NeuralExplorer: State Space Exploration of Closed-loop Control Systems using NN

Manish Goyal, Parasara **Sridhar** Duggirala

Department of Computer Science, University of North Carolina at Chapel Hill

Automated Technology for Verification and Analysis (ATVA)

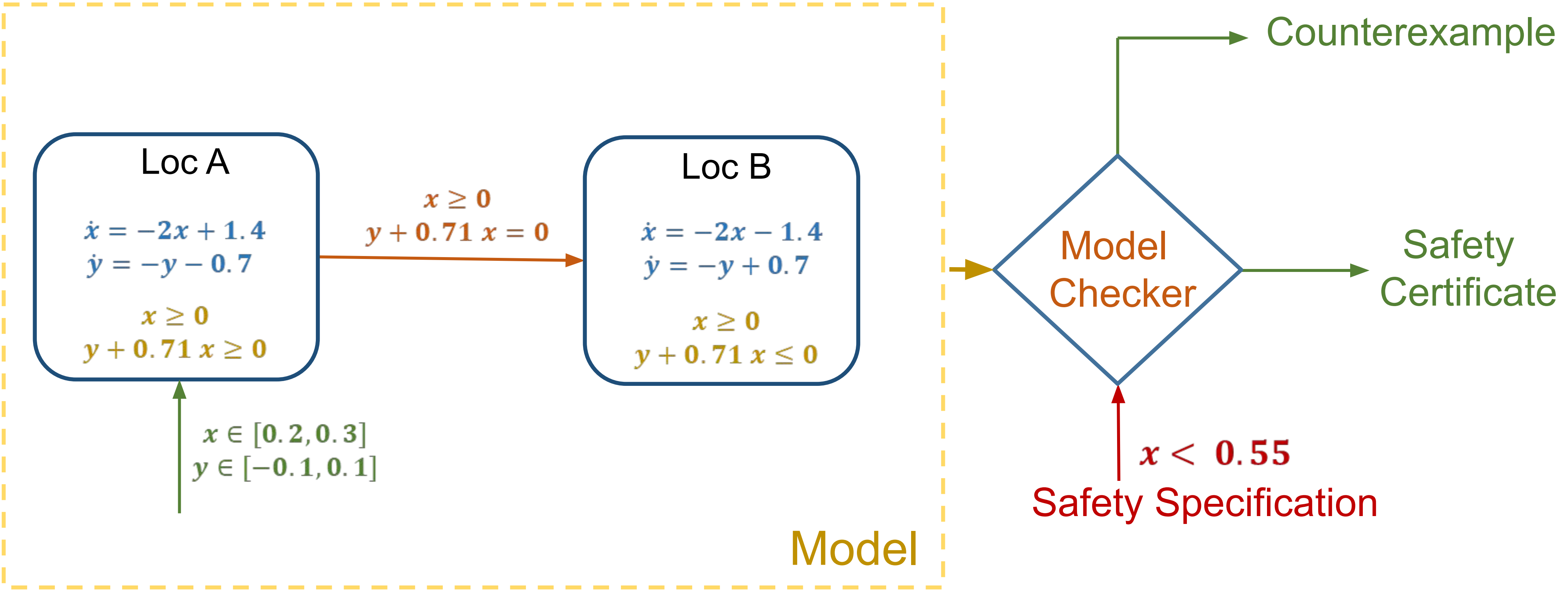
Oct 22, 2020

Outline

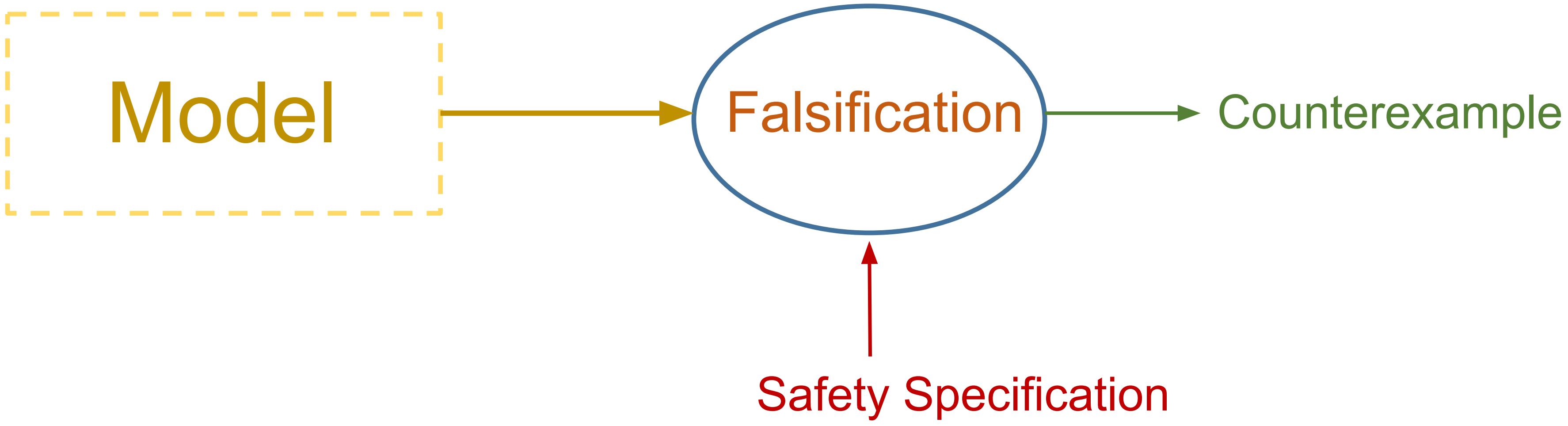
- Introduction
- Preliminaries
- Methodology
- Evaluation
- Applications
- Discussion

Verification

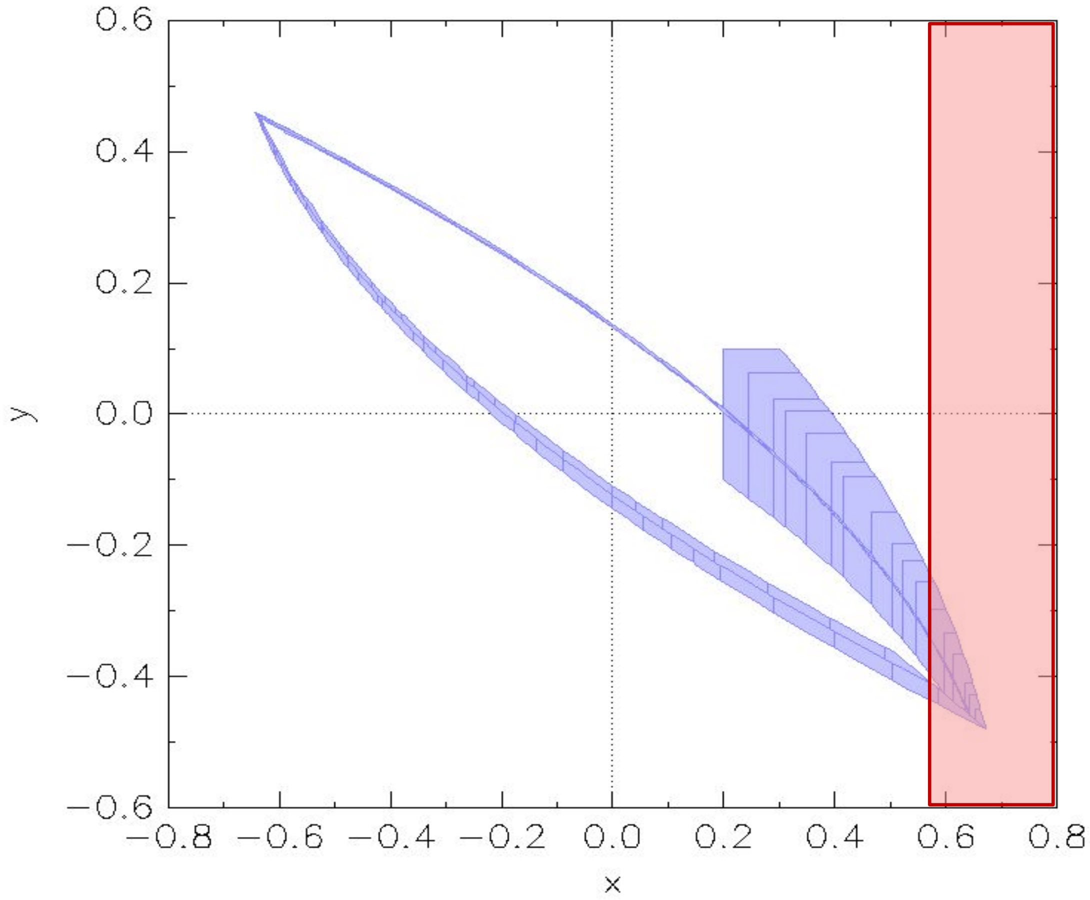
Analogous to Reachability



Falsification

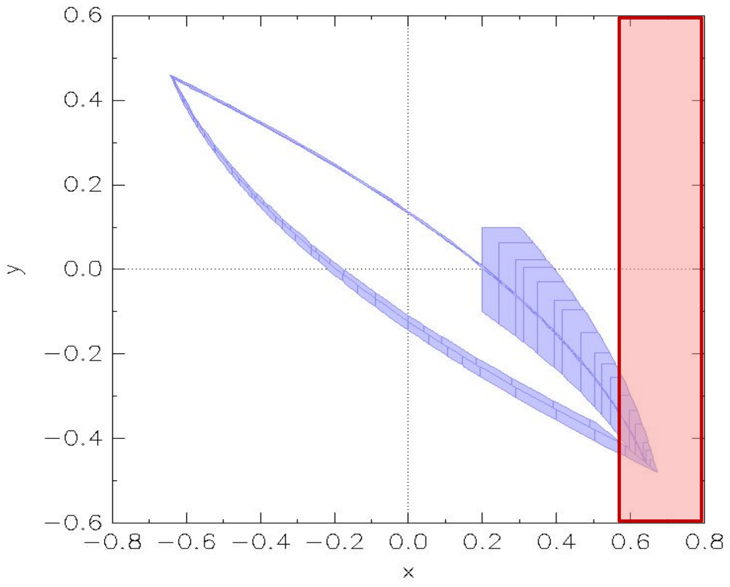


Verification

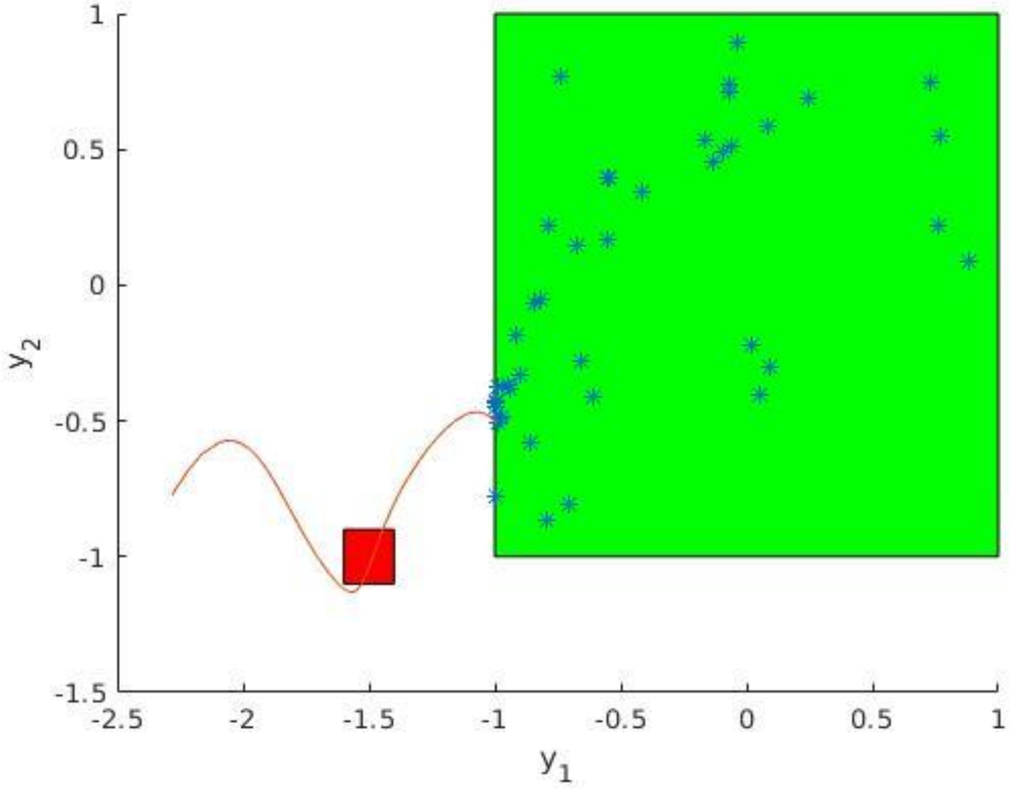


Verification

- SpaceEx
- HyLAA
- Flow*
- CORA

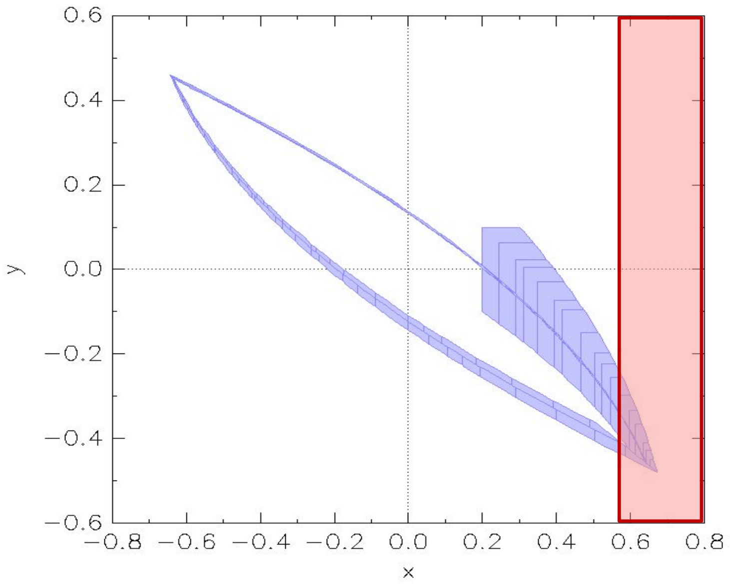


Falsification



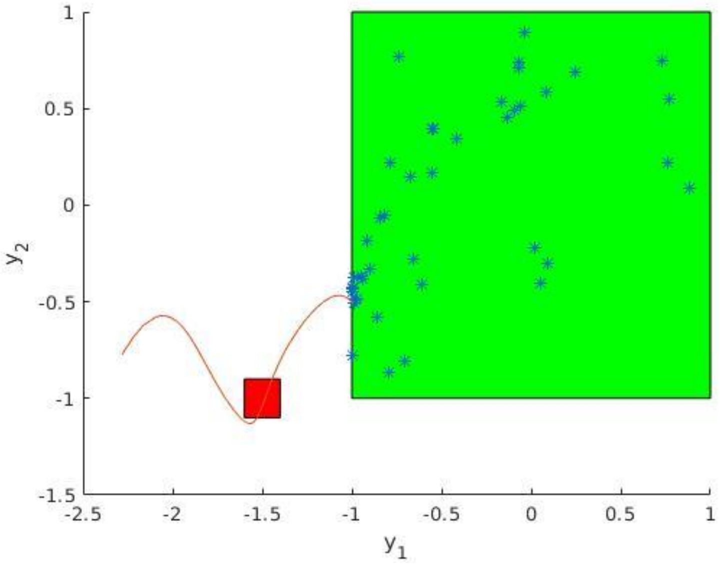
Verification

- SpaceEx
- HyLAA
- Flow*
- CORA

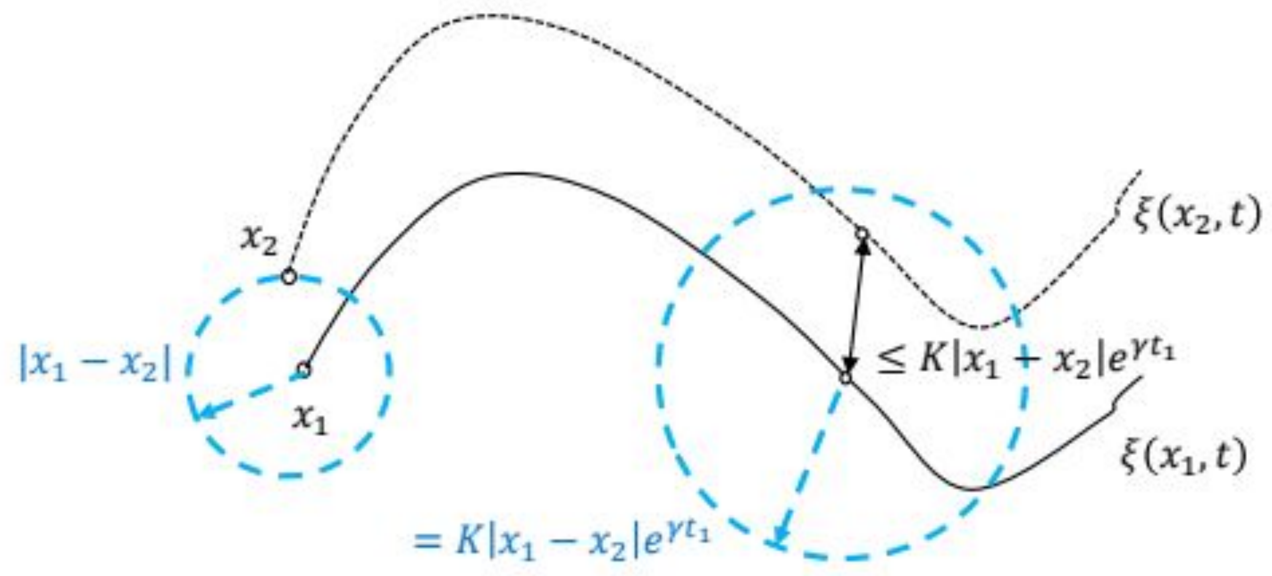


Falsification

- S-Taliro
- Breach

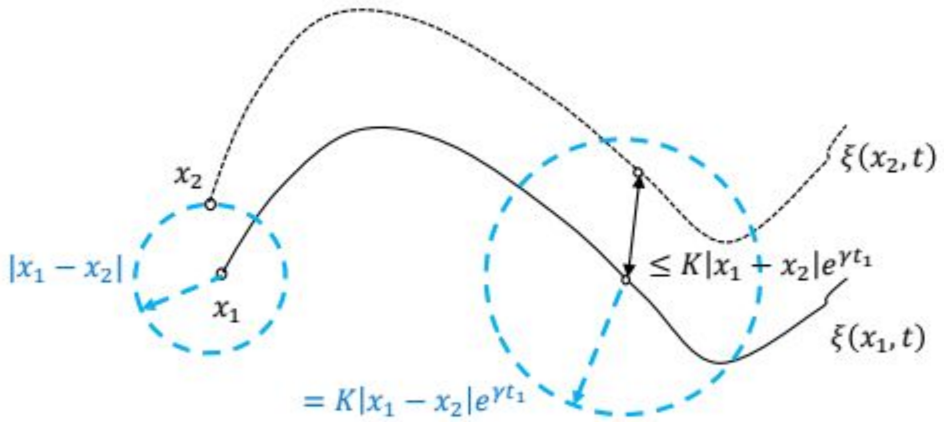


Simulation driven verification



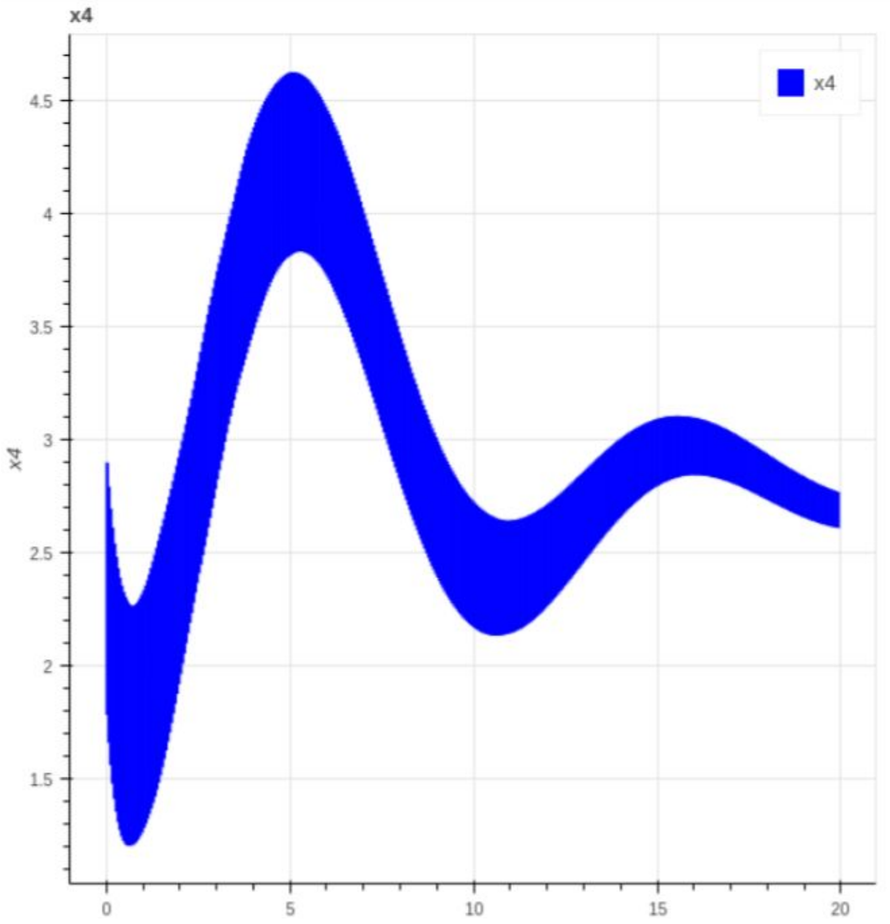
Discrepancy function

Simulation driven verification

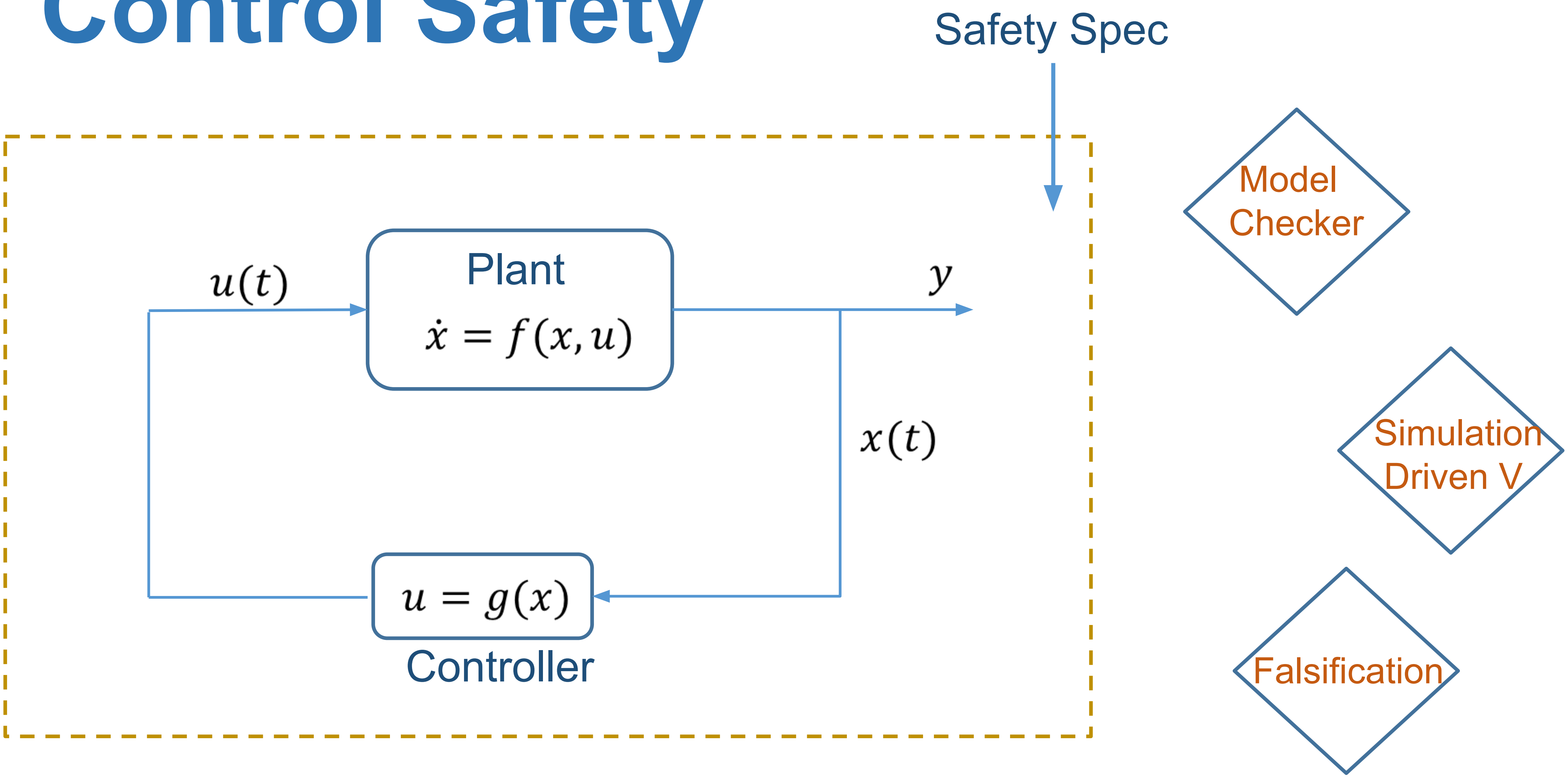


Discrepancy function

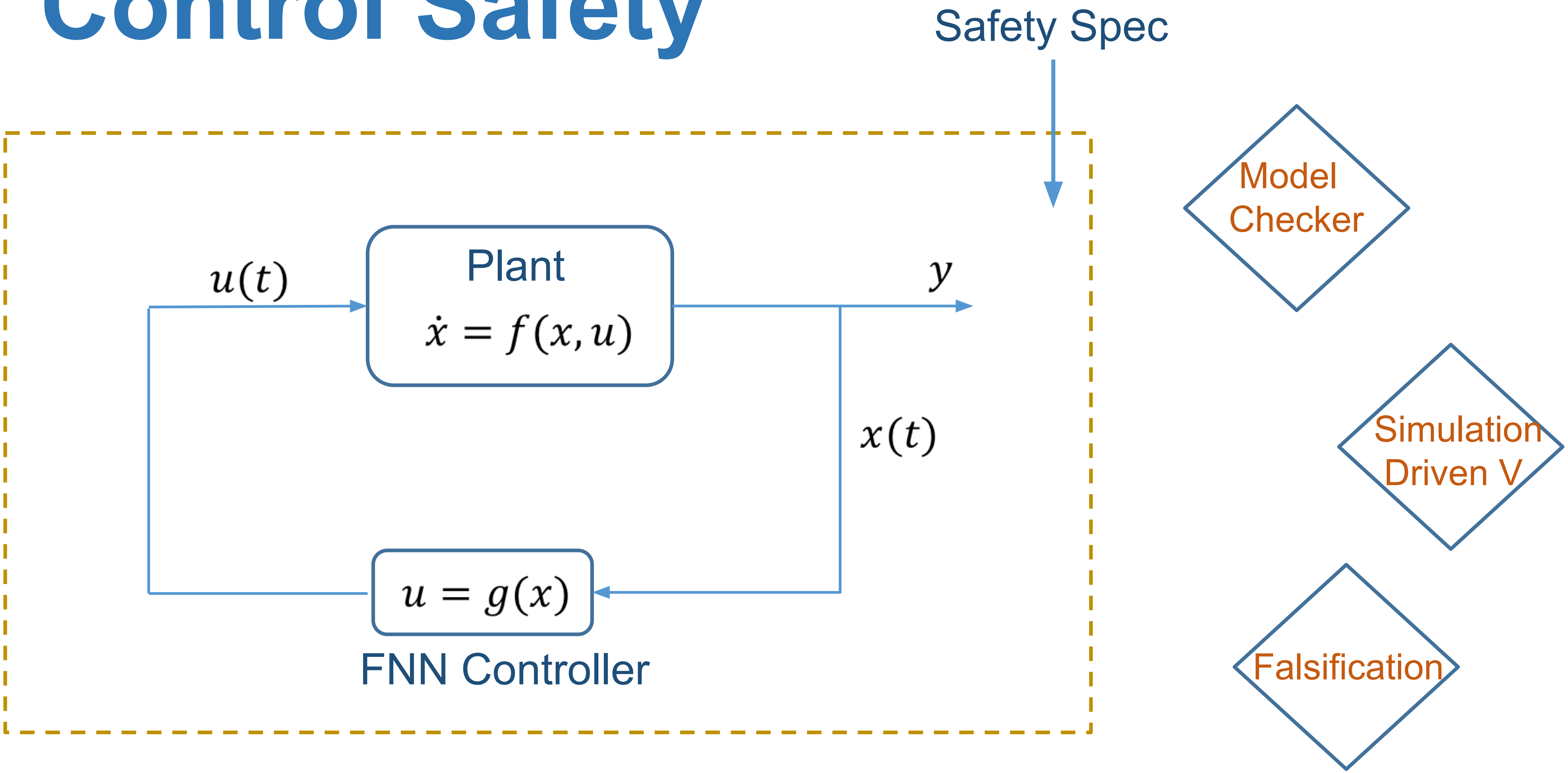
- C2E2
- DryVR



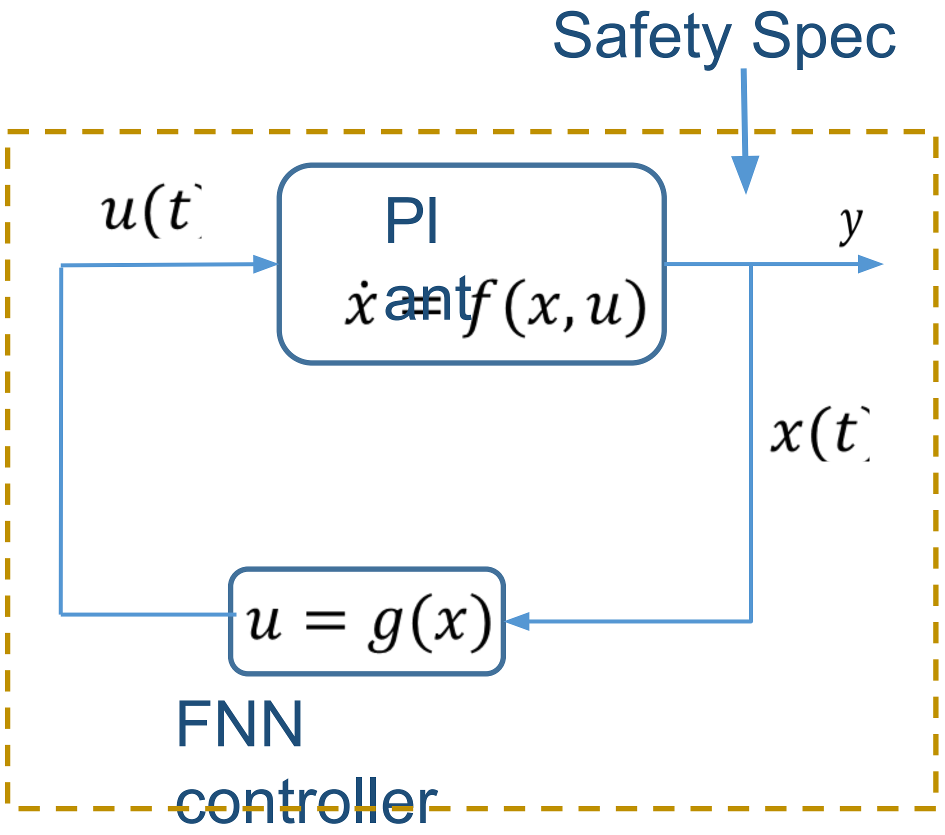
Control Safety



Control Safety



NN Control Safety



- Sherlock
- Verisig
- NNV
- S-Taliro

Motivation

- Complexity of systems
- Test case based exploration
- Abundance of data
- Application of neural networks

Motivation

- Complexity of systems
- Test case based exploration
- Abundance of data
- Application of neural networks → Learning system dynamics; Learning Barrier function; State classification, etc.

Motivation

- Complexity of systems
- Test case based exploration
- Abundance of data
- Application of neural networks

Our Approach

Sensitivity function

Learning system dynamics

Learning Barrier function

State classification

Outline

- ✔ Introduction
- Preliminaries
- Methodology
- Evaluation
- Applications
- Discussion

System Trajectory

Definition 1 (Unique Trajectory Feedback Functions). *A feedback function* $u = g(x)$ *is said to be unique trajectory feedback function if the closed loop system* $\dot{x} = f(x, g(x))$ *is guaranteed existence and uniqueness of the solution for the initial value problem for all initial points* $x_0 \in \mathbb{R}^n$.

System Trajectory

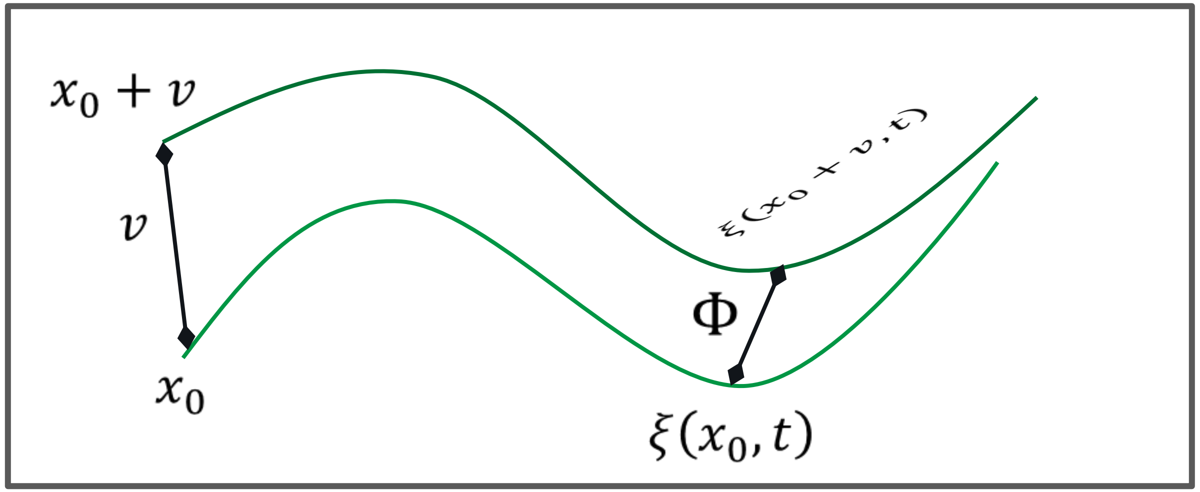
Definition 1 (Unique Trajectory Feedback Functions). *A feedback function* $u = g(x)$ *is said to be unique trajectory feedback function if the closed loop system* $\dot{x} = f(x, g(x))$ *is guaranteed existence and uniqueness of the solution for the initial value problem for all initial points* $x_0 \in \mathbb{R}^n$.

Definition 2 (Trajectories of Closed Loop System). *Given a unique trajectory feedback function* $u = g(x)$, *a trajectory of closed loop system* $\dot{x} = f(x, g(x))$, *denoted as* $\xi_g(x_0, t)$ $(t \geq 0)$, *is the solution of the initial value problem of the differential equation* $\dot{x} = f(x, g(x))$ *with initial condition* x_0. *We often drop the feedback function* g *when it is clear from the context.*

Sensitivity

Definition 3 (Sensitivity of Trajectories). *Given an initial state x_0, vector v, and time t, the sensitivity of the trajectories, denoted as $\Phi(x_0, v, t)$ is defined as.*

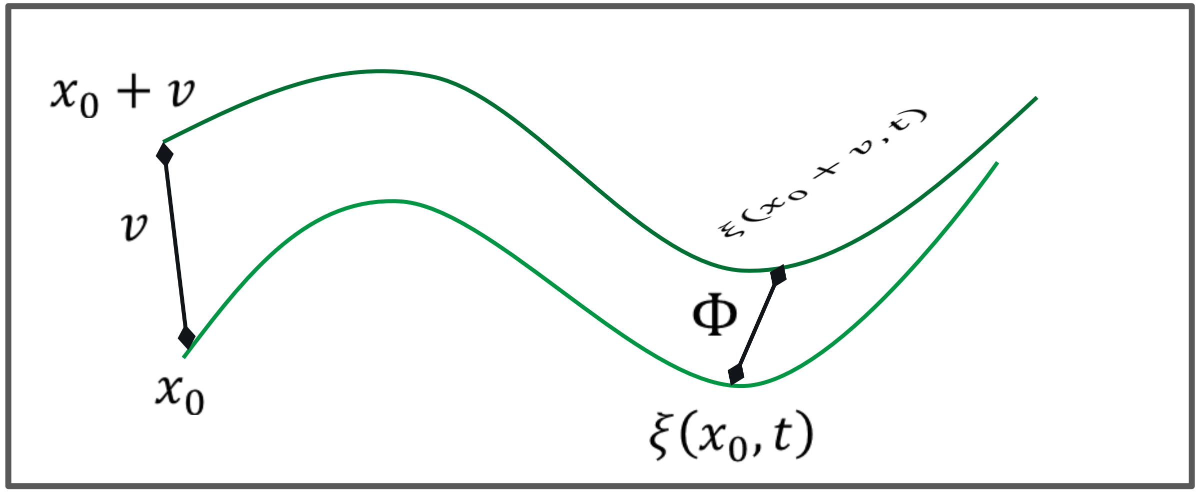
$$\Phi(x_0, v, t) = \xi(x_0 + v, t) - \xi(x_0, t).$$



Sensitivity

Definition 3 (Sensitivity of Trajectories). *Given an initial state x_0, vector v, and time t, the sensitivity of the trajectories, denoted as $\Phi(x_0, v, t)$ is defined as.*

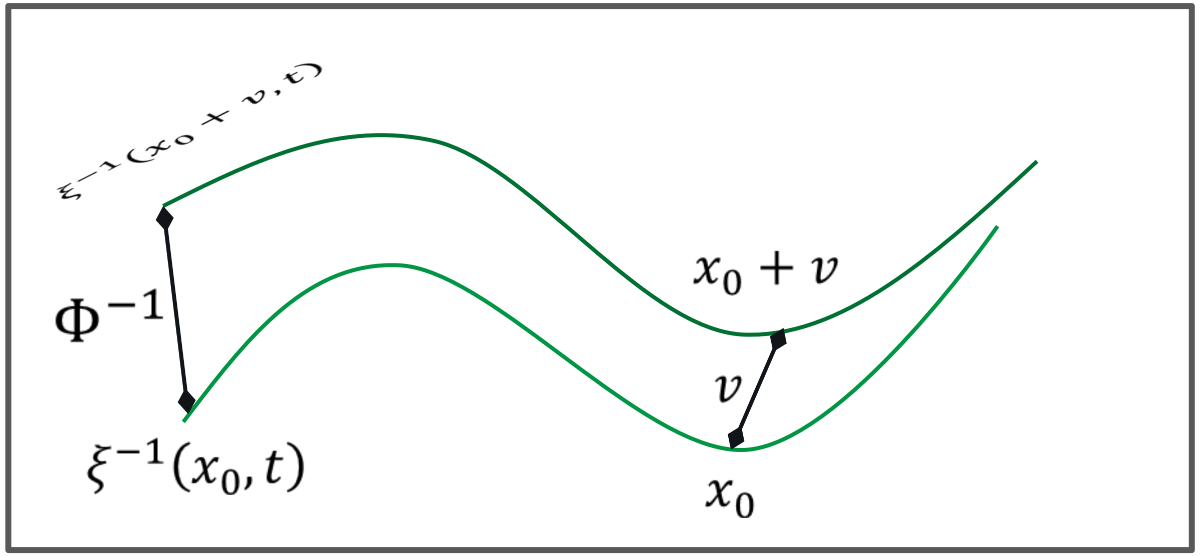
$$\Phi(x_0, v, t) = \xi(x_0 + v, t) - \xi(x_0, t).$$



Vector difference between trajectories at time t

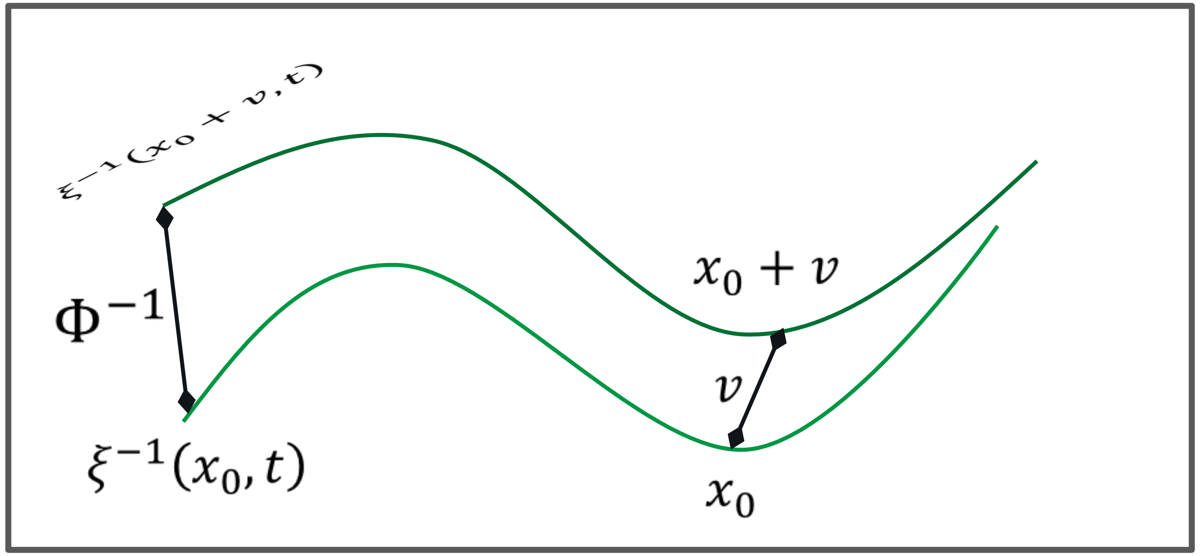
Inverse Sensitivity

$$\Phi^{-1}(x_0, v, t) = \xi^{-1}(x_0 + v, t) - \xi^{-1}(x_0, t).$$



Inverse Sensitivity

$$\Phi^{-1}(x_0, v, t) = \xi^{-1}(x_0 + v, t) - \xi^{-1}(x_0, t).$$



Initial perturbation required to displace the trajectory by v

Outline

- ✔ Introduction
- ✔ Preliminaries
- Methodology
- Evaluation
- Applications
- Discussion

Learning Φ and Φ^{-1}

- Generate finite set of time bounded trajectories
- Recall $v_{sen} = \Phi(x_0, v, t)$ and $v_{isen} = \Phi^{-1}(x_0, v, t)$
- For each pair of (real or virtual) trajectories
- Generate tuples $< x_0, v, t, v_{sen} >$ and $< x_0, v, t, v_{isen} >$
- Use tuples for training to approximate sensitivity and inverse sensitivity
- Denote these networks as NN_{sen} and NN_{isen}, resp.

Results: Learning Φ^{-1}

MSE: Mean Squared error
MRE: Mean Relative error

Benchmark		Dims	Step size (sec)	Time bound	Training Time (min)	MSE	MRE
Continuous Nonlinear Dynamics	Brussellator	2	0.01	500	67.0	1.01	0.29
	Buckling	2	0.01	500	42.0	0.59	0.17
	Lotka	2	0.01	500	40.0	0.50	0.13
	Jetengine	2	0.01	300	34.0	1.002	0.26
Hybrid/NN Systems	HybridOsc.	2	0.01	1000	77.0	0.31	0.077
	SmoothOsc.	2	0.01	1000	77.5	0.23	0.063
	Mountain Car	2	-	100	10.0	0.005	0.70
	Quadrotor	6	0.01	120	25.0	0.0011	0.16

Outline

- ✔ Introduction
- ✔ Preliminaries
- ✔ Methodology
- Evaluation
- Applications
- Discussion

Reaching a state z with Φ^{-1}

- If the function is learnt without error, we should be able to reach a destination state in one shot

- Since we use an approximation, we need to iterate for a couple of times to get to the destination

Reaching a state z with Φ^{-1}

- Generate a trajectory ξ from random state x
- For the given time t, compute the vector $v = z - \xi(t)$
- Repeat while $||v||_2 \geq \delta$ and $iter \leq max$

a. Estimate $v_{isen} = NN_{isen}(\xi(t), v, t)$
b. Generate a new trajectory ξ from $x = x + v_{isen}$
c. Compute $v = z - \xi(t)$

- Return $(x, ||v||_2)$

Reaching a state z with Φ^{-1}

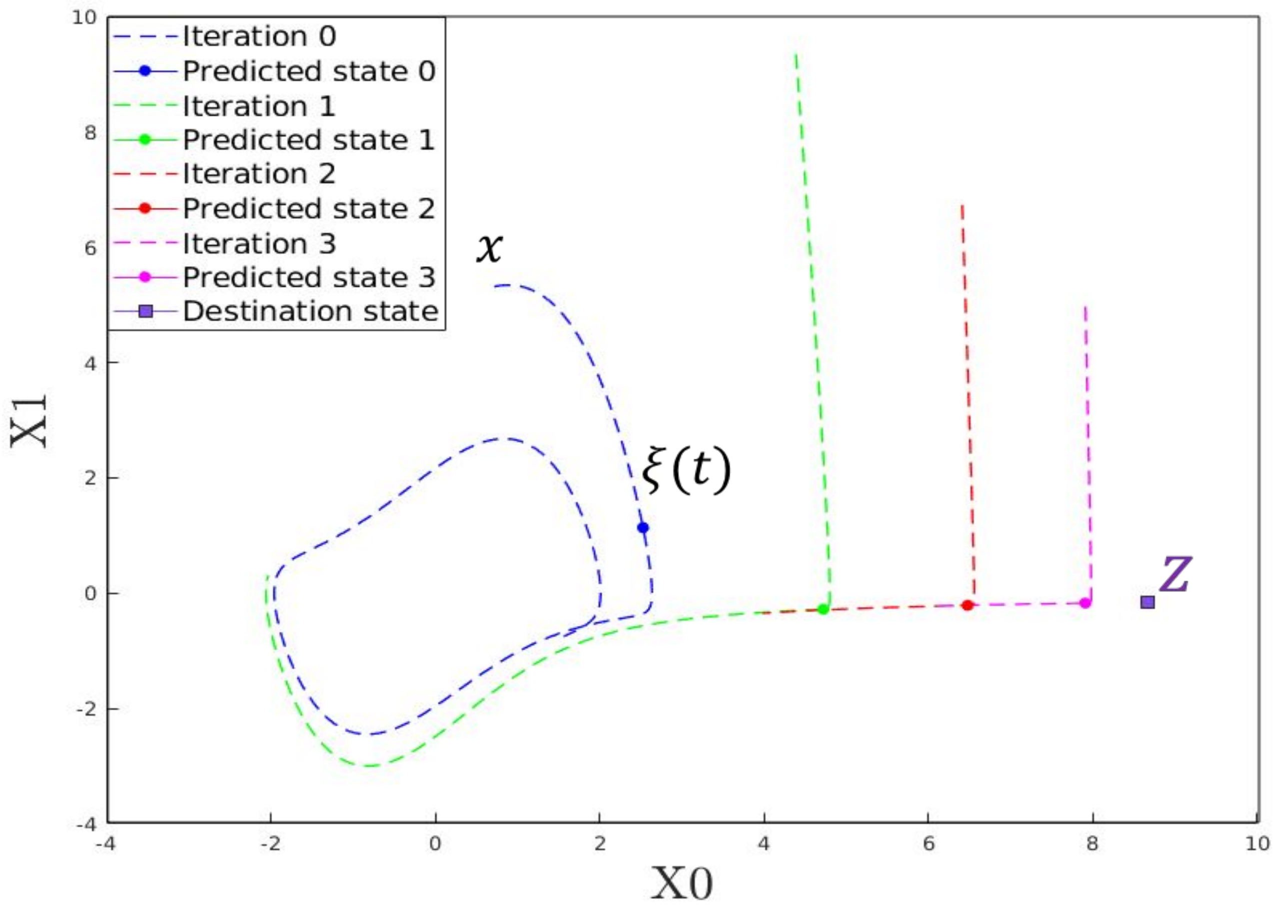
- Generate a trajectory ξ from random state x
- For the given time t, compute the vector $v = z - \xi(t)$
- Repeat while $||v||_2 \geq \delta$ and $iter \leq max$

ReachDestination

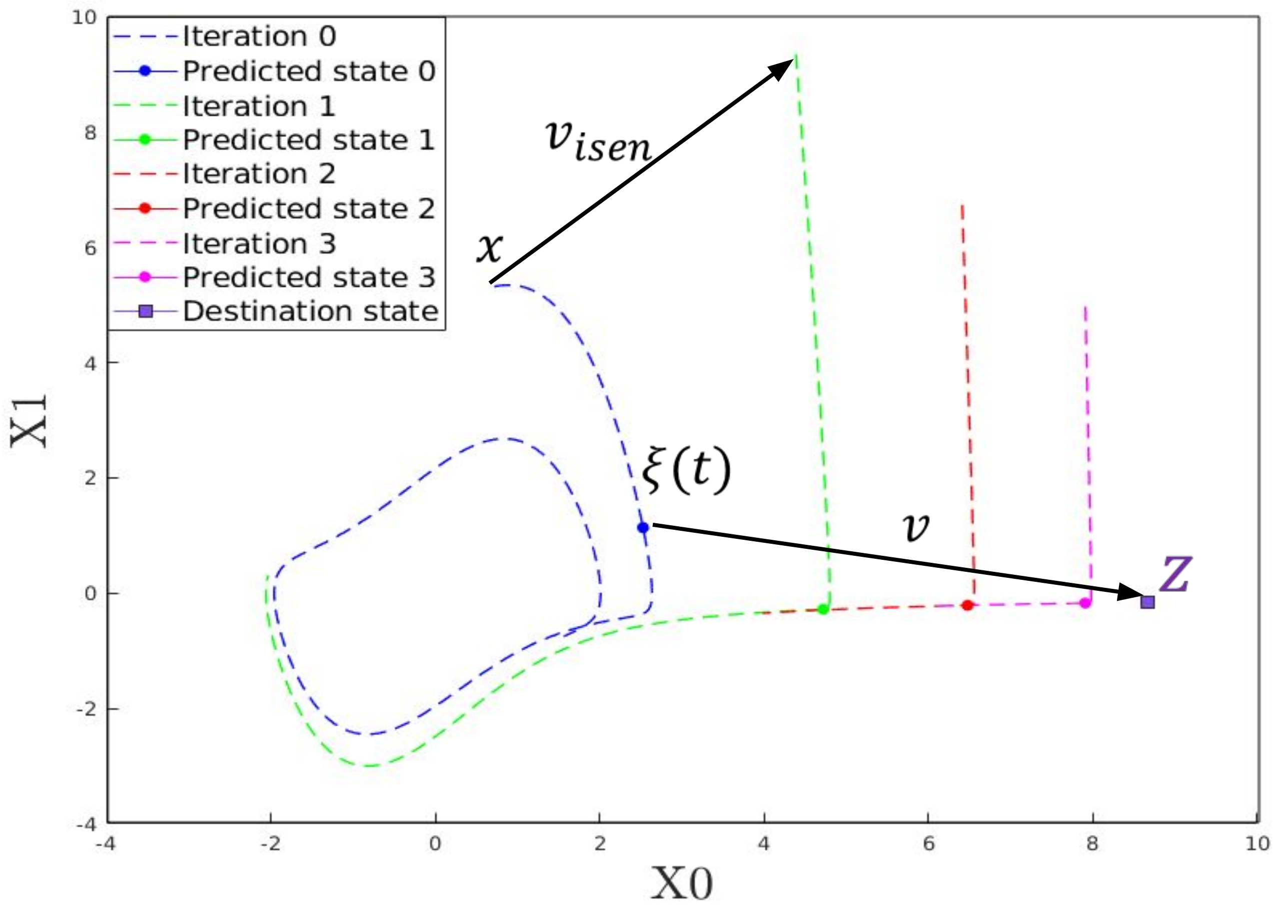
a. Estimate $v_{isen} = NN_{isen}(\xi(t), v, t)$
b. Generate a new trajectory ξ from $x = x + v_{isen}$
c. Compute $v = z - \xi(t)$

- Return $(x, ||v||_2)$

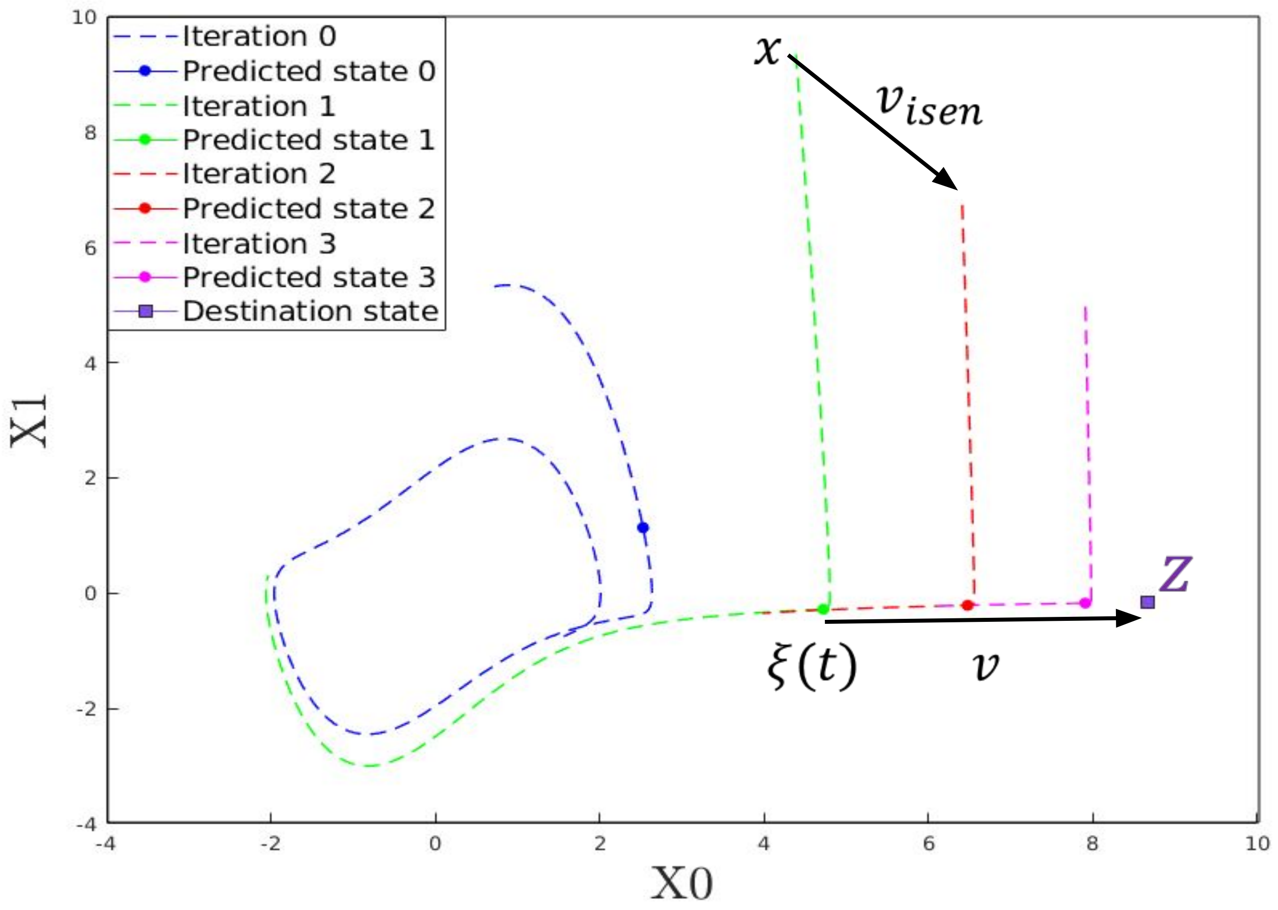
Evaluation: ReachDestination



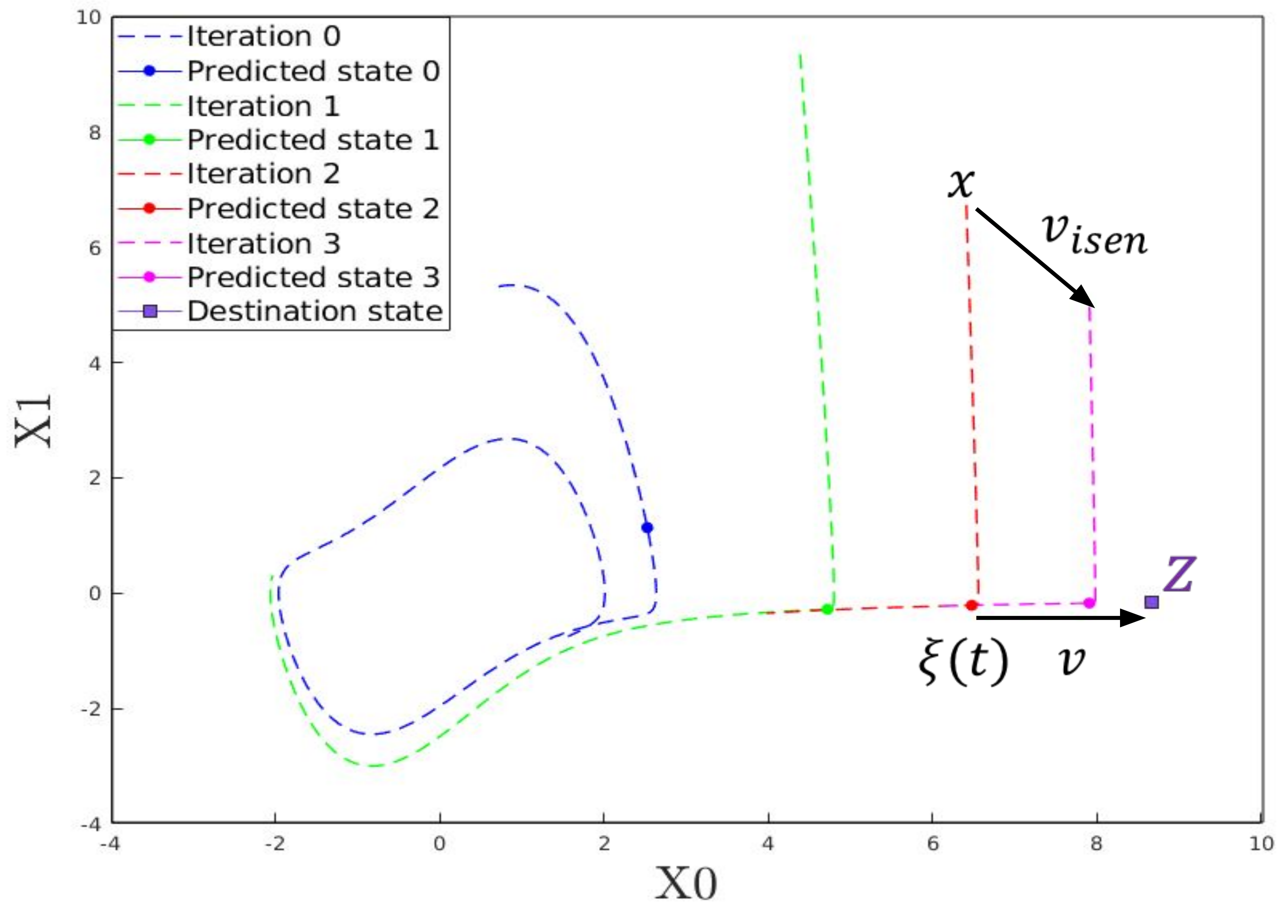
Evaluation: ReachDestination



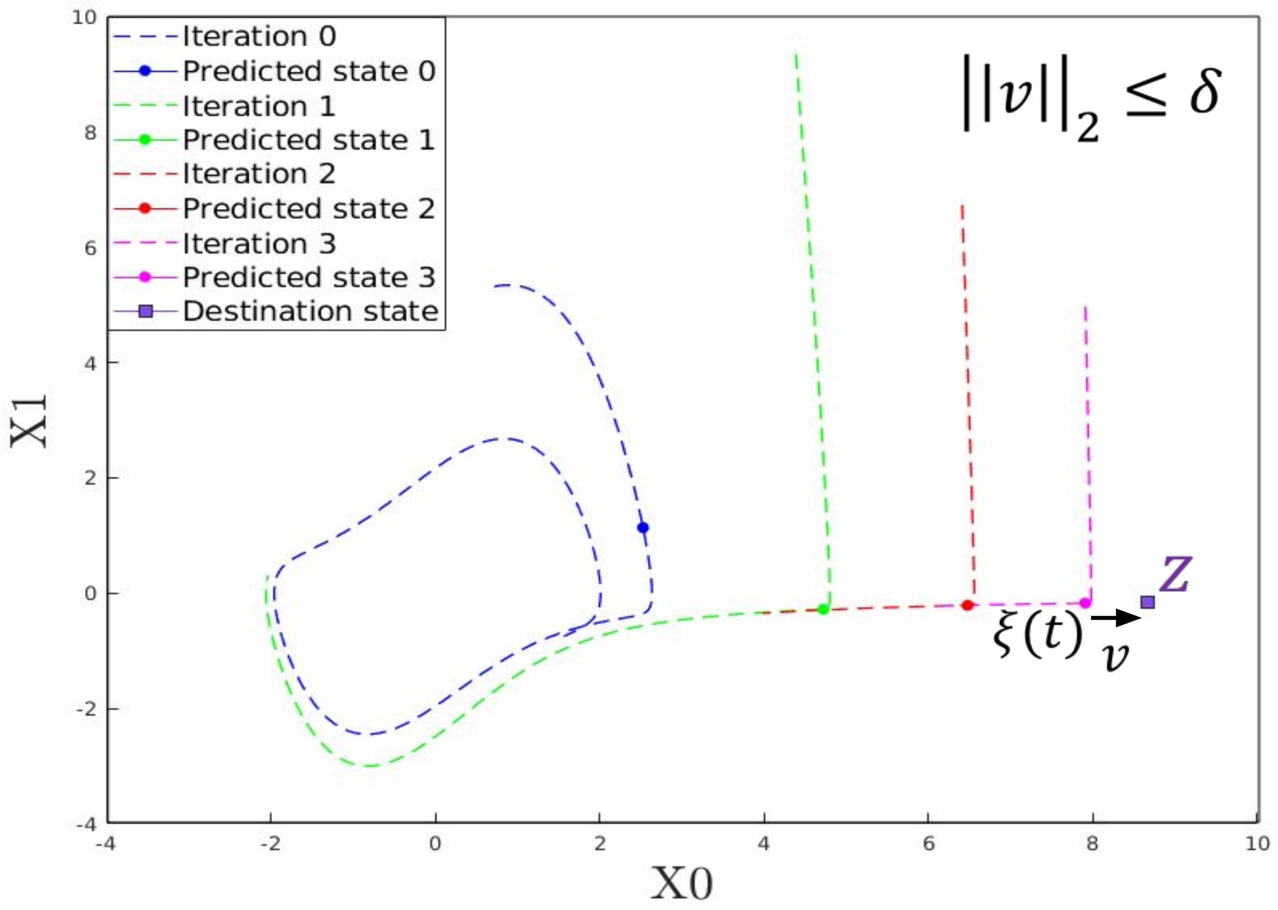
Evaluation: ReachDestination



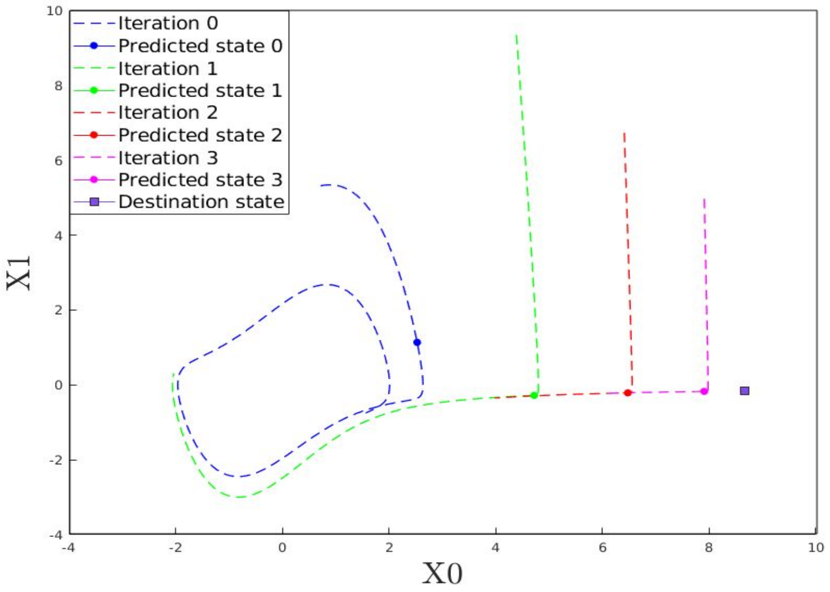
Evaluation: ReachDestination



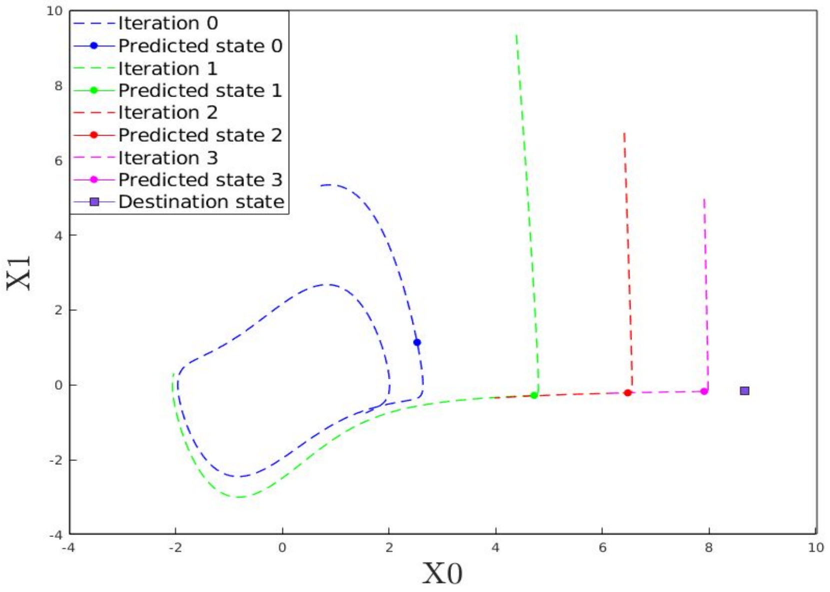
Evaluation: ReachDestination

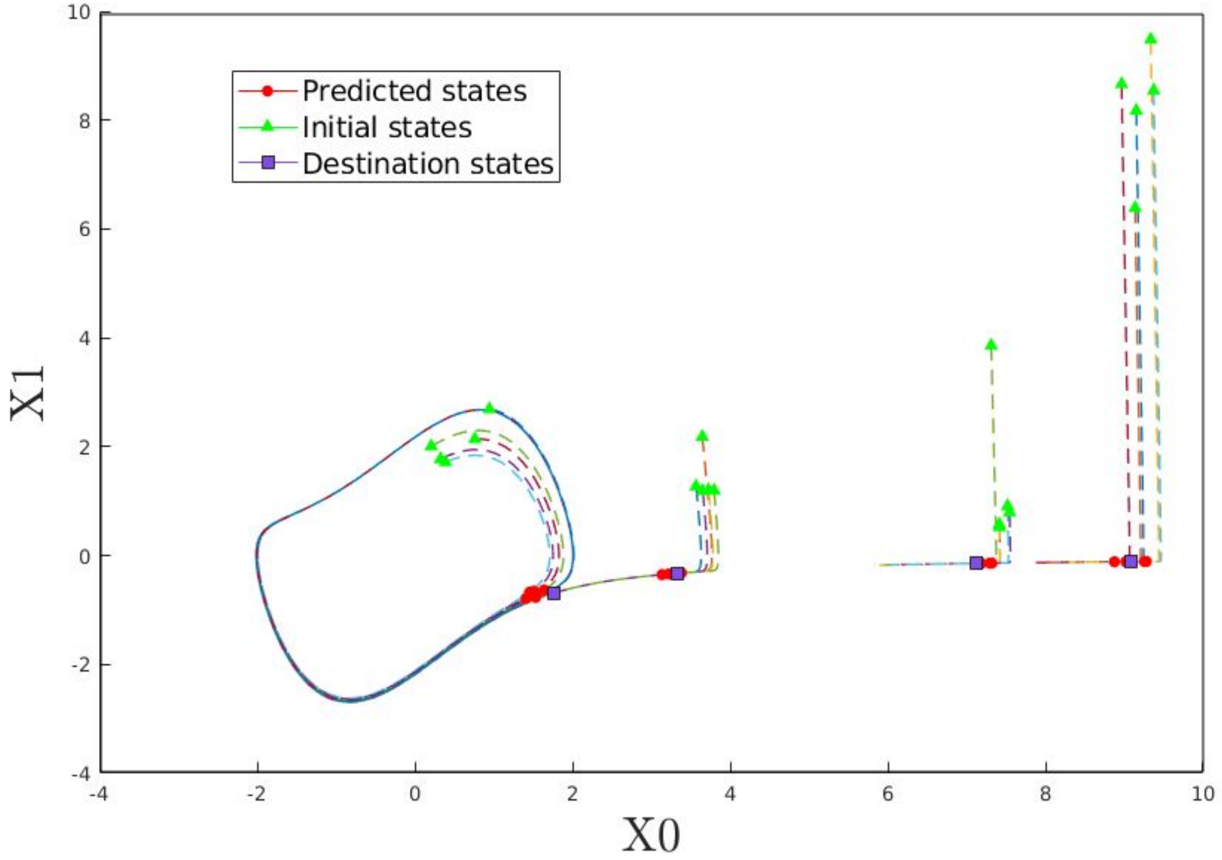


Evaluation: ReachDestination



Evaluation: ReachDestination





Evaluation: ReachDestination

Benchmark	Dims	Iteration count = 1			Iteration count = 5		
		d_a	d_r	Time (ms)	d_a	d_r	Time (ms)
Brussellator	2	[0.19–1.87]	[0.23–0.74]	11.38	[0.003–0.22]	[0.01–0.12]	31.34
Buckling	2	[1.67–11.52	[0.17–0.45]	13.61	[0.36- 2.09]	[0.06–0.31]	34.51
Lotka	2	[0.08–0.24]	[0.21–0.45]	12.38	[0.02–0.07]	[0.09–0.22]	34.28
Jetengine	2	[0.05 -0.20]	[0.19–0.28]	15.96	[0.0004–0.05]	[0.006–0.14]	38.26
HybridOsc	2	[0.28–0.92]	[0.13–0.29]	16.70	[0.03–0.31]	[0.01–0.10]	45.82
SmoothOsc	2	[0.37–1.09]	[0.13- 0.23]	52.22	[0.04–0.42]	[0.02–0.18]	136.72
Mountain Car	2	[0.004–0.24]	[0.08–0.22]	138.90	[0.0002–0.005]	[0.03–0.12]	266.76
Quadrotor	6	[0.014–1.09]	[0.10–0.67]	284.96	[0.004–0.04]	[0.02–0.13]	668.78

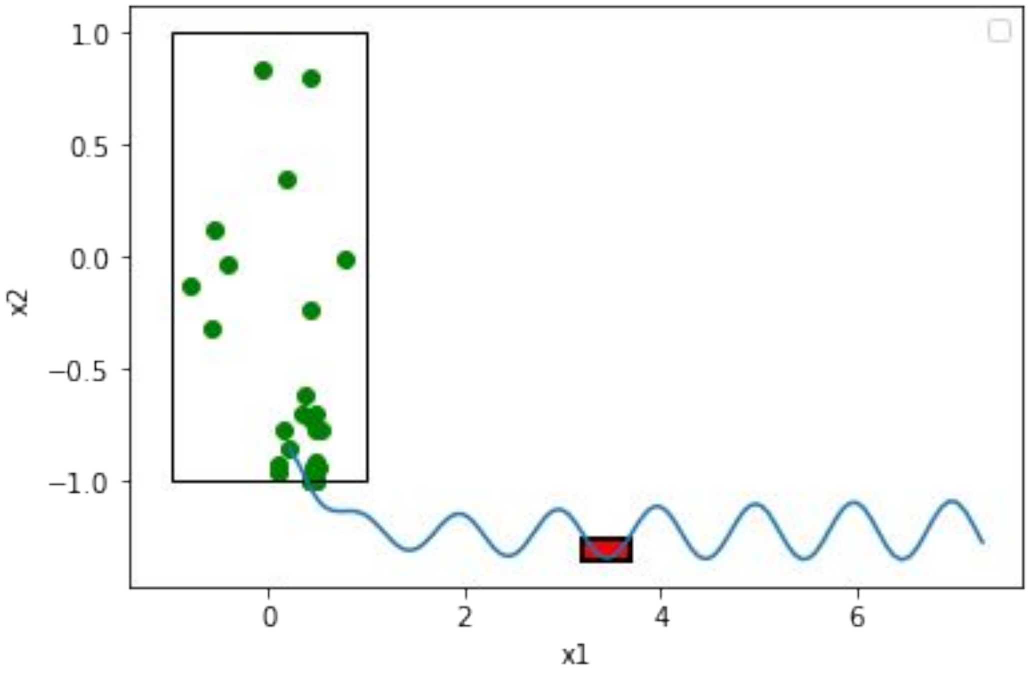
Outline

- ✔ Introduction
- ✔ Preliminaries
- ✔ Methodology
- ✔ Evaluation
- Applications
- Discussion

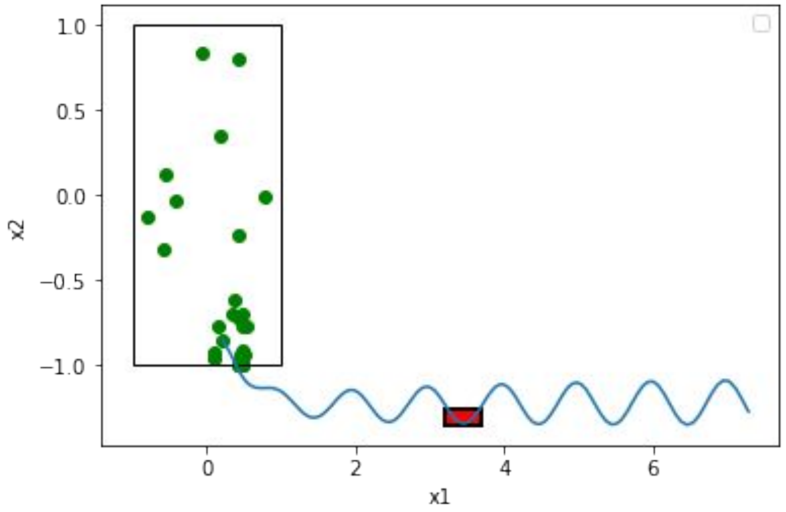
Falsification using Φ^{-1}

- Generate a set of random states in the unsafe set U
- Perform **ReachDestination** for each of those random states
- Terminate once a falsifying execution is obtained

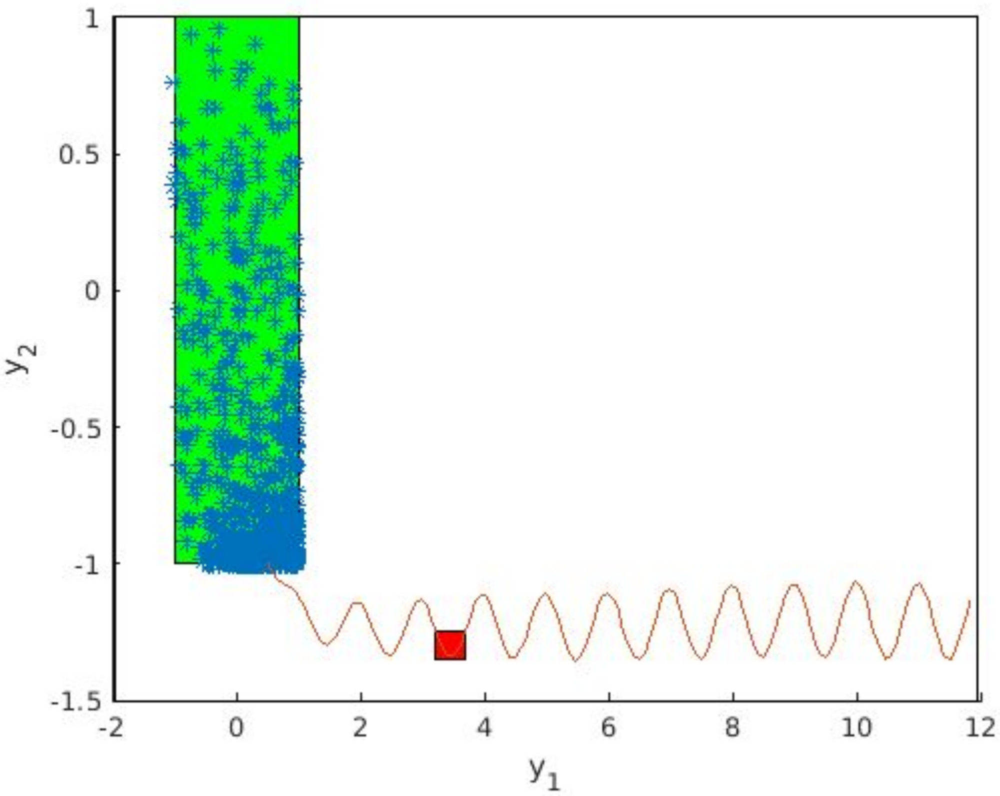
Evaluation: Falsification



Evaluation: Falsification

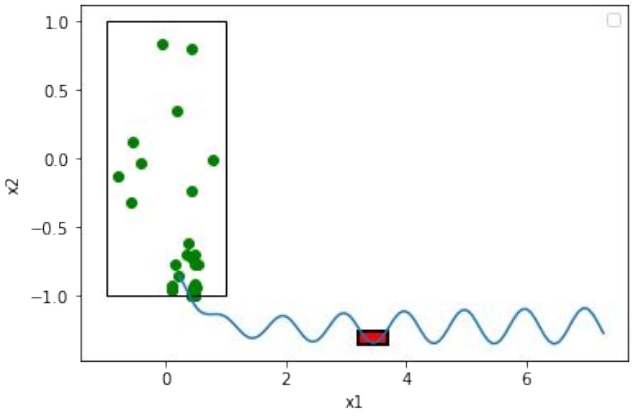


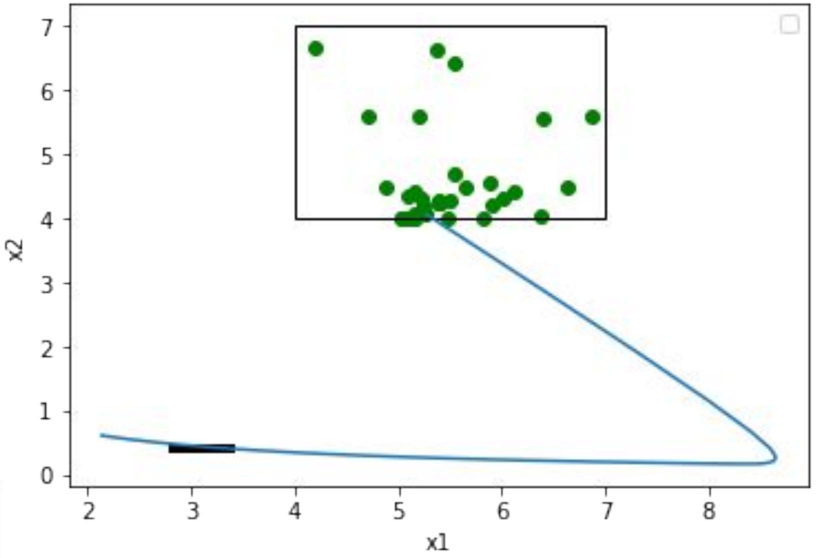
NeuralExplorer

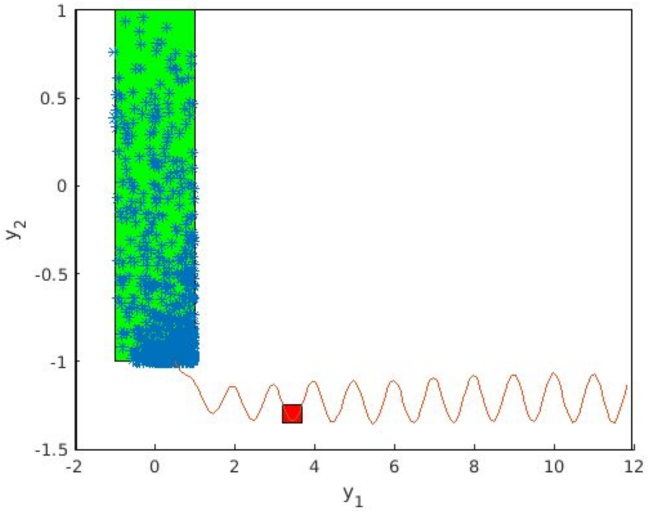


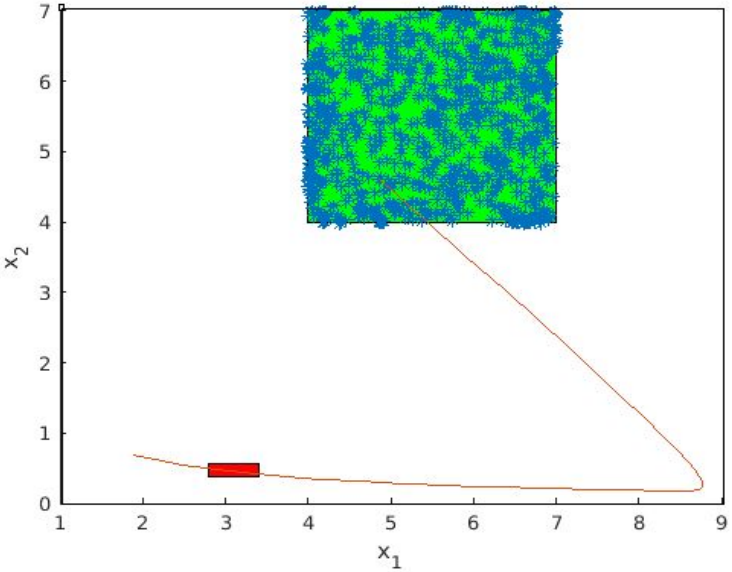
S-Taliro

Evaluation: Falsification

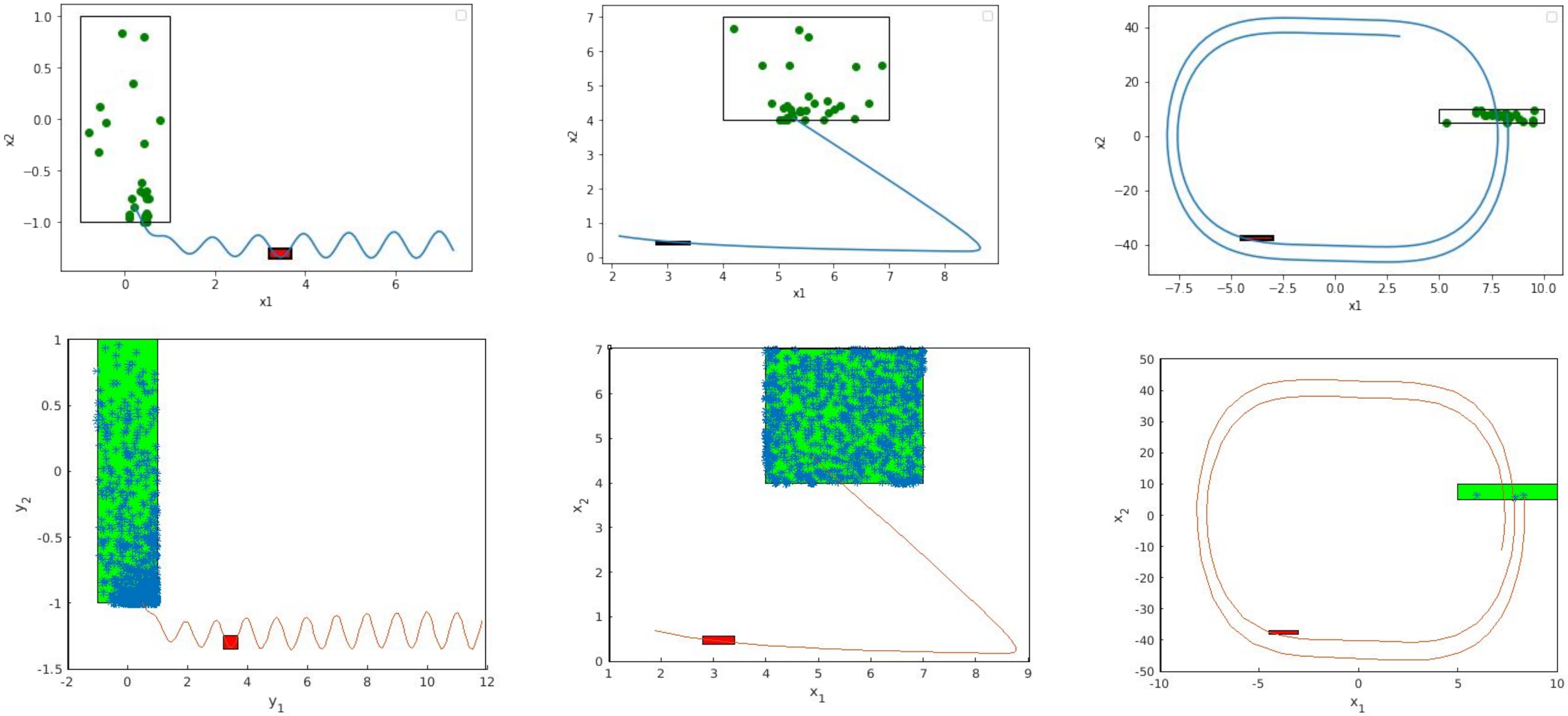








Evaluation: Falsification

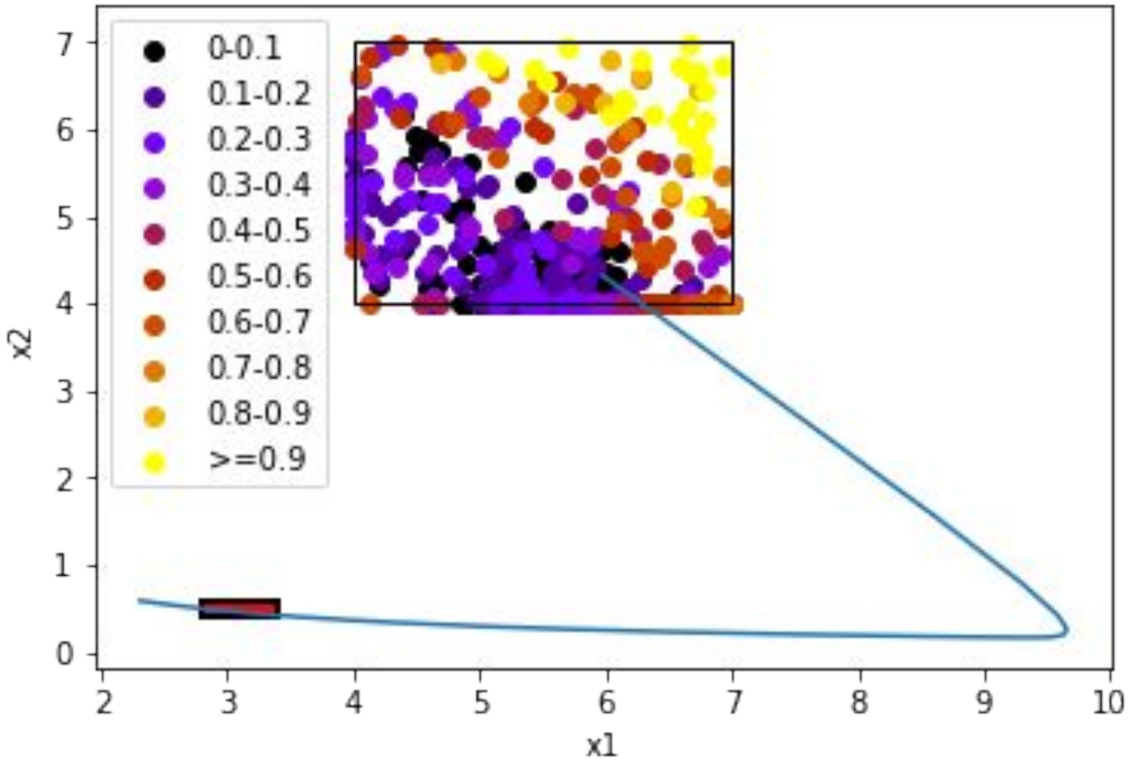


Density based estimation with Φ^{-1}

- Sample random states in the unsafe set
- Run **ReachDestination** on each state for a fixed number of times
- Maintain distance profiles of states explored in the initial set

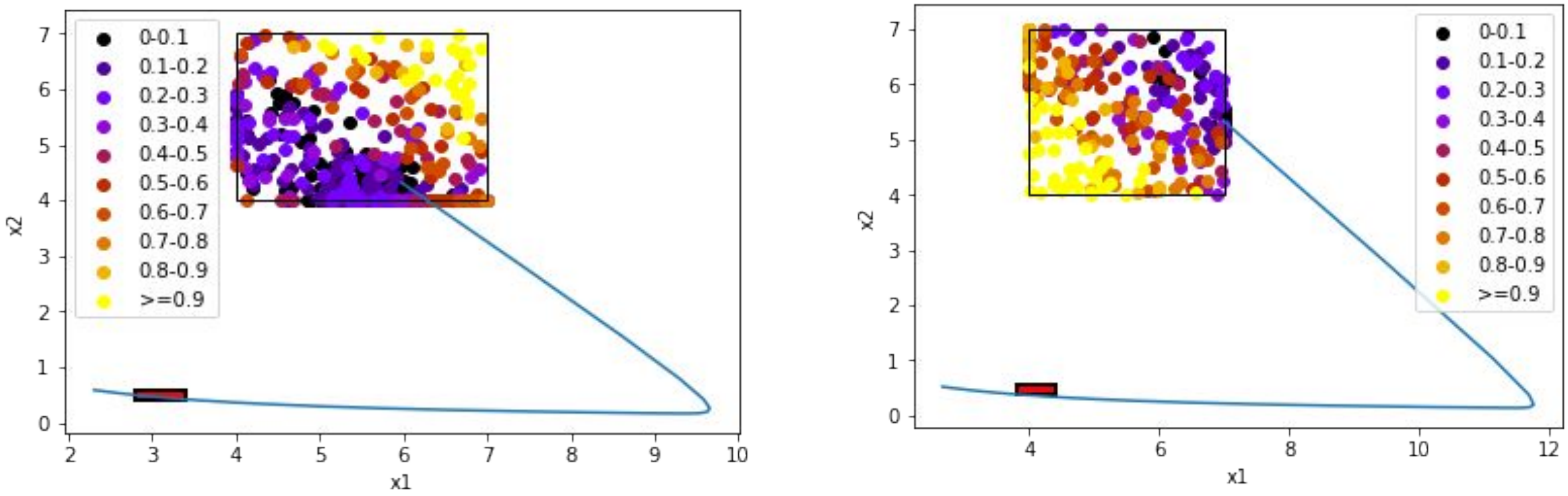
Density based estimation with Φ^{-1}

- Sample random states in the unsafe set
- Run **ReachDestination** on each state for a fixed number of times
- Maintain distance profiles of states explored in the initial set



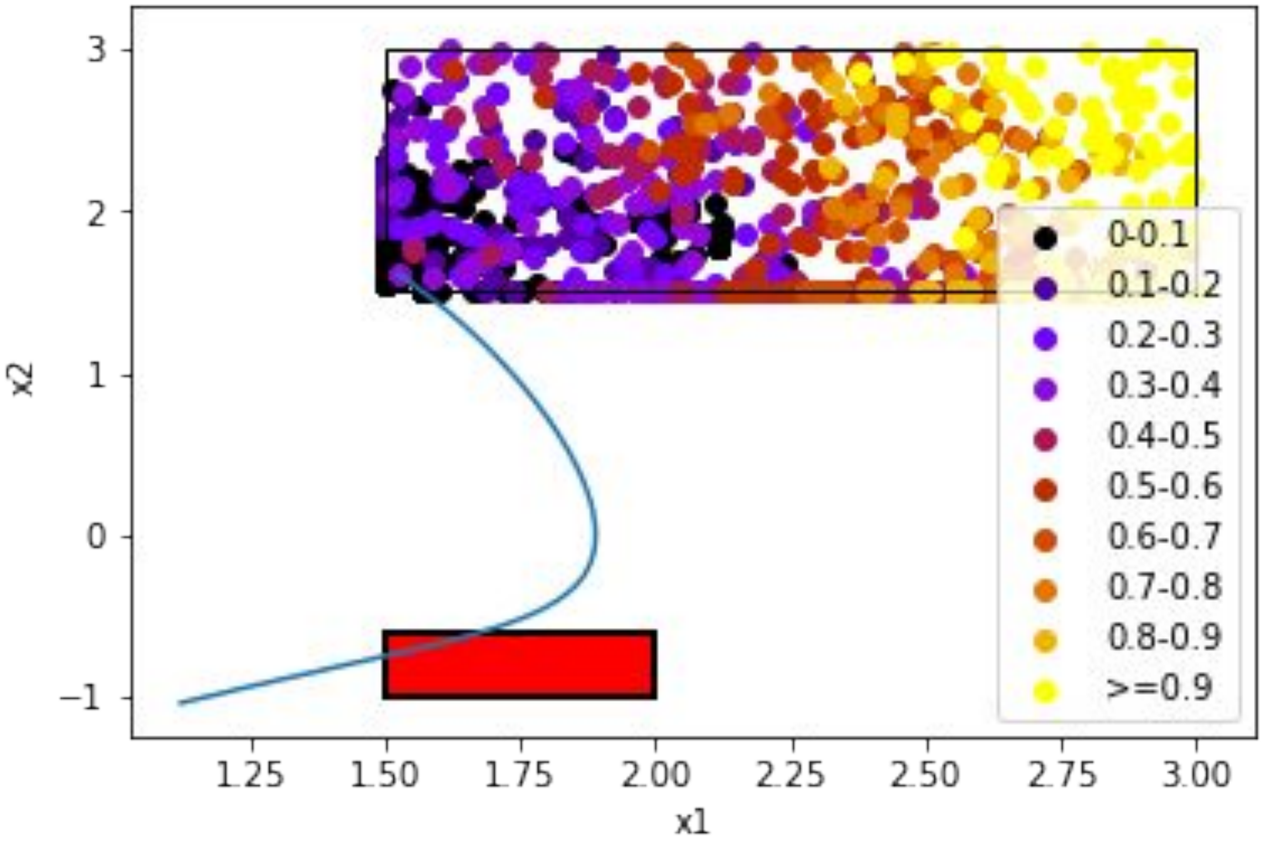
Brussellator

Density based estimation with Φ^{-1}



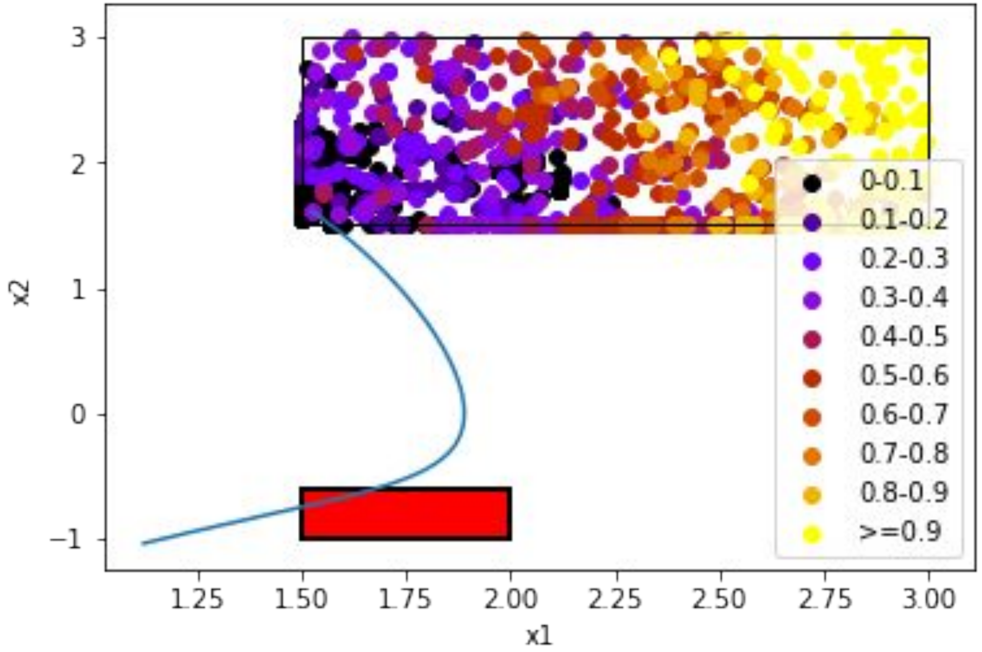
Brussellator: Change in spec

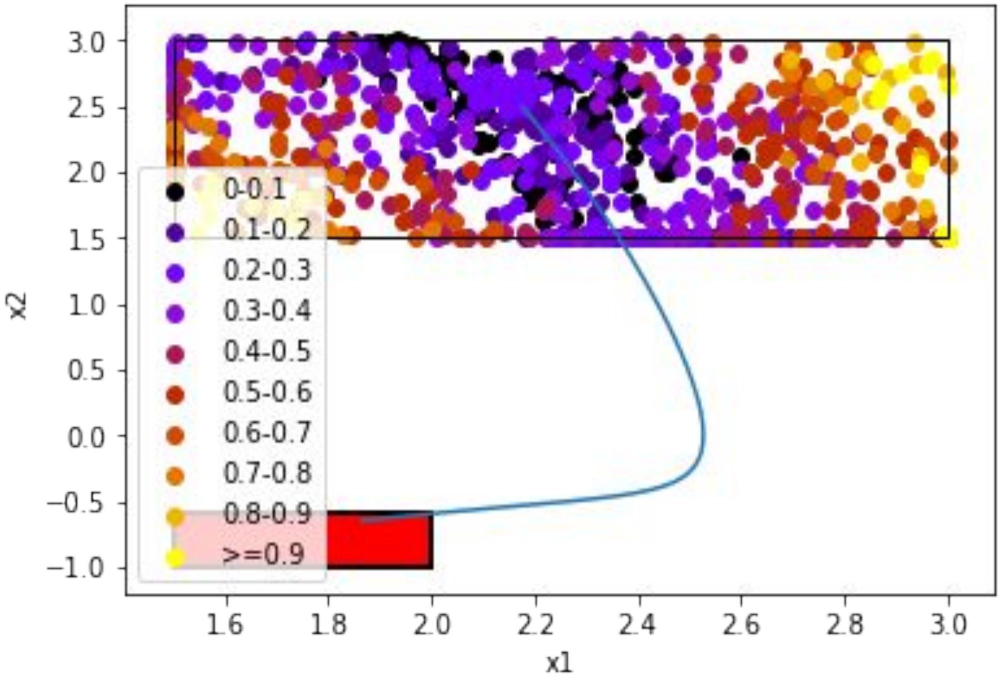
Density based estimation with Φ^{-1}



Vanderpol

Density based estimation with Φ^{-1}

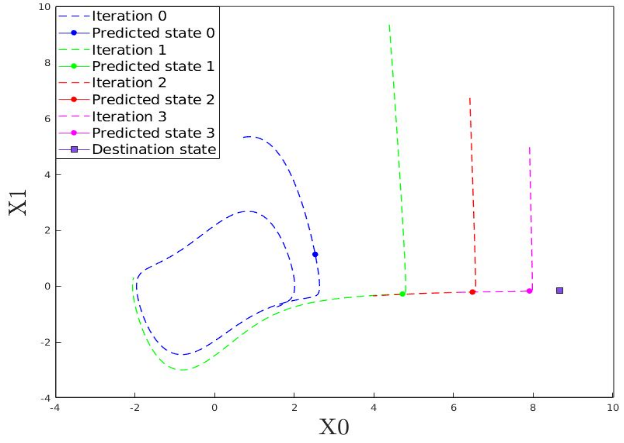




Vanderpol: Change in time instance

Advantages

- Each subsequent trajectory would make progress towards the destination
- Effective when the safety specification is modified
- Provides geometric insight

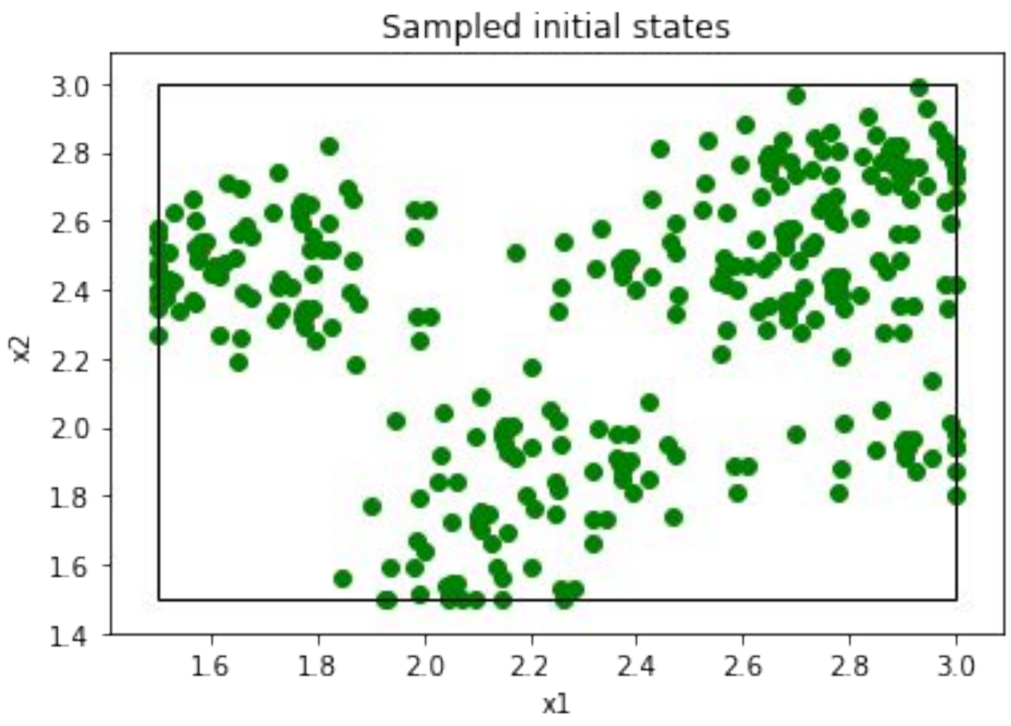


State space exploration with Φ

- Sample a set of states in the initial set
- Generate anchor trajectories from these initial states
- Sample a fixed number of states around each initial state
- Use NN_{sen} to predict the trajectories starting from new states

State space exploration with Φ

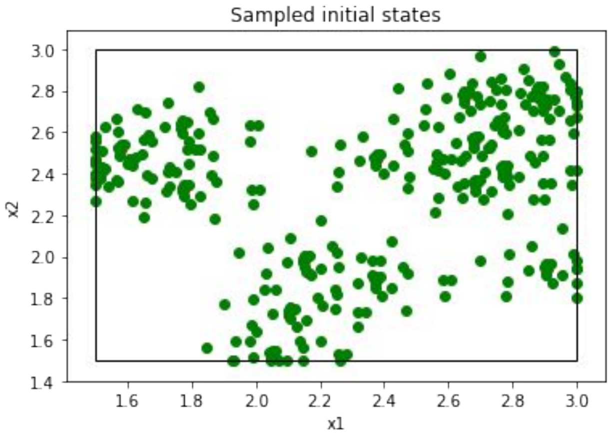
- Sample a set of states in the initial set
- Generate anchor trajectories from these initial states
- Sample a fixed number of states around each initial state
- Use NN_{sen} to predict the trajectories starting from new states



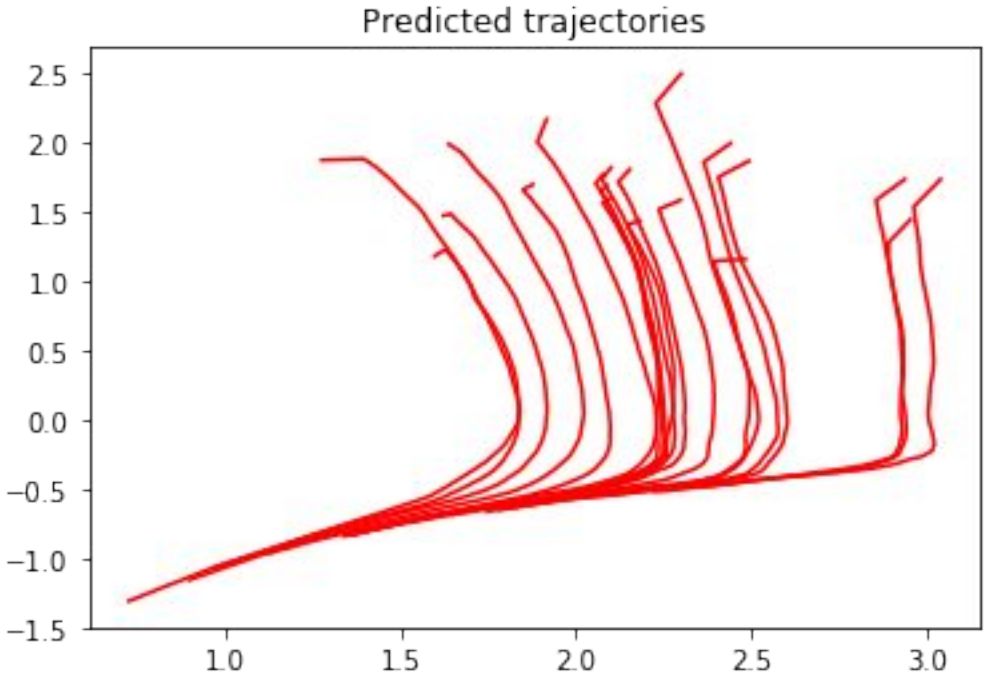
Vanderpol

State space exploration with Φ

- Sample a set of states in the initial set
- Generate anchor trajectories from these initial states
- Sample a fixed number of states around each initial state
- Use NN_{sen} to predict the trajectories starting from new states

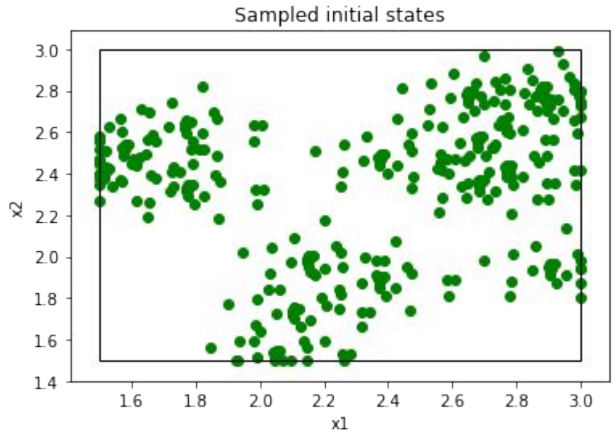


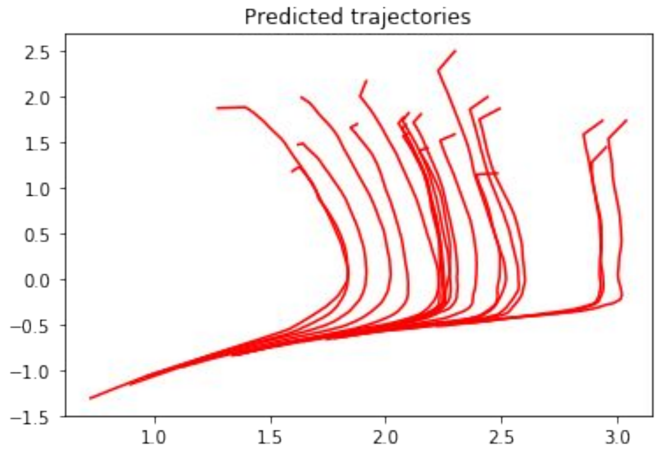
Vanderpol

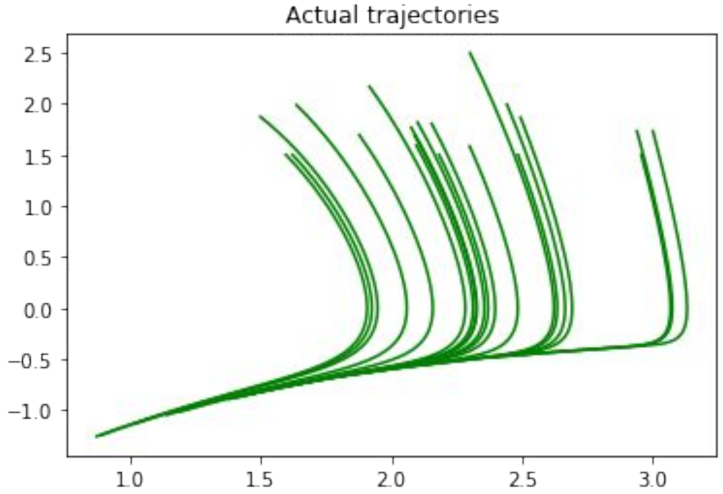


State space exploration with Φ

- Sample a set of states in the initial set
- Generate anchor trajectories from these initial states
- Sample a fixed number of states around each initial state
- Use NN_{sen} to predict the trajectories starting from new states







Vanderpol

Takeaways

- Approximation of sensitivity and inverse sensitivity using Neural networks
- Robust falsification *wrt* change in unsafe spec
- Density based state space exploration
- Provides geometric insights into the system behavior

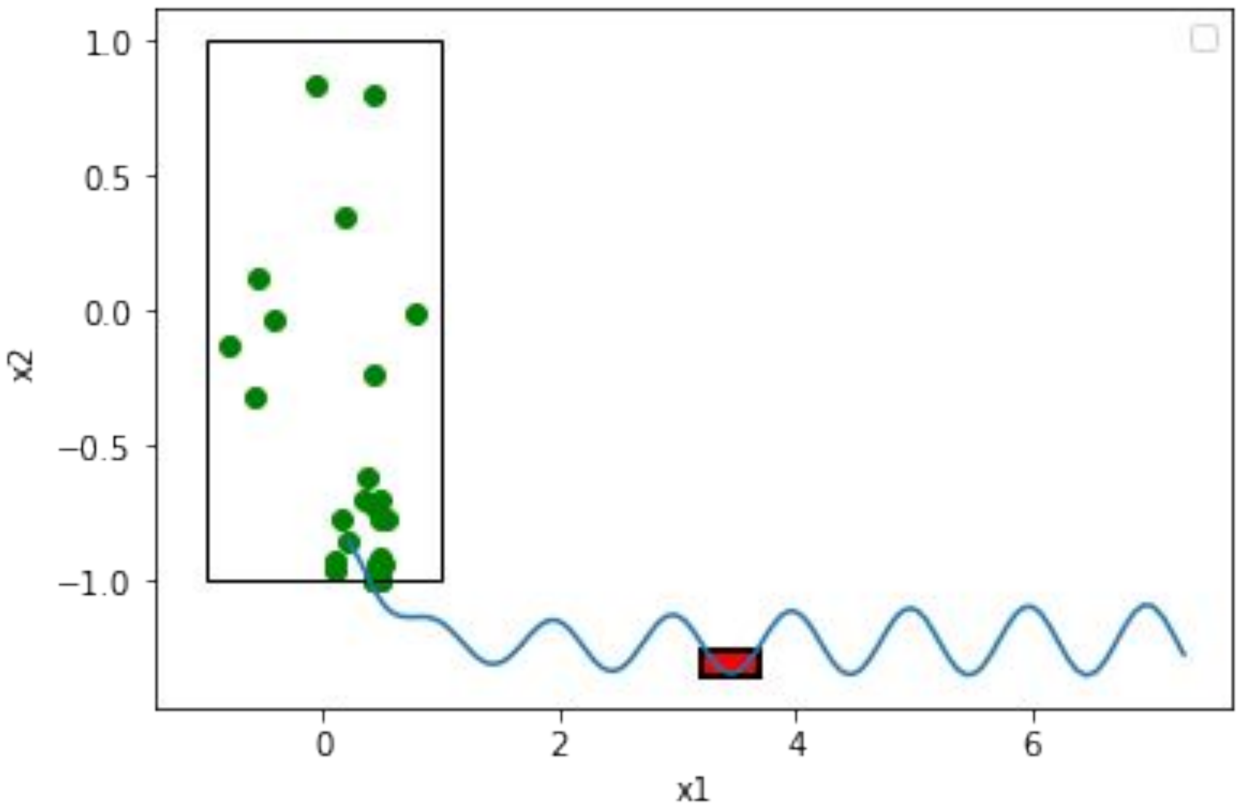
Takeaways

- Approximation of sensitivity and inverse sensitivity using Neural networks
- Robust falsification *wrt* change in unsafe spec
- Density based state space exploration
- Provides geometric insights into the system behavior

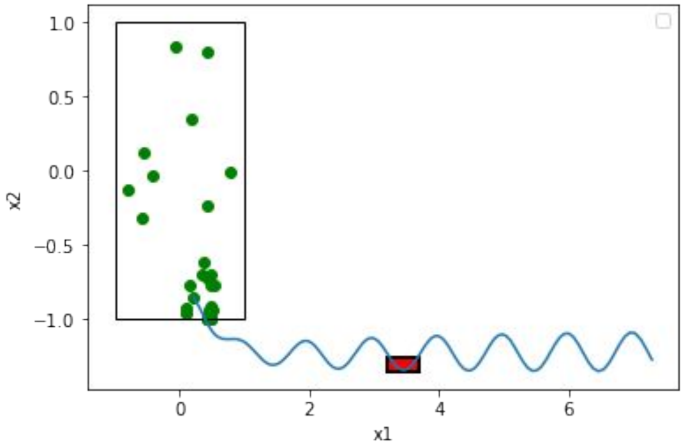
Future Work

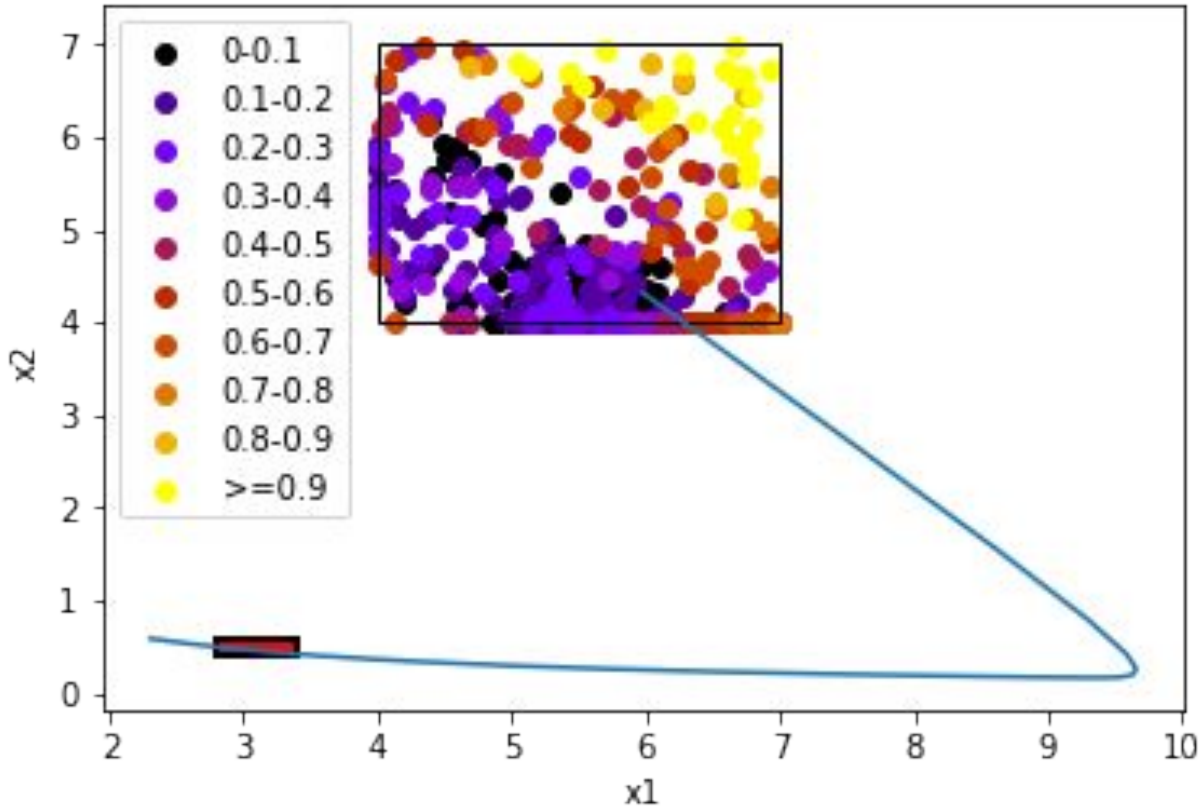
- Handle generic systems such as feedback systems with environmental inputs
- Devise a better framework to reduce training time
- Explore techniques to mask timing information for **reachDestination** subroutine

NeuralExplorer: State Space Exploration of Closed-loop Control Systems using NN

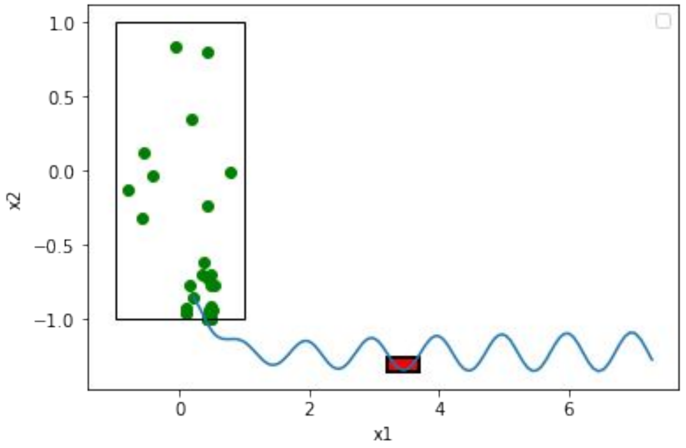


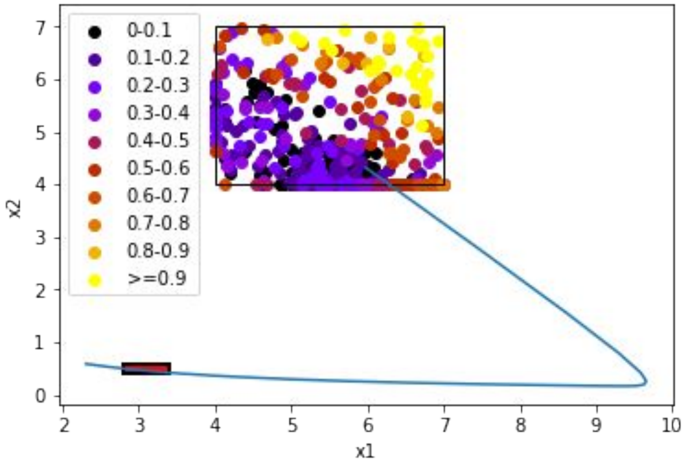
NeuralExplorer: State Space Exploration of Closed-loop Control Systems using NN



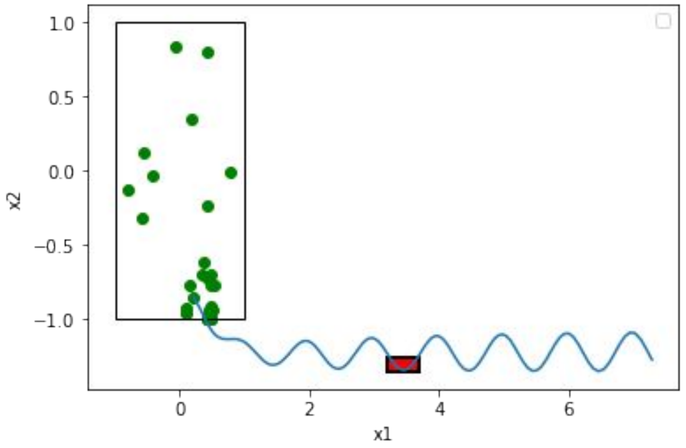


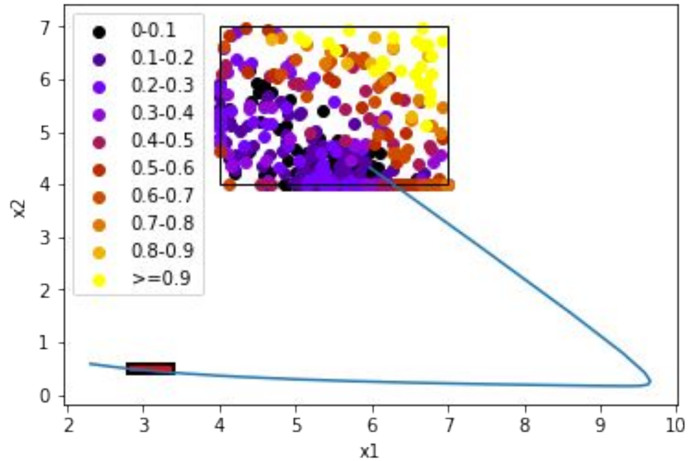
NeuralExplorer: State Space Exploration of Closed-loop Control Systems using NN

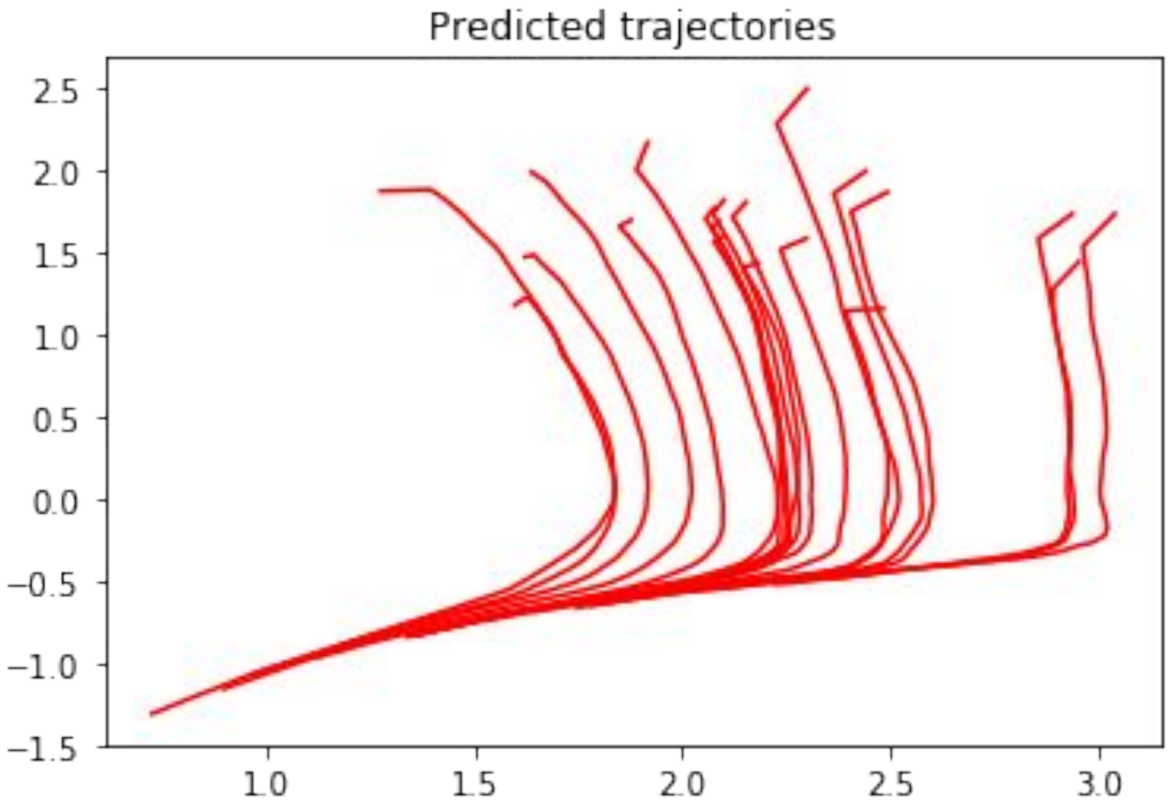




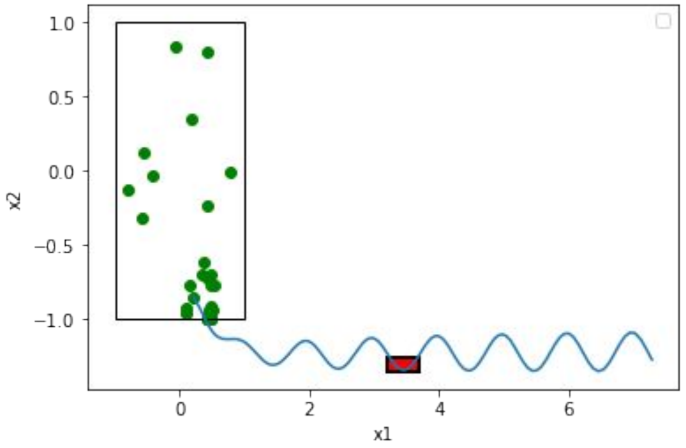
NeuralExplorer: State Space Exploration of Closed-loop Control Systems using NN

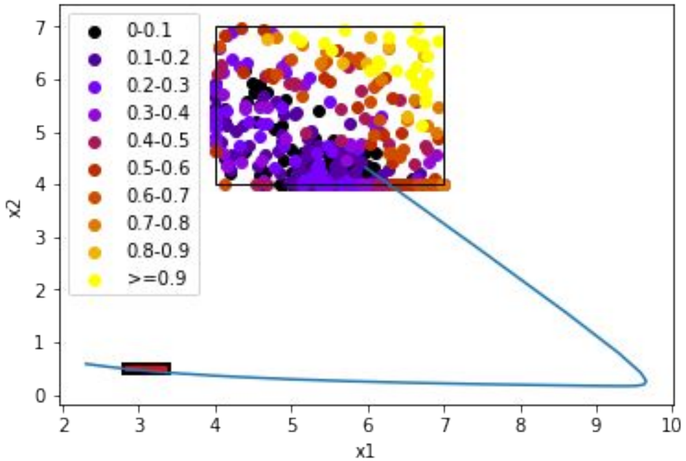


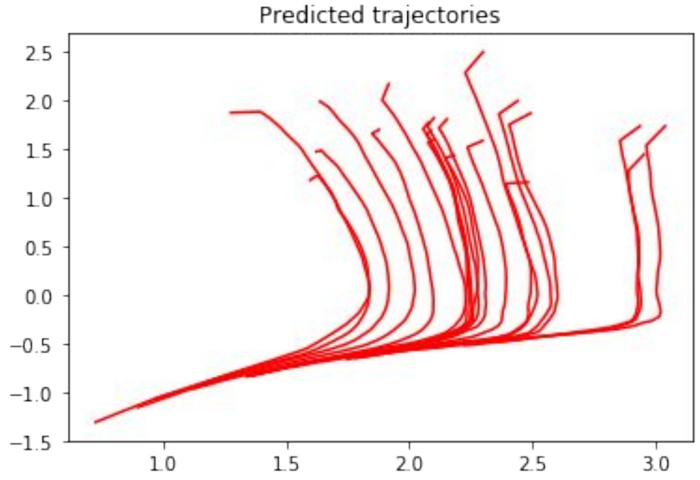




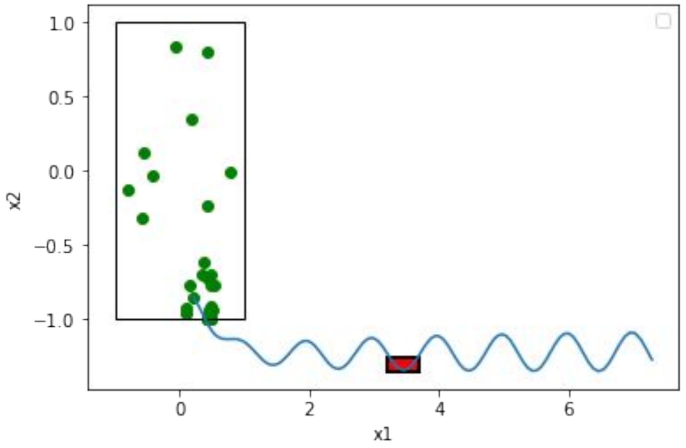
NeuralExplorer: State Space Exploration of Closed-loop Control Systems using NN

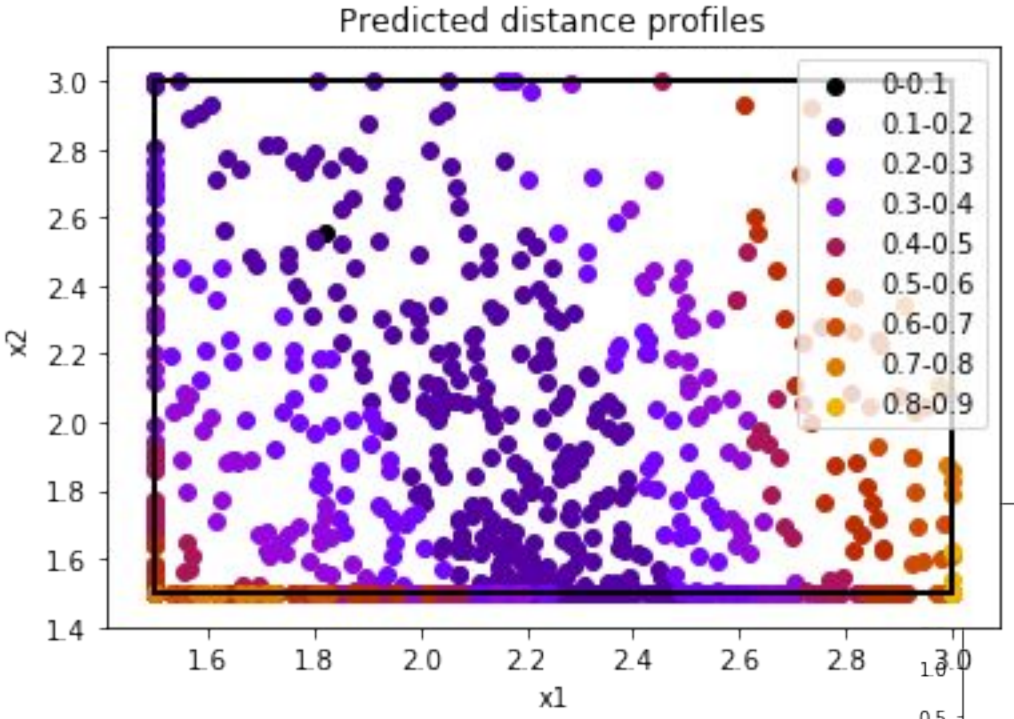


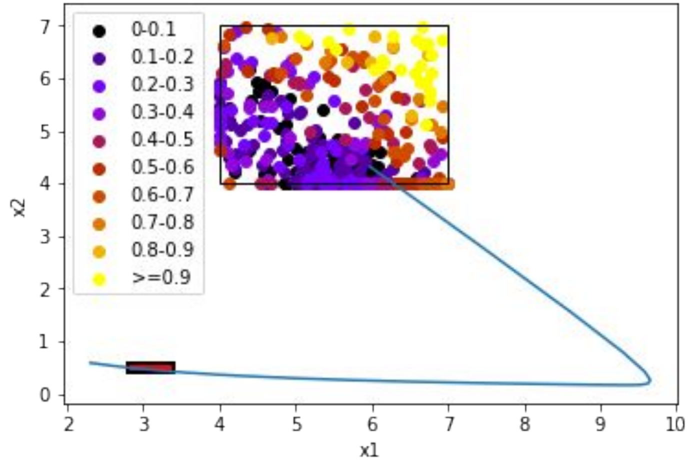


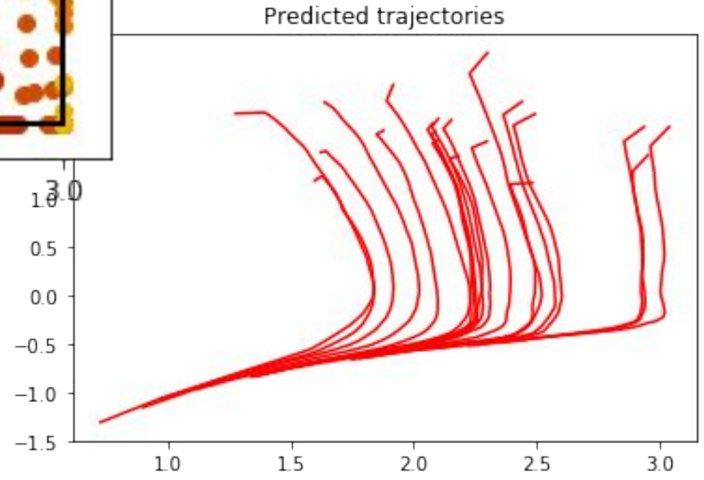


NeuralExplorer: State Space Exploration of Closed-loop Control Systems using NN

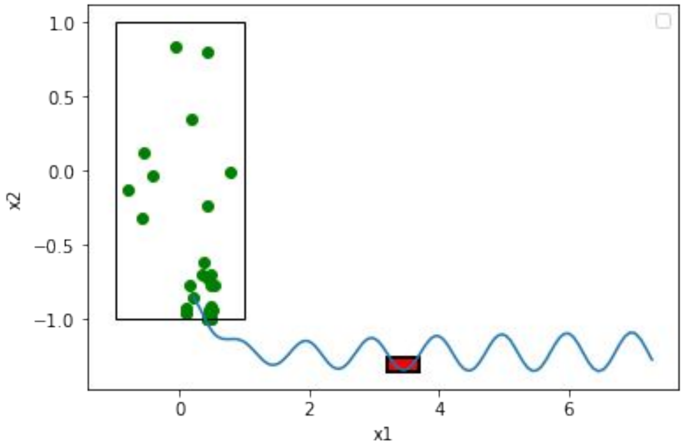




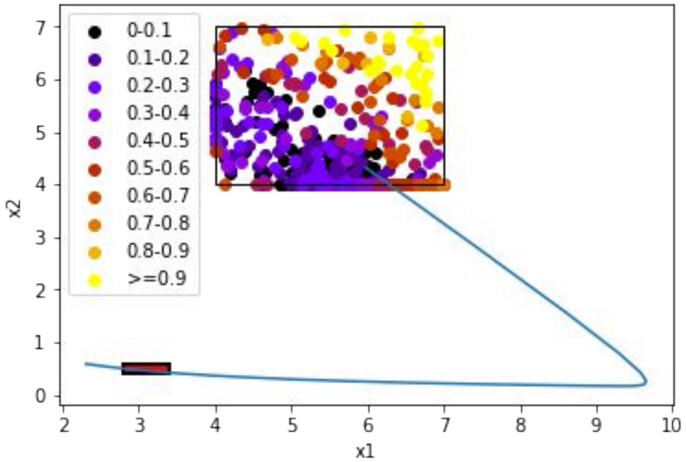


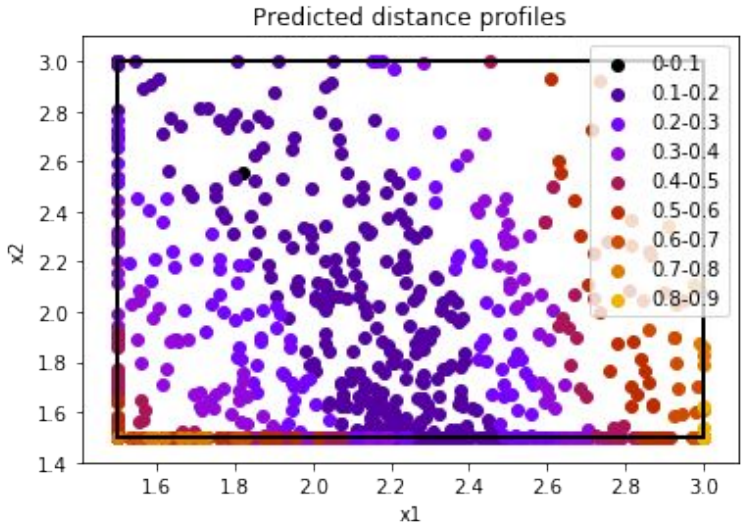


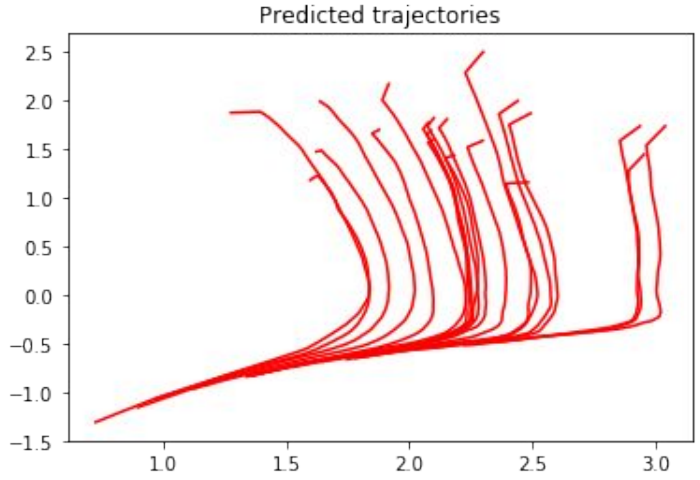
NeuralExplorer: State Space Exploration of Closed-loop Control Systems using NN



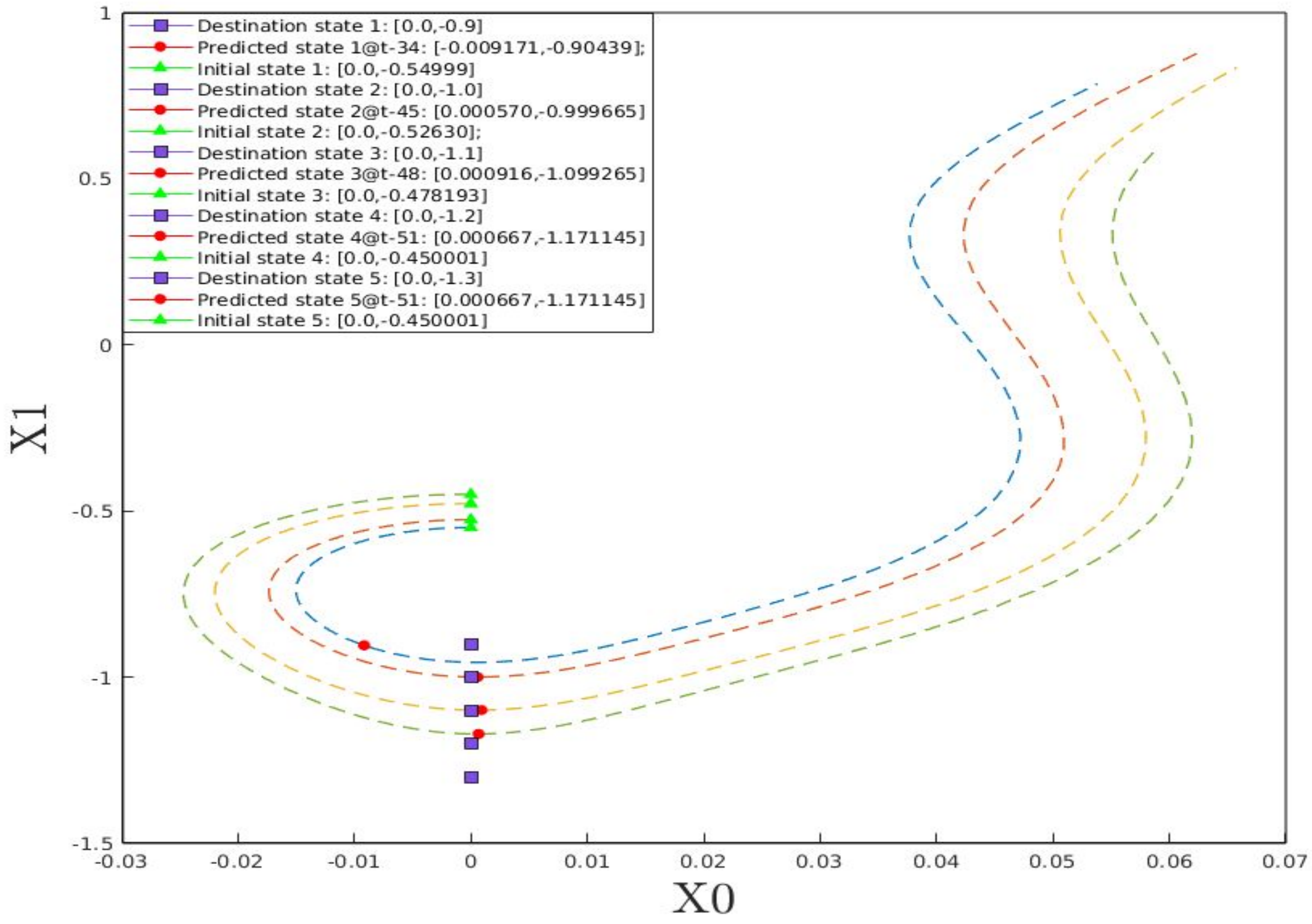
https://github.com/mag16154/NeuralExplorer







ReachTarget on a time interval



Mountain Car