

TRUST: Stability and Safety Controller Synthesis for Unknown Dynamical Models Using a Single Trajectory

Jamie Gardner
School of Computing
Newcastle University
j.gardner3@newcastle.ac.uk

Ben Wooding
School of Computing
Newcastle University
ben.wooding@newcastle.ac.uk

Amy Nejati
School of Computing
Newcastle University
amy.nejati@newcastle.ac.uk

Abolfazl Lavaei
School of Computing
Newcastle University
abolfazl.lavaei@newcastle.ac.uk

Abstract

TRUST is an open-source software tool developed for *data-driven controller synthesis* of dynamical systems with *unknown* mathematical models, ensuring either stability or safety properties. By collecting only a *single input-state trajectory* from the unknown system and satisfying a rank condition that ensures the system is persistently excited according to the Willems *et al.*'s fundamental lemma, TRUST aims to design either *control Lyapunov functions (CLF)* or *control barrier certificates (CBC)*, along with their corresponding stability or safety controllers. The tool implements sum-of-squares (SOS) optimization programs solely based on data to enforce stability or safety properties across four system classes: (i) *continuous-time nonlinear polynomial systems*, (ii) *continuous-time linear systems*, (iii) *discrete-time nonlinear polynomial systems*, and (iv) *discrete-time linear systems*. TRUST is a Python-based web application featuring an intuitive, reactive graphic user interface (GUI) built with web technologies. It can be accessed at <https://trust.tgo.dev> or installed locally, and supports both manual data entry and data file uploads. Leveraging the power of the Python backend and a JavaScript frontend, TRUST is designed to be highly user-friendly and accessible across desktop, laptop, tablet, and mobile devices. We apply TRUST to a set of physical benchmarks with unknown dynamics, ensuring either stability or safety properties across the four supported classes of models.

Keywords

Data-driven controller synthesis, stability and safety properties, control Lyapunov functions, control barrier certificates, Willems *et al.*'s fundamental lemma, single trajectory

1 Introduction

The formal synthesis of controllers that ensure stability or safety of dynamical systems is a cornerstone of control theory, especially in safety-critical applications where failures can lead to loss of life or significant financial consequences [1]. These applications include robotics, aerospace, autonomous vehicles, and medical devices, where system reliability is paramount. Traditionally, control synthesis has relied heavily on model-based approaches, which require precise mathematical representations of system dynamics. However, obtaining models with closed-form expressions of their dynamic can be highly challenging, as the identified models may

not fully capture the complexities of real-world systems [2]. In response to this key obstacle, *direct* data-driven techniques have emerged in the literature as a compelling alternative [3]. These techniques leverage system data to directly design controllers without the need for an explicit mathematical model, offering a more flexible and practical approach for complex dynamical systems.

While control Lyapunov functions (CLF) are essential methods for ensuring stability in dynamical systems [4], *control barrier certificates (CBC)* have been introduced as a powerful method for guaranteeing safety [5–7]. Similar to Lyapunov functions, CBCs are defined over the system's state space, but they focus on satisfying specific inequalities on both the system dynamics and the function itself. By identifying an appropriate *level set* of CBC, unsafe regions can be separated from the system's trajectories, starting from a given set of initial conditions. As a result, the existence of such a function not only provides formal safety certification but also facilitates the design of a controller that enforces safety throughout the system's operation.

1.1 Data-Driven Techniques

Since constructing CLF or CBC typically requires precise mathematical models of the system dynamics, the development of data-driven techniques has become crucial in control theory. Two promising approaches in the literature offer formal controller synthesis using collected data without relying on explicit models. The first is the *scenario approach* [8, 9], which solves the problem by leveraging collected data and then translating the results back to unknown models using intermediate steps involving *chance constraints* [10]. While this method shows significant potential for providing formal guarantees for systems with unknown models, it requires the data to be independent and identically distributed (i.i.d.). This restriction means that only one input-output data pair can be collected from each trajectory [8], necessitating the collection of *multiple independent trajectories*—potentially up to millions in real-world scenarios—to achieve a desired confidence level, based on a well-defined closed-form relationship between them.

An alternative to the scenario method is the *non-i.i.d. trajectory-based approach*, which complements the former by requiring only a *single input-output trajectory* from the unknown system over a specific time horizon [11]. This approach utilizes the concept of *persistent excitation*, where the trajectory should meet a rank condition to sufficiently excite the system's dynamics, as outlined

by Willems *et al.*'s fundamental lemma [12]. Specifically, when a system is persistently excited, the trajectory provides enough information about the system's behavior to enable its analysis. This method is advantageous since it eliminates the need for multiple independent trajectories, making it more practical in cases where obtaining several distinct trajectories is challenging.

1.2 Data-Driven Literature

A substantial body of research has investigated data-driven analysis of unknown systems using the scenario approach. This includes stability analysis of unknown systems (e.g., [13–16]), the construction of barrier certificates (e.g., [17–20]) and finite abstractions (e.g., [21–25]), both applied in safety verification and controller synthesis. Additionally, there is a rich literature on data-driven methods based on the *single-trajectory approach*. These approaches have been used to ensure stability and invariance properties in unknown systems (e.g., [26–29]) and to construct control barrier certificates for both continuous- and discrete-time systems (e.g., [30–32]).

1.3 Related Software Tools

There are only a few software tools available for constructing either control Lyapunov functions or (control) barrier certificates. One such tool is the class library developed in [33], which computes Lyapunov functions for nonlinear systems in C++. Another tool, LyZNet [34], utilizes neural networks (NN) to learn and verify stability functions and regions of attraction, leveraging a physical model to inform the NN. The tool FOSSIL 2.0 [35] synthesizes stability and safety barrier functions for both discrete and continuous systems using a model-driven NN. PROTECT [36] is another tool that designs safety barrier certificates for discrete- and continuous-time systems. While these tools [33–36] demonstrate significant potential, they all require *precise mathematical models of dynamical systems* to construct either stability Lyapunov functions or safety barrier certificates, along with their corresponding controllers. This requirement potentially contrasts with real-world scenarios where precise system models are often unavailable, making the aforementioned tools impractical for those applications.

1.4 Central Contributions

Motivated by the central challenge of unknown system models, this tool paper introduces the following innovative contributions:

- (i) TRUST is a first-of-its-kind tool that leverages *data-driven techniques* to synthesize stability Lyapunov functions and safety barrier certificates, along with their corresponding controllers, using only a *single input-state trajectory* from unknown systems.
- (ii) TRUST leverages the sum of squares (SOS) optimization toolbox [37], powered by MOSEK¹, and supports *four classes of dynamical systems*: (i) continuous-time nonlinear polynomial systems (ct-NPS), (ii) continuous-time linear systems (ct-LS), (iii) discrete-time nonlinear polynomial systems (dt-NPS), and (iv) discrete-time linear systems (dt-LS).
- (iii) Implemented as a responsive and reactive Python Flask² web application, TRUST offers an intuitive, user-friendly interface,

allowing users to seamlessly interact with the tool. Users can directly access and work with the application through the web without the need for downloads or installations.

- (iv) TRUST also offers a *Docker-based version* that can be downloaded and run locally, potentially achieving higher speeds depending on the capabilities of the local machine.
- (v) The server-side Python application, built on the Flask web framework, follows the Model-View-Controller (MVC) [38] architecture and adheres to Test-Driven Development (TDD) [39] practices, ensuring high-quality and maintainable code.
- (vi) TRUST leverages a monolithic architecture, using InertiaJS³ to enable real-time client-side updates and effective error handling, ensuring a seamless user experience across all platforms without the need to build a separate Application Programming Interface (API).
- (vii) The tool supports both manual data entry and data file uploads for collected trajectories, offering a user-friendly GUI for inputting the required information, such as the state space, and initial and multiple unsafe sets for designing CBC.
- (viii) We demonstrate the effectiveness of TRUST through a series of physical benchmarks, covering the four classes of dynamical systems and showcasing their respective stability or safety properties.

The web-based version of TRUST is accessible at:

<https://trust.tgo.dev>

The TRUST source code, accompanied by comprehensive installation and usage instructions for the Docker-based version, can be accessed at:

<https://github.com/Kiguli/TRUST>

1.5 Notation

The symbols \mathbb{R} , \mathbb{R}_0^+ , and \mathbb{R}^+ represent the sets of real numbers, non-negative and positive real numbers, respectively. Similarly, the symbols \mathbb{N} and \mathbb{N}^+ denote the sets of natural numbers, including and excluding zero, respectively. The notation $\mathbb{R}^{n \times m}$ denotes a matrix of size $n \times m$ with real values, while \mathbb{R}^n represents a vector of size n . The symbols A^{-1} and A^T represent the inverse and the transpose of matrix $A \in \mathbb{R}^{n \times n}$, respectively. The notation $[a, b]$ refers to the closed interval between a and b . The identity matrix in $\mathbb{R}^{n \times n}$ is denoted by \mathbb{I}_n . We denote a *symmetric* positive-definite matrix $P \in \mathbb{R}^{n \times n}$ as $P > 0$.

2 Overview of TRUST

Modes of the Tool. TRUST is capable of solving two main types of control problems:

- (i) **Stability problems** – synthesizing a control Lyapunov function and its corresponding controller to enforce the stability of the unknown system. The stability property implies that as time approaches infinity, the system's trajectories converge to the origin, which serves as the equilibrium point.
- (ii) **Safety problems** – synthesizing a control barrier certificate and its corresponding controllers to ensure the safe behavior of the unknown system. The safety property ensures that the system's trajectories, starting from an initial region, do not

¹<https://www.mosek.com/documentation/>

²<https://flask.palletsprojects.com/en/stable/>

³<https://inertiajs.com/>

reach (potentially multiple) unsafe regions within an infinite time horizon.

Classes of Systems. TRUST can support *four classes of systems*: (i) continuous-time nonlinear polynomial systems (ct-NPS), (ii) continuous-time linear systems (ct-LS), (iii) discrete-time nonlinear polynomial systems (dt-NPS), and (iv) discrete-time linear systems (dt-LS). For each of these system classes, TRUST can synthesize both CLF and CBC, along with their respective stability and safety controllers.

Datasets (*.csv, *.txt, *.json). TRUST accepts input datasets that include a *persistently excited* data trajectory, which should be provided in one of the following common file formats: *.csv, *.txt, or *.json. For discrete-time systems, the data should be collected over a time horizon $[0, 1, \dots, T-1]$, where $T \in \mathbb{N}^+$ is the number of collected samples:

$$\mathcal{U}_0^d = [u(0), u(1), u(2), \dots, u(T-1)] \in \mathbb{R}^{m \times T}, \quad (1a)$$

$$\mathcal{X}_0^d = [x(0), x(1), x(2), \dots, x(T-1)] \in \mathbb{R}^{n \times T}, \quad (1b)$$

$$\mathcal{X}_1^d = [x(1), x(2), x(3), \dots, x(T)] \in \mathbb{R}^{n \times T}, \quad (1c)$$

$$\mathcal{N}_0^d = [\mathcal{M}(x(0)), \mathcal{M}(x(1)), \dots, \mathcal{M}(x(T-1))] \in \mathbb{R}^{N \times T}, \quad (1d)$$

where $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$ are the state and input variables of the system. Moreover, $\mathcal{M}(x) \in \mathbb{R}^N$ is a vector of monomials in state $x \in \mathbb{R}^n$, which shapes the dynamics of nonlinear polynomial systems. We call the collected data in (1a)-(1c) a *single input-state trajectory*. Note that the trajectory \mathcal{N}_0^d in (1d) is constructed from the trajectory \mathcal{X}_0^d and the form of the monomial vector $\mathcal{M}(x) \in \mathbb{R}^N$.

Similarly, the *continuous-time data* can be collected over the time interval $[t_0, t_0 + (T-1)\tau]$, with $T \in \mathbb{N}^+$ being the number of collected samples, and $\tau \in \mathbb{R}^+$ as the sampling time:

$$\mathcal{U}_0^c = [u(t_0), u(t_0 + \tau), \dots, u(t_0 + (T-1)\tau)] \in \mathbb{R}^{m \times T}, \quad (2a)$$

$$\mathcal{X}_0^c = [x(t_0), x(t_0 + \tau), \dots, x(t_0 + (T-1)\tau)] \in \mathbb{R}^{n \times T}, \quad (2b)$$

$$\mathcal{X}_1^c = [\dot{x}(t_0), \dot{x}(t_0 + \tau), \dots, \dot{x}(t_0 + (T-1)\tau)] \in \mathbb{R}^{n \times T}, \quad (2c)$$

$$\mathcal{N}_0^c = [\mathcal{M}(x(t_0)) \dots \mathcal{M}(x(t_0 + (T-1)\tau))] \in \mathbb{R}^{N \times T}. \quad (2d)$$

To establish the theoretical aspects of the single-trajectory approach, it is required that the trajectories \mathcal{N}_0^c and \mathcal{N}_0^d for nonlinear polynomial cases, and \mathcal{X}_0^c and \mathcal{X}_0^d for linear cases, have *full row-rank* (cf. Lemmas 3.2, 3.6, 4.2 for different cases). To ensure this, the number of samples T must be greater than N in nonlinear cases and n in linear scenarios. Given that \mathcal{N}_0^c , \mathcal{N}_0^d , \mathcal{X}_0^c and \mathcal{X}_0^d are all derived from sampled data, this condition can be readily verified during data collection. This approach avoids explicitly identifying the system, which would otherwise require $\begin{bmatrix} \mathcal{U}_0^d \\ \mathcal{X}_0^d \end{bmatrix}$ (for discrete-time linear systems) to be full row-rank, see [40].

TRUST *automatically verifies* that the collected data satisfies the following rank condition, ensuring that the data is persistently excited [11, 12]. In case the collected data does not fulfill the required full row-rank condition, the tool will return an error to the user stating: “Error: The collected data must be full row-rank”.

Outputs. The outputs of TRUST depend on the selected mode. For stability problems, TRUST aims to return a CLF $\mathcal{V}(x)$ along with the corresponding stability controller, both derived from the collected

data. For safety problems, it provides a CBC $\mathcal{B}(x)$, along with its initial and unsafe level sets, γ and λ , as well as the corresponding safety controller, all based on the data. If the tool is unable to synthesize either, TRUST will return an error message and guide the user on potential issues.

3 Data-Driven Results for Continuous-Time Systems

Here, we present the corresponding results for the data-driven safety and stability controller synthesis of continuous-time systems with both nonlinear polynomial and linear dynamics. Since our work is a tool paper, we aim to focus on the features and implementation of the software, while we refer to the existing literature for the technical details of the data-driven approach. We remind the reader that the collected data used in this section follows the form described in (2).

3.1 Safety and Stability of ct-NPS

We consider continuous-time nonlinear polynomial systems defined as follows.

Definition 3.1. [ct-NPS] A continuous-time nonlinear polynomial system (ct-NPS) is described by

$$\Sigma^c: \dot{x} = AM(x) + Bu, \quad (3)$$

where $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$ are *unknown* system and control matrices, $\mathcal{M}(x) \in \mathbb{R}^N$ is a vector of monomials in state $x \in X$, and $u \in U$ is a control input, with $X \subset \mathbb{R}^n$, and $U \subset \mathbb{R}^m$ being the state and input sets, respectively.

The following lemma, taken from [30, 41], offers a data-based representation of closed-loop ct-NPS (3.1) with a controller $u = K(x)\mathcal{M}(x)$, where $K(x) \in \mathbb{R}^{m \times N}$ is a matrix polynomial.

LEMMA 3.2 (Data-based Representation of ct-NPS [30, 41]). Let $Q(x)$ be a $(T \times N)$ matrix polynomial such that

$$\mathbb{I}_N = \mathcal{N}_0^c Q(x), \quad (4)$$

with \mathcal{N}_0^c as in (2d) being an $(N \times T)$ full row-rank matrix. If one synthesizes $u = K(x)\mathcal{M}(x) = \mathcal{U}_0^c Q(x)\mathcal{M}(x)$, then the closed-loop system $\dot{x} = AM(x) + Bu$ has the following data-based representation:

$$\dot{x} = \mathcal{X}_1^c Q(x)\mathcal{M}(x), \quad \text{equivalently, } A + BK(x) = \mathcal{X}_1^c Q(x). \quad (5)$$

Using the above lemma, one can obtain a data-driven representation of ct-NPS with unknown matrices A and B as $\mathcal{X}_1^c Q(x)\mathcal{M}(x)$, which will be employed to design controllers for both CBC and CLF in the following theorems. It is worth noting that to ensure \mathcal{N}_0^c has full row rank, the number of samples T must exceed N [41], a condition that can be readily fulfilled during data collection.

The following theorem, borrowed from [30, Theorem 8], leverages the data-driven representation of ct-NPS from Lemma 3.2 to design a CBC $\mathcal{B}(x) = \mathcal{M}(x)^\top P \mathcal{M}(x)$, with $P \in \mathbb{R}^{N \times N} > 0$, and a safety controller $u = \mathcal{U}_0^c Q(x)\mathcal{M}(x)$ based on the collected data.

THEOREM 3.3 (Data-Driven CBC for ct-NPS [30]). Consider the ct-NPS in (3) with unknown matrices A, B , and its data-based representation $\dot{x} = \mathcal{X}_1^c Q(x)\mathcal{M}(x)$. Let $X_I, X_O \subset X$ represent the initial and unsafe regions of the ct-NPS, respectively. Suppose there

Algorithm 1 Data-driven design of *CBC and safety* controller for *ct-NPS*

Require: Regions of interest X, X_I, X_O , collected trajectories $\mathcal{U}_0^c, \mathcal{X}_0^c, \mathcal{X}_1^c$, a choice of monomials $\mathcal{M}(x)$ ¹

- 1: Check that the full row-rank condition for \mathcal{N}_0^c is satisfied
- 2: Solve (6a) and (7c) for P and $H(x)$ simultaneously²
- 3: Given the constructed $H(x)$, solve (7a) and (7b) to design level sets γ and λ , where $\lambda > \gamma$

Ensure: CBC $\mathcal{B}(x) = \mathcal{M}(x)^\top [\mathcal{N}_0^c H(x)]^{-1} \mathcal{M}(x)$ and its corresponding safety controller $u = \mathcal{U}_0^c H(x) [\mathcal{N}_0^c H(x)]^{-1} \mathcal{M}(x)$

exist constants $\gamma, \lambda \in \mathbb{R}^+$, with $\lambda > \gamma$, and a matrix polynomial $H(x) \in \mathbb{R}^{T \times N}$ such that the following constraints are fulfilled:

$$\mathcal{N}_0^c H(x) = P^{-1}, \quad \text{with } P > 0, \quad (6a)$$

$$\mathcal{M}(x)^\top [\mathcal{N}_0^c H(x)]^{-1} \mathcal{M}(x) \leq \gamma, \quad \forall x \in X_I, \quad (6b)$$

$$\mathcal{M}(x)^\top [\mathcal{N}_0^c H(x)]^{-1} \mathcal{M}(x) \geq \lambda, \quad \forall x \in X_O, \quad (6c)$$

$$-\left[\frac{\partial \mathcal{M}}{\partial x} \mathcal{X}_1^c H(x) + H(x)^\top \mathcal{X}_1^{c\top} \left(\frac{\partial \mathcal{M}}{\partial x} \right)^\top \right] \geq 0, \quad \forall x \in X. \quad (6d)$$

Then $\mathcal{B}(x) = \mathcal{M}(x)^\top [\mathcal{N}_0^c H(x)]^{-1} \mathcal{M}(x)$ is a CBC and $u = \mathcal{U}_0^c H(x) [\mathcal{N}_0^c H(x)]^{-1} \mathcal{M}(x)$ is its corresponding safety controller for the unknown *ct-NPS*, ensuring that any trajectory of the *ct-NPS* starting from X_I will not reach X_O within an infinite time horizon.

The following lemma, borrowed from [30, Corollary 11], reformulates conditions (6a)-(6d) as sum-of-squares (SOS) optimization programs, assuming that the regions of interest, X, X_I , and X_O , are semi-algebraic [42]. Specifically, a semi-algebraic set $X \subseteq \mathbb{R}^n$ is described by a vector of polynomials $a(x)$, meaning $X = \{x \in \mathbb{R}^n \mid a(x) \geq 0\}$, with the inequalities applied element-wise.

LEMMA 3.4 (Data-Driven SOS Reformulation of CBC for ct-NPS [30]). Consider the state set X , the initial set X_I , and the unsafe set X_O as semi-algebraic sets, each defined by vectors of polynomial inequalities $g(x)$, $g_I(x)$, and $g_O(x)$, respectively. Suppose there exist constants $\gamma, \lambda \in \mathbb{R}^+$, with $\lambda > \gamma$, a matrix polynomial $H \in \mathbb{R}^{T \times N}$, and vectors of SOS polynomials $L_I(x)$, $L_O(x)$, and $L(x)$ such that the following conditions

$$-\mathcal{M}(x)^\top [\mathcal{N}_0^c H(x)]^{-1} \mathcal{M}(x) - L_I^\top(x) g_I(x) + \gamma, \quad (7a)$$

$$\mathcal{M}(x)^\top [\mathcal{N}_0^c H(x)]^{-1} \mathcal{M}(x) - L_O^\top(x) g_O(x) - \lambda, \quad (7b)$$

$$-\left[\frac{\partial \mathcal{M}}{\partial x} \mathcal{X}_1^c H(x) + H(x)^\top \mathcal{X}_1^{c\top} \left(\frac{\partial \mathcal{M}}{\partial x} \right)^\top \right] - L^\top(x) g(x). \quad (7c)$$

are all SOS polynomials, while condition (6a) is also fulfilled. Then, $\mathcal{B}(x) = \mathcal{M}(x)^\top [\mathcal{N}_0^c H(x)]^{-1} \mathcal{M}(x)$ is a CBC, and $u = \mathcal{U}_0^c H(x) [\mathcal{N}_0^c H(x)]^{-1} \mathcal{M}(x)$ is its corresponding safety controller for the unknown *ct-NPS*.

¹The selection of $\mathcal{M}(x)$ is the choice of the user. Based on the foundational work [11], if an upper bound on the maximum degree of $\mathcal{M}(x)$ can be inferred using physical insights about the unknown system, $\mathcal{M}(x)$ should be chosen to encompass all possible combinations of states up to that upper bound. This ensures that $\mathcal{M}(x)$ potentially contains all monomial terms present in the actual unknown system.

²To satisfy condition (6a), we define $Z = P^{-1}$ and enforce that it is a symmetric positive-definite matrix, i.e., $Z > 0$. Once condition (6a) is met and Z is designed, the matrix P is computed as the inverse of Z , i.e., $P = Z^{-1}$.

Algorithm 2 Data-driven design of *CLF and stability* controller for *ct-NPS*

Require: Collected trajectories $\mathcal{U}_0^c, \mathcal{X}_0^c, \mathcal{X}_1^c$, a choice of monomials $\mathcal{M}(x)$

- 1: Check that the full row-rank condition for \mathcal{N}_0^c is satisfied
- 2: Solve (8a) and (8b) for P and $H(x)$ simultaneously

Ensure: CLF $\mathcal{V}(x) = \mathcal{M}(x)^\top [\mathcal{N}_0^c H(x)]^{-1} \mathcal{M}(x)$ and its corresponding stability controller $u = \mathcal{U}_0^c H(x) [\mathcal{N}_0^c H(x)]^{-1} \mathcal{M}(x)$

REMARK 1. To accommodate an arbitrary number of unsafe regions X_{O_i} , where $i \in \{1, \dots, z\}$, condition (7b) should be repeated and enforced for each distinct unsafe region, a capability fully supported by TRUST.

Algorithm 1 outlines the necessary steps for designing a CBC and its corresponding safety controller based solely on data.

The following theorem, borrowed from [41, Theorem 1], provides the required conditions for designing a CLF $\mathcal{V}(x) = \mathcal{M}(x)^\top P \mathcal{M}(x)$, with $P > 0$, along with a stability controller $u = \mathcal{U}_0^c Q(x) \mathcal{M}(x)$ based on collected data.

THEOREM 3.5 (Data-Driven CLF for ct-NPS [41]). Consider the *ct-NPS* in (3) with unknown matrices A, B , and its data-based representation $\dot{x} = \mathcal{X}_1^c Q(x) \mathcal{M}(x)$. Suppose there exists a polynomial matrix $H(x) \in \mathbb{R}^{T \times N}$ such that the following constrains are satisfied:

$$\mathcal{N}_0^c H(x) = P^{-1}, \quad \text{with } P > 0, \quad (8a)$$

$$-\left[\frac{\partial \mathcal{M}}{\partial x} \mathcal{X}_1^c H(x) + H(x)^\top \mathcal{X}_1^{c\top} \left(\frac{\partial \mathcal{M}}{\partial x} \right)^\top \right] > 0 \quad (8b)$$

Then, $\mathcal{V}(x) = \mathcal{M}(x)^\top [\mathcal{N}_0^c H(x)]^{-1} \mathcal{M}(x)$ is a CLF, and $u = \mathcal{U}_0^c H(x) [\mathcal{N}_0^c H(x)]^{-1} \mathcal{M}(x)$ is its corresponding stability controller for the unknown *ct-NPS*, ensuring that any trajectory of the *ct-NPS* converges to the origin as its equilibrium point when time approaches infinity.

The pseudocode for constructing the CLF and its corresponding stability controllers for *ct-NPS* is outlined in Algorithm 2.

Graphical User Interface (GUI) in TRUST. To maximize accessibility and ensure a highly user-friendly experience, TRUST offers an intuitive, accessible and responsive GUI which utilizes web-based technologies to enable its use without the user having to download or install an application. The GUI enhances usability by abstracting the underlying technical complexities, allowing users to construct either a CLF or CBC using data through a *reactive push-button* interface. Note that for more demanding use cases, or where the data is private, the tool can also be installed on local hardware via Docker. While TRUST provides GUIs for all four system classes, each focusing on either stability or safety properties, we only depict it for *dt-NPS* (see Subsection 4.1), as it offers *the most comprehensive inputs* to the GUI (see Figure 1). Labels in this figure are referenced with $\langle \cdot \rangle$.

As noted earlier, this iteration of TRUST relies on external packages which require a license for their use of MOSEK. Users must upload their license to run the tool $\langle 1 \rangle$. Note that the license is *not* stored; it is only used on a per-session basis, meaning users must re-upload their license for each new session. Moreover, future

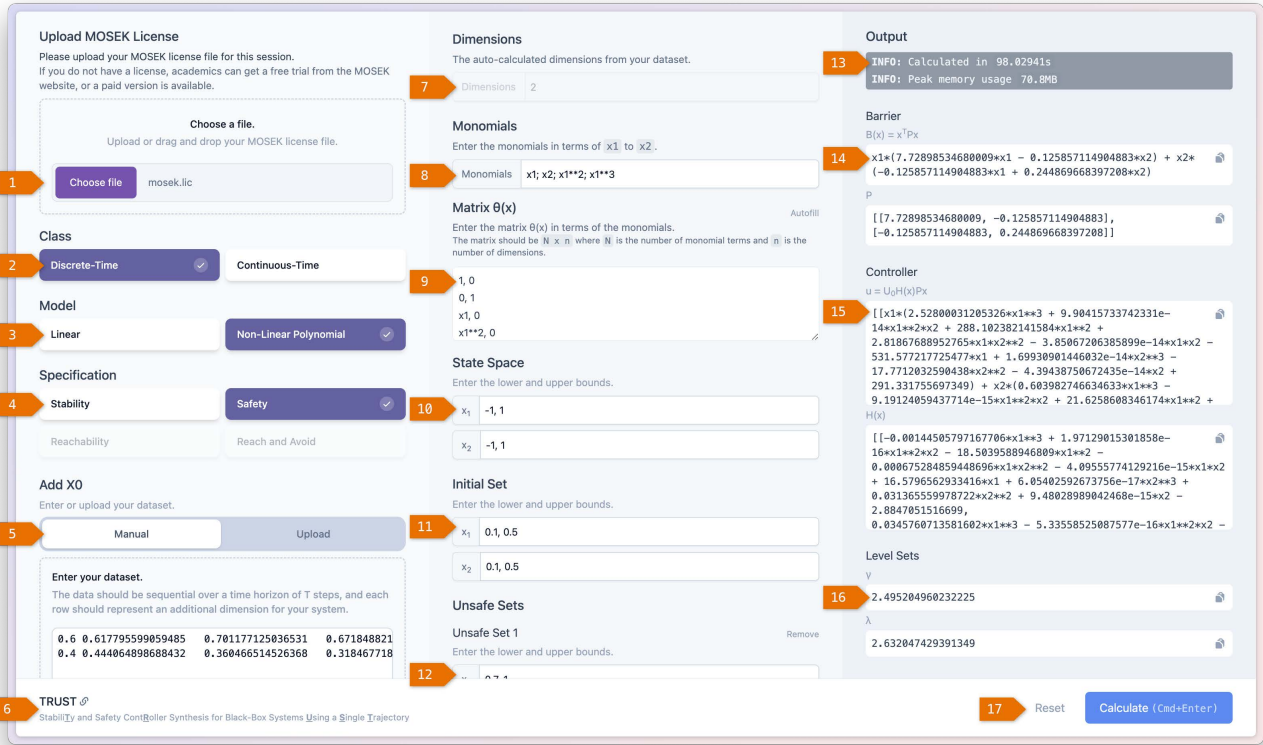


Figure 1: TRUST GUI with numbered annotations indicating respective sections.

iterations of the tool are anticipated to include optimal implementations of other solvers; <https://cvxopt.org/> is already supported by our tool, but given the recognition and performance of MOSEK, as well as the free academic license available, we have focused on optimizing the tool to utilize MOSEK. As future iterations are released, the GUI will include the additional input option to select a preferred solver.

TRUST supports both discrete-time and continuous-time system classes < 2 > as well as linear and nonlinear polynomial models < 3 >. It can design controllers to enforce either stability or safety properties < 4 >. Users may upload datasets for \mathcal{X}_0 , \mathcal{U}_0 , and \mathcal{X}_1 or enter them manually < 5 >. A link to the tool’s source code is provided in the footer < 6 >. The GUI is dynamically rendered, hiding unnecessary elements to avoid user confusion.

The system dimension is automatically detected based on the shape of \mathcal{X}_0 < 7 >. For nonlinear polynomial systems, users must specify the monomials $\mathcal{M}(x)$ < 8 >, separated by semicolons. Note that SymPy notation requires “*” for *multiplication* and “**” for *exponents*. For dt-NPS, users also need to define their desired $\Theta(x)$ (cf. (16)) in < 9 >. If desired, users can enable an “autofill” option for $\Theta(x)$ in < 9 >, allowing TRUST to automatically compute its value (cf. Remark 2). If the property is safety, TRUST uses hypercubes to define the state space < 10 >, initial sets < 11 >, and one or more unsafe sets < 12 >, with both lower and upper bounds required for each dimension. For stability properties, items < 10 > – < 12 > are hidden.

Finally, the output is displayed in the far-right section, showing a brief summary with the total computation time and peak memory usage < 13 >. If TRUST fails to find a valid solution, this is prominently displayed in red in the “INFO” section, and no other output is shown. For successful runs, the CLF or CBC is designed, along with the underlying matrix P < 14 >. This is followed by the controller, including the matrix H < 15 >. For the CBC, the designed level sets are also displayed < 16 >. All results are formatted in Python syntax.

To run the tool, the user clicks the “Calculate” button to initiate computation or “Reset” to clear their input < 17 >. The keyboard shortcut `Cmd+Enter` or `Ctrl+Enter`, depending on the operating system, can also start the computation.

Folder Structure in TRUST. Built with industry-standard software engineering practices, TRUST follows the typical Model-View-Controller folder structure. The root directory contains essential files such as the project LICENSE, a README.md, and configuration documents including `docker-compose.yml`, `requirements.txt`, and the `.env` setup file. The core application logic is organized within the app folder, housing controller classes in `http/controllers/` and data models in `models/`. For the frontend GUI, the vite directory includes the HTML, CSS, and VueJS files, along with configuration settings for building UI components. While the tool does not use a database, storage needs are handled by the storage directory. Unit tests are structured under `tests/`, ensuring comprehensive coverage across components. Docker configurations and startup scripts

are located in the docker folder, enabling straightforward deployment via Dockerfiles and Compose files. This structure not only enhances maintainability but also leverages modern development practices to streamline integration and testing processes.

```

1 TRUST/
2 - LICENSE
3 - README.md
4 - app/
5 - http/
6 - controllers/
7 - dashboard_controller.py
8 - models/
9 - __init__.py
10 - barrier.py
11 - safety_barrier.py
12 - stability.py
13 - docker/
14 - flask
15 - Dockerfile
16 - start-container
17 - supervisord.conf
18 - docker-compose.yml
19 - main.py
20 - node_modules/
21 - requirements.txt
22 - storage/
23 - tests/
24 - __init__.py
25 - http/
26 - models/
27 - pytest.ini
28 - vite/
29 - css/
30 - index.html
31 - js/
32 - main.js
33 - node_modules/
34 - package-lock.json
35 - package.json
36 - postcss.config.js
37 - tailwind.config.js
38 - vite.config.js
    
```

Listing 1: Folder Structure in TRUST.

Error Handling. TRUST is developed as a responsive and reactive Python Flask web application, offering an *intuitive, user-friendly* interface that allows seamless interaction. If a user error occurs, TRUST provides responsive error messages to guide the user in correcting their input. Listed below are some common errors that may be returned to the user:

- (i) For an invalid “state space”, “initial set” or “unsafe set(s)”: “Provided spaces are not valid. Please provide valid lower and upper bounds”.
- (ii) For an invalid shape of $\Theta(x)$: “Theta_x should be of shape (N, n)”.
- (iii) If monomials are provided with commas: “Monomial terms should be split by semicolon”; if they are not suitable for the set dimensions: “Monomials must be in terms of x1 (to xn)”; if some unspecified error has occurred with the monomials: “Invalid monomial terms”.
- (iv) If the rank condition is not met for nonlinear polynomial systems: “The number of samples, T, must be greater than the

number of monomial terms, N”, or “The N0 data is not full row-rank”, depending on which part of the rank condition failed. Similarly for linear systems: “The number of samples, T, must be greater than the number of states, n”, or “The X0 data is not full row-rank”, again depending on which part of the rank condition failed.

- (v) If data files are uploaded with an invalid format: “Unable to parse uploaded file(s)”.
- (vi) If the MOSEK solver cannot find a solution for the given values: “Solution Failure”, with a dynamic error description provided by the solver. If the MOSEK solver did find a solution but the solution does not contain an SOS decomposition: “No SOS decomposition found” with a dynamic error description. Similarly, if the solution does not contain valid SOS constraints: “Constraints are not sum-of-squares”.
- (vii) Any other errors in the tool will be caught with the generic error message: “An unknown error occurred” and a brief description that can be reported.

3.2 Safety and Stability of ct-LS

We consider continuous-time linear systems (ct-LS) defined as

$$\Sigma^c : \dot{x} = Ax + Bu \quad (9)$$

where $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$ are both *unknown*, and $x \in X$ and $u \in U$ represent the system’s state and control input, respectively, with $X \subset \mathbb{R}^n$ and $U \subset \mathbb{R}^m$ being the state and input sets.

The following lemma, borrowed from [11], provides a data-based representation of closed-loop ct-LS with a controller $u = Kx$, where $K \in \mathbb{R}^{m \times n}$.

LEMMA 3.6 (Data-based Representation of ct-LS [11]). *Let Q be a $(T \times n)$ matrix such that*

$$\mathbb{I}_n = X_0^c Q, \quad (10)$$

with X_0^c being a full row-rank matrix. If one synthesizes $u = Kx = \mathcal{U}_0^c Qx$, then the closed-loop system $\dot{x} = Ax + Bu$ has the following data-based representation:

$$\dot{x} = X_1^c Qx, \quad \text{equivalently, } A + BK = X_1^c Q. \quad (11)$$

The following theorem, adapted from [43], utilizes the data-driven representation of ct-LS from Lemma 3.6 to design a CBC $\mathcal{B}(x) = x^\top Px$, with $P > 0$, and a safety controller $u = \mathcal{U}_0^c Qx$ based on the collected data.

THEOREM 3.7 (Data-Driven CBC for ct-LS [43]). *Consider the ct-LS in (9) with unknown matrices A, B , and its data-based representation $\dot{x} = X_1^c Qx$. Let $X_I, X_O \subset X$ represent the initial and unsafe regions of the ct-NPS, respectively. Suppose there exist constants $\gamma, \lambda \in \mathbb{R}^+$, with $\lambda > \gamma$, and a matrix $H \in \mathbb{R}^{T \times n}$ such that the following constraints are fulfilled:*

$$X_0^c H = P^{-1}, \quad \text{with } P > 0, \quad (12a)$$

$$x^\top [X_0^c H]^{-1} x \leq \gamma, \quad \forall x \in X_I, \quad (12b)$$

$$x^\top [X_0^c H]^{-1} x \geq \lambda, \quad \forall x \in X_O, \quad (12c)$$

$$-[X_1^c H + H^\top X_1^{c\top}] \geq 0. \quad (12d)$$

Then $\mathcal{B}(x) = x^\top [X_0^c H]^{-1} x$ is a CBC and $u = \mathcal{U}_0^c H [X_0^c H]^{-1} x$ is its corresponding safety controller for the unknown ct-LS.

Algorithm 3 Data-driven design of *CBC and safety* controller for *ct-LS*

Require: Regions of interest X, X_I, X_O , collected trajectories $\mathcal{U}_0^c, \mathcal{X}_0^c, \mathcal{X}_1^c$

- 1: Check that the full row-rank condition for \mathcal{X}_0^c is satisfied
- 2: Solve (12a) and (12d) for P and H , simultaneously, using SDP solvers
- 3: Given the constructed H , solve (12b) and (12c) via SOS optimization to design γ and λ , where $\lambda > \gamma$

Ensure: CBC $\mathcal{B}(x) = x^\top [\mathcal{X}_0^c H]^{-1} x$ and its corresponding safety controller $u = \mathcal{U}_0^c H [\mathcal{X}_0^c H]^{-1} x$

Since condition (12d) for the ct-LS reduces to a linear matrix inequality (LMI), it can be solved using semidefinite programming (SDP) solvers such as SeDuMi [44]. Algorithm 3 details the steps required to design a CBC and its safety controller based entirely on data for ct-LS.

The following theorem, borrowed from [11], provides the required conditions for designing a CLF $\mathcal{V}(x) = x^\top P x$ with $P > 0$, along with a stability controller $u = \mathcal{U}_0^c Q x$ based on collected data.

THEOREM 3.8 (Data-Driven CLF for ct-LS [11]). Consider the ct-LS in (9) with unknown matrices A, B , and its data-based representation $\dot{x} = \mathcal{X}_1^c Q x$. Suppose there exists a matrix $H \in \mathbb{R}^{T \times n}$ such that the following constraints are satisfied:

$$\mathcal{X}_0^c H = P^{-1}, \quad \text{with } P > 0, \quad (13a)$$

$$\mathcal{X}_1^c H + H^\top \mathcal{X}_1^{c\top} < 0. \quad (13b)$$

Then $\mathcal{V}(x) = x^\top [\mathcal{X}_0^c H]^{-1} x$ is a CLF and $u = \mathcal{U}_0^c H [\mathcal{X}_0^c H]^{-1} x$ is its corresponding stability controller for the unknown ct-LS.

The pseudocode for designing a CLF and its stability controller for ct-LS is provided in Algorithm 4.

TRUST Implementation for ct-LS. For the ct-LS, inputs <8> and <9> in Figure 1 are hidden, as they are not required.

4 Data-Driven Results for Discrete-Time Systems

We now present the corresponding results for the data-driven safety and stability controller synthesis of *discrete-time systems* with both nonlinear polynomial and linear dynamics. We remind the reader that the collected data used in this section follows the form described in (1).

4.1 Safety and Stability of dt-NPS

Definition 4.1 (dt-NPS). A discrete-time nonlinear polynomial system (dt-NPS) is described by

$$\Sigma^d: x^+ = AM(x) + Bu, \quad (14)$$

where $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$ are both *unknown*, $M(x) \in \mathbb{R}^n$ is a vector of monomials in state $x \in X$, and $u \in U$ is a control input, with $X \subset \mathbb{R}^n$ and $U \subset \mathbb{R}^m$ being the state and input sets, respectively. Note that x^+ denotes the state one step ahead, i.e., $x(k+1)$, where $k \in \mathbb{N}$.

We now present the following lemma, borrowed from [31, Lemma 3.2], to obtain a data-based representation of closed-loop dt-NPS

Algorithm 4 Data-driven design of *CLF and stability* controller for *ct-LS*

Require: Collected trajectories $\mathcal{U}_0^c, \mathcal{X}_0^c, \mathcal{X}_1^c$

- 1: Check that the full row-rank condition for \mathcal{X}_0^c is satisfied
- 2: Solve (13a) and (13b) for P and H , simultaneously, using SDP solvers

Ensure: CLF $\mathcal{V}(x) = x^\top [\mathcal{X}_0^c H]^{-1} x$ and its corresponding stability controller $u = \mathcal{U}_0^c H [\mathcal{X}_0^c H]^{-1} x$

in (14) with a controller $u = K(x)x$, where $K(x) \in \mathbb{R}^{m \times n}$ is a matrix polynomial.

LEMMA 4.2 (Data-based Representation of dt-NPS [31]). Let $Q(x)$ be a $(T \times n)$ matrix polynomial such that

$$\Theta(x) = \mathcal{N}_0^d Q(x), \quad (15)$$

where $\Theta(x)$ is an $(N \times n)$ matrix polynomial such that

$$M(x) = \Theta(x)x, \quad (16)$$

and \mathcal{N}_0^d is an $(N \times T)$ full row-rank matrix as in (1d). If one synthesizes $u = K(x)x = \mathcal{U}_0^d Q(x)x$, then the closed-loop system $x^+ = AM(x) + Bu$ has the following data-based representation:

$$x^+ = \mathcal{X}_1^d Q(x)x, \quad \text{equivalently } A\Theta(x) + BK(x) = \mathcal{X}_1^d Q(x). \quad (17)$$

Using the above lemma, we now have a data-driven representation of dt-NPS in the form $\mathcal{X}_1^d Q(x)x$, which is utilized to design a CBC and its safety controller, as outlined in the following theorems [31, Theorem 3.5].

THEOREM 4.3 (Data-Driven CBC for dt-NPS [31]). Consider the dt-NPS in (14) with unknown matrices A, B , and its data-based representation $x^+ = \mathcal{X}_1^d Q(x)x$. Let $X_I, X_O \subset X$ represent the initial and unsafe regions of the dt-NPS, respectively. Suppose there exist constants $\gamma, \lambda \in \mathbb{R}^+$, with $\lambda > \gamma$, and a matrix polynomial $H(x) \in \mathbb{R}^{T \times n}$ such that the following constraints are fulfilled:

$$\mathcal{N}_0^d H(x) = \Theta(x)P^{-1}, \quad \text{with } P > 0, \quad (18a)$$

$$x^\top [\Theta^\dagger \mathcal{N}_0^d H(x)]^{-1} x \leq \gamma, \quad \forall x \in X_I, \quad (18b)$$

$$x^\top [\Theta^\dagger \mathcal{N}_0^d H(x)]^{-1} x \geq \lambda, \quad \forall x \in X_O, \quad (18c)$$

$$\begin{bmatrix} P^{-1} & \mathcal{X}_1^d H(x) \\ H(x)^\top \mathcal{X}_1^{d\top} & P^{-1} \end{bmatrix} \geq 0, \quad \forall x \in X. \quad (18d)$$

Then $\mathcal{B}(x) = x^\top [\Theta^\dagger \mathcal{N}_0^d H(x)]^{-1} x$ is a CBC and $u = \mathcal{U}_0^d H(x) [\Theta^\dagger \mathcal{N}_0^d H(x)]^{-1} x$ is its corresponding safety controller for the unknown dt-NPS, with Θ^\dagger being the left pseudoinverse of Θ .

The SOS optimization program can be applied to enforce conditions (18a)-(18d), similar to those in Lemma 3.4. The corresponding pseudocode for designing a CBC and its safety controller is provided in Algorithm 5.

REMARK 2. Note that the choice of $\Theta(x)$ satisfying the equality condition in (16) is not unique, and different choices may influence the proposed conditions in Theorem 4.3. To accommodate this flexibility, we allow the user to input this $\Theta(x)$ as an additional parameter (see < 9 > in Figure 1). Alternatively, the user can enable the autofill option, prompting the tool to solve (16) and design $\Theta(x)$ automatically.

Algorithm 5 Data-driven design of *CBC and safety* controller for *dt-NPS*

Require: Regions of interest X, X_I, X_O , collected trajectories $\mathcal{U}_0^d, \mathcal{X}_0^d, \mathcal{X}_1^d$, a choice of monomials $\mathcal{M}(x)$

- 1: Check that the full row-rank condition for \mathcal{N}_0^d is satisfied
- 2: Provide $\Theta(x)$ or select the autofill option to let TRUST solve it based on (16)
- 3: Solve (18a) and (18d) for P and $H(x)$, simultaneously³
- 4: Given the constructed $H(x)$, solve (18b) and (18c) to design γ and λ , where $\lambda > \gamma$

Ensure: CBC $\mathcal{B}(x) = x^\top [\Theta^\dagger \mathcal{N}_0^d H(x)]^{-1} x$ and its corresponding safety controller $u = \mathcal{U}_0^d H(x) [\Theta^\dagger \mathcal{N}_0^d H(x)]^{-1} x$

The following theorem, adapted from [31], provides the sufficient conditions for designing a CLF $\mathcal{V}(x) = x^\top P x$, with $P > 0$, along with a stability controller $u = \mathcal{U}_0^d Q(x)x$ based on collected data.

THEOREM 4.4 (Data-Driven CLF for dt-NPS [31]). *Consider the dt-NPS in (14) with unknown matrices A, B , and its data-based representation $x^+ = \mathcal{X}_1^d Q(x)x$. Suppose there exists a polynomial matrix $H(x) \in \mathbb{R}^{T \times n}$ such that the following constrains are satisfied:*

$$\mathcal{N}_0^d H(x) = \Theta(x) P^{-1}, \quad \text{with } P > 0, \quad (19a)$$

$$\begin{bmatrix} P^{-1} & \mathcal{X}_1^d H(x) \\ H(x)^\top \mathcal{X}_1^{d\top} & P^{-1} \end{bmatrix} > 0. \quad (19b)$$

Then $\mathcal{V}(x) = x^\top [\Theta^\dagger \mathcal{N}_0^d H(x)]^{-1} x$ is a CLF and $u = \mathcal{U}_0^d H(x) [\Theta^\dagger \mathcal{N}_0^d H(x)]^{-1} x$ is its corresponding stability controller for the unknown dt-NPS, with Θ^\dagger being the left pseudoinverse of Θ .

The pseudocode for constructing the CLF and its stability controllers for dt-NPS is outlined in Algorithm 6.

4.2 Safety and Stability of dt-LS

We consider discrete-time linear systems (dt-LS), defined as

$$\Sigma^d : x^+ = Ax + Bu, \quad (20)$$

where $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$ are both *unknown*.

Similar to Lemma 3.6 under condition (10) and full row-rank assumption of \mathcal{X}_0^d , it can be shown that the closed-loop system $x^+ = Ax + Bu$ has the following data-based representation [11, Theorem 2]:

$$x^+ = \mathcal{X}_1^d Q x, \quad \text{equivalently, } A + BK = \mathcal{X}_1^d Q. \quad (21)$$

The following theorem, adapted from [11, Theorem 2], utilizes the data-driven representation of dt-LS to design a CBC $\mathcal{B}(x) = x^\top P x$, with $P > 0$, and its safety controller $u = \mathcal{U}_0^d Q x$ based on the collected data.

THEOREM 4.5 (Data-Driven CBC for dt-LS [11]). *Consider the dt-LS in (20) with unknown matrices A, B , and its data-based representation $x^+ = \mathcal{X}_1^d Q x$. Let $X_I, X_O \subset X$ represent the initial and unsafe regions of dt-LS, respectively. Suppose there exist constants*

³To satisfy conditions (18a) and (18d), we define $Z = P^{-1}$ and enforce that it is a symmetric positive-definite matrix, i.e., $Z > 0$. Once conditions (18a),(18d) are met and Z is designed, the matrix P is computed as the inverse of Z , i.e., $P = Z^{-1}$.

Algorithm 6 Data-driven design of *CLF and stability* controllers for *dt-NPS*

Require: collected trajectories $\mathcal{U}_0^d, \mathcal{X}_0^d, \mathcal{X}_1^d$, a choice of monomials $\mathcal{M}(x)$

- 1: Check that the full row-rank condition for \mathcal{N}_0^d is satisfied
- 2: Provide $\Theta(x)$ or select the autofill option to let TRUST solve it based on (16)
- 3: Solve (19a) and (19b) for P and $H(x)$, simultaneously

Ensure: CLF $\mathcal{V}(x) = x^\top [\Theta^\dagger \mathcal{N}_0^d H(x)]^{-1} x$ and its corresponding stability controller $u = \mathcal{U}_0^d H(x) [\Theta^\dagger \mathcal{N}_0^d H(x)]^{-1} x$

$\gamma, \lambda \in \mathbb{R}^+$, with $\lambda > \gamma$, and a matrix $H \in \mathbb{R}^{T \times n}$ such that the following constraints are fulfilled:

$$\mathcal{X}_0^d H = P^{-1}, \quad \text{with } P > 0, \quad (22a)$$

$$x^\top [\mathcal{X}_0^d H]^{-1} x \leq \gamma, \quad \forall x \in X_I, \quad (22b)$$

$$x^\top [\mathcal{X}_0^d H]^{-1} x \geq \lambda, \quad \forall x \in X_O, \quad (22c)$$

$$\begin{bmatrix} P^{-1} & \mathcal{X}_1^d H \\ H^\top \mathcal{X}_1^{d\top} & P^{-1} \end{bmatrix} \geq 0. \quad (22d)$$

Then $\mathcal{B}(x) = x^\top [\mathcal{X}_0^d H]^{-1} x$ is a CBC and $u = \mathcal{U}_0^d H [\mathcal{X}_0^d H]^{-1} x$ is its corresponding safety controller for the unknown dt-LS.

The linear matrix (in)equalities in (22a), (22d) can be solved using SDP solvers such as SeDuMi [44], while conditions (22b), (22c) can be solved using SOSTOOLS [45]. The pseudocode for designing a CBC and its safety controller for dt-LS is provided in Algorithm 7.

The following theorem, borrowed from [11, Theorem 2], provides the required conditions for designing a CLF $\mathcal{V}(x) = x^\top P x$, with $P > 0$, along with a stability controller $u = \mathcal{U}_0^d Q x$ based on collected data.

THEOREM 4.6 (Data-Driven CLF for dt-LS [11]). *Consider the dt-LS in (20) with unknown matrices A, B , and its data-based representation $x^+ = \mathcal{X}_1^d Q x$. Suppose there exists a matrix $H \in \mathbb{R}^{T \times n}$ such that the following constrains are satisfied:*

$$\mathcal{X}_0^d H = P^{-1}, \quad \text{with } P > 0, \quad (23a)$$

$$\begin{bmatrix} P^{-1} & \mathcal{X}_1^d H \\ H^\top \mathcal{X}_1^{d\top} & P^{-1} \end{bmatrix} \geq 0. \quad (23b)$$

Then $\mathcal{V}(x) = x^\top [\mathcal{X}_0^d H]^{-1} x$ is a CLF and $u = \mathcal{U}_0^d H [\mathcal{X}_0^d H]^{-1} x$ is its corresponding stability controller for the unknown dt-LS.

The pseudocode for designing a CLF and its stability controller for dt-LS is provided in Algorithm 8.

TRUST Implementation for dt-LS. For the dt-LS, inputs <8> and <9> in Figure 1 are hidden, as they are not required.

5 Benchmarks and Evaluations

We demonstrate the effectiveness of TRUST through a series of physical benchmarks, covering the *four classes* of dynamical systems and showcasing their respective *stability or safety* properties. The mathematical models for all case studies are provided in the Appendix. However, we assume these models are unknown, relying solely on the trajectories collected from them for analysis. Table 1

Algorithm 7 Data-driven design of *CBC and safety* controller for *dt-LS***Require:** Regions of interest X, X_I, X_O , collected trajectories $\mathcal{U}_0^d, \mathcal{X}_0^d, \mathcal{X}_1^d$

- 1: Check that the full row-rank condition for \mathcal{X}_0^d is satisfied
- 2: Solve (22a) and (22d) for P and H , simultaneously
- 3: Given the constructed H , solve (22b) and (22c) via SOS optimization to design γ and λ , where $\lambda > \gamma$

Ensure: CBC $\mathcal{B}(x) = x^\top [\mathcal{X}_0^d H]^{-1} x$ and its corresponding safety controller $u = \mathcal{U}_0^d H [\mathcal{X}_0^d H]^{-1} x$

presents the results for the construction of CBC and safety controllers, while Table 2 shows the results for the design of CLF and stability controllers.

Details of the simulations, including the number of collected samples, computation time, and memory usage, are reported in both tables. As observed, solving linear cases is very fast (under a second for 2-dimensional cases) due to the use of *semidefinite programs* for satisfying the required conditions (apart from designing level sets for CBC if the property is safety-related). Nonlinear cases, however, require more computation time, as expected, due to solving SOS *programs* and designing Lagrange multipliers in addition to CBC and CLF. For high-dimensional linear cases (4, 6, and 8 dimensions), solving the stability problem is pretty fast using semidefinite programs. However, the safety problem takes longer due to the need to design level sets for the CBC using an SOS optimization program. Simulation results for different case studies, illustrating the design of CBCs and CLFs, are presented in Figures 2 and 3, respectively. Another observation is that, although both ct-NPS models—the Lotka-Volterra Predator-Prey Model and the Van der Pol Oscillator—have a dimension of two, the computational time and memory usage for the latter are greater due to having higher-degree monomial terms. A future update to TRUST could include optimizing certain SOS algorithms to speed up the tool for solving nonlinear cases with potentially higher-order monomials. Additionally, whilst the tool is currently designed for scenarios without noise, future iterations of the tool will increase the scope to handle noisy data.

6 Conclusion

We developed an open-source software tool, TRUST, for *data-driven controller synthesis* of dynamical systems with *unknown* mathematical models, to guarantee stability or safety properties. Using only a *single input-state trajectory* from the unknown system and by meeting a rank condition for persistent excitation, TRUST is designed to construct either *control Lyapunov functions* or *control barrier certificates*, along with corresponding stability or safety controllers. The tool employs SOS optimization programs based solely on data to enforce these properties across four system classes: (i) *continuous-time nonlinear polynomial systems*, (ii) *continuous-time linear systems*, (iii) *discrete-time nonlinear polynomial systems*, and (iv) *discrete-time linear systems*. We applied TRUST to a set of physical benchmarks with unknown dynamics, ensuring their stability or safety properties within the supported model classes. Future directions involve

Algorithm 8 Data-driven design of *CLF and stability* controller for *dt-LS***Require:** Collected trajectories $\mathcal{U}_0^d, \mathcal{X}_0^d, \mathcal{X}_1^d$

- 1: Check that the full row-rank condition for \mathcal{X}_0^d is satisfied
- 2: Solve (23a) and (23b) for P and H , simultaneously

Ensure: CLF $\mathcal{V}(x) = x^\top [\mathcal{X}_0^d H]^{-1} x$ and its corresponding stability controller $u = \mathcal{U}_0^d H [\mathcal{X}_0^d H]^{-1} x$

extending TRUST to support a *wider range of nonlinear systems* and to accommodate unknown systems with *stochastic dynamics*.

7 Acknowledgment

The authors would like to thank the MOSEK Team for their support, which allowed the use of the academic MOSEK license as part of the web application for our tool. Additionally, the authors are grateful to Chenyang Yuan for his assistance with the SOS toolbox [46]. Ben Wooding is supported by an EPSRC Doctoral Prize Research Fellowship.

References

- [1] J. McGregor, D. Gluch, and P. Feiler, "Analysis and design of safety-critical, cyber-physical systems," *ACM SIGAda Ada Letters*, vol. 36, no. 2, pp. 31–38, 2017.
- [2] Z.-S. Hou and Z. Wang, "From model-based control to data-driven control: Survey, classification and perspective," *Information Sciences*, vol. 235, pp. 3–35, 2013.
- [3] F. Dörfler, J. Coulson, and I. Markovskiy, "Bridging direct and indirect data-driven control formulations via regularizations and relaxations," *IEEE Transactions on Automatic Control*, vol. 68, no. 2, pp. 883–897, 2022.
- [4] H. Khalil, "Nonlinear systems," 2002.
- [5] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *Hybrid Systems: Computation and Control*. Springer Berlin Heidelberg, 2004, pp. 477–492.
- [6] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control Barrier Functions: Theory and Applications," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 3420–3431.
- [7] W. Xiao, C. G. Cassandras, and C. Belta, *Safe Autonomy with Control Barrier Functions: Theory and Applications*. Springer, 2023.
- [8] G. C. Calafiore and M. C. Campi, "The scenario approach to robust control design," *IEEE Transactions on automatic control*, vol. 51, no. 5, pp. 742–753, 2006.
- [9] M. C. Campi, S. Garatti, and M. Prandini, "The scenario approach for systems and control design," *Annual Reviews in Control*, vol. 33, no. 2, pp. 149–157, 2009.
- [10] P. Mohajerin Esfahani, T. Sutter, and J. Lygeros, "Performance bounds for the scenario approach and an extension to a class of non-convex programs," *IEEE Transactions on Automatic Control*, vol. 60, no. 1, pp. 46–58, 2014.
- [11] C. De Persis and P. Tesi, "Formulas for data-driven control: Stabilization, optimality, and robustness," *IEEE Transactions on Automatic Control*, vol. 65, no. 3, pp. 909–924, 2019.
- [12] J. C. Willems, P. Rapisarda, I. Markovskiy, and B. L. De Moor, "A note on persistency of excitation," *Systems & Control Letters*, vol. 54, no. 4, pp. 325–329, 2005.
- [13] J. Kenanian, A. Balkan, R. M. Jungers, and P. Tabuada, "Data driven stability analysis of black-box switched linear systems," *Automatica*, vol. 109, 2019.
- [14] N. Boffi, S. Tu, N. Matni, J. J. Slotine, and V. Sindhvani, "Learning stability certificates from data," in *Proceedings of Conference on Robot Learning*, 2021, pp. 1341–1350.
- [15] A. Lavaei, P. M. Esfahani, and M. Zamani, "Data-driven stability verification of homogeneous nonlinear systems with unknown dynamics," in *61st Conference on Decision and Control (CDC)*, 2022, pp. 7296–7301.
- [16] A. Lavaei and D. Angeli, "Data-driven stability certificate of interconnected homogeneous networks via ISS properties," *IEEE Control Systems Letters*, vol. 7, pp. 2395–2400, 2023.
- [17] A. Nejati, A. Lavaei, P. Jagtap, S. Soudjani, and M. Zamani, "Formal verification of unknown discrete- and continuous-time systems: A data-driven approach," *IEEE Transactions on Automatic Control*, vol. 68, no. 5, pp. 3011–3024, 2023.
- [18] L. Lindemann, H. Hu, A. Robey, H. Zhang, D. Dimarogonas, S. Tu, and N. Matni, "Learning hybrid control barrier functions from data," in *Conference on robot learning*, 2021, pp. 1351–1370.
- [19] A. Nejati and M. Zamani, "Data-driven synthesis of safety controllers via multiple control barrier certificates," *IEEE Control Systems Letters*, vol. 7, pp. 2497–2502, 2023.

Table 1: Data-driven design of CBCs and safety controllers. The symbol n denotes the dimension, while T represents the number of collected samples. All cases were run on a MacBook Pro (Apple M3 Max with 36 GB RAM).

Experiment Name	System	n	T	γ	λ	Time (s)	Memory (MB)
Lotka-Volterra Predator-Prey Model	ct-NPS	2	12	0.11	0.14	50.92	45.0
Van der Pol Oscillator	ct-NPS	2	15	5.73	14.26	345.79	105.5
DC Motor	ct-LS	2	15	3.11	3.43	0.53	1.2
Room Temperature System 1	ct-LS	2	15	631.02	1838.62	0.34	0.8
Two Tank System	ct-LS	2	12	5.48	6.17	0.35	10.3
High Order 4	ct-LS	4	16	13,910.89	14,374.75	5.98	6.2
High Order 6	ct-LS	6	16	20,072.83	20,600.83	91.06	17.0
High Order 8	ct-LS	8	20	47,437.80	66,261.32	1008.80	44.0
Lotka-Volterra Predator Prey	dt-NPS	2	12	0.46	0.57	59.86	58.3
Lorenz Attractor	dt-NPS	3	12	1,052.68	3,931.40	948.52	323.7
DC Motor	dt-LS	2	15	0.64	0.70	0.61	1.4
Room Temperature System 1	dt-LS	2	15	8.87	9.07	0.77	1.8
Room Temperature System 2	dt-LS	3	15	8.02	12.00	2.58	9.5
Two Tank System	dt-LS	2	8	1.80	3.17	1.05	2.3
High Order 4	dt-LS	4	16	261.63	264.94	5.59	5.6
High Order 6	dt-LS	6	16	22.50	23.14	85.01	16.0
High Order 8	dt-LS	8	20	21.93	30.46	1003.39	48.5

Table 2: Data-driven design of CLFs and stability controllers. The symbol n denotes the dimension, while T represents the number of collected samples. All cases were run on a MacBook Pro (Apple M3 Max with 36 GB RAM).

Experiment Name	System	n	T	Time (s)	Memory (MB)
Lotka-Volterra Predator-Prey Model	ct-NPS	2	12	49.76	33.8
Van der Pol Oscillator	ct-NPS	2	15	347.67	139.7
DC Motor	ct-LS	2	15	0.02	0.1
Room Temperature System 1	ct-LS	2	15	0.05	0.1
Two Tank System	ct-LS	2	12	0.06	0.1
High Order 4	ct-LS	4	16	0.10	0.2
High Order 6	ct-LS	6	16	0.17	0.4
High Order 8	ct-LS	8	16	0.35	0.8
Academic System	dt-NPS	2	12	44.72	43.9
Lorenz Attractor	dt-NPS	3	12	720.81	251
DC Motor	dt-LS	2	15	0.06	0.2
Room Temperature System 1	dt-LS	2	15	0.05	0.1
Room Temperature System 2	dt-LS	3	15	0.08	0.2
Two Tank System	dt-LS	2	8	0.06	0.1
High Order 4	dt-LS	4	16	0.15	0.3
High Order 6	dt-LS	6	16	0.29	0.6
High Order 8	dt-LS	8	16	0.49	1.2

- [20] A. Aminzadeh, M. Ashoori, A. Nejati, and A. Lavaei, "A physics-informed scenario approach with data mitigation for safety verification of nonlinear systems," *arXiv:2412.03932*, 2024.
- [21] M. Kazemi, R. Majumdar, M. Salamaty, S. Soudjani, and B. Wooding, "Data-driven abstraction-based control synthesis," *Nonlinear Analysis: Hybrid Systems*, vol. 52, p. 101467, 2024.
- [22] A. Makdesi, A. Girard, and L. Fribourg, "Data-driven models of monotone systems," *IEEE Transactions on Automatic Control*, 2023.
- [23] A. Devonport, A. Saoud, and M. Arcak, "Symbolic abstractions from data: A pac learning approach," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 599–604.
- [24] R. Coppola, A. Peruffo, and M. Mazo, "Data-driven abstractions for verification of deterministic systems," *arXiv:2211.01793*, 2022.
- [25] D. Ajeleye, A. Lavaei, and M. Zamani, "Data-driven controller synthesis via finite abstractions with formal guarantees," *IEEE Control Systems Letters*, vol. 7, pp. 3453–3458, 2023.
- [26] J. Berberich, J. Köhler, M. A. Müller, and F. Allgöwer, "Data-driven model predictive control with stability and robustness guarantees," *IEEE Transactions on Automatic Control*, vol. 66, no. 4, pp. 1702–1717, 2020.
- [27] X. Dai, C. De Persis, and N. Monshizadeh, "Data-driven optimal output feedback control of linear systems from input-output data," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 1376–1381, 2023.
- [28] A. Luppi, A. Bisoffi, C. De Persis, and P. Tesi, "Data-driven design of safe control for polynomial systems," *European Journal of Control*, vol. 75.
- [29] M. Zaker, D. Angeli, and A. Lavaei, "Certified learning of incremental iss controllers for unknown nonlinear polynomial dynamics," *arXiv:2412.03901*, 2024.
- [30] A. Nejati, B. Zhong, M. Caccamo, and M. Zamani, "Data-driven controller synthesis of unknown nonlinear polynomial systems via control barrier certificates," in *Learning for Dynamics and Control Conference*, 2022, pp. 763–776.
- [31] B. Samari, O. Akbarzadeh, M. Zaker, and A. Lavaei, "From a single trajectory to safety controller synthesis of discrete-time nonlinear polynomial systems," vol. 8, pp. 3123–3128, 2024.
- [32] B. Wooding and A. Lavaei, "Learning k-inductive control barrier certificates for unknown nonlinear dynamics beyond polynomials," *arXiv:2412.07232*, 2024.
- [33] J. Björnsson, S. Gudmundsson, and S. Hafstein, "Class library in C++ to compute lyapunov functions for nonlinear systems," *IFAC-PapersOnLine*, vol. 48, no. 11, pp. 778–783, 2015.
- [34] J. Liu, Y. Meng, M. Fitzsimmons, and R. Zhou, "TOOL LyZNet: A lightweight python tool for learning and verifying neural lyapunov functions and regions of attraction," in *Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control*, 2024, pp. 1–8.
- [35] A. Edwards, A. Peruffo, and A. Abate, "Fossil 2.0: Formal certificate synthesis for the verification and control of dynamical models," in *Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control*, 2024, pp.

- 1–10.
- [36] B. Wooding, V. Horbanov, and A. Lavaei, "PRoTECT: Parallelized construction of safety barrier certificates for nonlinear polynomial systems," *arXiv preprint arXiv:2404.14804*, 2024.
- [37] S. Prajna, A. Papachristodoulou, and P. A. Parrilo, "Introducing SOSTOOLS: A general purpose sum of squares programming solver," in *Proceedings of the 41st IEEE Conference on Decision and Control*, 2002., vol. 1, 2002, pp. 741–746.
- [38] J. Bucanek, "Model-view-controller pattern," *Learn Objective-C for Java Developers*, pp. 353–402, 2009.
- [39] K. Beck, *Test driven development: By example*. Addison-Wesley Professional, 2022.
- [40] C. De Persis, M. Rotulo, and P. Tesi, "Learning controllers from data via approximate nonlinearity cancellation," *IEEE Transactions on Automatic Control*, vol. 68, no. 10, pp. 6082–6097, 2023.
- [41] M. Guo, C. De Persis, and P. Tesi, "Learning control for polynomial systems using sum of squares relaxations," in *2020 59th IEEE conference on decision and control (CDC)*, 2020, pp. 2436–2441.
- [42] J. Bochnak, M. Coste, and M.-F. Roy, *Real algebraic geometry*. Springer Science & Business Media, 2013, vol. 36.
- [43] H. Wang, K. Margellos, A. Papachristodoulou, and C. De Persis, "Convex co-design of control barrier functions and feedback controllers for linear systems," 2024.
- [44] J. F. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization methods and software*, vol. 11, no. 1-4, pp. 625–653, 1999.
- [45] S. Prajna, A. Papachristodoulou, P. Seiler, and P. A. Parrilo, "SOSTOOLS: Control applications and new developments," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2004, pp. 315–320.
- [46] C. Yuan, "SumOfSquares.py." [Online]. Available: <https://github.com/yuanchenyang/SumOfSquares.py>
- [47] P. J. Wangersky, "Lotka-volterra population models," *Annual Review of Ecology and Systematics*, vol. 9, pp. 189–218, 1978.
- [48] A. Abate, H. Blom, N. Cauchi, J. Delicaris, A. Hartmanns, M. Khaled, A. Lavaei, C. Pilch, A. Remke, S. Schupp, F. Shmarov, S. Soudjani, A. Vinod, B. Wooding, M. Zamani, and P. Zuliani, "ARCH-COMP20 Category Report: Stochastic Models," 2020.
- [49] P. A. Adewuyi, "Dc motor speed control: A case between pid controller and fuzzy logic controller," *international journal of multidisciplinary sciences and engineering*, vol. 4, no. 4, pp. 36–40, 2013.
- [50] A. Abate, D. Ahmed, A. Edwards, M. Giacobbe, and A. Peruffo, "FOSSIL: a software tool for the formal synthesis of lyapunov functions and barrier certificates using neural networks," in *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, 2021, pp. 1–11.
- [51] J. A. Ramos and P. L. Dos Santos, "Mathematical modeling, system identification, and controller design of a two tank system," in *2007 46th IEEE Conference on Decision and Control*. IEEE, 2007, pp. 2838–2843.

8 Appendix

The mathematical models for each case study are provided below. Note that these models are assumed to be unknown, and our analysis relies solely on the trajectories collected from them.

ct-NPS: Lotka-Volterra Predator-Prey Model. The Lotka-Volterra equations are to describe the dynamics of biological systems in which two species interact [47]. The state variable x_1 represents the population density of the prey and x_2 the population density of the predator:

$$\Sigma^c : \begin{cases} \dot{x}_1(t) = \alpha x_1(t) - \beta x_1(t)x_2(t) - u_1(t), \\ \dot{x}_2(t) = -\eta x_2(t) + \delta x_1(t)x_2(t) + u_2(t), \end{cases}$$

which is of the form (3) with

$$A = \begin{bmatrix} \alpha & 0 & -\beta \\ 0 & -\eta & \delta \end{bmatrix}, \quad B = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \quad M(x) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_1(t)x_2(t) \end{bmatrix},$$

where $\alpha = \eta = 0.6$ are the prey growth and predator death rates, while $\beta = \delta = 1$ are the effect of the presence of the predator and prey on each other. The inputs above represent a pest control scenario where either pesticides or the introduction of predators is used to manage the prey population. For safety problems, the

regions of interest are given as follows: state space $X = [-2, 2] \times [-1, 1]$, initial set $X_I = [0.5, 1] \times [0.2, 0.4]$ and unsafe sets $X_O = [-2, -1.5] \times [0.8, 1] \cup [-2, -1.5] \times [-1, -0.8] \cup [1.6, 2] \times [0.85, 1] \cup [1.6, 2] \times [-1, -0.8]$.

ct-NPS: Van der Pol Oscillator. We consider the Van der Pol oscillator benchmark from the ARCH competition [48], with the following dynamics:

$$\Sigma^c : \begin{cases} \dot{x}_1(t) = x_2(t), \\ \dot{x}_2(t) = -x_1(t) + (1 - x_1(t)^2)x_2(t) + u(t), \end{cases}$$

which is of the form (3) with

$$A = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 1 & -1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 1 \end{bmatrix}^\top, \quad M(x) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_1^2(t)x_2(t) \end{bmatrix}.$$

For safety problems, we consider $X = [-2, 2]^2$, $X_I = [-0.2, 0.2]^2$ and $X_O = [-2, -1.5]^2 \cup [1.5, 2]^2$.

ct-LS: DC Motor. We consider a DC Motor, based on [49], with the dynamics

$$\Sigma^c : \begin{cases} \dot{x}_1 = -(\frac{R}{L}x_1 + \frac{k_{dc}}{L}x_2 + u_1), \\ \dot{x}_2 = \frac{k_{dc}}{J}x_1 - \frac{b}{J}x_2 + u_2, \end{cases} \quad (24)$$

which is of the form (9) with

$$A = \begin{bmatrix} -\frac{R}{L} & -\frac{k_{dc}}{L} \\ \frac{k_{dc}}{J} & -\frac{b}{J} \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

where $x_1, x_2, R = 1, L = 0.01, J = 0.01$ are the armature current, the rotational speed of the shaft, the electrical resistance, the electrical inductance, and the moment of inertia of the rotor, respectively. In addition, $b = 1$, and $k_{dc} = 0.01$, represent, respectively, the motor torque, and the back electromotive force. For safety problems, the state space is given as $X = [-1, 1]^2$, with initial set $X_I = [0.1, 0.4] \times [0.1, 0.55]$ and four unsafe sets $X_O = [0.5, 1] \times [0.6, 1] \cup [-1, -0.6] \times [0.6, 1] \cup [-1, -0.7] \times [-1, -0.6] \cup [0.6, 1] \times [-1, -0.6]$.

ct-LS: Room Temperature System 1. We consider the two-room system [50], with dynamics

$$\Sigma^c : \begin{cases} \dot{x}_1 = -(\alpha + \alpha_{e1})x_1 + \alpha x_2 + \alpha_{e1}u, \\ \dot{x}_2 = -(\alpha + \alpha_{e2})x_2 + \alpha x_1 + \alpha_{e2}u, \end{cases} \quad (25)$$

which is of the form (9) with

$$A = \begin{bmatrix} -(\alpha + \alpha_{e1}) & \alpha \\ \alpha & -(\alpha + \alpha_{e2}) \end{bmatrix}, \quad B = \begin{bmatrix} \alpha_{e1} \\ \alpha_{e2} \end{bmatrix},$$

where heat exchange constants are $\alpha = 5 \times 10^{-2}$, $\alpha_{e1} = 5 \times 10^{-3}$, $\alpha_{e2} = 8 \times 10^{-3}$. For safety problems, the state space is provided as $X = [-2, 3]^2$, with the initial set $X_I = [-0.5, 0.5]^2$ and two unsafe sets $X_O = [-2, -1] \times [2, 3] \cup [2, 3] \times [-2, -1]$.

ct-LS: Two Tank System. Consider a two-tank system [51], characterized by differential equations

$$\Sigma^c : \begin{cases} \dot{x}_1 = -\frac{\alpha_1}{a_1}x_1 + \frac{u_1}{a_1}, \\ \dot{x}_2 = \frac{\alpha_1}{a_2}x_1 - \frac{\alpha_2}{a_2}x_2 + \frac{u_2}{a_2}, \end{cases} \quad (26)$$

which is of the form (9) with

$$A = \begin{bmatrix} -\frac{\alpha_1}{a_1} & 0 \\ \frac{\alpha_1}{a_2} & -\frac{\alpha_2}{a_2} \end{bmatrix}, \quad B = \begin{bmatrix} \frac{1}{a_1} & 0 \\ 0 & \frac{1}{a_2} \end{bmatrix},$$

where x_1, x_2 are heights of the fluid in two tanks. Additionally, α_i and a_i are the valve coefficient and area of the tank i , and u_1 and u_2 are the inflow and outflow rate of tank 1 and 2, respectively. Furthermore, $\alpha_1 = \alpha_2 = a_1 = a_2 = 2$. For safety problems, we consider the state space $X = [-3, 3]^2$, initial set $X_I = [-1, 1]^2$ and two unsafe sets $X_O = [1.5, 3]^2 \cup [-3, -1.5]^2$.

ct-LS: 4D Academic Example. We consider the following 4-dimensional benchmark adapted from [50]

$$\Sigma^c : \begin{cases} \dot{x}_1(t) = x_2(t), \\ \dot{x}_2(t) = x_3(t), \\ \dot{x}_3(t) = x_4(t), \\ \dot{x}_4(t) = -3980x_4(t) - 4180x_3(t) - 2400x_2(t) - 576x_1(t) + u(t), \end{cases}$$

which is of the form (9) with

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -576 & -2400 & -4180 & -3980 \end{bmatrix}, \\ B = [0 \ 0 \ 0 \ 0 \ 0 \ 1]^\top,$$

with the state space $X = [-2, 2]^4$, initial region $X_I = [0.5, 1.5]^4$, and unsafe region $X_O = [-2.4, -1.6]^4$ for safety problems.

ct-LS: 6D Academic Example. We consider the following 6-dimensional benchmark adapted from [50]

$$\Sigma^c : \begin{cases} \dot{x}_1(t) = x_2(t), \\ \dot{x}_2(t) = x_3(t), \\ \dot{x}_3(t) = x_4(t), \\ \dot{x}_4(t) = x_5(t), \\ \dot{x}_5(t) = x_6(t), \\ \dot{x}_6(t) = -800x_6(t) - 2273x_5(t) - 3980x_4(t) - 4180x_3(t) \\ \quad - 2400x_2(t) - 576x_1(t) + u(t), \end{cases}$$

which is of the form (9) with

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -576 & -2400 & -4180 & -3980 & -2273 & -800 \end{bmatrix}, \\ B = [0 \ 0 \ 0 \ 0 \ 0 \ 1]^\top,$$

with the state space $X = [-2, 2]^6$, initial region $X_I = [0.5, 1.5]^6$, and unsafe region $X_O = [-2, -1.6]^6$ for safety problems.

ct-LS: 8D Academic Example. We consider the following 8-dimensional benchmark adapted from [50]

$$\Sigma^c : \begin{cases} \dot{x}_1(t) = x_2(t), \\ \dot{x}_2(t) = x_3(t), \\ \dot{x}_3(t) = x_4(t), \\ \dot{x}_4(t) = x_5(t), \\ \dot{x}_5(t) = x_6(t), \\ \dot{x}_6(t) = x_7(t), \\ \dot{x}_7(t) = x_8(t), \\ \dot{x}_8(t) = -20x_8(t) - 170x_7(t) - 800x_6(t) - 2273x_5(t) \\ \quad - 3980x_4(t) - 4180x_3(t) - 2400x_2(t) - 576x_1(t) + u(t), \end{cases}$$

which is of the form (9) with

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -576 & -2400 & -4180 & -3980 & -2273 & -800 & -170 & -20 \end{bmatrix}, \\ B = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1]^\top,$$

with the state space $X = [-2.2, 2.2]^8$, initial region $X_I = [0.9, 1.1]^8$, and unsafe region $X_O = [-2.2, -1.8]^8$ for safety problems.

dt-NPS: Lotka-Volterra Predator-Prey Model. We consider the Lotka-Volterra Predator-Prey Model [47], with dynamics

$$\Sigma^d : \begin{cases} \dot{x}_1^+ = x_1 + \tau(\alpha x_1 - \beta x_1 x_2 - u_1), \\ \dot{x}_2^+ = x_2 + \tau(-\eta x_2 + \delta x_1 x_2 + u_2), \end{cases}$$

which is of the form (14) with

$$A = \begin{bmatrix} 1 + \tau\alpha & 0 & -\tau\beta \\ 0 & 1 - \tau\eta & \tau\delta \end{bmatrix}, \quad B = \begin{bmatrix} -\tau \\ \tau \end{bmatrix}, \quad \mathcal{M}(x) = \begin{bmatrix} x_1 \\ x_2 \\ x_1 x_2 \end{bmatrix},$$

where $\tau = 0.01, \alpha = 0.6, \beta = 1.0, \eta = 0.6, \delta = 1.0$. For safety problems, we consider $X = [-2, 2] \times [-1, 1]$, $X_I = [0.5, 1] \times [0.2, 0.4]$, and $X_O = [-2, -1.5] \times [0.8, 1] \cup [-2, -1.5] \times [-1, -0.8] \cup [1.6, 2] \times [0.85, 1] \cup [1.6, 2] \times [-1, -0.8]$.

dt-NPS: Academic System. We focus on the following nonlinear polynomial system borrowed from [41]:

$$\Sigma^d : \begin{cases} \dot{x}_1^+ = x_1 + \tau x_2, \\ \dot{x}_2^+ = x_2 + \tau(x_1^2 + u), \end{cases}$$

which is of the form of (14), with

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathcal{M}(x) = \begin{bmatrix} x_2 \\ x_1^2 \end{bmatrix}.$$

with $\tau = 0.1$. Note that this case study serves solely for stability; therefore, no regions of interest for safety are provided.

dt-NPS: Lorenz Attractor. We consider the Lorenz attractor, a well-studied dynamical system with chaotic behavior, with dynamics

$$\Sigma^d : \begin{cases} x_1^+ = x_1 + \tau(\sigma x_1 + \sigma x_2 + u_1), \\ x_2^+ = x_2 + \tau(\rho x_1 - x_2 - x_1 x_3 + u_2), \\ x_3^+ = x_3 + \tau(x_1 x_2 - \beta x_3 + u_3), \end{cases}$$

which is of the form (14) with

$$A = \begin{bmatrix} 1 + \tau\sigma & \tau\sigma & 0 & 0 & 0 & 0 \\ \tau\rho & 1 - \tau & 0 & -\tau & 0 & 0 \\ 0 & 0 & 1 - \tau\beta & \tau & 0 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} \tau \\ \tau \\ \tau \end{bmatrix}, \quad \mathcal{M}(x) = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_1 x_2 \\ x_2 x_3 \\ x_1 x_3 \end{bmatrix},$$

where $\rho = 28, \sigma = 10, \beta = \frac{8}{3}, \tau = 10^{-3}$. For safety problems, the regions of interest are given as: state space $X = [-5, 5]^3$, initial set $X_I = [-1, 1]^3$ and two unsafe sets $X_O = [-5, -3.5]^3 \cup [3.5, 5]^3$.

dt-LS: DC Motor. We consider a DC Motor, based on [49], with the dynamics

$$\Sigma^d : \begin{cases} x_1^+ = x_1 - \tau\left(\frac{R}{L}x_1 + \frac{k_{dc}}{L}x_2 + u_1\right), \\ x_2^+ = x_2 + \tau\left(\frac{k_{dc}}{J}x_1 - \frac{b}{J}x_2 + u_2\right), \end{cases}$$

which is of the form (20) with

$$A = \begin{bmatrix} 1 - \frac{\tau R}{L} & -\frac{\tau k_{dc}}{L} \\ \frac{\tau k_{dc}}{J} & 1 - \frac{\tau b}{J} \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

where $\tau = 0.01$ is the sampling time, and other parameters are as in (24). For safety problems, the state space is given as $X = [-1, 1]^2$, with initial set $X_I = [0.1, 0.4] \times [0.1, 0.55]$ and two unsafe sets $X_O = [0.45, 1] \times [0.6, 1] \cup [-1, -0.6] \times [0.6, 1]$.

dt-LS: Room Temperature System 1. We consider the two-room system [50], with dynamics

$$\Sigma^d : \begin{cases} x_1^+ = (1 - \tau(\alpha + \alpha_{e1}))x_1 + \tau\alpha x_2 + \tau\alpha_{e1}u, \\ x_2^+ = (1 - \tau(\alpha + \alpha_{e2}))x_2 + \tau\alpha x_1 + \tau\alpha_{e2}u, \end{cases}$$

which is of the form (20) with

$$A = \begin{bmatrix} 1 - \tau(\alpha + \alpha_{e1}) & \tau\alpha \\ \tau\alpha & 1 - \tau(\alpha + \alpha_{e2}) \end{bmatrix}, \quad B = \begin{bmatrix} \tau\alpha_{e1} \\ \tau\alpha_{e2} \end{bmatrix},$$

where $\tau = 5$, and other parameters are as in (25). For safety problems, the regions of interest as given as $X = [-2, 3]^2$, $X_I = [-0.5, 0.5]^2$, and $X_O = [2, 3]^2 \cup [-2, -0.5] \times [1.5, 3] \cup [1.5, 3] \times [-2, -0.5]$.

dt-LS: Room Temperature System 2. We consider the three-room system [50], with dynamics

$$\Sigma^d : \begin{cases} x_1^+ = (1 - \tau(\alpha - \alpha_{e1}))x_1 + \tau\alpha x_3 + \tau\alpha_{e1}u, \\ x_2^+ = (1 - \tau(\alpha - \alpha_{e2}))x_2 + \tau\alpha(x_1 + x_3) + \tau\alpha_{e2}u, \\ x_3^+ = (1 - \tau(\alpha - \alpha_{e3}))x_3 + \tau\alpha x_1 + \tau\alpha_{e3}u, \end{cases}$$

which is of the form (20) with

$$A = \begin{bmatrix} 1 - \tau(\alpha - \alpha_{e1}) & 0 & \tau\alpha \\ \tau\alpha & 1 - \tau(2\alpha - \alpha_{e2}) & \tau\alpha \\ \tau\alpha & 0 & 1 - \tau(\alpha - \alpha_{e3}) \end{bmatrix}, \quad B = \begin{bmatrix} \tau\alpha_{e1} \\ \tau\alpha_{e2} \\ \tau\alpha_{e3} \end{bmatrix},$$

where $\tau = 5$, and other parameters are as in (25). For safety problems, the regions of interest are given as: state space $X = [-2, 3]^3$, initial set $X_I = [-0.5, 0.5]^3$ and two unsafe sets $X_O = [2, 3] \times [-2, -3] \times [2, 3] \cup [-2, -3] \times [2, 3] \times [-2, -3]^3$.

dt-LS: Two Tank System. Consider a two-tank system [51], characterized by difference equations

$$\Sigma^d : \begin{cases} x_1^+ = (1 - \tau\frac{\alpha_1}{a_1})x_1 + \tau\frac{u_1}{a_1}, \\ x_2^+ = \tau\frac{\alpha_1}{a_2}x_1 + (1 - \tau\frac{\alpha_2}{a_2})x_2 + \tau\frac{u_2}{a_2}, \end{cases}$$

which is of the form (20) with

$$A = \begin{bmatrix} 1 - \tau\frac{\alpha_1}{a_1} & 0 \\ \tau\frac{\alpha_1}{a_2} & 1 - \tau\frac{\alpha_2}{a_2} \end{bmatrix}, \quad B = \begin{bmatrix} \tau\frac{1}{a_1} & 0 \\ 0 & \tau\frac{1}{a_2} \end{bmatrix},$$

where $\tau = 0.1$, and other parameters are as in (26). For safety problems, the regions of interest are $X = [-2, 2]^2$, $X_I = [-0.5, 0.5]^2$, $X_O = [1.5, 2]^2 \cup [-2, -1.5] \times [1, 2] \cup [-1.5, -1] \times [1.5, 2] \cup [1.5, 2] \times [-2, -1]$.

dt-LS: 4D Academic Example. We consider the following 4-dimensional benchmark adapted from [50]

$$\Sigma^d : \begin{cases} x_1^+ = x_1 + \tau(x_2 + u_1), \\ x_2^+ = x_2 + \tau(x_3 + u_2), \\ x_3^+ = x_3 + \tau(x_4 + u_3), \\ x_4^+ = x_4 + \tau(-3980x_4 - 4180x_3 - 2400x_2 - 576x_1 + u_4), \end{cases}$$

which is of the form (20) with

$$A = \begin{bmatrix} 1 & \tau & 0 & 0 \\ 0 & 1 & \tau & 0 \\ 0 & 0 & 1 & \tau \\ -576\tau & -2400\tau & -4180\tau & 1 - 3980\tau \end{bmatrix}, \quad B = \tau\mathbb{I}_4,$$

where $\tau = 0.001$, with the state space $X = [-2, 2]^4$, initial region $X_I = [0.5, 1.5]^4$, and unsafe region $X_O = [-2.4, -1.6]^4$ for safety problems.

dt-LS: 6D Academic Example. We consider the following 6-dimensional benchmark adapted from [50]

$$\Sigma^d : \begin{cases} x_1^+ = x_1 + \tau(x_2 + u_1), \\ x_2^+ = x_2 + \tau(x_3 + u_2), \\ x_3^+ = x_3 + \tau(x_4 + u_3), \\ x_4^+ = x_4 + \tau(x_5 + u_4), \\ x_5^+ = x_5 + \tau(x_6 + u_5), \\ x_6^+ = x_6 + \tau(-800x_6 - 2273x_5 - 3980x_4 - 4180x_3 - 2400x_2 - 576x_1 + u_6), \end{cases}$$

which is of the form (20) with

$$A = \begin{bmatrix} 1 & \tau & 0 & 0 & 0 & 0 \\ 0 & 1 & \tau & 0 & 0 & 0 \\ 0 & 0 & 1 & \tau & 0 & 0 \\ 0 & 0 & 0 & 1 & \tau & 0 \\ 0 & 0 & 0 & 0 & 1 & \tau \\ -576\tau & -2400\tau & -4180\tau & -3980\tau & -2273\tau & 1 - 800\tau \end{bmatrix},$$

$$B = \tau \mathbb{I}_6,$$

where $\tau = 0.1$, with the state space $X = [-2, 2]^6$, initial region $X_I = [0.5, 1.5]^6$, and unsafe region $X_O = [-2, -1.6]^6$ for safety problems.

dt-LS: 8D Academic Example. We consider the following 8-dimensional benchmark adapted from [50]

$$\Sigma^d : \begin{cases} x_1^+ = x_1 + \tau(x_2 + u_1), \\ x_2^+ = x_2 + \tau(x_3 + u_2), \\ x_3^+ = x_3 + \tau(x_4 + u_3), \\ x_4^+ = x_4 + \tau(x_5 + u_4), \\ x_5^+ = x_5 + \tau(x_6 + u_5), \\ x_6^+ = x_6 + \tau(x_7 + u_6), \\ x_7^+ = x_7 + \tau(x_8 + u_7), \\ x_8^+ = x_8 + \tau(-20x_8 - 170x_7 - 800x_6 - 2273x_5 - 3980x_4 - 4180x_3 \\ \quad - 2400x_2 - 576x_1 + u_8), \end{cases}$$

which is of the form (20) with

$$A = \begin{bmatrix} 1 & \tau & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & \tau & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \tau & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \tau & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \tau & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & \tau & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \tau \\ -576\tau & -2400\tau & -4180\tau & -3980\tau & -2273\tau & -800\tau & -170\tau & 1 - 20\tau \end{bmatrix},$$

$$B = \tau \mathbb{I}_8,$$

where $\tau = 0.1$, with the state space $X = [-2.2, 2.2]^8$, initial region $X_I = [0.9, 1.1]^8$, and unsafe region $X_O = [-2.2, -1.8]^8$ for safety problems.

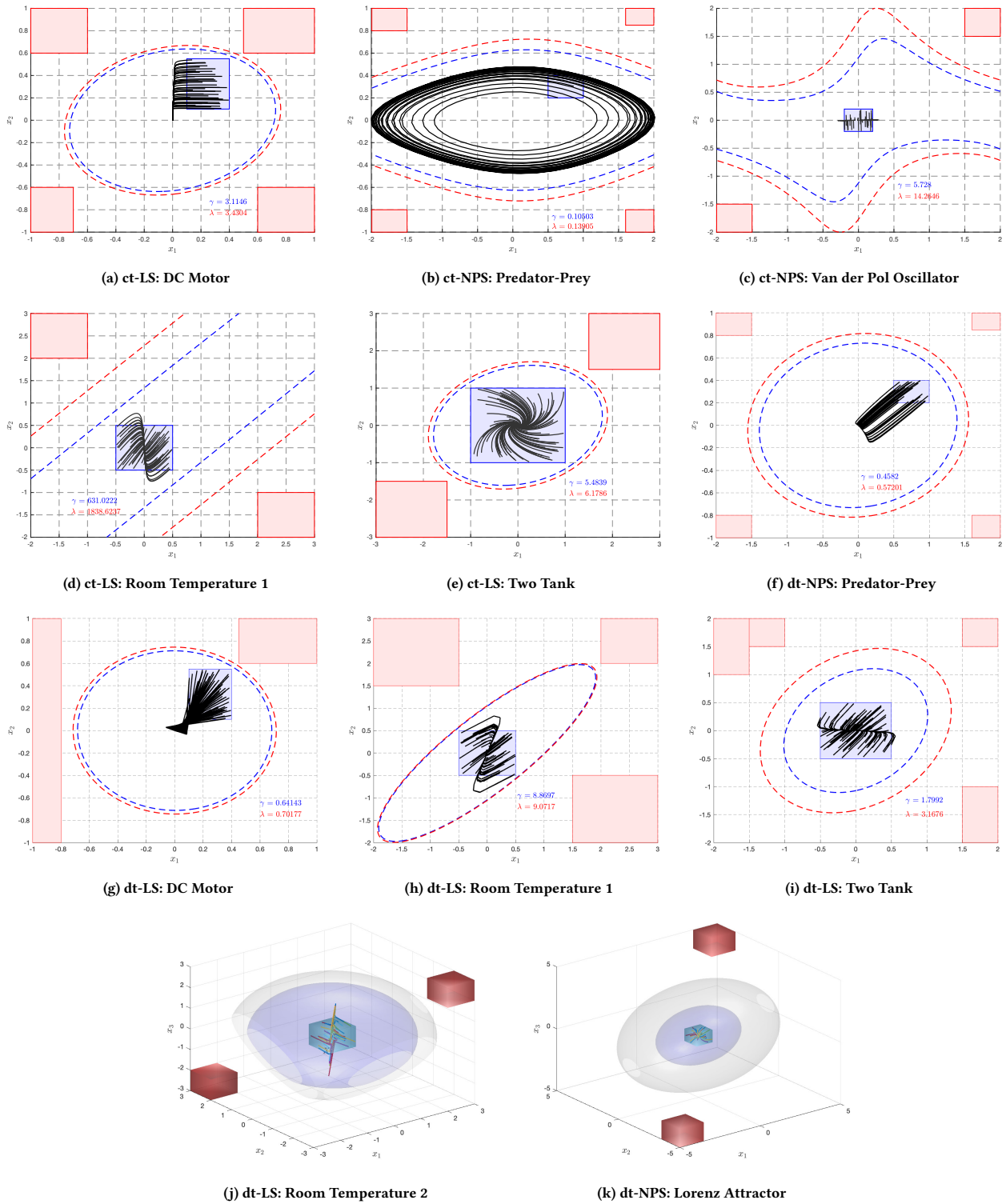
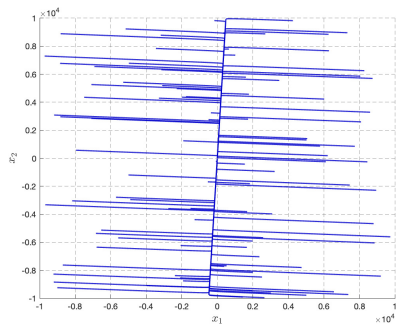
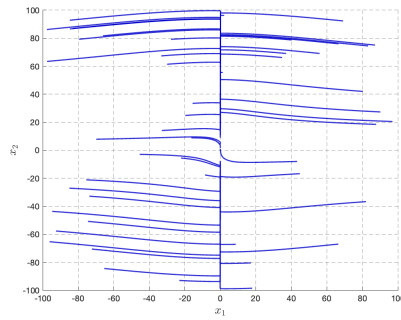


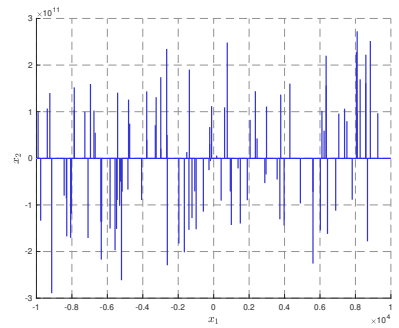
Figure 2: Simulation results for designing CBCs. The purple box represents the initial region, while the red boxes indicate multiple unsafe regions. For the 2D figures, the blue and red dashed lines show the initial and unsafe level sets, respectively. For the 3D figures, the purple bubble indicates the initial level set, while the gray bubble represents the unsafe level set.



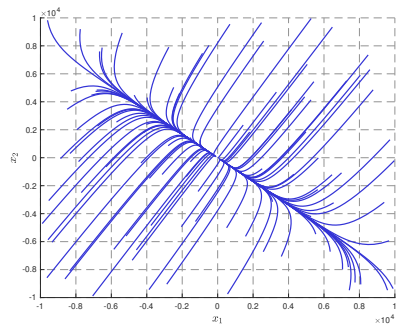
(a) ct-LS: DC Motor



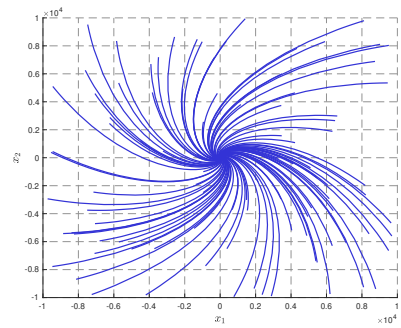
(b) ct-NPS: Predator Prey



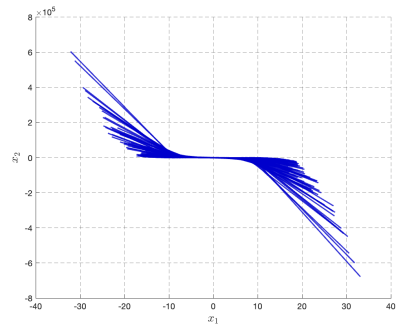
(c) ct-NPS: Van der Pol Oscillator



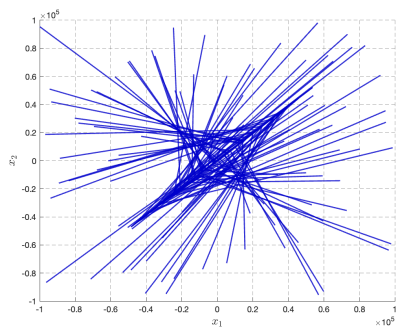
(d) ct-LS: Room Temperature 1



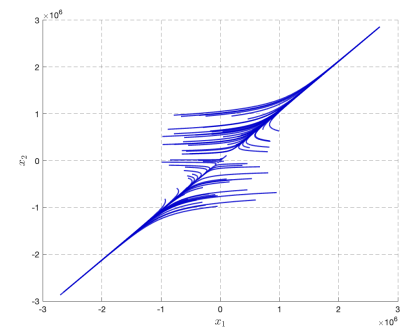
(e) ct-LS: Two Tank



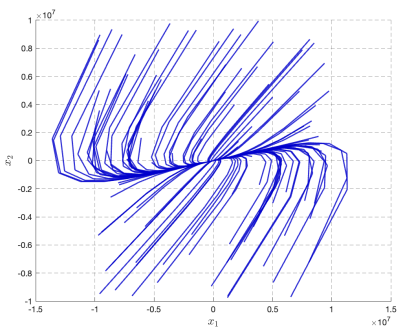
(f) dt-NPS: Academic System



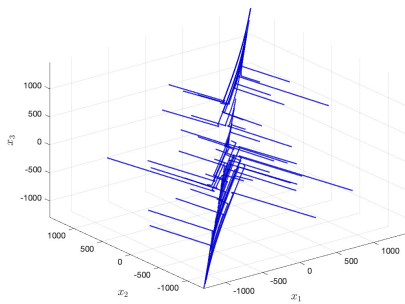
(g) dt-LS: DC Motor



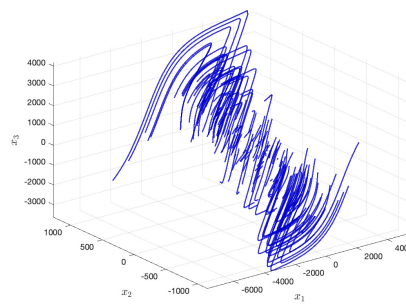
(h) dt-LS: Room Temperature 1



(i) dt-LS: Two Tank



(j) dt-LS: Room Temperature 2



(k) dt-NPS: Lorenz Attractor

Figure 3: Simulation results for designing CLFs.