Safe Control with Neural Network Dynamic Models

Tianhao Wei TWEI2@ANDREW.CMU.EDU
Carnegie Mellon University

Changliu Liu CLIU6@ANDREW.CMU.EDU
Carnegie Mellon University

Editors: R. Firoozi, N. Mehr, E. Yel, R. Antonova, J. Bohg, M. Schwager, M. Kochenderfer

Abstract

Safety is critical in autonomous robotic systems. A safe control law should ensure forward invariance of a safe set (a subset in the state space). It has been extensively studied regarding how to derive a safe control law with a control-affine analytical dynamic model. However, how to formally derive a safe control law with Neural Network Dynamic Models (NNDM) remains unclear due to the lack of computationally tractable methods to deal with these black-box functions. In fact, even finding a control that minimizes an objective for NNDM without any safety constraint is still challenging. In this work, we propose MIND-SIS (Mixed Integer for Neural network Dynamic models with Safety Index Synthesis), the first method to synthesize safe control for NNDM. The method includes two parts: 1) SIS: an algorithm for the offline synthesis of the safety index (also called as a barrier function) using evolutionary methods and 2) MIND: an algorithm that computes the optimal safe control input online by solving a constrained optimization with a computationally efficient encoding of neural networks. It has been theoretically proved that MIND-SIS guarantees forward invariance and finite-time convergence to a subset of the user-defined safe set. It has also been numerically validated that MIND-SIS achieves optimal safe control of NNDM with less than 10^{-8} optimality gap and zero safety constraint violation.

Keywords: safe control, neural network dynamic model

1. Introduction

Robot safety depends on the correct functioning of all system components, such as accurate perception, safe motion planning, and safe control. Safe control, as the last defense of system safety, has been widely studied in the context of dynamical systems Nagumo (1942); Blanchini (1999). A safe control law ensures the forward invariance of a subset inside the user-defined safety constraint, meaning that any agent entering that subset will remain in it. There are many methods to derive the safe control laws for control-affine analytical dynamic model Wei and Liu (2019); Liu and Tomizuka (2014). However, constructing such an analytical dynamic model for complex systems can be difficult, time-consuming, and sometimes impossible Nguyen-Tuong and Peters (2011). Recent works adopt data-driven approaches to learn these dynamic models, and most of the learned models are encoded in neural networks, e.g. virtual world models of video games or dynamic models of a robot, etc. Nagabandi et al. (2018); Janner et al. (2019). Although neural network dynamic models (NNDMs) can greatly alleviate human efforts in modeling, they are less interpretable than analytical models. It is more challenging to derive control laws, especially safe control laws, for these NNDMs than for analytical models.

This paper focuses on safe tracking tasks with NNDMs, which is formulated as a constrained optimization that minimizes the state tracking error given the safety constraint and the neural network dynamics constraint. Even without the safety constraint, the tracking control with NNDMs is already challenging. Since NNDMs are complex and highly nonlinear, there is no computationally efficient method to compute its model inverse, which is required by most existing white-box methods Tolani et al. (2000). On the other hand, black-box methods, such as the shooting method which chooses control from randomly generated candidates, can not guarantee to find the optimal solution in finite time. Moreover, the safety constraint adds another layer of difficulty to the problem. The robot should select an action that not only satisfies the safety constraint at the current time step, but also ensures that in the future, the agent will not enter any state where no action is safe. This property is called *persistent feasibility*. To ensure persistent feasibility, we need to compute the control invariant set inside the original user-specified safety constraint and constrain the robot motion in this more restrictive control invariant set. For an analytical model, we can manually craft this control invariant set to meet the requirement Liu and Tomizuka (2014) based on our understanding of the dynamics. The same task becomes difficult for NNDM due to its poor interpretability.

In this work, we address these challenges by introducing an integrated method, mixed integer for neural network dynamic models with safety index synthesis (MIND-SIS), to handle both the offline synthesis of the control invariant set and the online computation of the constrained optimization with NNDM constraints. First, inspired by an algorithm for neural network verification Tjeng et al. (2017), we use mixed integer programming (MIP) to encode the NNDM constraint, which greatly reduces the complexity of the optimization problem. Importantly, the MIP method is complete and guarantees optimality. Second, to synthesize the control invariant set, we use evolutionary algorithms to optimize a parameterized safety index. Using the learned safety index, the resulting control solved by the constrained optimization will ensure persistent feasibility and hence forward invariance inside the user-specified safety constraint.

The remaining of the paper is organized as follows. Section 2 provides a formal description of the problem and introduces notations. Section 3 introduces prior works on safe control, NNDM, and neural network verification, which inspire our method. Section 4 discusses the proposed method in detail. Section 5 shows experimental results that validate our method. And section 6 discusses possible future directions. Additional results and discussions can be found in the appendix in the arxiv version https://arxiv.org/abs/2110.01110. The code is at https://github.com/intelligent-control-lab/NNDM-safe-control.

2. Formulation

Dynamic model Consider a discrete time dynamic system with m_x state and m_u controls.

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

where k is the time step, $\mathbf{x}_k \in X \subset \mathbb{R}^{m_x}$ is the state, $\mathbf{u}_k \in U \subset \mathbb{R}^{m_u}$ is the control, $\mathbf{f} : \mathbb{R}^{m_x} \mapsto \mathbb{R}^{m_u}$ is the dynamic model, and dt is the sampling time. We assume the legal state set X and control set U are both defined by linear constraints. This assumption covers most cases in practice.

In the NNDM case, the dynamic model $\mathbf{f}$ is encoded by a n-layer feedforward neural network. Each layer in $\mathbf{f}$ corresponds to a function $\mathbf{f}_i : \mathbb{R}^{k_{i-1}} \mapsto \mathbb{R}^{k_i}$, where k_i is the dimension of the hidden variable $\mathbf{z}_i$ in layer i, and $k_0 = m_x + m_u$, $k_n = m_x$. The network can be represented by $\mathbf{f} = \mathbf{f}_n \circ \mathbf{f}_{n-1} \circ \cdots \circ \mathbf{f}_1$, where $\mathbf{f}_i$ is the mapping for layer i. And $\mathbf{z}_i = \mathbf{f}_i(\mathbf{z}_{i-1}) = \boldsymbol{\sigma}_i(\hat{\mathbf{z}}_i) = \boldsymbol{\sigma}_i(\mathbf{W}_i\mathbf{z}_{i-1} + \mathbf{b}_i)$

where $\mathbf{W}_i \in \mathbb{R}^{k_i \times k_{i-1}}$ is the weight matrix, $\mathbf{b}_i \in \mathbb{R}^{k_i}$ is the bias vector, and $\boldsymbol{\sigma}_i : \mathbb{R}^{k_i} \mapsto \mathbb{R}^{k_i}$ is the activation function. We only consider ReLU activation in this work. For simplicity, denote $\mathbf{W}_i\mathbf{z}_{i-1} + \mathbf{b}_i$ by $\hat{\mathbf{z}}_i$. Let $z_{i,j}$ be the value of the j^{th} node in the i^{th} layer, $\mathbf{w}_{i,j} \in \mathbb{R}^{1\times k_{i-1}}$ be the j^{th} row in $\mathbf{W}_i$, and $b_{i,j}$ be the j^{th} entry in $\mathbf{b}_j$.

Safety specification We consider the safety specification as a requirement that the system state should be constrained in a connected and closed set $\mathcal{X}_0 \subseteq X$. $\mathcal{X}_0$ is called the safe set. $\mathcal{X}_0$ should be a zero-sublevel set of an initial safety index $\phi_0 : X \mapsto \mathbb{R}$, *i.e.* $\mathcal{X}_0 = \{\mathbf{x} \mid \phi_0(\mathbf{x}) \leq 0\}$. ϕ_0 can be defined differently for a given $\mathcal{X}_0$. Ideally, a safe control law should guarantee forward invariance and finite-time convergence to the safe set. Forward invariance requires that $\phi_0(\mathbf{x}_k) \leq 0 \implies \phi_0(\mathbf{x}_{k+1}) \leq 0$. And finite-time convergence can be enforced by requiring that $\phi_0(\mathbf{x}_k) > 0 \implies \phi_0(\mathbf{x}_{k+1}) \leq \phi_0(\mathbf{x}_k) - \gamma dt$. Hence the number of time steps for an unsafe state to return to the safe set is bounded above by $\phi_0(\mathbf{x}_k)/\gamma dt$. These two conditions can be written compactly as one:

$$\phi_0(\mathbf{x}_{k+1}) \leq \max\{0, \phi_0(\mathbf{x}_k) - \gamma dt\}. \tag{2}$$

The safe tracking problem This paper considers the following constrained optimization for safe tracking, where the problem is solved at every time step k:

$$\begin{aligned} \min_{\mathbf{u}_k, \mathbf{x}_{k+1}} \ & \|\mathbf{x}_{k+1} - \mathbf{x}^r_{k+1}\|_p \\ \textit{s.t. } & \mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)dt, \quad \mathbf{u}_k \in U \\ & \phi_0(\mathbf{x}_{k+1}) \leq \max\{0, \phi_0(\mathbf{x}_k) - \gamma dt\} \end{aligned} \tag{3}$$

where $\mathbf{x}^r_{k+1}$ is the reference state at time step $k+1$, $\|\cdot\|_p$ can be either $\ell 1$-norm or $\ell 2$-norm. This formulation can be viewed as a one-step model predictive control (MPC). The extension to multi-step MPC is straightforward, which we leave for future work. At a given step k, (3) is a nonlinear programming problem. However, existing nonlinear solvers have poor performance for constraints involving neural networks (which will be shown in section 5). The reason is that neural networks (with ReLU activation) are piece-wise linear, whose second-order derivatives are not informative. New techniques are needed to solve this problem.

Persistent feasibility Persistent feasibility requires that there always exists $\mathbf{u}_k \in U$ that satisfies (2) for all time step k. However, this may not be true for some $\mathbf{x}_k \in \mathcal{X}_0$. For example, if ϕ_0 measures the distance between the ego vehicle and the leading vehicle. It is possible that the ego vehicle is still far from the leading vehicle ($\phi_0(\mathbf{x}_k) < 0$), but has big relative speed toward the leading vehicle. Then collision is inevitable ($\phi_0(\mathbf{x}_{k+1}) > 0$ for all possible $\mathbf{u}_k \in U$). This situation may happen when the relative degree from ϕ_0 to $\mathbf{u}$ is greater than one or when the control inputs are bounded. In these cases, $\mathcal{X}_0$ may not be forward invariant or finite-time convergent. We call this situation as *loosing control feasibility*, which further leads to *loosing persistent feasibility*. To address this problem, we want to prevent the system from getting into those control-infeasible states in $\mathcal{X}_0$. That is to find a subset $\mathcal{X}_s \subseteq \mathcal{X}_0$ such that there exists a feasible control law to make $\mathcal{X}_s$ forward invariant and finite-time convergent. We call $\mathcal{X}_s$ a *control invariant set* within $\mathcal{X}_0$.

3. Related work

Optimization with Neural Network Constraints Recent progress in nonlinear optimization involving neural network constraints can be classified as primal optimization methods and dual optimization methods. The primal optimization methods encode the nonlinear activation functions

(e.g., ReLU) as mixed-integer linear programmings Tjeng et al. (2017), relaxed linear programmings Ehlers (2017) or semidefinite programmings Raghunathan et al. (2018). Our method to encode NNDM is inspired by MIPVerify Tjeng et al. (2017), which uses mixed integer programming to compute maximum allowable disturbances to the input. MIPVerify is complete and sound, meaning that the encoding is equivalent to the original problem.

QP-based safe control When the system dynamics are analytical and control-affine, the safe tracking problem can be decomposed into two steps: 1) computing a reference control $\mathbf{u}^r$ without the safety constraint; 2) projecting $\mathbf{u}^r$ to the safe control set Ames et al. (2019). For analytical control-affine dynamic models, the safe control set that satisfies (3) is a half-space intersecting with U. Therefore, the second step is essentially a quadratic projection of the reference control to that linear space, which can be efficiently computed by calling a quadratic programming (QP) solver. Existing methods include CBF-QP Ames et al. (2016), SSA-QP Liu and Tomizuka (2014), etc. However, to our best knowledge, there has not been any quadratic projection method that projects a reference control to a safety constraint with non-analytical and non-control-affine dynamic models, in which case the safe control set can be non-convex. Moreover, our work solves both the computation of reference control and its projection onto safe control set in an integrated manner. Hence, our work is not limited to the quadratic projection of the reference control.

Persistent feasibility in MPC There are different approaches in MPC literature to compute the control invariant set to ensure persistent feasibility, such as Lyapunov function Danielson et al. (2016), linearization-convexification Jalalmaab et al. (2017), and grid-based reachability analysis Bansal et al. (2017). However, most of the non-grid-based methods approximate the control invariant set by convex set, which greatly limit the expressiveness of the geometries. Although grid-based methods have better expressiveness and may be able to extend to non-analytical models, they have limited scalability due to the curse of dimensionality and the fact they are usually non-parameterized. Our method can synthesize the control invariant set with nonlinear boundaries for non-analytical models using parameterized functions, hence more computationally efficient.

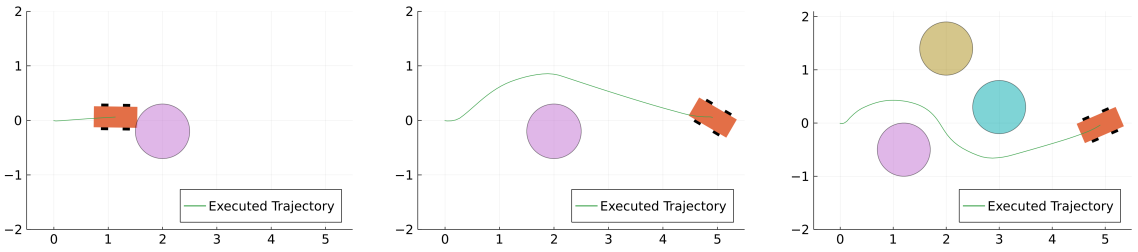
4. Method

In this section, we discuss how to efficiently solve the constrained optimization (3) and ensure it is persistently feasible. First, we introduce MIND, a way to find the optimal solution of eq. (3) by encoding NNDM constraints as mixed integer constraints. Then we present SIS, a method to find the control invariant set by learning a new safety index ϕ that maximizes control feasibility. Finally, we present the reformulated problem.

4.1. MIND: Encode NNDM constraints

To overcome the complexity of NNDM constraints, we first add all hidden nodes in the neural network as decision variables and turn (3) into the following equivalent form:

$$\begin{aligned}
&\min_{\mathbf{u}_k, \mathbf{x}_{k+1}, \mathbf{z}_i} \|\mathbf{x}_{k+1} - \mathbf{x}^r_{k+1}\|_p \\
&\quad s.t.\ \mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{z}_n dt, \mathbf{z}_0 = [\mathbf{x}_k, \mathbf{u}_k], \quad \mathbf{u}_k \in U \\
&\qquad z_{i,j} = \max\{\hat{z}_{i,j}, 0\}, \hat{z}_{i,j} = \mathbf{w}_{i,j}\mathbf{z}_{i-1} + b_{i,j}, \forall i \in \{1, \dots, n\}, \forall j \in \{1, \dots, k_i\} \\
&\qquad \phi_0(\mathbf{x}_{k+1}) \leq \max\{0, \phi_0(\mathbf{x}_k) - \gamma dt\}.
\end{aligned} \tag{4}$$



(*a*) (*b*) (*c*)

Figure 1: MIND-SIS in collision avoidance. (a) shows the performance of MIND with ϕ_0. The vehicle collides with the obstacle due to loss of control feasibility (too late to break). (b) and (c) show the performance of MIND-SIS. The SIS-synthesized safety index ϕ^l guarantees persistent feasibility and is general enough to be directly applied to multi-obstacle scenarios without any modification.

Nevertheless, the nonlinear non-smooth constraints introduced by the ReLU activation $z_{i,j} = \max\{\hat{z}_{i,j}, 0\}$ in (4) is still challenging to handle. Inspired by MIPVerify Tjeng et al. (2017), we use mixed integer formulation to rewrite these constraints. We first introduce an auxiliary variable $\delta_{i,j}$ to denote the activation status of the ReLU node:

$$\delta_{i,j} = 1 \Rightarrow z_{i,j} = \hat{z}_{i,j}, \quad \delta_{i,j} = 0 \Rightarrow z_{i,j} = 0. \tag{5}$$

Then we compute the pre-activation upper bounds $\hat{u}_{i,j}$ and lower bound $\hat{l}_{i,j}$ of every node in the neural network using interval arithmetics Moore et al. (2009). Given the input ranges (e.g., $x \in [1, 2]$ and $y \in [3, 4]$), interval arithmetics compute the output range using the lower and upper bounds (e.g., $x - y \in [1-4, 2-3] = [-3, -1]$). When $\hat{u}_{i,j} \leq 0$, the constraint for ReLU activation reduces to $z_{i,j} = 0$. When $\hat{l}_{i,j} \geq 0$, the constraint reduces to $z_{i,j} = \hat{z}_{i,j}$. Otherwise, the constraint can be represented as the following linear inequalities Liu et al. (2021):

$$z_{i,j} \geq \hat{z}_{i,j}, z_{i,j} \geq 0, z_{i,j} \leq \hat{z}_{i,j} - \hat{\ell}_{i,j}\left(1 - \delta_{i,j}\right), z_{i,j} \leq \hat{u}_{i,j}\delta_{i,j}, \delta_{i,j} \in \{0, 1\}. \tag{6}$$

With this encoding, the constrained optimization is converted into a MIP, which can be solved efficiently. Theoretically, MIP is a NP-complete problem. The worst case computation time grows exponentially with the number of integer variables, which is the total number of ReLU activation functions. However, in practice, the computation can be greatly accelerated by various techniques developed in recent years Gurobi Optimization, LLC (2021). The evaluation in section 5 shows that the actual computation time for a network with 100 ReLUs is only 0.36 seconds while that for a network with 200 ReLUs is 0.8 seconds. Besides, it is worth noting that MIND scales well with dimensions of state and control when the total number of neurons in the hidden layers is fixed.

Nevertheless, successfully obtaining the solution for time k and executing the control does not necessarily ensure we will have a solution to the constrained optimization in future time steps as shown in fig. 1(a). The safety constraint needs to be modified to ensure persistent feasibility.

4.2. SIS: Guaranteed persistent feasibility

Unlike analytical models, it is challenging to design a safety index for NNDM because of its poor interpretability. Therefore, we introduce Safety Index Synthesis (SIS), which automatically synthesizes a safety index ϕ that results in a forward invariant and finite-time convergent $\mathcal{X}_s$ to guarantee persistent feasibility.

Liu and Tomizuka (2014) introduced a form of ϕ that can improve control feasibility, $\phi(\mathbf{x}) = \phi_0^*(\boldsymbol{\alpha}_0, \mathbf{x}) + \sum_{i=1}^{q} \alpha_i \phi_0^{(i)}(\mathbf{x}) + \beta$, where $\phi_0^*(\boldsymbol{\alpha}_0, \mathbf{x})$ defines the same sublevel set as ϕ_0 and is parameterized by $\boldsymbol{\alpha}_0$, $\phi_0^{(i)}(\mathbf{x})$ is the i-th order derivative of ϕ_0, q is the order such that the relative degree from $\phi_0^{(q)}$ to $\mathbf{u}$ is 1, and β is a constant. We denote the concatenation $[\boldsymbol{\alpha}_0, \alpha_1, \cdots, \alpha_q]$ by $\boldsymbol{\alpha}$. Liu and Tomizuka (2014) showed that this ϕ could result in a forward invariant and finite-time convergent $\mathcal{X}_s$ when the control input is unbounded. When the control input is bounded, we argue that persistent feasibility can be achieved by optimizing $\boldsymbol{\alpha}$ and β under Assumption 1.

Assumption 1 *$\mathbf{f}$ and ϕ are Lipschitz continuous functions with Lipschitz constants k_f and k_ϕ respectively. The Euclidean norm of $\mathbf{f}(\mathbf{x}, \mathbf{u})$ is bounded by M_f.*

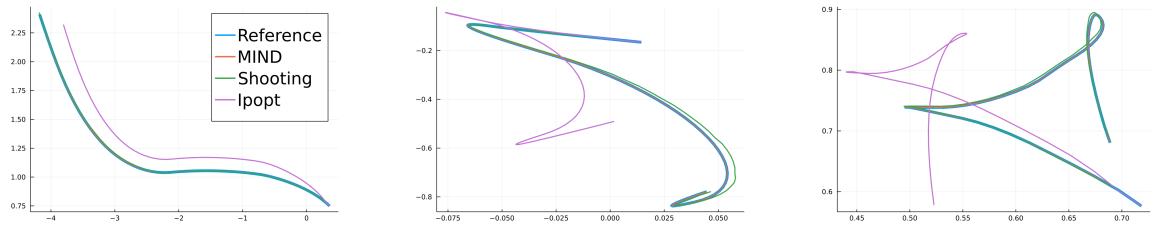
To ensure forward invariance, it suffices to enforce that there always exists a feasible control for all states near the zero-level set of ϕ (as proved in appendix A.2). Define the state-of-interest set $B = \{\mathbf{x} \mid |\phi(\mathbf{x})| \leq k_\phi M_f dt\}$ (states near the boundary $\phi = 0$) and infeasible-state-of-interest set $B^* = \{\mathbf{x} \mid \mathbf{x} \in B, \forall \mathbf{u}, \phi(\mathbf{x} + \mathbf{f}(\mathbf{x}, \mathbf{u})dt) > \max\{0, \phi(\mathbf{x}) - \gamma dt\}\}$. B contains all the states that can cross the boundary $\phi = 0$ in one step because $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| \leq \|\mathbf{f}(\mathbf{x}, \mathbf{u})\| dt \leq M_f dt$. To achieve forward invariance, we need all states in B to have feasible control (i.e., when B^* is empty). Then the problem can be formulated as $\min_{\boldsymbol{\alpha}, \beta} |B^*|/|B|$. The expression of the corresponding control invariant set $\mathcal{X}_s$ is derived in appendix A.2. We can also let $B = X$ to learn a safety index that further achieves finite time convergence of $\mathcal{X}_s$ at the cost of a potentially more conservative policy (see appendix A.2). Since the gradient from $|B^*|/|B|$ to $\boldsymbol{\alpha}, \beta$ is usually difficult to compute, we use a derivative-free evolutionary approach, CMA-ES Hansen (2016) to optimize the parameters. CMA-ES runs for multiple generations. In each generation, the algorithm samples many parameter candidates (called members) from a multivariate Gaussian distribution and evaluates their performance. A proportion of candidates with the best performance will be used to update the mean and covariance of the Gaussian distribution. To evaluate the parameter candidates, we sample a subset $S \subset B$ and minimize the infeasible rate $r := \frac{|S \cap B^*|}{|S|}$ as a surrogate for the original objective function $|B^*|/|B|$. We prove that if the sampling is dense enough and $r = 0$, the safety constraint with the learned safety index is guaranteed to be feasible for arbitrary states in B.

Lemma 2 *Suppose 1) we sample a state subset $S \subset B$ such that $\forall \mathbf{x} \in B$, $\min_{\mathbf{x}' \in S} \|\mathbf{x} - \mathbf{x}'\| \leq \delta$, where δ is an arbitrary constant representing the sampling density.; and 2) $\forall \mathbf{x}' \in S$, there exists a safe control $\mathbf{u}$, s.t. $\phi(\mathbf{x}' + \mathbf{f}(\mathbf{x}', \mathbf{u})dt) \leq \max\{-\epsilon, \phi(\mathbf{x}') - \gamma dt - \epsilon\}$, where $\epsilon = k_\phi(1 + k_f dt)\delta$. Then $\forall \mathbf{x} \in B, \exists \mathbf{u}$, s.t.*

$$\phi(\mathbf{x} + \mathbf{f}(\mathbf{x}, \mathbf{u})dt) \leq \max\{0, \phi(\mathbf{x}) - \gamma dt\}. \tag{7}$$

Proof According to condition 1), $\forall \mathbf{x} \in B$, we can find $\mathbf{x}' \in S$ such that $\|\mathbf{x} - \mathbf{x}'\| \leq \delta$. According to condition 2), for this $\mathbf{x}'$, we can find $\mathbf{u}$ such that $\phi(\mathbf{x}' + \mathbf{f}(\mathbf{x}', \mathbf{u})dt) \leq \max\{0, \phi(\mathbf{x}') - \gamma dt\} - \epsilon$. Next we show $\mathbf{x}$ and $\mathbf{u}$ satisfy (7) using Lipschitz condition and triangle inequality.

$$\begin{aligned}
\phi(\mathbf{x} + \mathbf{f}(\mathbf{x}, \mathbf{u})dt) &= \phi(\mathbf{x} + \mathbf{f}(\mathbf{x}, \mathbf{u})dt) - \phi(\mathbf{x}' + \mathbf{f}(\mathbf{x}', \mathbf{u})dt) + \phi(\mathbf{x}' + \mathbf{f}(\mathbf{x}', \mathbf{u})dt) && \text{(8a)}\\
&\leq k_\phi \|\mathbf{x} - \mathbf{x}' + [\mathbf{f}(\mathbf{x}, \mathbf{u}) - \mathbf{f}(\mathbf{x}', \mathbf{u})]dt\| + \max\{0, \phi(\mathbf{x}') - \gamma dt\} - \epsilon && \text{(8b)}\\
&\leq k_\phi \|\mathbf{x} - \mathbf{x}'\| + k_\phi \|\mathbf{f}(\mathbf{x}, \mathbf{u}) - \mathbf{f}(\mathbf{x}', \mathbf{u})\| dt + \max\{0, \phi(\mathbf{x}') - \gamma dt\} - \epsilon && \text{(8c)}\\
&\leq k_\phi \delta + k_\phi k_f \delta dt - \epsilon + \max\{0, \phi(\mathbf{x}') - \gamma dt\} && \text{(8d)}\\
&= \max\{0, \phi(\mathbf{x}') - \gamma dt\}. && \text{(8e)}
\end{aligned}$$



(a) (b) (c)

Figure 2: Trajectory tracking with NNDM II using different optimization methods: MIND (our method), shooting method with sample size of 100, and Ipopt. (a-c) show 3 randomly generated trajectories. MIND has the smallest tracking error in all three cases.

Hence (7) is verified. ■

The computation time of SIS depends on the number of CMA-ES iterations and the time spent in finding S in each iteration. Although a rigorous proof is missing, the convergence rate of CMA-ES is empirically exponential Hansen and Ostermeier (2001). We can find S in each iteration by uniformly sampling X. This process can be time consuming if the state dimension is high (e.g. $n > 10$), but we may accelerate the process by high dimensional Breadth-First-Search, which we leave for future work.

4.3. MIND-SIS: Safe control with NNDM

Once the safety index is synthesized, we substitute ϕ_0 with ϕ in (4) to guarantee persistent feasibility. To address the nonlinearity in the safety constraint (2), we approximate it with first order Taylor expansion at the current state $\mathbf{x}_k$ (appendix A.1 discusses how the safety guarantee is preserved):

$$\phi(\mathbf{x}_{k+1}) = \phi(\mathbf{x}_k) + \nabla_{\mathbf{x}}\phi \cdot \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)dt + o(\|\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)dt\|), \tag{9}$$

where $\lim_{dt\to 0} o(\|\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)dt\|) = 0$. Then (4) is transformed into a mixed integer problem:

$$\begin{aligned}
\min_{\mathbf{u}_k, \mathbf{x}_{k+1}, \mathbf{z}_i, \delta_{i,j} \in \{0,1\}} & \|\mathbf{x}_{k+1} - \mathbf{x}^r_{k+1}\|_p \\
s.t.\ & \mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{z}_n dt, \mathbf{z}_0 = [\mathbf{x}_k, \mathbf{u}_k], \quad \mathbf{u}_k \in U, \\
& z_{i,j} \geq \hat{z}_{i,j}, z_{i,j} \geq 0, z_{i,j} \leq \hat{z}_{i,j} - \hat{\ell}_{i,j}(1 - \delta_{i,j}), z_{i,j} \leq \hat{u}_{i,j}\delta_{i,j}, \\
& \hat{z}_{i,j} = \mathbf{w}_{i,j}\mathbf{z}_{i-1} + b_{i,j}, \forall i \in \{1, \ldots, n\}, \forall j \in \{1, \ldots, k_i\} \\
& \nabla_{\mathbf{x}}\phi \cdot \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \leq \max\{-\frac{\phi(\mathbf{x}_k)}{dt}, -\gamma\}.
\end{aligned} \tag{10}$$

Depending on the norm $\|\cdot\|_p$, (10) is either a Mixed Integer Linear Programming or Quadratic Programming, which both can be solved by existing solvers, such as GLPK, CPLEX, and Gurobi.

5. Experiment

5.1. Experiment set-up

The evaluation is designed to answer the following questions: 1) How does our method (by solving (10)) compare to the shooting method and regular nonlinear solvers in terms of optimality and

computational efficiency on problems without safety constraints? 2) Does the safety index synthesis improve persistent feasibility? 3) Does our method ensure safety in terms of forward invariance.

We evaluate our method on a system with NNDMs for 2D vehicles. The NNDMs are learned from a second order unicycle dynamic model with 4 state inputs (2D position, velocity, and heading angle), 2 control inputs (angular velocity, acceleration), and 4 state outputs (2D velocity, angular velocity, and acceleration). All the states and controls are bounded, where $X : [-10, 10] \times [-10, 10] \times [-2, 2] \times [-\pi, \pi]$ and $U : [-4, 4] \times [-\pi, \pi]$. We learn 3 different fully connected NNDMs to show the generalizability of our method, which are: I. 3-layer with 50 hidden neurons per layer. II. 3-layer with 100 hidden neurons per layer. III. 4-layer with 50 hidden neurons per layer. Scalability analysis with more models can be found in appendix A.4.

When evaluating the control performance, we roll-out the closed-loop trajectory directly using NNDM to avoid model mismatch. Our evaluation aims to show that the proposed method can provide provably safe controls efficiently for the learned model. The safe control computed by NNDM may be unsafe for the actual dynamics under model mismatch. We will extend our work to robust safe control Liu and Tomizuka (2015); Noren and Liu (2019), which can guarantee safety even with model mismatch in the future.

To answer the questions we raised in the beginning, we design the following two tasks. The first task is trajectory tracking without safety constraints, which can test how our method performs comparing to other methods in terms of optimality and computational efficiency. And the second task is trajectory tracking under safety constraints. It is to test whether the learned safety index improves the feasibility and whether MIND-SIS ensures forward invariance and finite-time convergence.

	NNDM I			NNDM II			NNDM III		
Method	Mean	Std	Time (s)	Mean	Std	Time (s)	Mean	Std	Time (s)
MIND	$\mathbf{< 10^{-8}}$	$\mathbf{< 10^{-7}}$	0.364	$\mathbf{< 10^{-8}}$	$\mathbf{< 10^{-7}}$	0.838	$\mathbf{< 10^{-8}}$	$\mathbf{< 10^{-7}}$	1.235
Shooting-10^3	0.129	0.080	0.021	0.128	0.080	0.100	0.128	0.080	0.029
Shooting-10^4	0.041	0.026	0.209	0.041	0.026	1.002	0.041	0.026	0.283
Shooting-10^5	0.012	0.007	2.084	0.012	0.007	10.063	0.012	0.007	2.822
Ipopt	1.871	0.626	0.032	1.852	0.619	0.040	1.865	0.623	0.033

Table 1: Average tracking error and average computation time of different methods in the trajectory tracking task (without safety constraint). The table shows mean and standard deviation of the average tracking error. The number after "Shooting" denotes the sampling size. Our method can always find the optimal solution, therefore achieves almost zero tracking error. The actual trajectories are illustrated in fig. 2.

5.2. Trajectory tracking

In this task, we randomly generate 500 reference trajectory waypoints for each NNDM by rolling out the NNDM with some random control inputs. We compare our method (MIND) with 1) shooting methods with different sampling sizes and 2) Interior Point OPTimizer (Ipopt), a popular nonlinear solver. We use CPLEX to solve the MIND formulation. This experiment is done on a computer with AMD® Ryzen threadripper 3960x 24-core processor, 128 GB memory. Some results are shown in fig. 2. Detailed settings and comparison of control sequences can be found in appendix A.3.

As shown in table 1, MIND achieves an average tracking error less than 10^{-8}. The tracking error is less than the resolution of single-precision floats. We can conclude that MIND finds the optimal solution, which is a significant improvement comparing to other methods. The shooting method achieves lower tracking error with a larger sampling size but that also takes longer. To achieve the

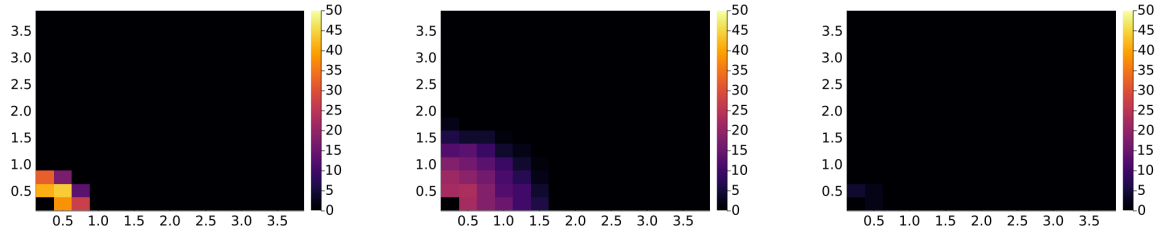


Figure 3: Distribution of infeasible-state-of-interest states B^* when we optimize the safety index on the whole state space ($B = X$). An obstacle is located at $(0, 0)$. Each grid in the graph corresponds to a location. We sample 100 states at each location (with different heading angle and velocity). The color denote how many states at this location are in B^*. (a) shows that the original safety index ϕ_0 has many infeasible states near the obstacle. (b) shows that a handcrafted safety index ϕ^h can not find feasible control for some states near the obstacle. (c) shows that the learned index ϕ has no state in B^*, thus can always find a feasible safe control.

same tracking error as MIND, the sampling size and computation time will be unacceptably large. Ipopt takes shorter time because it gets stuck at local optima quickly.

5.3. Trajectory tracking under safety constraints

This task considers two safety constraints corresponding to different scenarios: collision avoidance and safe following. When control infeasibility happens for a poorly designed safety index, we relax the safety constraint by adding a slack variable.

5.3.1. COLLISION AVOIDANCE

Collision avoidance is one of the most common safety requirement in real-world applications. In this experiment, we consider one static obstacle. The safety index is given as $\phi_0(\mathbf{x}) = d_{min} - d(\mathbf{x}) < 0$, where $d(\mathbf{x})$ is the relative distance from the agent to the obstacle, and d_{min} is a constant. This constraint usually can not guarantee persistent feasibility. Therefore, we synthesize a safety index $\phi(\mathbf{x})$ that guarantees persistent feasibility by learning parameters of the following form:

$$\phi(\mathbf{x}) = d_{min}^{\alpha_1} - d(\mathbf{x})^{\alpha_1} - \alpha_2 \dot{d}(\mathbf{x}) + \beta,$$

where $\dot{d}(\mathbf{x})$ is the relative velocity, α_1, α_2 and β are parameters to learn. This form guarantees forward invariance and finite-time convergence for second-order systems when there is no control limits as shown in Liu and Tomizuka (2014), (see appendix A.2 for more discussion). The learned index can generalize to multiple obstacles case when we consider one constraint to each obstacle.

The search ranges for the parameters are: $\alpha_1 \in (0.1, 5)$, $\alpha_2 \in (0.1, 5)$, $\beta \in (0.001, 1)$. For each set of parameters, we test how many sampled states are in B^*. We place an obstacle at $(0, 0)$, and uniformly sample 40000 states around the obstacle to find S, then test whether the safe control set of each state is empty. Figure 3 shows the distribution of B^* of ϕ_0, a manually tuned safety index ϕ^h given by previous work Liu and Tomizuka (2014) ($\alpha_1 = 2, \alpha_2 = 1, \beta = 0.1$, did not consider control limits), and a synthesized safety index ϕ^l with learned parameters ($\alpha_1 = 0.172, \alpha_2 = 4.107, \beta = 0.447$). ϕ^l achieves 0 infeasible rate. Additional comparison is in appendix A.2. To demonstrate the effect of the synthesized safety index, we visualize the behavior of the agent with ϕ_0 and ϕ^l in fig. 1. The figure also shows that the synthesized safety index can be directly applied to

unseen multi-obstacle scenarios without any change. The persistent feasibility is preserved if there is always at most one obstacle becoming safety critical Zhao et al. (2021).

We evaluate these safety indices on 100 randomly generated collision avoidance tasks. The agent has to track a trajectory while avoiding collision (keep $\phi_0 \leq 0$). A task succeeds if there is no collision and no control-infeasible states throughout the trajectory. The evaluation results are shown in table 2. The learned safety index achieves 0% ϕ_0-violation rate and 0% infeasible rate. It is worth mentioning that Zhao et al. (2021) proposes a safety index design rule that guarantees feasibility for 2D collision avoidance. We verified that the ϕ^l satisfies this rule.

	Collision avoidance			Safe following		
Metric	ϕ_0	ϕ^h	ϕ^l	ϕ_0	ϕ^h	ϕ^l
Success rate	0%	89%	$\mathbf{100}\%$	43%	82%	$\mathbf{100}\%$
ϕ_0-violation rate	100%	0%	0%	56%	0%	0%
Infeasible rate	100%	11%	0%	57%	18%	0%

Table 2: Performance comparison of the original safety index ϕ_0, a manually tuned safety index ϕ^h, and a learned safety index ϕ^l on 100 randomly generated tasks. One trial is successful if there is no safety violation or infeasible state. We can see that ϕ_0 always violates the constraints due to infeasibility (due to our choice of the initial state). ϕ^h fails to find control for some states due to infeasibility. Finally, the learned index ϕ^l can always find a safe control and avoid ϕ_0 violation.

5.3.2. SAFE FOLLOWING

The safe following constraint appears when an agent is following a target while keeping a safe distance, such as in adaptive cruise control, nap-of-the-earth flying, etc. The initial safety index is $\phi_0(\mathbf{x}) = (d(\mathbf{x}) - d_l)(d(\mathbf{x}) - d_u)$, where d_l and d_u are the lower and upper bound of the relative distance. We design the safety index to be of the form:

$$\phi(\mathbf{x}) = \left| d(\mathbf{x}) - \frac{d_l + d_u}{2} \right|^{\alpha_1} - \left(\frac{d_u - d_l}{2} + \beta \right)^{\alpha_1} + \alpha_2 \left[2d(\mathbf{x}) - (d_l + d_u) \right] \dot{d}(\mathbf{x}). \tag{11}$$

The search range for the parameters to learn are $\alpha_1 \in (0.1, 10), \alpha_2 \in (0.1, 10), \beta \in (0.001, 0.5)$. The learned parameters for ϕ^l are $\alpha_1 = 8.092, \alpha_2 = 9.826, \beta = 0.489$. The human designed parameters are $\alpha_1 = 2, \alpha_2 = 1, \beta = 0.01$ We also randomly generate 100 following tasks for evaluation. We consider a task successful if there is no ϕ_0 constraint violation or infeasible state during the following. The evaluation results are shown in table 2. The learned index achieves 0% ϕ_0-violation rate and 0% infeasible rate.

6. Discussion

In this work, we propose MIND-SIS, the first method to derive safe control law for NNDM. MIND finds the optimal solution for safe tracking problems involving NNDM constraints, and SIS synthesizes a safety index that guarantees forward invariance and finite-time convergence. Theoretical guarantees of optimality and feasibility are provided. However, safety violation may still exist if the NNDM does not align with the true dynamics. As a future work, we will explore how to guarantee safety under model mismatch and uncertainty. One limitation of SIS is that to guarantee persistent feasibility, the theoretical sampling rate grows exponentially with the dimension of states. We will study how to adaptively adjust the sampling rate to overcome the curse of dimensionality.

References

Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8): 3861–3876, 2016.

Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pages 3420–3431. IEEE, 2019.

Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253. IEEE, 2017.

Franco Blanchini. Set invariance in control. *Automatica*, 35(11):1747–1767, 1999.

Claus Danielson, Avishai Weiss, Karl Berntorp, and Stefano Di Cairano. Path planning using positive invariant sets. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 5986–5991. IEEE, 2016.

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.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2021. URL https://www.gurobi.com.

Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

Mehdi Jalalmaab, Bariş Fidan, Soo Jeon, and Paolo Falcone. Guaranteeing persistent feasibility of model predictive motion planning for autonomous vehicles. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 843–848. IEEE, 2017.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *arXiv preprint arXiv:1906.08253*, 2019.

Changliu Liu and Masayoshi Tomizuka. Control in a safe set: Addressing safety in human-robot interactions. In *ASME 2014 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers Digital Collection, 2014.

Changliu Liu and Masayoshi Tomizuka. Safe exploration: Addressing various uncertainty levels in human robot interactions. In *2015 American Control Conference (ACC)*, pages 465–470. IEEE, 2015.

Changliu Liu, Tomer Arnon, Christopher Lazarus, Christopher Strong, Clark Barrett, Mykel J Kochenderfer, et al. Algorithms for verifying deep neural networks. *Foundations and Trends® in Optimization*, 4, 2021.

Ramon E Moore, R Baker Kearfott, and Michael J Cloud. *Introduction to interval analysis*. SIAM, 2009.

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.

Mitio Nagumo. Über die lage der integralkurven gewöhnlicher differentialgleichungen. *Proceedings of the Physico-Mathematical Society of Japan. 3rd Series*, 24:551–559, 1942.

Duy Nguyen-Tuong and Jan Peters. Model learning for robot control: a survey. *Cognitive processing*, 12(4):319–340, 2011.

Charles Noren and Changliu Liu. Safe adaptation in confined environments using energy functions. *arXiv preprint arXiv:1912.09095*, 2019.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Semidefinite relaxations for certifying robustness to adversarial examples. *arXiv preprint arXiv:1811.01057*, 2018.

Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

Deepak Tolani, Ambarish Goswami, and Norman I Badler. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical models*, 62(5):353–388, 2000.

Tianhao Wei and Changliu Liu. Safe control algorithms using energy functions: A uni ed framework, benchmark, and new directions. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 238–243. IEEE, 2019.

Weiye Zhao, Tairan He, and Changliu Liu. Model-free safe control for zero-violation reinforcement learning. In *Conference on Robot Learning*, pages 784–793. PMLR, 2021.