variables

$$\widetilde{\mathbf{\Phi}}_x = \mathbf{\Phi}_x\mathbf{\Sigma}, \quad \widetilde{\mathbf{\Phi}}_u = \mathbf{\Phi}_u\mathbf{\Sigma} \tag{5.21}$$

where $\widetilde{\mathbf{\Phi}}_x, \widetilde{\mathbf{\Phi}}_u \in \mathcal{L}_{TV}$ can be interpreted as system responses mapping $\widetilde{\mathbf{w}}$ to $(\mathbf{x}, \mathbf{u})$ for the system $\mathbf{x} = Z\hat{\mathbf{A}}\mathbf{x} + Z\hat{\mathbf{B}}\mathbf{u} + \mathbf{\Sigma}\widetilde{\mathbf{w}}$ in closed-loop with $\mathbf{u} = \mathbf{K}\mathbf{x}$. According to [157, Theorem 1], all achievable closed-loop system responses $(\widetilde{\mathbf{\Phi}}_x, \widetilde{\mathbf{\Phi}}_u)$ are parameterized by

$$\begin{bmatrix} I - Z\hat{\mathbf{A}} & -Z\hat{\mathbf{B}} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{\Phi}}_x \\ \widetilde{\mathbf{\Phi}}_u \end{bmatrix} = \mathbf{\Sigma}, \tag{5.22}$$

which is jointly affine in $(\widetilde{\mathbf{\Phi}}_x, \widetilde{\mathbf{\Phi}}_u, \mathbf{\Sigma})$. Under the affine constraint (5.22), Problem 3 is equivalently transformed into finding $(\widetilde{\mathbf{\Phi}}_x, \widetilde{\mathbf{\Phi}}_u, \mathbf{\Sigma})$ such that

$$Z \begin{bmatrix} \mathbf{\Delta}_A & \mathbf{\Delta}_B \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{\Phi}}_x \\ \widetilde{\mathbf{\Phi}}_u \end{bmatrix} \widetilde{\mathbf{w}} + \mathbf{w} = \mathbf{\Sigma}\widetilde{\mathbf{w}}, \ \widetilde{\mathbf{w}} \in \mathcal{W}_{\widetilde{\mathbf{w}}} \tag{5.23}$$

is feasible for all $(\Delta_A, \Delta_B) \in \mathcal{P}$ and $w_t \in \mathcal{W}$. The invertibility of $\mathbf{\Sigma}$ guarantees the equivalence between searching $(\mathbf{\Phi}_x, \mathbf{\Phi}_u, \mathbf{\Sigma})$ under constraint (5.13) and searching $(\widetilde{\mathbf{\Phi}}_x, \widetilde{\mathbf{\Phi}}_u, \mathbf{\Sigma})$ under constraint (5.22) (see [157, Remark 1] for more details).

5.4.3. Convex over-approximation constraints

We now present convex sufficient conditions on $(\widetilde{\mathbf{\Phi}}_x, \widetilde{\mathbf{\Phi}}_u, \mathbf{\Sigma})$ such that the robust feasibility of (5.23) is guaranteed. The block-down-shifting operator Z in (5.23) makes it possible to decompose and analyze the equality constraints for $t = 0, \cdots, T-1$ sequentially.

Case: $t = 0$

For bounding η_0 at $t = 0$, we have the following constraint from (5.23):

$$\Delta_A x_0 + \Delta_B \tilde{\Phi}_u^{0,0} x_0 + w_0 = \Sigma^{1,1} x_0 + \Sigma^{1,0} \tilde{w}_0 \tag{5.24}$$

where we plug in $\tilde{\Phi}_x^{0,0} = I$. The above constraint is robustly feasible with a solution $\|\tilde{w}_0\|_\infty \leq 1$ if and only if

$$\|{\Sigma^{1,0}}^{-1}(\Delta_A x_0 + \Delta_B \tilde{\Phi}_u^{0,0} x_0 - \Sigma^{1,1} x_0 + w_0)\|_\infty \leq 1 \tag{5.25}$$

for all $(\Delta_A, \Delta_B) \in \mathcal{P}$ and $\|w_0\|_\infty \leq \sigma_w$. Constraint (5.25) is non-convex in the design parameter $\Sigma^{1,0}$, but under our diagonal matrix parameterization $\Sigma^{1,0} = \text{diag}(d_0)$ (see Section 5.4.1) it can be rewritten as

$$\|e_i^\top (\Delta_A x_0 + \Delta_B \tilde{\Phi}_u^{0,0} x_0 - \Sigma^{1,1} x_0 + w_0)\|_1 \leq d_{0,i} \tag{5.26}$$

for $i = 1, \cdots, n_x$ where $d_{1,i}$ denotes the i-th entry of d_1 and e_i is the i-th standard basis. The change of norm type in (5.26) is due to the fact the left-hand side (LHS) of (5.26) is now a vector, and therefore the matrix ∞-induced norm reduces to the ℓ_1 norm. Then, the following constraints

$$\begin{aligned} &\|e_i^\top (\Delta_A x_0 + \Delta_B \tilde{\Phi}_u^{0,0} x_0 - \Sigma^{1,1} x_0)\|_1 + \sigma_w \leq d_{0,i}, \\ &\forall (\Delta_A, \Delta_B) \in \text{Vert}(\mathcal{P}), \quad i = 1, \cdots, n_x \end{aligned} \tag{5.27}$$

guarantee (5.26) is robustly feasible, where $\text{Vert}(\mathcal{P}) = \{(\Delta_{A,j}, \Delta_{B,j}), j = 1, \cdots, M\}$ denotes the set of vertices of the uncertainty set $\mathcal{P}$ defined in (5.3). The robust feasibility guarantee follows from the triangle inequality of $\|\cdot\|_1$, the norm bound σ_w on w_0, and the fact that the LHS of (5.26) is convex in (Δ_A, Δ_B), and that the maximum of a convex function over a convex polytope is achieved at the polytope vertices [128].

Now constraint (5.27) is convex in the design parameters $\tilde{\Phi}_u^{0,0}, \Sigma^{1,1}$ and d_1. It guarantees that for all possible realizations of $(\Delta_A, \Delta_B, w_0)$ and the generated lumped uncertainty η_0 generated, we can always find $\tilde{w}_0^*$ such that $\eta_0 = \Sigma^{1,1}x_0 + \Sigma^{1,0}\tilde{w}_0^*$ with $\|\tilde{w}_0^*\|_\infty \leq 1$. We fix $\tilde{w}_0^*$ as the $\tilde{w}_0$-component of the solution $\widetilde{\mathbf{w}}^*$ to (5.23).

Case: $t = 1$

To bound η_1, similarly we first write down the relevant equality constraints from (5.23) as

$$\Delta_A(\tilde{\Phi}_x^{1,1}x_0 + \tilde{\Phi}_x^{1,0}\tilde{w}_0^*) + \Delta_B(\tilde{\Phi}_u^{1,1}x_0 + \tilde{\Phi}_u^{1,0}\tilde{w}_0^*) + w_1 = \Sigma^{2,2}x_0 + \Sigma^{2,1}\tilde{w}_0^* + \Sigma^{2,0}\tilde{w}_1 \tag{5.28}$$

where $\tilde{w}_0^*$ is the solution from the previous step and captures the effects of η_0 on future perturbations η_t for $t \geq 1$. Using the diagonal matrix parameterization of $\Sigma^{2,0} = \mathrm{diag}(d_1)$, following the same steps as in the case $t = 0$ and grouping the terms in (5.28) by $(x_0, \tilde{w}_0^*, \tilde{w}_1)$, we have that if the inequalities

$$\|e_i^\top(\Delta_A\tilde{\Phi}_x^{1,1} + \Delta_B\tilde{\Phi}_u^{1,1} - \Sigma^{2,2})x_0\|_1 + \|e_i^\top(\Delta_A\tilde{\Phi}_x^{1,0} + \Delta_B\tilde{\Phi}_u^{1,0} - \Sigma^{2,1})\tilde{w}_0^*\|_1 + \|e_i^\top w_1\|_1 \leq d_{1,i} \tag{5.29}$$

for $1 \leq i \leq n_x$ hold robustly, then the robust feasibility of (5.28) is guaranteed. However, the exact value of $\tilde{w}_0^*$ is unknown to us since it depends on $(\Delta_A, \Delta_B, w_0)$. To address this issue, we treat $\tilde{w}_0^*$ as uncertainty satisfying $\|\tilde{w}_0^*\|_\infty \leq 1$ and further tighten the constraint (5.29) as

$$\begin{aligned} &\|e_i^\top(\Delta_A\tilde{\Phi}_x^{1,1} + \Delta_B\tilde{\Phi}_u^{1,1} - \Sigma^{2,2})x_0\|_1 + \|e_i^\top(\Delta_A\tilde{\Phi}_x^{1,0} + \Delta_B\tilde{\Phi}_u^{1,0} - \Sigma^{2,1})\|_1 + \sigma_w \leq d_{1,i}, \\ &\forall(\Delta_A, \Delta_B) \in \mathrm{Vert}(\mathcal{P}), \quad i = 1, \cdots, n_x \end{aligned} \tag{5.30}$$

by applying the Hölder's inequality on $\|a^\top\tilde{w}_0^*\|_1 \leq \|a\|_1\|\tilde{w}_0^*\|_\infty \leq \|a\|_1$ and $\|e_i^\top w_1\|_1 \leq \|e_i\|_1\|w_1\|_\infty \leq \sigma_w$.

General case

We repeat this process from $t = 0$ to $t = T - 1$ to obtain a set of convex constraints on $(\widetilde{\mathbf{\Phi}}_x, \widetilde{\mathbf{\Phi}}_u, \mathbf{\Sigma})$:

$$\begin{aligned}
&\|e_i^\top(\Delta_A\tilde{\Phi}_x^{t,t} + \Delta_B\tilde{\Phi}_u^{t,t} - \Sigma^{t+1,t+1})x_0\|_1 + \sigma_w + \\
&\sum_{i=1}^{t}\|e_i^\top(\Delta_A\tilde{\Phi}_x^{t,t-i} + \Delta_B\tilde{\Phi}_u^{t,t-i} - \Sigma^{t+1,t+1-i})\|_1 \leq d_{t,i}, \qquad (5.31)\\
&\forall(\Delta_A, \Delta_B) \in \mathrm{Vert}(\mathcal{P}), i = 1, \cdots, n_x, t = 0, \cdots, T-1.
\end{aligned}$$

Any feasible solution $(\widetilde{\mathbf{\Phi}}_x, \widetilde{\mathbf{\Phi}}_u, \mathbf{\Sigma})$ to constraints (5.31) guarantees the existence of a virtual disturbance signal $\widetilde{\mathbf{w}}^* \in \mathcal{W}_{\widetilde{\mathbf{w}}}$ such that $\boldsymbol{\eta} = \mathbf{\Sigma}\widetilde{\mathbf{w}}^*$ for all possible realization of uncertainties. In other words, $\mathcal{R}(\boldsymbol{\eta}; \{\mathbf{\Phi}_x, \mathbf{\Phi}_u\}) \subseteq \mathcal{R}(\mathbf{\Sigma}\widetilde{\mathbf{w}})$ holds, and it suffices to consider the uncertain dynamical system $\mathbf{x} = Z\hat{\mathbf{A}}\mathbf{x} + Z\hat{\mathbf{B}}\mathbf{u} + \mathbf{\Sigma}\widetilde{\mathbf{w}}$ with the surrogate additive disturbance $\mathbf{\Sigma}\widetilde{\mathbf{w}}$ for solving the robust OCP (5.4). The synthesized controller $\mathbf{K} = \widetilde{\mathbf{\Phi}}_u\widetilde{\mathbf{\Phi}}_x^{-1}$ enjoys robustness guarantees for the original uncertain dynamical system (5.2). We present the details of the robust OCP solutions in the next section.

5.5. Convex Formulation of Robust Optimal Control Problem

In this section, we summarize our tools and present our solution to the robust OCP (5.4).

5.5.1. Constraint tightening of robust OCP

Recall that with SLS, it suffices to consider the uncertain system $\mathbf{x} = Z\hat{\mathbf{A}}\mathbf{x} + Z\hat{\mathbf{B}}\mathbf{u} + \mathbf{\Sigma}\widetilde{\mathbf{w}}$ for robust control since under constraint (5.31), the additive filtered disturbances $\mathbf{\Sigma}\widetilde{\mathbf{w}}$ can realize all possible values of the lumped uncertainties $\eta_t = \Delta_A x_t + \Delta_B u_t + w_t$ at each time t. Compared with the actual uncertain perturbation η_t, the over-approximating additive disturbance $\mathbf{\Sigma}\widetilde{\mathbf{w}}$ has a simple structure with $\|\tilde{w}_t\|_\infty \leq 1$ which makes constraint tightening of the robust OCP (5.4) straightforward.

Let us take the state constraint tightening of $x_t \in \mathcal{X}$ as an example. The state constraint is a polyhedral set $\mathcal{X} = \{x \in \mathbb{R}^{n_x} \mid F_x x \leq b_x\}$. Denote the number of linear constraints in defining $\mathcal{X}$ as $n_\mathcal{X}$ and $\mathrm{facet}(\mathcal{X}) = \{(F_x(:,i), b_x(i)) \mid i = 1, \cdots, n_\mathcal{X}\}$ as the set of all

linear constraint parameters of $\mathcal{X}$. Then, an arbitrary state constraint is represented by $f^\top x \leq b, (f,b) \in \text{facet}(\mathcal{X})$.

For the dynamical system (5.18) with filtered disturbance $\mathbf{\Sigma}\widetilde{\mathbf{w}}$, the affine constraint (5.22) parameterizes all achievable closed-loop system responses $(\widetilde{\mathbf{\Phi}}_x, \widetilde{\mathbf{\Phi}}_u)$ such that $\mathbf{x} = \widetilde{\mathbf{\Phi}}_x \widetilde{\mathbf{w}}, \mathbf{u} = \widetilde{\mathbf{\Phi}}_u \widetilde{\mathbf{w}}$ under a linear state feedback controller $\mathbf{u} = \mathbf{K}\mathbf{x}$ [157, Theorem 1]. Plugging in $x_t = \tilde{\Phi}_x^{t,t} x_0 + \sum_{i=1}^{t} \tilde{\Phi}_x^{t,t-i} \tilde{w}_{i-1}$, the tightening of the state constraint is given by

$$f^\top \widetilde{\Phi}_x^{t,t} x_0 + \sum_{i=1}^{t} \|f^\top \widetilde{\Phi}_x^{t,t-i}\|_1 \leq b, \forall (f,b) \in \text{facet}(\mathcal{X}), \quad t = 0, \cdots, T-1 \tag{5.32}$$

which follows from a direct application of the Hölder's inequality and the fact $\|\tilde{w}_t\|_\infty \leq 1$. Similarly, we can tighten the terminal constraint as

$$f^\top \widetilde{\Phi}_x^{T,T} x_0 + \sum_{i=1}^{T} \|f^\top \widetilde{\Phi}_x^{T,T-i}\|_1 \leq b, \ \forall (f,b) \in \text{facet}(\mathcal{X}_T), \tag{5.33}$$

and tighten the control input constraints as

$$f^\top \widetilde{\Phi}_u^{t,t} x_0 + \sum_{i=1}^{t} \|f^\top \widetilde{\Phi}_u^{t,t-i}\|_1 \leq b, \forall (f,b) \in \text{facet}(\mathcal{U}), \quad t = 0, \cdots, T-1. \tag{5.34}$$

5.5.2. Convex tightening of the robust OCP

Finally, we can synthesize a robust state feedback controller $\mathbf{K}$ as shown in the following theorem.

Theorem 5. *Consider the convex quadratic program*

$$\begin{aligned} \underset{\widetilde{\mathbf{\Phi}}_x, \widetilde{\mathbf{\Phi}}_u, \mathbf{\Sigma}}{\text{minimize}} \quad & \left\| \begin{bmatrix} \mathbf{Q}^{1/2} & \\ & \mathbf{R}^{1/2} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{\Phi}}_x(:,0) \\ \widetilde{\mathbf{\Phi}}_u(:,0) \end{bmatrix} x_0 \right\|_2^2 \\ \text{subject to} \quad & \text{affine constraint } (5.22) \\ & \text{uncertainty over-approximation constraint } (5.31) \\ & \text{tightened constraints } (5.32), (5.33), (5.34) \\ & x_0 = x(k) \end{aligned} \tag{5.35}$$

where $\widetilde{\mathbf{\Phi}}_x \in \mathcal{L}_{TV}^{T,n_x \times n_x}$, $\widetilde{\mathbf{\Phi}}_u \in \mathcal{L}_{TV}^{T,n_u \times n_x}$, $\mathbf{\Sigma} \in \mathcal{L}_{TV}^{T,n_x \times n_x}$ *is parameterized in Section 5.4.1, and* $\mathbf{Q} = blkdiag(Q, \cdots, Q)$, $\mathbf{R} = blkdiag(R, \cdots, R)$. *For any feasible solution* $(\widetilde{\mathbf{\Phi}}_x, \widetilde{\mathbf{\Phi}}_u, \mathbf{\Sigma})$ *of the above problem* (5.35)*, the linear state feedback controller* $\mathbf{K} = \widetilde{\mathbf{\Phi}}_u \widetilde{\mathbf{\Phi}}_x^{-1}$ *is feasible for the robust OCP* (5.4) *with polytopic model uncertainty.*

The proof of Theorem 5 directly follows from our derivation of the constraints in (5.35) from the previous sections. Note that since each entry of d_t is lower bounded by $\sigma_w > 0$ from (5.31), any feasible $\mathbf{\Sigma}$ of problem (5.35) is invertible. Some remarks about formulation (5.35) are given as follows.

Remark 5. *The objective function in* (5.35) *is a quadratic nominal function for the system* $\mathbf{x} = Z\hat{\mathbf{A}}\mathbf{x} + Z\hat{\mathbf{B}}\mathbf{u} + \mathbf{\Sigma}\tilde{\mathbf{w}}$ *considering surrogate disturbances where* $\widetilde{\mathbf{\Phi}}_x(:,0)x_0$ *and* $\widetilde{\mathbf{\Phi}}_u(:,0)x_0$ *model the nominal predicted states and control inputs, respectively. It aims to stabilize the closed-loop system to the origin, and it equals to the nominal cost considered in* (5.5) *when the sub-diagonal blocks in the filter* $\mathbf{\Sigma}$ *are restricted to be zero in which case* $\widetilde{\mathbf{\Phi}}_x(:,0) = \mathbf{\Phi}_x(:,0)$ *and* $\widetilde{\mathbf{\Phi}}_u(:,0) = \mathbf{\Phi}_u(:,0)$ *following from* (5.21).

Remark 6. *One important feature of our method is that the robust controller synthesis is decomposed into two steps: uncertainty set over-approximation by a surrogate additive disturbance and constraint tightening. This framework enjoys flexibility to adapt to different uncertainty assumptions. For example, when the actual additive disturbances* w_t *are*

bounded in terms of 2*-norm, i.e.,* $\|w_t\|_2 \leq \sigma_w$*, it is a natural choice to assume our virtual disturbances* $\tilde{w}_t$ *are bounded by* $\|\tilde{w}_t\|_2 \leq 1$ *in* (5.17)*. A similar robust OCP formulation to* (5.35) *can be derived following the same process but using* $\|\cdot\|_2$ *instead. Another main difference will be that the diagonal matrices* $\Sigma^{t,0}$ *of the filter have to be parameterized as* $\Sigma^{t,0} = \sigma_{t-1} I$ *in this case since* $\|\cdot\|_2$ *does not allow processing the constraint* (5.25) *in a row-wise manner.*

5.6. Numerical Examples

We evaluate the performance of our proposed method numerically. All the experiments were implemented in MATLAB R2019b with YALMIP [158] and MOSEK [139] on an Intel i7-6700K CPU.

5.6.1. Test example

We evaluate the conservatism of our proposed method and other baseline methods on a 2-dimensional system adapted from [64]. The main difference from the example in [64] is that we use a simpler model uncertainty assumption, i.e., we only consider uncertainty in one entry of the A and B matrix, to allow varying the uncertainty levels in a wide range and highlighting the differences between different methods.

The system nominal dynamics and problem constraints are given as

$$\begin{aligned} &\hat{A} = \begin{bmatrix} 1 & 0.15 \\ 0.1 & 1 \end{bmatrix}, \quad \hat{B} = \begin{bmatrix} 0.1 \\ 1.1 \end{bmatrix}, \\ &\mathcal{X} = \Big\{ x \in \mathbb{R}^2 | \begin{bmatrix} -8 \\ -8 \end{bmatrix} \leq x \leq \begin{bmatrix} 8 \\ 8 \end{bmatrix} \Big\}, \quad \mathcal{U} = \{ u \in \mathbb{R} | -4 \leq u \leq 4 \}. \end{aligned} \tag{5.36}$$

Then we consider the following polytopic model uncertainty:

$$\Delta_A \in \text{Conv}\Big\{ \begin{bmatrix} \epsilon_A & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} -\epsilon_A & 0 \\ 0 & 0 \end{bmatrix} \Big\}, \quad \Delta_B \in \text{Conv}\Big\{ \begin{bmatrix} 0 \\ \epsilon_B \end{bmatrix}, \begin{bmatrix} 0 \\ -\epsilon_B \end{bmatrix} \Big\} \tag{5.37}$$

and the norm bounded additive disturbances $\|w_t\|_\infty \leq \sigma_w$ for robust MPC. The uncertainty parameters ϵ_A, ϵ_B and σ_w are to be specified. The cost weights are chosen as $Q = 10I, R = 1, Q_T = 10I$. The horizon is chosen as $T = 10$. The robust MPC methods tested in this section are described next.

5.6.2. Overview of robust MPC methods

The focus of this section is on comparing the conservatism and complexity of various representative robust MPC methods that solve the robust OCP. These methods mainly vary in (i) how to parameterize the control policy π_t and (ii) how to handle the polytopic model uncertainty such that the state and input constraints are robustly satisfied under the synthesized policy π_t. In the following, we discuss the tube-based and non-tube-based robust MPC methods separately.

Tube-based robust MPC

Tube-based approaches, which aim to synthesize a robust controller and an associated tube to contain all possible realization of system trajectories, are popular in robust MPC. Tube-based robust MPC typically adopts the following scheme:

1. Parameterize the control policy π_t.

2. Parameterize the tube cross sections $\mathbb{X}_t$.

3. Enforce the tube containment constraints:

$$x_t \in \mathbb{X}_t \Rightarrow x_{t+1} \in \mathbb{X}_{t+1}, \forall (\Delta_A, \Delta_B) \in \mathcal{P}, \forall w_t \in \mathcal{W}, t = 0, \cdots, T-1. \tag{5.38}$$

   and the robustness constraints:

$$\begin{aligned} &\mathbb{X}_t \subseteq \mathcal{X}, \quad u_t = \pi_t(x_t) \in \mathcal{U} \ \forall x_t \in \mathbb{X}_t, t = 0, \cdots, T-1 \\ &x_0 \in \mathbb{X}_0, \quad \mathbb{X}_T \in \mathcal{X}_T. \end{aligned} \tag{5.39}$$

4. Specify the cost function and solve a robust OCP with constraints (5.38) and (5.39).

By induction, it is straight-forward to conclude from constraints (5.38) and (5.39) that the tube $\{\mathbb{X}_t\}_{t=0}^{T}$ contains all possible trajectories under π_t. Steps 1 to 3 in the above scheme jointly determine the conservatism and computational complexity of a tube-based robust MPC method. Next, we review four baseline tube-based robust MPC methods from the literature with distinctive designs.

Tube-A The first method which we denote `Tube-A` is from [59]. In `Tube-A`, a *homothetic* tube [144] is applied:

$$\mathbb{X}_t = z_t + \alpha_t \mathbb{X}_0, t = 1, \cdots, T \tag{5.40}$$

where $z_t \in \mathbb{R}^{n_x}$ and $\alpha_t \in \mathbb{R}_{\geq 0}$. The tube cross sections $\mathbb{X}_t$ at different time instants are restricted to be transitions and dilations of a given polytope $\mathbb{X}_0$.

Let n_J denote the number of vertices of $\mathbb{X}_0$ and $\{x_0^j\}_{j=1}^{n_J}$ denote the vertices of $\mathbb{X}_0$. Since $\mathbb{X}_t$ has the same number of vertices as $\mathbb{X}_0$, we can define $\{x_t^j\}_{j=1}^{n_J}$ as the vertices of $\mathbb{X}_t$ similarly. For controller design, `Tube-MPC` associate a control input u_t^j with each vertex x_t^j of $\mathbb{X}_t$, and parameterize the feedback policy π_t in a barycentric manner:

$$u_t = \pi_t(x_t) = \sum_{j=1}^{n_J} \lambda_j u_t^j, \text{where } \lambda_j \text{ satisfies } x_t = \sum_{j=1}^{n_J} \lambda_j x_t^j, \lambda_j \geq 0, \sum_{j=1}^{n_J} \lambda_j = 1. \tag{5.41}$$

With the homothetic tube and barycentric controller parameterization, the tube containment constraint (5.38) and robustness constraint (5.39) can be enforced by considering the vertex states x_t^j and control inputs u_t^j. In this process, both the vertices of the tube cross section $\mathbb{X}_t$ and the polytopic model uncertainty set $\mathcal{P}$ are enumerated.

Remark 7 (Choice of $\mathbb{X}_0$)**.** *Standard methods to compute a suitable template polytope $\mathbb{X}_0$ include the minimal robust forward invariant set [159], the maximal robust forward invariant set [160], the λ-contractive set [161, Chapter 5.6]. These methods first require computing a robustly stabilizing controller $u_t = Kx_t$ such that for the uncertain system $x_{t+1} = (\hat{A} + \Delta_A)x_t + (\hat{B} + \Delta_B)u_t$ with $(\Delta_A, \Delta_B) \in \mathcal{P}$, which can be achieved by solving an SDP [162].*

Tube-B In [60], the homothetic tube parameterization (5.40) is applied but with a pre-stabilizing controller $\pi_t(x_t) = Kx_t + v_t$. We denote this method `Tube-B`. In this method, the time-invariant feedback gain K is found offline by solving an SDP such that $u_t = Kx_t$ is robustly stabilizing for the uncertain system (5.2) with polytopic model uncertainty. The bias terms v_t of the control inputs are optimized online together with the center z_t and scaling parameter α_k of the cross sections. Constraints (5.38) and (5.39) are enforced using a dual formulation and enumeration of the vertices of each cross section $\mathbb{X}_t$.

Tube-C Based on the constraint tightening method from [163], the authors in [61] propose a tube-based robust MPC method that applies a homothetic tube (5.40) and a pre-stabilizing feedback controller $u_t = Kx_t + v_t$. Different from `Tube-B`, this method enforces the tube containment constraint (5.38) by offline computation of parameters that bound the effects of model uncertainty on the dilation of the cross sections as opposed to using vertex enumeration in `Tube-B`. Furthermore, in [61] the centers of the cross sections $\mathbb{X}_t$ are chosen as the nominal trajectory $\hat{x}_t$ evolved according to $\hat{x}_{t+1} = \hat{A}\hat{x}_t + \hat{B}(K\hat{x}_t + v_t)$; instead, in `Tube-A` and `Tube-B`, the centers z_t of $\mathbb{X}_t$ are treated as free variables. We label the robust MPC method proposed in [61] as `Tube-C`.

Tube-D `Tube-D` [62] uses the pre-stabilizing feedback controller $u_t = Kx_t + v_t$ but parameterizes the cross sections as $\mathbb{X}_t = \{x \in \mathbb{R}^{n_x} \mid Vx \leq \beta_t\}$ where $V \in \mathbb{R}^{r \times n_x}$ denotes a fixed set of hyperplanes that define a polytope and $\beta_t \in \mathbb{R}^r$ is optimized online. This formulation of $\mathbb{X}_t$ is more flexible than the homothetic tube (5.40), but treating β_t as optimization variables give rise to bilinear terms of the form $\Lambda\beta_t$ in enforcing the tube containment and robustness constraints where Λ denotes the dual variables. To eliminate the numerical intractability, the dual variables Λ are fixed offline.

The main features of the above tube-based robust MPC methods are summarized in Table 5.1.

Table 5.1: Summary of tube-based robust MPC methods.

Method	Controller	Tube	Vertex enumeration of tube
`Tube-A` [59]	Barycentric	Homothetic	Yes
`Tube-B` [60]	$Kx_t + v_t$	Homothetic	Yes
`Tube-C` [61]	$Kx_t + v_t$	Homothetic	No
`Tube-D` [62]	$Kx_t + v_t$	Hyperplane	No

Non-tube-based MPC

In robust MPC of systems with only additive disturbances, state feedback and disturbance feedback approaches [148] have been well studied and shown to be effective. Essentially, for the uncertain system $x_{t+1} = \hat{A}x_t + \hat{B}u_t + w_t$, Goulart et al. [148] prove that the linear time-varying (LTV) state feedback controller

$$\underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{T-1} \end{bmatrix}}_{\mathbf{u}} = \underbrace{\begin{bmatrix} K^{0,0} & & & \\ K^{1,1} & K^{1,0} & & \\ \vdots & \ddots & \ddots & \\ K^{T-1,T-1} & \cdots & K^{T-1,1} & K^{T-1,0} \end{bmatrix}}_{\mathbf{K}} \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{T-1} \end{bmatrix}}_{\mathbf{x}} + \underbrace{\begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{T-1} \end{bmatrix}}_{\mathbf{g}} \tag{5.42}$$

is equivalent to the disturbance feedback controller

$$\underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{T-1} \end{bmatrix}}_{\mathbf{u}} = \underbrace{\begin{bmatrix} 0 & & & \\ M^{1,1} & 0 & & \\ \vdots & \ddots & \ddots & \\ M^{T-1,T-1} & \cdots & M^{T-1,1} & 0 \end{bmatrix}}_{\mathbf{M}} \underbrace{\begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{T-1} \end{bmatrix}}_{\mathbf{w}} + \underbrace{\begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{T-1} \end{bmatrix}}_{\mathbf{v}} \tag{5.43}$$

in solving the robust OCP involving the uncertain dynamics with only additive disturbances. Since the feedback gains $\mathbf{K}$ in (5.42) are time-varying and optimized online instead of being fixed offline, the state/disturbance feedback controller parameterization in general gives tighter results than tube MPC using the pre-stabilizing controller $u_t = Kx_t + v_t$ [149].

However, extending the state/disturbance feedback approach to handle both the polytopic model uncertainty and the additive disturbances remains a challenge. We now introduce three robust MPC methods that utilize state/disturbance feedback controllers to solving the robust OCP (5.4).

Lumped-Disturbance-MPC A naive extension of state/disturbance feedback controllers to handle model uncertainties is to uniformly over-approximate the perturbation $\Delta_A x_t + \Delta_B u_t + w_t$ by a norm-bounded disturbance $\bar{w}_t$, and apply standard state/disturbance feedback controller design on the system $x_{t+1} = \hat{A}x_t + \hat{B}u_t + \bar{w}_t$. Since the state constraint $\mathcal{X}$ and input constraint $\mathcal{U}$ are compact sets, an upper bound on $\|\Delta_A x_t + \Delta_B u_t + w_t\|$ can be easily obtained using the triangle inequality and submultiplicativity of norms as shown in [63]. We denote this method `Lumped-Dist-MPC`. Despite its simplicity, the uniform bound on the state and input-dependent perturbation term over the entire state space can be overly conservative.

Offline-Tightening-MPC One key challenge of handling model uncertainty is that the future states and inputs depend on the controller parameters in a complex manner. With an LTV state/disturbance feedback controller shown in (5.42) and (5.43), formulating constraints on the controller parameters such that the robust satisfaction of state and input constraints is guaranteed requires careful characterization of such dependence. In this spirit, Bujarbaruah et al. [64] propose a constraint tightening method for the robust OCP (5.4) such that the disturbance feedback controller parameters $\mathbf{M}$ and $\mathbf{v}$ explicitly control the tightening margin. However, the constraint tightening parameters also heavily rely on offline computed relaxation bounds that hold for all possible open-loop control inputs, which could already become conservative before the disturbance feedback controller is plugged in. We denote this method `Offline-Tightening-MPC`. In the implementation of this method, instead of using vertex enumeration which is intractable for the horizon $T = 10$ considered in our test example, we adopt the computationally cheaper alternative bounds for constraint tightening proposed by the authors [164, Appendix A.4] with an enumeration horizon 3.

SLS-MPC Our proposed method `SLS-MPC` solves the robust OCP (5.4) using an LTV state feedback controller. By System Level Synthesis, `SLS-MPC` transforms the controller design problem into the space of closed-loop system responses where the dependence of effects of uncertainty on the controller parameters is exactly characterized. A novel constraint tightening approach is proposed which over-approximates the perturbation terms by a filtered additive disturbance signal. `SLS-MPC` can jointly optimize the controller parameters and the constraint tightening parameters online.

In the next subsection, we compare the conservatism and computational complexity of the discussed robust MPC methods above on the test example.

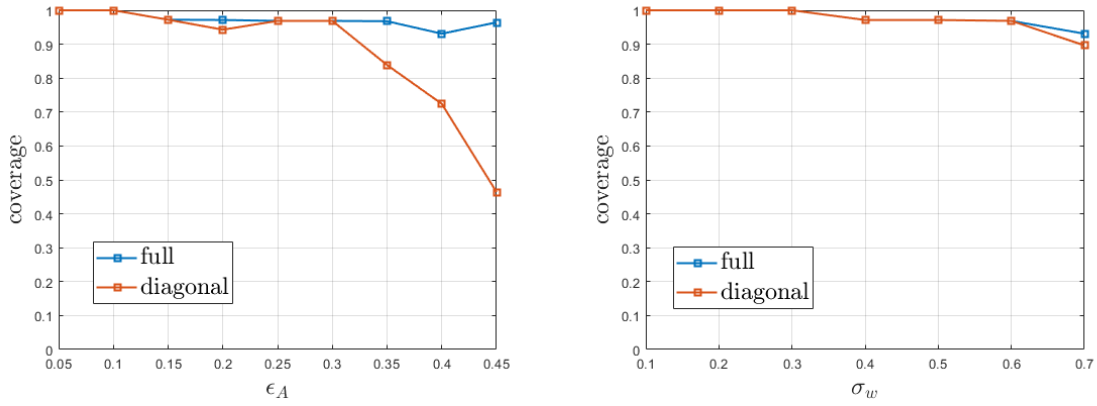
5.6.3. Conservatism and complexity comparison

Evaluation of the filter parameterization in `SLS-MPC`

We first investigate how the parameterization of the filter $\mathbf{\Sigma}$ affects the conservatism of `SLS-MPC`. With the full parameterization, the filter $\mathbf{\Sigma}$ is a block-lower triangular matrix while with the diagonal parameterization, all off diagonal blocks are set zero. Figure 5.2 shows that the full parameterization of the filter reduces the conservatism of `SLS-MPC`. When either the model uncertainty or the additive disturabnces becomes large, the diagonal parameterization of the filter is not sufficient to mitigating the effects of uncertainties. The simulation results are consistent with our analysis shown in Fig. 5.1.

Conservatism comparison of robust MPC methods

We compare the conservatism of our proposed method, `SLS-MPC` and the aforementioned robust MPC methods for varying values of $(\epsilon_A, \epsilon_B, \sigma_w)$. For each fixed $(\epsilon_A, \epsilon_B, \sigma_w)$, we first apply an iterative algorithm [165] to find the maximal robust control invariant set and use it as the terminal set $\mathcal{X}_T$. By construction, $\mathcal{X}_T$ gives the largest feasible domain for any robust MPC algorithm and can be achieved by the exact yet computationally expensive dynamic programming approach [97, Chapter 15]. We do a 10×10 uniform grid search of initial conditions x_0 over $\mathcal{X}_T$ and solve all robust MPC formulations with the sampled x_0 and horizon $T = 10$. The coverage of the feasible domain of each MPC method is given by



(a) $\epsilon_B = 0.1, \sigma_w = 0.1$ (b) $\epsilon_A = 0.1, \epsilon_B = 0.1$.

Figure 5.2: Feasible domain coverage of `SLS-MPC` using the full (blue) and diagonal (orange) parameterization of the filter for different uncertainty parameters.

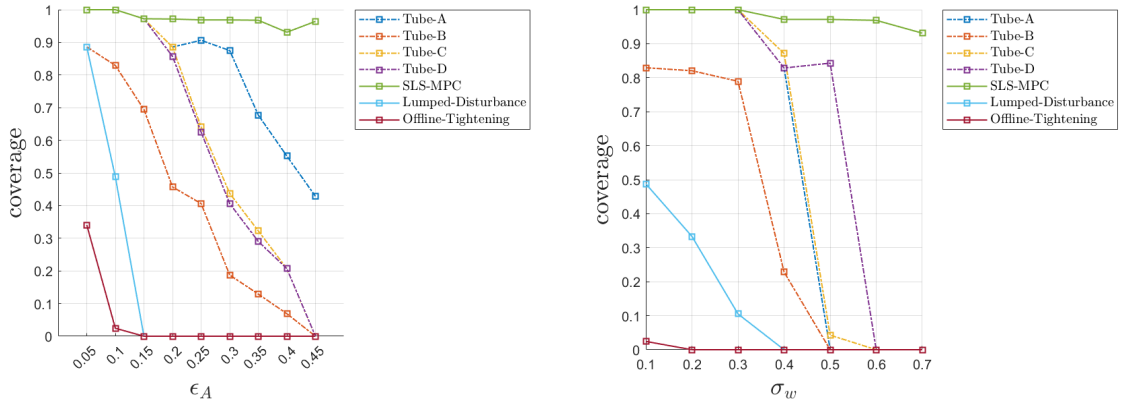
the ratio of sampled feasible initial conditions, and a coverage close to 1 indicates minimal conservatism of the method even compared with the maximal robust control invariant set.

In Figure 5.3, we plot the coverage of the aforementioned RMPC methods under varying $(\epsilon_A, \epsilon_B, \sigma_w)$ values. For all tube-based MPC approaches, we evaluate their coverages for all three constructions of tube cross section discussed in Remark 7 and report the *best coverage*. Note that the design of the tube cross section can significantly affect the performance of tube-based MPC, and the computation of the cross sections in Remark 7 can become numerically challenging when the system dimension increases. The effects of the tube cross sections are reported in the next subsection.

Figure 5.3 shows that `SLS-MPC` consistently outperforms all other methods for a wide range of uncertainty parameters. In all cases it remains a coverage greater than 90%.

Effects of tube cross sections

Figure 5.4 demonstrates the dependence of the conservatism of tube-based robust MPC methods on the choice of the tube cross section. In our experiments, the tube cross section can be chosen as the minimal nominal forward invariant set, maximal robust forward invariant set, or the maximal robust contractive set (see Remark 7 for details). We observe that



(a) $\epsilon_B = 0.1, \sigma_w = 0.1$ (b) $\epsilon_A = 0.1, \epsilon_B = 0.1$.

Figure 5.3: Coverage comparison of different robust MPC methods under different uncertainty parameters. `SLS-MPC` consistently outperforms all other baselines in all cases.

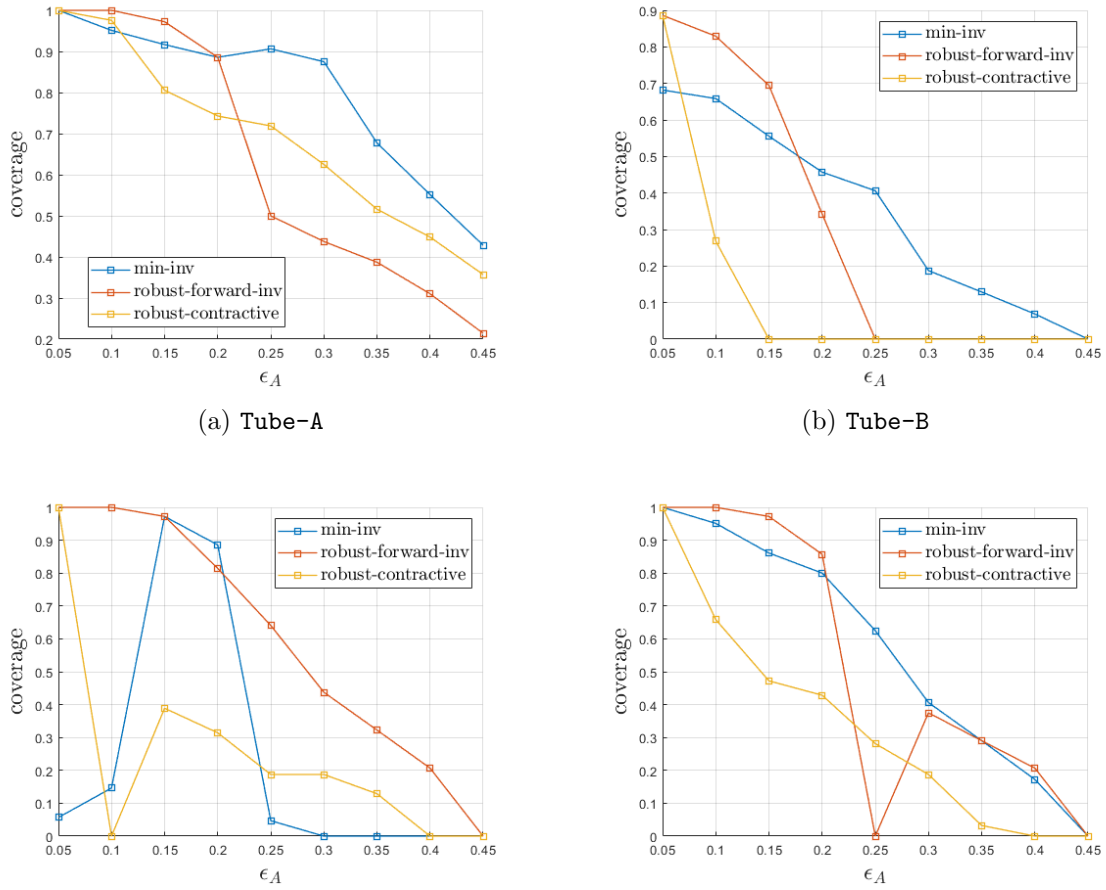
the choice of the cross section can greatly affect the conservatism. It remains an open problem in practice which cross section to use since Figure 5.4 shows that there is no particular choice that consistently performs better than the others for different levels of uncertainties. In the conservatism comparison in Figure 5.3, we adopt the best coverage for tube-based MPC methods.

Solver time comparison

In Figure 5.5, we plot the average solver time of each robust MPC method with varying uncertainty parameters. Data is left blank if a robust MPC method was not feasible on any sampled initial conditions. We observe that `Tube-B` and `Tube-C` normally have an order of magnitude smaller solver time than the other methods. The average solver time of the non-tube-based RMPC methods including `SLS-MPC`, `Lumped-Disturbance-MPC`, `Offline-Tightening-MPC` are similar to that of `Tube-D`. We also observe that the solver time of robust MPC methods can significantly increase when a challenging problem instance (e.g., those with large uncertainty parameters) is encountered.

5.7. Chapter Summary

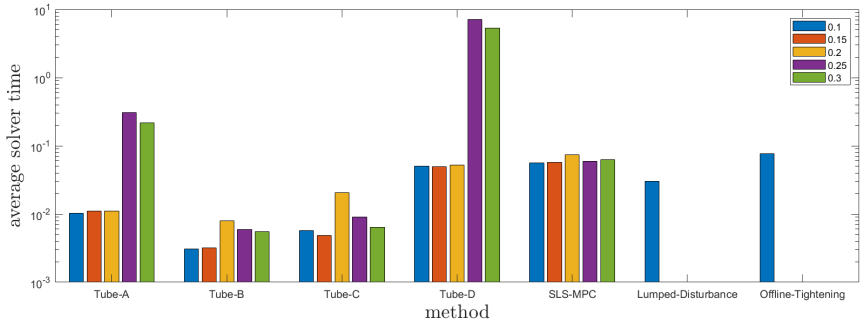
We proposed a novel robust MPC method for uncertain linear systems subject to both polytopic model uncertainty and additive disturbances. Using System Level Synthesis, our



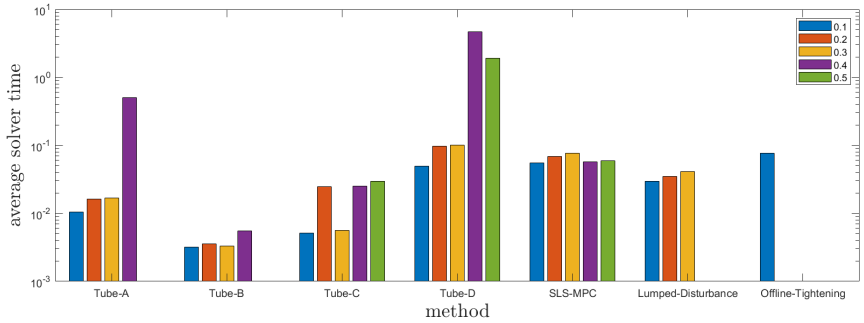
(a) `Tube-A` (b) `Tube-B`

(c) `Tube-C` (d) `Tube-D`

Figure 5.4: The coverage of tube-based robust MPC methods with different choices of the tube cross section. The tube cross section can be chosen as the minimal nominal forward invariant set (min-inv), maximal robust forward invariant set (robust-foward-inv), or the maximal robust contractive set (robust-contractive). See Remark 7 for details. The coverage is reported for $\epsilon_B = 0.1, \sigma_w = 0.1$ and varying sizes of ϵ_A.



(a) Average solver time of each RMPC method with $\epsilon_B = 0.1, \sigma_w = 0.1$ and varying values of ϵ_A as labeled in the legend.



(b) Average solver time of each RMPC method with $\epsilon_A = 0.1, \epsilon_B = 0.1$ and varying values of σ_w as labeled in the legend.

Figure 5.5: Solver time comparison of different robust MPC methods under different uncertainty parameters.

method optimizes in the space of closed-loop system responses which allows efficient over-approximation of the effects of uncertainties. Numerical examples demonstrate that our method can be almost as tight as the exact dynamic programming approach for robust MPC, and significantly outperforms both tube-based and non-tube-based robust MPC methods in terms of conservatism.

CHAPTER 6

Conclusion and Open Problems

In this dissertation, we studied how to provide formal guarantees on neural network dynamical systems. Our methodology is combining control theoretical tools and specialized optimization algorithm design such that it is tractable or efficient to analyze or control a neural network dynamical system.

From Chapter 2 to Chapter 4, we solved a hierarchy of analysis problems regarding properties of neural networks in isolation, reachability and stability of dynamical systems with neural networks in the loop. Neural network verification considers the fundamental problem of providing formal guarantees on a neural network. As discussed in Chapter 2, there is a rich set of tools available for neural network verification which can be used as building blocks for reachability and stability analysis of neural network dynamical systems. Neural network verification remains an active area of research since there is a constant need to develop more scalable and tighter methods to handle neural networks with ever growing sizes that are deployed in the real world. In this spirit, our work in Chapter 2 makes a relatively tight linear program-based verification method more scalable through operator splitting. Other paths towards scalable and effective neural network verifiers are also viable. The relative strengths and weaknesses of our proposed method and the baselines are illustrated through the numerical examples in Chapter 2.

Chapter 3 and 4 respectively demonstrate how reachability and stability analysis of neural network dynamical systems can be formulated and solved using neural network verification tools. Compared with the numerous works in neural network verification, the design of high level reachability analysis algorithms such as the one-shot method in Chapter 3 and solving neural network verification problems with nonlinear objectives as shown in Chapter 4 have received less attention. Our works contributed in these directions which are particularly useful for dynamical system analysis.

In Chapter 5, we developed a novel robust model predictive control method using a state feedback controller and achieved significant conservatism reduction compared with existing baselines. This opens the door for combining robust model predictive control and neural network verification tools for safe control of neural network dynamical systems. By obtaining local sound bounds on the neural network outputs, we can treat the neural network dynamics in a system as an uncertainty function with simplified bounds amenable to robust model predictive control. This will be an interesting direction for future research.

While in this dissertation we primarily focused on providing deterministic guarantees, the analysis of machine learning algorithms is statistical in nature such as the sample complexity and generalization analysis. The deterministic guarantees can be challenging to obtain when neural networks are updated online to operate in a dynamic environment, or when diverse learning models are deployed in addition to neural networks. Establishing a theoretical framework that allows closed-loop analysis of learning-enabled systems through the statistical properties of each learning modules in the loop will be useful for many real world applications.

BIBLIOGRAPHY

[1] Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy Mann, and Pushmeet Kohli. A dual approach to scalable verification of deep networks. *Proceedings of the 34th Annual Conference on Uncertainty in Artificial Intelligence*, 2018.

[2] Rudy Bunel, Jingyue Lu, Ilker Turkaslan, Pushmeet Kohli, P Torr, and P Mudigonda. Branch and bound for piecewise linear neural network verification. *Journal of Machine Learning Research*, 21(2020), 2020.

[3] Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J Zico Kolter. Scaling provable adversarial defenses. *Advances in Neural Information Processing Systems*, 31, 2018.

[4] Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond. *Advances in Neural Information Processing Systems*, 33:1129–1141, 2020.

[5] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

[6] Jaume Giné and Claudia Valls. On the dynamics of the rayleigh–duffing oscillator. *Nonlinear Analysis: Real World Applications*, 45:309–319, 2019.

[7] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[8] Harry A Pierson and Michael S Gashler. Deep learning in robotics: a review of recent research. *Advanced Robotics*, 31(16):821–835, 2017.

[9] Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721, 2021.

[10] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018.

[11] Jemin Hwangbo, Inkyu Sa, Roland Siegwart, and Marco Hutter. Control of a quadrotor with reinforcement learning. *IEEE Robotics and Automation Letters*, 2(4):2096–2103, 2017.

[12] Ravi Teja Mullapudi, William R Mark, Noam Shazeer, and Kayvon Fatahalian. Hydranets: Specialized dynamic architectures for efficient inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8080–8089, 2018.

[13] Mayank Bansal, Alex Krizhevsky, and Abhijit S. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. In Antonio Bicchi, Hadas Kress-Gazit, and Seth Hutchinson, editors, *Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, June 22-26, 2019*, 2019.

[14] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[15] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[16] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[18] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[22] Hyeon-Joong Yoo. Deep convolution neural networks in computer vision: a review. *IEIE Transactions on Smart Processing and Computing*, 4(1):35–43, 2015.

[23] Li Deng and Yang Liu. *Deep learning in natural language processing*. Springer, 2018.

[24] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Na-*

ture, 596(7873):583–589, 2021.

[25] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[26] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International conference on computer aided verification*, pages 97–117. Springer, 2017.

[27] Vincent Tjeng, Kai Yuanqing Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[28] Radoslav Ivanov, James Weimer, Rajeev Alur, George J Pappas, and Insup Lee. Verisig: verifying safety properties of hybrid systems with neural network controllers. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 169–178, 2019.

[29] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5286–5295. PMLR, 2018.

[30] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Efficient formal safety analysis of neural networks. *Advances in Neural Information Processing Systems*, 31, 2018.

[31] Gagandeep Singh, Rupanshu Ganvir, Markus Püschel, and Martin Vechev. Beyond the single neuron convex barrier for neural network certification. *Advances in Neural Information Processing Systems*, 32, 2019.

[32] Alessandro De Palma, Harkirat S Behl, Rudy Bunel, Philip Torr, and M Pawan Kumar. Scaling the convex barrier with active sets. In *Proceedings of the ICLR 2021 Conference*. Open Review, 2021.

[33] Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. Towards fast computation of certified robustness for relu networks. In *International Conference on Machine Learning*, pages 5276–5285. PMLR, 2018.

[34] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. *Advances in neural information processing systems*, 31, 2018.

[35] Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems*, pages 10900–10910, 2018.

[36] Mahyar Fazlyab, Manfred Morari, and George J Pappas. Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. *IEEE Transactions on Automatic Control*, 2020.

[37] Rudy R Bunel, Ilker Turkaslan, Philip Torr, Pushmeet Kohli, and Pawan K Mudigonda. A unified view of piecewise linear neural network verification. *Advances in Neural Information Processing Systems*, 31, 2018.

[38] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. *Advances in Neural Information Processing Systems*, 34:29909–29921, 2021.

[39] Alessandro De Palma, Rudy Bunel, Alban Desmaison, Krishnamurthy Dvijotham, Pushmeet Kohli, Philip HS Torr, and M Pawan Kumar. Improved branch and bound for neural network verification via lagrangian decomposition. *arXiv preprint arXiv:2104.06718*, 2021.

[40] Weiming Xiang, Hoang-Dung Tran, and Taylor T Johnson. Output reachable set estimation and verification for multilayer neural networks. *IEEE transactions on neural networks and learning systems*, 29(11):5777–5783, 2018.

[41] Brendon G Anderson, Ziye Ma, Jingqi Li, and Somayeh Sojoudi. Tightened convex relaxations for neural network robustness certification. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 2190–2197. IEEE, 2020.

[42] Shaoru Chen, Eric Wong, J Zico Kolter, and Mahyar Fazlyab. Deepsplit: Scalable verification of deep neural networks via operator splitting. *IEEE Open Journal of Control Systems*, 1:126–140, 2022.

[43] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[44] Michael Akintunde, Alessio Lomuscio, Lalit Maganti, and Edoardo Pirovano. Reachability analysis for neural agent-environment systems. In *Sixteenth international conference on principles of knowledge representation and reasoning*, 2018.

[45] Chao Huang, Jiameng Fan, Wenchao Li, Xin Chen, and Qi Zhu. Reachnn: Reachability analysis of neural-network controlled systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(5s):1–22, 2019.

[46] Niklas Kochdumper, Christian Schilling, Matthias Althoff, and Stanley Bak. Open- and closed-loop neural network verification using polynomial zonotopes. *arXiv preprint arXiv:2207.02715*, 2022.

[47] Yuhao Zhang and Xiangru Xu. Safety verification of neural feedback systems based on constrained zonotopes. *arXiv preprint arXiv:2204.00903*, 2022.

[48] Michael Everett, Golnaz Habibi, and Jonathan P How. Efficient reachability analysis of closed-loop systems with neural network controllers. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4384–4390. IEEE, 2021.

[49] Haimin Hu, Mahyar Fazlyab, Manfred Morari, and George J Pappas. Reach-sdp: Reachability analysis of closed-loop systems with neural network controllers via semidefinite programming. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 5929–5934. IEEE, 2020.

[50] Chelsea Sidrane, Amir Maleki, Ahmed Irfan, and Mykel J Kochenderfer. Overt: An algorithm for safety verification of neural network control policies for nonlinear systems. *Journal of Machine Learning Research*, 23(117):1–45, 2022.

[51] Benjamin Karg and Sergio Lucia. Stability and feasibility of neural network-based controllers via output range analysis. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 4947–4954. IEEE, 2020.

[52] Eduardo Sontag. Nonlinear regulation: The piecewise linear approach. *IEEE Transactions on automatic control*, 26(2):346–358, 1981.

[53] Pratik Biswas, Pascal Grieder, Johan Löfberg, and Manfred Morari. A survey on stability analysis of discrete-time piecewise affine systems. *IFAC Proceedings Volumes*, 38(1):283–294, 2005.

[54] Hai Lin and Panos J Antsaklis. Stability and stabilizability of switched linear systems: a survey of recent results. *IEEE Transactions on Automatic control*, 54(2):308–322, 2009.

[55] Zhendong Sun. Stability of piecewise linear systems revisited. *Annual Reviews in Control*, 34(2):221–231, 2010.

[56] Razvan Pascanu, Guido Montufar, and Yoshua Bengio. On the number of response regions of deep feed forward networks with piece-wise linear activations. *arXiv preprint arXiv:1312.6098*, 2013.

[57] Alberto Bemporad and Manfred Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[58] Arjan J van der Schaft and Johannes Maria Schumacher. Complementarity modeling of hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):483–490, 1998.

[59] Wilbur Langson, Ioannis Chryssochoos, SV Raković, and David Q Mayne. Robust model predictive control using tubes. *Automatica*, 40(1):125–133, 2004.

[60] Matthias Lorenzen, Mark Cannon, and Frank Allgöwer. Robust mpc with recursive model update. *Automatica*, 103:461–471, 2019.

[61] Johannes Köhler, Elisa Andina, Raffaele Soloperto, Matthias A Müller, and Frank Allgöwer. Linear robust adaptive model predictive control: Computational complexity and conservatism. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 1383–1388. IEEE, 2019.

[62] Xiaonan Lu and Mark Cannon. Robust adaptive tube model predictive control. In *2019 American Control Conference (ACC)*, pages 3695–3701. IEEE, 2019.

[63] Monimoy Bujarbaruah, Ugo Rosolia, Yvonne R Stürz, and Francesco Borrelli. A simple robust mpc for linear systems with parametric and additive uncertainty. In *2021 American Control Conference (ACC)*, pages 2108–2113. IEEE, 2021.

[64] Monimoy Bujarbaruah, Ugo Rosolia, Yvonne R Stürz, Xiaojing Zhang, and Francesco Borrelli. Robust mpc for lpv systems via a novel optimization-based constraint tightening. *Automatica*, 143:110459, 2022.

[65] James Anderson, John C Doyle, Steven H Low, and Nikolai Matni. System level synthesis. *Annual Reviews in Control*, 47:364–393, 2019.

[66] Alessio Lomuscio and Lalit Maganti. An approach to reachability analysis for feedforward relu neural networks. *arXiv preprint arXiv:1706.07351*, 2017.

[67] Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. Maximum resilience of artificial neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 251–268. Springer, 2017.

[68] Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. Output range analysis for deep feedforward neural networks. In *NASA Formal Methods Symposium*, pages 121–138. Springer, 2018.

[69] Matteo Fischetti and Jason Jo. Deep neural networks and mixed integer linear optimization. *Constraints*, 23(3):296–309, 2018.

[70] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[71] Rudy Bunel, Alessandro De Palma, Alban Desmaison, Krishnamurthy Dvijotham, Pushmeet Kohli, Philip Torr, and M Pawan Kumar. Lagrangian decomposition for neural network verification. In *Conference on Uncertainty in Artificial Intelligence*, pages 370–379. PMLR, 2020.

[72] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Duane Boning, and Cho-Jui Hsieh. Robust deep reinforcement learning against adversarial perturbations on observations. *Advances in Neural Information Processing Systems*, 2020.

[73] Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. A convex relaxation barrier to tight robustness verification of neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[74] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279, 2008.

[75] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. In *International Conference on Learning Representations*, 2020.

[76] Ruediger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 269–286. Springer, 2017.

[77] BS He, Hai Yang, and SL Wang. Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities. *Journal of Optimization Theory and applications*, 106(2):337–356, 2000.

[78] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003.

[79] Bingsheng He and Xiaoming Yuan. On the o(1/n) convergence rate of the douglas–rachford alternating direction method. *SIAM Journal on Numerical Analysis*, 50(2):700–709, 2012.

[80] Sumanth Dathathri, Krishnamurthy Dvijotham, Alexey Kurakin, Aditi Raghunathan, Jonathan Uesato, Rudy R Bunel, Shreya Shankar, Jacob Steinhardt, Ian Goodfellow, Percy S Liang, et al. Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. *Advances in Neural Information Processing Systems*, 33:5318–5331, 2020.

[81] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In

International Conference on Learning Representations, 2018.

[82] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, 2020.

[83] Sergio Lucia, Alexandru Tătulea-Codrean, Christian Schoppmeyer, and Sebastian Engell. Rapid development of modular and sustainable nonlinear model predictive control solutions. *Control Engineering Practice*, 60:51–62, 2017.

[84] Ching-Yun Ko, Zhaoyang Lyu, Lily Weng, Luca Daniel, Ngai Wong, and Dahua Lin. Popqorn: Quantifying robustness of recurrent neural networks. In *International Conference on Machine Learning*, pages 3468–3477. PMLR, 2019.

[85] Wonryong Ryou, Jiayu Chen, Mislav Balunovic, Gagandeep Singh, Andrei Dan, and Martin Vechev. Scalable polyhedral verification of recurrent neural networks. In *International Conference on Computer Aided Verification*, pages 225–248. Springer, 2021.

[86] Sara Mohammadinejad, Brandon Paulsen, Jyotirmoy V Deshmukh, and Chao Wang. Diffrnn: Differential verification of recurrent neural networks. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 117–134. Springer, 2021.

[87] Weiming Xiang, Hoang-Dung Tran, Joel A Rosenfeld, and Taylor T Johnson. Reachable set estimation and safety verification for piecewise linear systems with neural network controllers. In *2018 Annual American Control Conference (ACC)*, pages 1574–1579. IEEE, 2018.

[88] Michael Everett, Golnaz Habibi, Chuangchuang Sun, and Jonathan P How. Reachability analysis of neural feedback loops. *IEEE Access*, 9:163938–163953, 2021.

[89] Hoang-Dung Tran, Feiyang Cai, Manzanas Lopez Diego, Patrick Musau, Taylor T Johnson, and Xenofon Koutsoukos. Safety verification of cyber-physical systems with reinforcement learning control. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(5s):1–22, 2019.

[90] Christian Schilling, Marcelo Forets, and Sebastián Guadalupe. Verification of neural-network control systems by integrating taylor models and zonotopes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8169–8177, 2022.

[91] Souradeep Dutta, Xin Chen, and Sriram Sankaranarayanan. Reachability analysis for neural feedback systems using regressive polynomial rule inference. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 157–168, 2019.

[92] Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. Learning and verification of feedback control systems using feedforward neural networks. *IFAC-PapersOnLine*, 51(16):151–156, 2018.

[93] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–30, 2019.

[94] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and effective robustness certification. *Advances in neural information processing systems*, 31, 2018.

[95] Razvan V Florian. Correct equations for the dynamics of the cart-pole system. *Center for Cognitive and Neural Studies (Coneural), Romania*, 2007.

[96] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022.

[97] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[98] WPMH Heemels, Johannes M Schumacher, and S Weiland. Linear complementarity systems. *SIAM journal on applied mathematics*, 60(4):1234–1269, 2000.

[99] Wilhemus PMH Heemels, Bart De Schutter, and Alberto Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091, 2001.

[100] Mikael K-J Johansson. *Piecewise linear control systems: a computational approach*, volume 284. Springer, 2003.

[101] Mikael Johansson and Anders Rantzer. Computation of piecewise quadratic lyapunov functions for hybrid systems. In *1997 European Control Conference (ECC)*, pages 2005–2010. IEEE, 1997.

[102] Stephen Prajna and Antonis Papachristodoulou. Analysis of switched and hybrid systems-beyond piecewise quadratic methods. In *Proceedings of the 2003 American Control Conference, 2003.*, volume 4, pages 2779–2784. IEEE, 2003.

[103] Armando Solar-Lezama, Liviu Tancau, Rastislav Bodik, Sanjit Seshia, and Vijay Saraswat. Combinatorial sketching for finite programs. In *Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, pages 404–415, 2006.

[104] Armando Solar-Lezama and Rastislav Bodik. *Program synthesis by sketching*. Citeseer, 2008.

[105] Daniele Ahmed, Andrea Peruffo, and Alessandro Abate. Automated and sound synthesis of lyapunov functions with smt solvers. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 97–114. Springer, 2020.

[106] Alessandro Abate, Daniele Ahmed, Mirco Giacobbe, and Andrea Peruffo. Formal synthesis of lyapunov neural networks. *IEEE Control Systems Letters*, 2020.

[107] James Kapinski, Jyotirmoy V Deshmukh, Sriram Sankaranarayanan, and Nikos Arechiga. Simulation-guided lyapunov analysis for hybrid dynamical systems. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pages 133–142, 2014.

[108] Hadi Ravanbakhsh and Sriram Sankaranarayanan. Learning control lyapunov functions from counterexamples and demonstrations. *Autonomous Robots*, 43(2):275–307, 2019.

[109] I. Erlich S. Tarasov, L. G. Khachiyan. The method of inscribed ellipsoids. *Soviet Mathematics Doklady*, 37, 1988.

[110] Lars Lindemann, Haimin Hu, Alexander Robey, Hanwen Zhang, Dimos Dimarogonas, Stephen Tu, and Nikolai Matni. Learning hybrid control barrier functions from data. In *Conference on Robot Learning*, 2020.

[111] Tobia Marcucci and Russ Tedrake. Mixed-integer formulations for optimal control of piecewise-affine systems. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 230–239, 2019.

[112] Juan Pablo Vielma. Mixed integer linear programming formulation techniques. *Siam Review*, 57(1):3–57, 2015.

[113] Thomas Kleinert, Martine Labbé, Fränk Plein, and Martin Schmidt. Technical note—there's no free lunch: On the hardness of choosing a correct big-m in bilevel optimization. *Operations Research*.

[114] Daniel Simon and Johan Löfberg. Stability analysis of model predictive controllers using mixed integer linear programming. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 7270–7275. IEEE, 2016.

[115] Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Advances in Neural Information Processing Systems*, pages 2266–2276, 2017.

[116] Wassim M Haddad and VijaySekhar Chellaboina. *Nonlinear dynamical systems and control: a Lyapunov-based approach*. Princeton university press, 2011.

[117] Dirk Aeyels and Joan Peuteman. A new asymptotic stability criterion for nonlinear time-variant differential equations. *IEEE Transactions on automatic control*, 43(7):968–971, 1998.

[118] Ruxandra Bobiti and Mircea Lazar. A sampling approach to finding lyapunov functions for nonlinear discrete-time systems. In *2016 European Control Conference (ECC)*, pages 561–566. IEEE, 2016.

[119] Amir Ali Ahmadi and Pablo A Parrilo. Non-monotonic lyapunov functions for stability of discrete time nonlinear and switched systems. In *2008 47th IEEE Conference on Decision and Control*, pages 614–621. IEEE, 2008.

[120] David S Atkinson and Pravin M Vaidya. A cutting plane algorithm for convex programming that uses analytic centers. *Mathematical Programming*, 69(1-3):1–43, 1995.

[121] Jack Elzinga and Thomas G Moore. A central cutting plane algorithm for the convex programming problem. *Mathematical Programming*, 8(1):134–145, 1975.

[122] Stephen Boyd and Lieven Vandenberghe. Localization and cutting-plane methods. *From Stanford EE 364b lecture notes*, 2007.

[123] A. Yu. Levin. An algorithm for minimizing convex functions. *Soviet Mathematics Doklady*, 160:1244–1247, 1965.

[124] Leonid G Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

[125] D.B. Yudin A.S. Nemirovskii. Problem complexity and method efficiency in optimization. 1983.

[126] Jean-Louis Goffin and Jean-Philippe Vial. On the computation of weighted analytic centers and dual ellipsoids with the projective algorithm. *Mathematical Programming*, 60(1-3):81–92, 1993.

[127] Yu Nesterov. Cutting plane algorithms from analytic centers: efficiency estimates. *Mathematical Programming*, 69(1):149–176, 1995.

[128] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[129] Yinyu Ye. A potential reduction algorithm allowing column generation. *SIAM Journal on Optimization*, 2(1):7–20, 1992.

[130] Zhi-Quan Luo and Jie Sun. A polynomial cutting surfaces algorithm for the convex feasibility problem defined by self-concordant inequalities. *Computational Optimiza-*

tion and Applications, 15(2):167–191, 2000.

[131] Jean-Louis Goffin, Zhi-Quan Luo, and Yinyu Ye. Complexity analysis of an interior cutting plane method for convex feasibility problems. *SIAM Journal on Optimization*, 6(3):638–652, 1996.

[132] Jie Sun, Kim-Chuan Toh, and Gongyun Zhao. An analytic center cutting plane method for semidefinite feasibility problems. *Mathematics of Operations Research*, 27(2):332–346, 2002.

[133] Pietro Belotti, Christian Kirches, Sven Leyffer, Jeff Linderoth, James Luedtke, and Ashutosh Mahajan. Mixed-integer nonlinear optimization. *Acta Numerica*, 22:1, 2013.

[134] Stefan Vigerske. Decomposition in multistage stochastic programming and a constraint integer programming approach to mixed-integer nonlinear programming. 2013.

[135] Mohit Tawarmalani and Nikolaos V Sahinidis. *Convexification and global optimization in continuous and mixed-integer nonlinear programming: theory, algorithms, software, and applications*, volume 65. Springer Science & Business Media, 2013.

[136] Pietro Belotti, Jon Lee, Leo Liberti, Francois Margot, and Andreas Wächter. Branching and bounds tighteningtechniques for non-convex minlp. *Optimization Methods & Software*, 24(4-5):597–634, 2009.

[137] Laurence A Wolsey and George L Nemhauser. *Integer and combinatorial optimization*, volume 55. John Wiley & Sons, 1999.

[138] M. Herceg, M. Kvasnica, C.N. Jones, and M. Morari. Multi-Parametric Toolbox 3.0. In *Proc. of the European Control Conference*, pages 502–510, Zürich, Switzerland, July 17–19 2013. http://control.ee.ethz.ch/~mpt.

[139] MOSEK ApS. *The MOSEK optimization toolbox for Python manual. Version 9.2*, 2020.

[140] Tobia Marcucci, Robin Deits, Marco Gabiccini, Antonio Bicchi, and Russ Tedrake. Approximate hybrid model predictive control for multi-contact push recovery in complex environments. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 31–38. IEEE, 2017.

[141] Francois Chollet et al. Keras, 2015. https://github.com/fchollet/keras.

[142] David Q Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50(12):2967–2986, 2014.

[143] David Q Mayne, María M Seron, and SV Raković. Robust model predictive control

of constrained linear systems with bounded disturbances. *Automatica*, 41(2):219–224, 2005.

[144] Saša V Raković, Basil Kouvaritakis, Rolf Findeisen, and Mark Cannon. Homothetic tube model predictive control. *Automatica*, 48(8):1631–1638, 2012.

[145] Saša V Raković, William S Levine, and Behçet Açikmese. Elastic tube model predictive control. In *2016 American Control Conference (ACC)*, pages 3594–3599. IEEE, 2016.

[146] Saša V Rakovic, Basil Kouvaritakis, Mark Cannon, Christos Panos, and Rolf Findeisen. Parameterized tube model predictive control. *IEEE Transactions on Automatic Control*, 57(11):2746–2761, 2012.

[147] Johan Lofberg. Approximations of closed-loop minimax mpc. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, volume 2, pages 1438–1442. IEEE, 2003.

[148] Paul J Goulart, Eric C Kerrigan, and Jan M Maciejowski. Optimization over state feedback policies for robust control with constraints. *Automatica*, 42(4):523–533, 2006.

[149] Jerome Sieber, Samir Bennani, and Melanie N Zeilinger. A system level approach to tube-based model predictive control. *IEEE Control Systems Letters*, 2021.

[150] Mayuresh V Kothare, Venkataramanan Balakrishnan, and Manfred Morari. Robust constrained model predictive control using linear matrix inequalities. *Automatica*, 32(10):1361–1379, 1996.

[151] J Schuurmans and JA Rossiter. Robust predictive control using tight sets of predicted states. *IEE proceedings-Control theory and applications*, 147(1):13–18, 2000.

[152] Basil Kouvaritakis, John Anthony Rossiter, and Jan Schuurmans. Efficient robust predictive control. *IEEE Transactions on automatic control*, 45(8):1545–1549, 2000.

[153] Diego Muñoz-Carpintero, Mark Cannon, and Basil Kouvaritakis. Robust mpc strategy with optimized polytopic dynamics for linear systems with additive and multiplicative uncertainty. *Systems & Control Letters*, 81:34–41, 2015.

[154] Rodrigo S Gesser, Daniel M Lima, and Julio E Normey-Rico. Robust model predictive control: Implementation issues with comparative analysis. *IFAC-PapersOnLine*, 51(25):478–483, 2018.

[155] James Fleming, Basil Kouvaritakis, and Mark Cannon. Robust tube mpc for linear systems with multiplicative uncertainty. *IEEE Transactions on Automatic Control*, 60(4):1087–1092, 2014.

[156] Hassan K Khalil. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, NJ, 2002.

[157] Shaoru Chen, Nikolai Matni, Manfred Morari, and Victor M. Preciado. System level synthesis-based robust model predictive control through convex inner approximation. In *arXiv:2111.05509*, 2021.

[158] Johan Lofberg. Yalmip: A toolbox for modeling and optimization in matlab. In *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*, pages 284–289. IEEE, 2004.

[159] SV Rakovic, KI Kouramas, EC Kerrigan, JC Allwright, and DQ Mayne. The minimal robust positively invariant set for linear difference inclusions and its robust positively invariant approximations. *Automatica*, 2005.

[160] Bert Pluymers, John A Rossiter, Johan AK Suykens, and Bart De Moor. The efficient computation of polyhedral invariant sets for linear systems with polytopic uncertainty. In *Proceedings of the 2005, American control conference, 2005.*, pages 804–809. IEEE, 2005.

[161] Saša V Raković. Model predictive control: classical, robust, and stochastic [bookshelf]. *IEEE Control Systems Magazine*, 36(6):102–105, 2016.

[162] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear matrix inequalities in system and control theory*. SIAM, 1994.

[163] Johannes Köhler, Raffaele Soloperto, Matthias A Müller, and Frank Allgöwer. A computationally efficient robust model predictive control framework for uncertain nonlinear systems. *IEEE Transactions on Automatic Control*, 66(2):794–801, 2020.

[164] Monimoy Bujarbaruah, Ugo Rosolia, Yvonne R Stürz, Xiaojing Zhang, and Francesco Borrelli. Robust mpc for linear systems with parametric and additive uncertainty: A novel constraint tightening approach. *arXiv preprint arXiv:2007.00930*, 2020.

[165] Pascal Grieder, Pablo A Parrilo, and Manfred Morari. Robust receding horizon control-analysis & synthesis. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, volume 1, pages 941–946. IEEE, 2003.