

Optimal Control-Based Falsification of Learnt Dynamics via Neural ODEs and Symbolic Regression

Downloaded from: https://research.chalmers.se, 2026-11-23 18:39 UTC

Citation for the original published paper (version of record):
Kötz, L., Sjöberg, J., Åkesson, K. (2026). Optimal Control-Based Falsification of Learnt Dynamics via Neural ODEs and Symbolic Regression. [Source Title missing].
http://dx.doi.org/10.1145/3801146.3805672

N.B. When citing this work, cite the original published paper.

research.chalmers.se offers the possibility of retrieving research publications produced at Chalmers University of Technology. It covers all kind of research output: articles, dissertations, conference papers, reports etc. since 2004. research.chalmers.se is administrated and maintained by Chalmers Library

(article starts on next page)

Optimal Control-Based Falsification of Learnt Dynamics via Neural ODEs and Symbolic Regression

Lasse Kötz
kotz@chalmers.se
Chalmers University of Technology
Gothenburg, Sweden

Jonas Sjöberg
Chalmers University of Technology
Gothenburg, Sweden

Knut Åkesson
knut@chalmers.se
Chalmers University of Technology
Gothenburg, Sweden

Abstract

We present a falsification framework that integrates learned surrogate dynamics with optimal control to efficiently generate counterexamples for cyber-physical systems specified in signal temporal logic (STL). The unknown system dynamics are identified using neural ODEs, while known a-priori structure is embedded directly into the model, reducing data requirements. The learned neural ODE is converted into an analytical form via symbolic regression, enabling fast and interpretable trajectory optimization. Falsification is cast as minimizing STL robustness over input trajectories; negative robustness yields candidate counterexamples, which are validated on the original system. Spurious traces are iteratively used to refine the surrogate, while true counterexamples are returned as final results. Experiments on ARCH-COMP 2024 benchmarks show that this method can require orders of magnitude fewer experiments on the system under test than optimization-based approaches that do not model system dynamics.

Keywords

Falsification, Optimal control, Surrogate modeling, Neural ODEs, Symbolic Regression

ACM Reference Format:
Lasse Kötz, Jonas Sjöberg, and Knut Åkesson. 2026. Optimal Control-Based Falsification of Learnt Dynamics via Neural ODEs and Symbolic Regression. In *29th ACM International Conference on Hybrid Systems: Computation and Control (HSCC '26), May 11–14, 2026, Saint Malo, France*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3801146.3805672

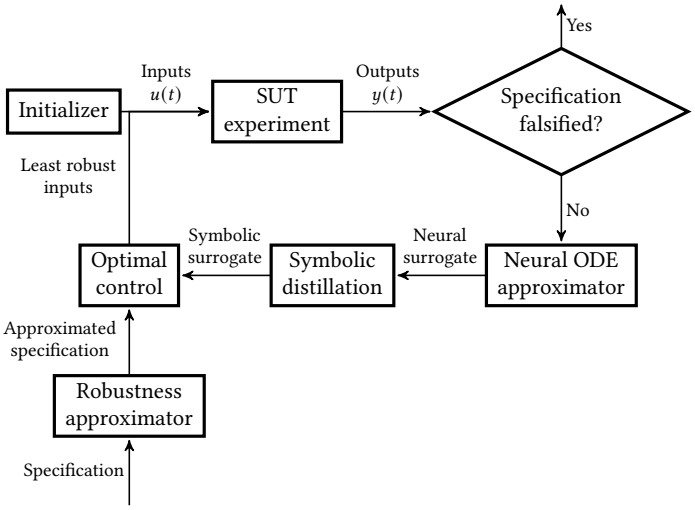


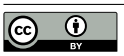
Figure 1: The SUT is approximated using neural surrogates with every new trajectory available, before being distilled into a symbolic representation. Counterexamples are computed using optimal control of the symbolic surrogate but must always be verified on the SUT to avoid spurious counterexamples.

1 INTRODUCTION

Rapid technological advancements have made systems increasingly complex. Cyber-physical systems (CPS) are now widely deployed in safety-critical products such as autonomous vehicles and healthcare devices, which puts even greater emphasis on ensuring adherence to desired behavior. Formal verification techniques—including model checking [1] and deductive verification [9]—have proven effective for verifying the correctness of smaller systems under test (SUT) or sub-systems. However, for sufficiently complex SUTs, formal verification techniques are generally not applicable or feasible.

Falsification has emerged as an alternative method that aims to find a *counterexample*, violating given safety specifications. A specification is typically expressed in signal temporal logic (STL), an expressive formalism capable of capturing a wide range of signal behaviors. A counterexample corresponds to a trace of input and output signals for the SUT that violates the specification. In classical falsification, this is done by performing experiments on the SUT, treating it as a black-box model and observing its input–output traces. Depending on the SUT, experiments can be carried out either through simulation or by conducting physical experimentation. In practice, model-in-the-loop simulations can be performed in early design phases and at a much larger scale than physical experiments, which are typically far more costly and time-consuming to conduct.

When a counterexample is found, it can be used to correct the identified safety violation in the SUT. Such a counterexample is an input–output trajectory that violates the specified safety property. Falsification avoids many of the challenges associated with formal verification, as it only requires that experiments can be conducted on the SUT—either in simulation or by physical experiments. However, existing falsification methods typically fail to exploit all information gained from experiments, instead relying on repeatedly running new experiments.



This work is licensed under a Creative Commons Attribution 4.0 International License.
HSCC '26, Saint Malo, France
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2566-1/2026/05
https://doi.org/10.1145/3801146.3805672

In this work, we propose an optimal control-based falsification method built on neural ODEs [5] and symbolic regression (SR), as shown in Figure 1. In short, a neural ODE models the system dynamics by representing the time derivative of the state with a neural network, yielding a nonlinear differential equation whose solution approximates the state evolution. The neural surrogate is then used to distill a symbolic representation through symbolic regression techniques. With this symbolic model available, an optimal control problem is formulated to steer the surrogate toward a trajectory that violates the STL specification, if possible.

As a final step, the counterexample is validated by running an experiment on the SUT using the optimal control input and evaluating the true robustness value. The robustness value is defined as positive if the specification is satisfied and negative if it is violated. If the SUT experiment also violates the specification, the algorithm terminates and returns the counterexample as the solution to the falsification problem. If not, the trajectory produced from the SUT experiment is added to the dataset used to learn the surrogate dynamics, and the loop is repeated with the expanded dataset. Using neural ODEs as a surrogate framework naturally supports nonlinear dynamics and full differentiability of the state evolution with respect to the input signal.

We outline the main contributions of this paper below:

- A falsification method is proposed that learns system dynamics from experiments using neural and symbolic surrogate models. The falsification step is performed via an optimal control formulation that relies on the symbolic representation of the learned dynamics. An approximation of the specification is incorporated into the optimal control problem to guide the search toward violating trajectories. To the authors' knowledge, this is the first method that combines learning-based nonlinear surrogate modeling with an optimal control approach for falsification.
- The performance is evaluated and compared for several benchmark specifications in the ARCH-COMP competition [15], including performance comparison to the participating tools in the competition.

2 RELATED WORK

The falsification problem has been approached in many different ways. In black-box falsification, specifications are often violated using heuristic or gradient-free optimization methods, where input generators are parametrized and the optimization procedure attempts to find parameter values that falsify the specification. Successful approaches include, for example, hybrid-corners random, Nelder–Mead, and line-search falsification.

More recent methods aim to learn a surrogate of the objective function—rather than the system dynamics—and balance exploration and exploitation, typically using Bayesian optimization (BO) [28]. These strategies [7, 27] guide sampling toward regions of the input-parameter space where low robustness is expected, thereby reducing the number of SUT evaluations required for falsification.

However, these methods lack the ability to model the dynamics explicitly. Instead, they rely on representing the surrogate as a simplified probabilistic mapping between input signal parameters and the expected objective value. Furthermore, they scale poorly with the dimensionality of the input parameter space, which further constrains parameterization.

A further step in the development has been to approximate a surrogate model for the underlying dynamical system iteratively, and subject it to optimization-based falsification methods instead of attempting to falsify the relatively expensive SUT directly. This approach offers flexibility but is only effective when the discrepancy between the SUT and the surrogate remains relatively small. Prior tools include ARIsTEO [22], which leverages system identification to identify a range of models and subjects the cheaper surrogate to black-box falsification.

A related approach [2] is to use Koopman operator theory to model the time evolution of a set of observables. This provides linear dynamics of the observables via dynamic mode decomposition (DMD), which enables the use of MILP-solvers to find the least robust trace and verifying them on the SUT. For nonlinear dynamics in general, however, there need not be a finite subspace of observables that is invariant under the Koopman operator, making them inherently inexact. Furthermore, DMD struggles with linear approximations for systems with multiple fixed points or periodic orbits, which are common properties of nonlinear dynamical systems.

Other notable surrogate model-based frameworks include NN-Fal [16] which trains a static depth deep neural network as a surrogate, mapping input vectors to output vectors. The falsification/verification step is carried out using adversarial attack algorithms and deep neural network verifiers. It is however, limited in what kind of temporal operators it supports during falsification and does not learn any continuous-time nonlinear dynamics in which prior knowledge can be embedded. Ψ−TaLiRo [31] is another framework with industrial adaptation that searches for minimal robustness trajectories from arbitrary Simulink and Stateflow diagrams based on stochastic optimization techniques.

Online diffusion (OD) [25] is an online test generation algorithm, based on Wasserstein generative adversarial networks. The algorithm works by denoising test vectors corresponding to parameterizations of piecewise-constant input signals. Moonlight [23] is a monitoring tool and falsification framework which in the ARCH-COMP 2024 competition leverages BO with trust regions. ForeSee [34] is a framework which manipulates the specification's syntax tree to avoid the so-called scale problem that arises when falsifying signals of widely different magnitudes [34]. It uses Monte Carlo tree search to traverse the specification syntax tree. In FalCAuN [33] the SUT is treated as a black-box transition system with arbitrarily many states. It uses model checking with active automata learning to represent the system as a Mealy machine. EXAM-Net uses an adversarial approach to have the discriminator learn a mapping between inputs and robustness value. ATheNA [10] is a framework using a mix of manual and automatic fitness functions. The toolbox allows the engineer to target certain areas of the input space which may be especially critical for falsification.

To summarize, existing frameworks either do not model the full nonlinear dynamics of the system explicitly or do not provide usable gradients for gradient-based falsification of the surrogate, motivating our proposed framework.

3 BACKGROUND

3.1 Preliminaries

Definition 3.1 (trace). A trace is defined as a tuple

$$\tau = (x_0, u(t), y(t)) \tag{1}$$

consisting of the initial state x_0, output signal y, and input signal u sampled at discrete time steps $\mathbf{t}$ when obtained from experiments; continuous-time traces are discretized for robustness evaluation. In this paper, a trace can arise from:

- solving an optimal control synthesis problem,
- solving an initial-value problem (IVP) of the surrogate, or
- performing experiments on the SUT to obtain an input-output trajectory.

Definition 3.2 (Signal Temporal Logic [21]). From a set of atomic predicates μ, the syntax of STL is defined as

$$\varphi ::= \mu \mid \neg\mu \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \Box_{[a,b]}\psi \mid \Diamond_{[a,b]}\psi \mid \varphi \mathcal{U}_{[a,b]}\psi, \tag{2}$$

where $a \leq b$ are time-bounds for non-negative scalar values. The $\Box_{[a,b]}$ and $\Diamond_{[a,b]}$ denote the *globally* and *finally* operators in the closed interval $[a, b]$, respectively. Lastly, $\mathcal{U}_{[a,b]}$ denotes the *until* operator.

Definition 3.3 (STL Semantics). The semantics of STL are recursively defined as

$$\begin{aligned}
(y,t) &\vDash \mu && \iff \mu(y(t)) > 0\\
(y,t) &\vDash \neg\mu && \iff \neg((y,t) \vDash \mu)\\
(y,t) &\vDash \varphi \wedge \psi && \iff (y,t) \vDash \varphi \wedge (y,t) \vDash \psi\\
(y,t) &\vDash \varphi \vee \psi && \iff (y,t) \vDash \varphi \vee (y,t) \vDash \psi\\
(y,t) &\vDash \Box_{[a,b]}\varphi && \iff \forall t' \in [t+a, t+b], (y,t') \vDash \varphi\\
(y,t) &\vDash \Diamond_{[a,b]}\varphi && \iff \exists t' \in [t+a, t+b], (y,t') \vDash \varphi\\
(y,t) &\vDash \varphi \mathcal{U}_{[a,b]}\psi && \iff \exists t' \in [t+a, t+b], (y,t') \vDash \psi\\
& && \quad \wedge \forall t'' \in [t, t'], (y,t'') \vDash \varphi.
\end{aligned}$$

Definition 3.4 (Robustness Semantics). Finding a counterexample as a solution to an optimization problem requires quantifying how strongly the specification is violated or not violated. This *robustness value* is quantified with the following quantitative semantics [19]:

$$\begin{aligned}
(y,t) \vDash \varphi &\iff \rho^{\varphi}(y,t) > 0\\
\rho^{\neg\varphi}(y,t) &= -\rho^{\varphi}(y,t)\\
\rho^{\varphi_1 \wedge \varphi_2}(y,t) &= \min(\rho^{\varphi_1}(y,t), \rho^{\varphi_2}(y,t))\\
\rho^{\varphi_1 \vee \varphi_2}(y,t) &= \max(\rho^{\varphi_1}(y,t), \rho^{\varphi_2}(y,t))\\
\rho^{\Diamond_{[a,b]}\varphi}(y,t) &= \max_{t' \in [t+a,t+b]}(\rho^{\varphi}(y,t'))\\
\rho^{\Box_{[a,b]}\varphi}(y,t) &= \min_{t' \in [t+a,t+b]}(\rho^{\varphi}(y,t'))\\
\rho^{\varphi_1 \mathcal{U}_{[a,b]}\varphi_2}(y,t) &= \max_{t' \in [t+a,t+b]}(\min([\rho^{\varphi_2}(y,t'),\\
&\qquad \min_{t'' \in [t+a,t']}(\rho^{\varphi_1}(y,t''))]))
\end{aligned}$$

for some real-valued function $\rho^{\varphi} : \mathbb{R}^{n_{signals}} \mapsto \mathbb{R}$.

3.2 Problem statement

Given a SUT $\bar{\mathcal{S}}$ and a specification of desired behavior φ, the falsification problem is to find a *counterexample* such that

$$\bar{\mathcal{S}}(x_0, u(t)) \nvDash \varphi, \tag{3}$$

Algorithm 1 Neural-symbolic falsification with optimal control

Input: experiment budget, $\bar{\mathcal{S}}$, φ
Output: u_{OC}

1: $k \leftarrow 0$ ▷ Number of experiments on $\bar{\mathcal{S}}$
2: $\mathcal{D}_0 \leftarrow$ InitializeData($\bar{\mathcal{S}}$)
3: $\rho_{inc} \leftarrow$ ComputeRobustness($\bar{\mathcal{S}}, \varphi, \mathcal{D}_0$)
4: **while** $k \leq$ experiment budget **and** $\rho_{inc} > 0$ **do**
5: $\hat{\mathcal{S}} \leftarrow$ EstimateNeuralSurrogate($\mathcal{D}_k$)
6: $\hat{\mathcal{S}}_{symb} \leftarrow$ DistillSymbolic($\hat{\mathcal{S}}, \mathcal{D}_k$)
7: $u_{OC} \leftarrow$ OptimizeControl($\hat{\mathcal{S}}_{symb}, \varphi$)
8: $\tau_{OC} \leftarrow$ Experiment($\bar{\mathcal{S}}, u_{OC}$)
9: $\rho_{inc} \leftarrow$ ComputeRobustness($\bar{\mathcal{S}}, \varphi, \tau_{OC}$)
10: $\mathcal{D}_{k+1} \leftarrow \mathcal{D}_k \cup \{\tau_{OC}\}$
11: $k \leftarrow k + 1$
12: **end while**
13: **Return:** u_{OC}

where x_0 is the initial state, φ is the specification, commonly expressed in signal temporal logic, and $u(t) : \mathbb{R}_{\geq 0} \mapsto \mathbb{R}^m$ are the inputs to $\bar{\mathcal{S}}$.

4 LEARNING-BASED FALSIFICATION

Our proposed framework, Falsification using optimal Control and Neural Surrogates (FalConN) solves the falsification problem by combining neural surrogate modeling techniques, symbolic distillation, and optimal control synthesis. Assuming no prior knowledge of the dynamics, the process begins by running an experiment using a randomized input signal, constrained by specified upper and lower bounds, to initialize the estimation dataset. For simpler specifications, the initial experiment can be selected using alternative heuristic strategies instead. In general, however, the initial experiments should be designed to persistently excite the system, ensuring that the collected traces are informative for learning the dynamics.

The proposed method's main loop is the iterative training-distillation-optimal-control loop (steps 4 through 12 in Algorithm 1), which terminates when a true falsifying trace is found or the number of iterations has reached a preset experiment budget.

For each new trajectory added to the estimation dataset $\mathcal{D}_k$, our method approximates a neural surrogate model as shown in Alg 2. Specifically, we use neural ODEs for estimation, parameterizing the unknown dynamics with a neural network. The neural ODE formulation allows prior knowledge about the system to be embedded directly into the model, so that only the unknown components of the dynamics are learned. This significantly improves data efficiency when estimating the full dynamics. The surrogate is trained by minimizing the discrepancy between the observed and simulated trajectories. Incorporating prior structural information also improves generalization beyond the estimation data and leads to faster training convergence.

Our algorithm distills a symbolic form from the estimated neural surrogate, according to Algorithm 3, by employing symbolic regression methods (SR). The resulting symbolic surrogate serves as a basis for the optimal control problem by allowing the use of the sparse nonlinear optimizer (SNOPT [11]).

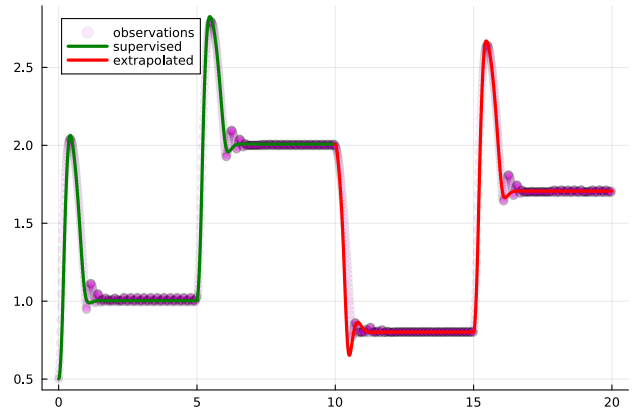


Figure 2: Simulation of neural ODE surrogate supervised training example. Training data lies in the interval $t \in [0, 10]$ and extrapolated simulation on unseen data in $t \in [10, 20]$. The trajectory is simulated from the NN-controller benchmark example [15].

In the last step, the symbolic surrogate representation and STL specification are jointly encoded as an optimal control problem. This optimal control problem can be solved for a least-robust input signal u_{OC} using sequential quadratic programming (SQP) solvers and specifically SNOPT by minimizing the robustness value. The resulting solution u_{OC} is verified by executing the SUT for u_{OC} to avoid returning spurious counterexamples resulting from surrogate inaccuracies. If u_{OC} also falsifies the specification on the SUT, it is returned as a solution to the falsification problem. If not, however, the trajectory is added to the estimation dataset $\mathcal{D}$ used to estimate the surrogate in subsequent iterations. By verifying counterexamples on the SUT, the proposed method is sound, although surrogate candidates alone may be inaccurate.

Alternatively, one could formulate the optimal control problem directly on the neural surrogate using a discretized neural ODE. This would avoid the loss of accuracy introduced by symbolic distillation, but also yield a substantially more difficult optimization problem, making it challenging to solve efficiently. A detailed exploration of this alternative is beyond the scope of this paper.

4.1 Neural ODEs

We assume that a trajectory can be obtained as the solution to the initial value problem of a neural ODE [5], potentially incorporating a priori assumptions of the dynamics

$$\begin{aligned} \frac{dx}{dt} &= f_k(x, u, t) + f_\theta(x, u, t) \\ x(0) &= x_0, \end{aligned} \tag{4}$$

where f_k and f_θ are the known and unknown dynamics, respectively. This is particularly useful when a priori dynamics f_k are available, as it improves generalizability beyond estimation data and reduces estimation time. In the specific case that $f_k = 0$, we are left with a vanilla neural ODE

$$x(T) = x_0 + \int_0^T f_\theta(x, u, t)\, dt. \tag{5}$$

Algorithm 2 EstimateNeuralSurrogate

Input: $\mathcal{D} = \{\tau\}_{i=1}^k$
Output: $\hat{\mathcal{S}}$
1: Initialize θ
2: **repeat**
3: $\quad \hat{x}_i(t) \leftarrow \text{ODESolve}(f_h, x_0^{(i)}, u^{(k)})$
4: $\quad L \leftarrow \sum_{i=1}^k \sum_{t=0}^T \ell\left(y_t^{(i)}, x^{(i)}(t; \theta)\right)$
5: $\quad$ Compute $\nabla_\theta L$ by the adjoint state method.
6: $\quad \theta \leftarrow \theta - \eta \nabla_\theta L$
7: **until** convergence

Neural ODEs can be integrated numerically using any standard ODE-solver, including adaptive step-size solvers, making them analogous to continuous-depth generalizations of residual networks [13] without fixed time steps. They locally extrapolate well beyond supervised training, as seen in Figure 2, where trajectories are accurate despite small perturbations in training data.

The unknown dynamics are commonly parameterized as a multilayer perceptron mapping the vector field $f_\theta : \mathbb{R}^{n \times m} \mapsto \mathbb{R}^n$ for an n-dimensional state-space with m input signals. For a solution to exist and be unique according to the Picard-Lindelöf theorem, we must choose activation functions such that

(1) f_θ is Lipschitz in z and
(2) f_θ is continuous in t.

The vanilla neural ODE is a first-order differential equation incapable of modeling higher order systems. Thus, for systems of order greater than one, we lift the state to include derivatives $x^{(1)}, x^{(2)}, \ldots$, forming an augmented state

$$z = \begin{bmatrix} x_1^{(0)} \\ \vdots \\ x_n^{(o_n)} \end{bmatrix}. \tag{6}$$

Here, $x_i^{(j)}$ denotes the j-th time derivative $\frac{d^j x}{dt^j}$ of the i-th state variable, up to order o_i. The augmented state z with outputs y evolves according to

$$\begin{aligned} \dot{z} &= Az + Bf_\theta(z) \\ y &= Cz \\ z(0) &= z_0, \end{aligned} \tag{7}$$

where A is the canonical companion matrix, B is an input mask and C is the output vector. The higher order initial values can either be estimated during training or specified a priori. The neural ODE is then efficiently trained using the adjoint sensitivity method [26] on some data-loss function at the sampled observations y and predictions $\hat{y}$

$$\mathcal{L}(\theta) = \ell(\hat{y}(\theta), y).$$

The adjoint sensitivity method works by first solving the dynamical system forward in time followed by computing the adjoint state $a(t_N) = \frac{d\mathcal{L}}{dx(t_N)}$ at the end of the solution interval. This adjoint state at t_N serves as the initial condition for the adjoint differential equation solved backwards in time. Solving the adjoint differential equation can be done in a single ODESolve, only scaling with the number of function evaluations required by the integrator.

Algorithm 3 DistillSymbolic

Input: $\mathcal{D} = \{\tau\}_{i=1}^{k}$, $\hat{\mathcal{S}}$
Output: $\hat{\mathcal{S}}_{symb}$
1: Generate $\dot{\mathbf{z}}, \mathbf{z}$ from $\hat{\mathcal{S}}$ and $\mathcal{D}$
2: $f^*(\mathbf{z}) \leftarrow \arg\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} (\dot{\mathbf{z}} - f(\mathbf{z}_i))^2$
3: $\hat{\mathcal{S}}_{symb} \leftarrow \arg\min_{f \in f^*} \mathcal{L}(\text{ODESolve}(\mathbf{z_0}, f), \mathbf{z})$
4: **Return:** $\hat{\mathcal{S}}_{symb}$

The MLP parameters θ are then updated iteratively using gradient descent until some convergence criterion is met, as shown in Algorithm 2. We begin training with ADAM and switch to a quasi-Newton method once ADAM satisfies a predefined convergence criterion. The quasi-Newton stage uses an approximate Hessian to accelerate convergence toward a local optimum, offering faster refinement than ADAM alone. To prevent the neural surrogate from bias accumulation towards trajectories collected early, we retrain it from scratch on the entire dataset $\mathcal{D}_k$ in Algorithm 1, for each iteration k.

Neural ODEs can also be trained in parallel, scaling independently of the number of trajectories, as each ODESolve runs independently of the other trajectories. This also enables estimation with multiple shooting [32], by partitioning trajectories into sub-trajectories with initial conditions at some of the observations.

A beneficial side-effect of estimating the dynamics as a neural ODE is that it provides continuous numerical derivatives. This is particularly useful in settings with measurement noise, where finite differencing or dynamic mode decomposition, used in Koopman theory, suffers inaccuracies. We emphasize the seamless ability to incorporate known dynamics and physical laws into the hybrid neural ODE and loss function, respectively, which reduces training time and improves generalization outside of training data.

4.2 Symbolic Distillation

With the neural ODE trained in accordance with Algorithm 2, we now estimate a symbolic surrogate using SR techniques [18, 20]. This is done by sampling new derivative data from the trained surrogate and minimizing the error between the symbolic expression and data. This is particularly useful when the outputs are noisy, as many SR methods are sensitive to noisy and sparsely sampled data. The surrogate dynamics of the neural ODE are continuous and smooth, which allows for smooth derivatives at arbitrary small sampling steps, aiding SR fitting. Sampling derivative data from more areas of the state space than just the training data is of particular importance during the falsification stage, as the dynamics in the SQP solver should be an accurate approximation of the neural surrogate over the entire domain.

Our proposed method is agnostic to the specific SR mechanism of choice, but some options include sparse identification of nonlinear dynamics (SINDy) [4] and genetic algorithms [6].

As a last step in SR distillation, we select the symbolic candidate which yields the lowest trajectory MSE-loss integrated over the horizon in $\mathcal{D}$. This avoids selecting a candidate which is over-fitted to derivatives instead of integration and regularizes against overly complex candidate expressions.

Using SR sparsifies the dynamics encoded by an otherwise dense MLP in the neural ODE into something symbolic. The main benefit of sparsifying the dynamics is to enable the use of sparse nonlinear optimizer (SNOPT) [11] and circumvent numerical gradient approximation in the following optimal control problem.

4.3 Optimal control synthesis

Given a symbolic approximation of the surrogate dynamics, the STL specification can be encoded as a nonlinear program, and SNOPT can then be used to solve the resulting optimal control problem to search for a counterexample. Access to a symbolic expression of the dynamics enables exact analytical gradients of the surrogate, which are both faster and more accurate than numerically approximated gradients. In addition to being computationally cheaper, analytical derivatives avoid the noise and sensitivity that numerical differentiation introduces, particularly in non-smooth objective landscapes.

Minimizing the robustness of a specification is equivalent to maximizing the robustness of its negation, yielding the optimal control formulation for the discretized dynamics

$$\begin{aligned} &\max_{\mathbf{x},\mathbf{u}} \rho^{\neg\varphi}(y_0, y_1, \ldots, y_T) \\ &\text{s.t. } x_0 \text{ fixed} \\ &\quad x_{k+1} = f(x_k, u_k) \\ &\quad y_k = g(x_k, u_k) \\ &\quad u_{min} \le u_k \le u_{max}, \quad \forall k \in [0, \ldots, T] \\ &\quad x_{min} \le x_k \le x_{max}, \quad \forall k \in [0, \ldots, T]. \end{aligned} \tag{8}$$

With a finite set of decision variables $u_0, \ldots, u_T$ and states $x_0, \ldots, x_T$, the optimal control problem is no longer a standard IVP that can make use of arbitrary ODE solvers. Instead, the dynamics must be transcribed using direct collocation methods. Furthermore, the decision variables are fixed to a set of predefined discrete time steps, which prevents the use of adaptive time stepping. Adding the step sizes $\Delta t_i = t_i - t_{i-1}$ as decision variables would also lead to spurious counterexamples, because the solver may exploit larger Δt_i if it falsifies the specification, regardless of whether or not such behavior is consistent with the learned dynamics.

This lack of adaptive time stepping and the absence of intermediate evaluations of f make the optimal control problem sensitive to stiffness in the dynamics. Moreover, the number of constraints required to encode the specification grows poorly with the depth of nested temporal operators.[1] Thus, choosing an appropriate collocation method is crucial.

To make the STL robustness fully differentiable, we must also make smooth approximations of the min and max operators in the quantitative semantics. We do this in accordance to [12, 24], by applying the log-sum-exponential (LSE) approximation:

$$\begin{aligned} \max([a_1, ..., a_m]^T) &\approx \frac{1}{k} \log\left(\sum_{i=1}^{m} e^{k a_i}\right) \\ \min([a_1, ..., a_m]^T) &\approx -\frac{1}{k} \log\left(\sum_{i=1}^{m} e^{-k a_i}\right). \end{aligned} \tag{9}$$

[1] For instance, a nested $\Box_{[0,T]} \Diamond_{[0,T]} \varphi$ scales with $O(T^2)$ compared to φ.

Here, k is a smoothing parameter for which the approximations become exact as $k \to \infty$, for a vector of discrete signal values $a_1, \ldots, a_m$. Such a robustness approximation does not change the non-convex nature of the problem, but local optima can now be solved for with sequential quadratic programming (SQP) solvers, such as SNOPT. The robustness function being asymptotically complete with respect to the specification entails that potential serendipitous solutions - falsifying the surrogate but not the SUT - only can arise from discretization errors, mislearned surrogate or a too-small smoothing coefficient.

5 EXPERIMENTAL SETUP

The implementation of our method was developed in the Julia programming language [3]. We use SymbolicRegression.jl [6] for symbolic regression. The optimal control problem is formulated using Drake [29] and the STL specifications are smoothly approximated and encoded using STLpy [17].

Lastly, we use Breach [8] with max semantics for robustness evaluation. All experiments and computations were carried out on a 13th Gen Intel(R) Core(TM) i7-13700H machine with 32GB memory. The framework is evaluated using the neural network controlled levitating magnet benchmark system [30]. The system consists of a feedback-controlled magnet levitating above a coil in which an electromagnetic field is induced. The inducing current is given from a NARMA-L2 feedback controller tasked with positioning the magnet at a reference value. The most important hyperparameters for our experimental setup are summarized in Table 1.

5.1 Learning

We initialize the data by parameterizing the input signal and sampling it using a uniform-random strategy. For both instances, we use piecewise-constant inputs and adhere to upper and lower bounds $1 \leq Ref \leq 3$ for NN_1 and NN_2, and $1.95 \leq Ref \leq 2.05$ for NN_x, respectively.

To ensure a fair comparison with other tools, no a priori assumptions of the dynamics are made, setting $f_k = 0$. Solving the IVPs is done using Tsit5, which is a 4th-order Runge-Kutta solver.

Table 1: Overview of the hyperparameters used in our falsification experiments.

Learning		Symbolic Regression		Falsification (OCP)	
Learning rate	$5e{-}2$	Complexity	< 30	Smoothing parameter	2
Epochs (ADAM)	300	Iterations	100	Experiment budget	10, 200
Iterations L-BFGS	20	Population size	50	Optimizer	SNOPT
Hidden neurons	16, 8	Unary operators	exp, sin, cos	Collocation Δt	0.2s, 0.1s
Loss function	MSE	Binary operators	$+, -, \times, \div$		
System order	2				

For the neural network controlled levitating magnet, we use a fully connected MLP for second order dynamics with one neuron per initial state and input signal, $f : \mathbb{R}^n \times \mathbb{R} \mapsto \mathbb{R}^n$, with two hidden layers consisting of 16 and 8 hidden neurons, respectively. For activations we use tanh functions in all layers except the output layer, which satisfies the Picard-Lindelöf conditions for uniqueness and existence of a solution to the initial-value problem (4), due to tanh being Lipschitz. No physics loss is incorporated for the benchmarks, implying a pure data-loss from the observations. We use the mean-squared error (MSE) to guide learning:

$$\ell(\theta) = \frac{1}{T} \sum_{t=0}^{T} (\hat{y}_t(\theta) - y_t)^2. \tag{10}$$

For training, a fixed number of epochs is set, instead of relying on early-stopping techniques. This is done to avoid spending experiments on acquiring costly validation data that does not explicitly improve training. The learning rate of 5e−2 was selected based on preliminary evaluations, which showed training instability for higher learning rates and slow convergence for lower values. Similarly, increasing the number of epochs beyond 300 did not display significant improvements, in particular for early iterations, when $\mathcal{D}$ contains only a few trajectories. Lastly, we train in a multiple-shooting fashion, splitting trajectories into segments and training on the batched segments.

5.2 Symbolic Distillation

We use SymbolicRegression.jl [6] for SR, which is an open-source Julia package for discovering analytical expressions of functions, from data. The method encodes the computation tree of a solution as a genetic individual and combines evolutionary with gradient-based optimization to arrive at an optimally fit computation tree. The search space of possible symbolic expressions is limited to have complexities less than 30. Preliminary results showed that both trajectory and derivative losses started to plateau around that value.

To keep the function space $\mathcal{F}$ general, we search for functions broadly, incorporating the unary operators $\{\exp(\cdot), \sin(\cdot), \cos(\cdot)\}$ and binary operators $\{+, -, (\cdot), \div\}$. We strategically omit operations which can yield undefined computations over the domain, such as $\log(\cdot)$ and $\sqrt{\cdot}$. Since our framework cannot guarantee that SNOPT stays in the definition set of f, it risks crashing if such operators are included.

The symbolic distillation step is solved for 100 iterations using 50 populations. The symbolic candidate is selected greedily by minimum trajectory loss. Candidate dynamics that are not driven by inputs, i.e., expressions for which $\frac{\partial f}{\partial u} = 0$ or expressions that are numerically unstable during simulation, are also filtered out.

5.3 Optimal Control Falsification

We set the experiment budget of the SUT to 10 in the parameterization-agnostic case and 200 in the parameterization-compliant case. If a counterexample on the SUT is not found by then, the experiment is considered unsuccessful. Robustness evaluations are done using Breach [8].

As shown in Table 2, the specifications NN and $\mathrm{NN}_{\beta=0.04}$ both have a triple nesting of temporal operators, with the outermost spanning almost the entire time horizon. To reduce the number of equality constraints, we use coarser discretization scheme for said specifications, setting $\Delta t = 0.2$s. Due to its shorter interval definition, NN_x can be solved for a much shorter horizon and less nested temporal operators, which allows for a finer discretization $\Delta t = 0.1$s. All discretizations are made with equidistant steps. A

Table 2: The specifications used in the levitating magnet benchmark problem.

Spec.	STL definition
NN	$\square_{[1,37]}\Big((\lvert Pos - Ref\rvert > 0.005 + 0.03\lvert Ref\rvert) \longrightarrow \Diamond_{[0,2]}\square_{[0,1]}\neg\,(0.005 + 0.03\lvert Ref\rvert \leq \lvert Pos - Ref\,\rvert)\Big)$
$\text{NN}_{\beta=0.04}$	$\square_{[1,37]}\Big((\lvert Pos - Ref\rvert > 0.005 + 0.04\lvert Ref\rvert) \longrightarrow \Diamond_{[0,2]}\square_{[0,1]}\neg\,(0.005 + 0.04\lvert Ref\rvert \leq \lvert Pos - Ref\,\rvert)\Big)$
NNx	$\Diamond_{[0,1]}(Pos > 3.2)$ $\wedge\, \Diamond_{[1,1.5]}(\square_{[0,0.5]}(1.75 < Pos < 2.25))$ $\wedge\, \square_{[2,3]}(1.825 < Pos < 2.175)$ $1.95 \leq Ref \leq 2.05$

coarser step size introduces larger truncation and global errors, in particular if using an explicit Euler discretization scheme.

To counter this, we assume piece-wise linear dynamics f in each interval $t \in [t_k, t_{k+1}]$ with interval length $h_k = t_{k+1} - t_k$. This is direct collocation using the trapezoidal method [14]:

$$f(t) = \dot{\mathbf{z}}(t) \approx f_k + \frac{t - t_k}{h_k}(f_{k+1} - f_k), \tag{11}$$

where $f_k = f(z_k, u_k)$. This produces a quadratic spline interpolation for the state trajectory in the segment:

$$x(t) \approx x_k + f_k \cdot (t - t_k) + \frac{(t - t_k)^2}{2h_k}(f_{k+1} - f_k), \tag{12}$$

which at $t = t_{k+1}$ yields

$$x_{k+1} \approx x_k + \frac{h_k}{2}(f_{k+1} + f_k). \tag{13}$$

This equality constraint is repeatedly applied between each pair of collocation points, encoding the discretized trajectory as a chain of equality constraints.

The local and global truncation errors of the trapezoidal collocation scale with $O(h_k^3)$ and $O(h_k^2)$, respectively. This can be leveraged to make larger steps, effectively reducing the number of decision variables and equality constraints for large specifications, or maintain higher accuracy for specifications that are difficult to falsify.

Furthermore, we warm-start the optimal control problem by setting the initial guess of states and inputs as a solution to the IVP of the surrogate. This is done by sampling P random-distributed parameter tuples $\phi_1, \ldots, \phi_P$, and selecting the one associated with the lowest robustness, normalized by Euclidean distance to the most recent parameters that appear in the training data:

$$\phi_u^{\text{init}} = \arg\min_{\phi \in \{\phi_1, \ldots, \phi_P\}} \frac{\rho^{\varphi}(\hat{\mathcal{S}}(x_0, u_\phi))}{||\phi - \phi_{\mathcal{D}}||_2}. \tag{14}$$

Initializing the optimal control problem with a solution to the surrogate IVP assures that equality constraints are satisfied before running the solver, instead of spending iterations in the solver to satisfy them. It also reduces the risk of initializing the decision variables in poor local optima. To ensure termination of SNOPT, we set a major iterations limit of 2000.

The optimal control problem is solved using STLpy [17] for approximating smooth robustness and pyDrake [29] to interface the SNOPT solver. We set the smoothing parameter to moderate value $k = 2$ for the LSE approximation, to balance gradient smoothness and stiffness.

For fair methodological comparison to the tools available in [15], we separate fluke counterexamples from converged counterexamples in the optimal control problem. A fluke counterexample falsifies the SUT, despite building on an optimal control solution that failed to converge. As this is an artifact rather than a trusted solution, we add the SUT trajectory to $\mathcal{D}$, but do not consider it as a valid counterexample.

6 RESULTS

We apply our method on a subset of the ARCH-COMP 2024 benchmark problems [15] for evaluation, namely all specifications related to the neural-network controlled levitating magnet [30]. The results are compared to other state-of-the-art methods from the competition, namely ARIsTEO, ATheNA, EXAM-Net, ForeSee, FReaK, Moonlight, OD and Ψ-TaLiRo. The performance metrics for these tools are retrieved from the ARCH-COMP 2024 paper and are not re-evaluated or independently confirmed by the authors of this paper. We specifically omit the tools NNFal and FalCAuN, as these do not have any associated performance metrics reported on the levitating magnet specifications.

The ARCH-COMP 2024 benchmark contains two different instances. The instances parameterize the input signal differently but share common specifications. Instance 1 allows arbitrary piecewise-continuous input signals, with segment lengths ≥ 3 time units, to be designed by participants of the competition, whereas instance 2 constrains the input to be parameterized as piecewise-constants with exactly three segments. However, to demonstrate our framework's agnosticism to input signal parameterization, we also let the optimal control problem construct the piecewise-constant input signal freely within upper and lower amplitude bounds, and compare it against both instances as reported in [15].

The benchmark problem we use for evaluation consists of a levitating magnet, controlled by a neural network feedback controller. It is the only benchmark system from the ARCH-COMP 2024 problem set with both multiple STL specifications and no mode switching or mixed discrete-continuous states. It has three different specifications shown in Table 2, all related to setpoint tracking. Specifications NN and $\text{NN}_{\beta=0.04}$ describe a convergence behavior of *Pos* towards *Ref* with some tolerance factor β. NN_x describes, in more detail, the settling process during the first three seconds as a conjunction of three sub-specifications.

Due to the algorithm's stochastic nature, we run the method 10 times for each benchmark problem and specification and report aggregate performance metrics. Performance is evaluated based on three main metrics: average and median number of experiments needed to falsify a given specification, $\bar{S}$ and $\tilde{S}$, as well as the falsification rate (FR). The FR describes how many of the repetitions that successfully find a counterexample within the experiment budget.

6.1 Learning

Training the neural ODE surrogate accurately requires trajectory data that sufficiently cover the relevant regions of the state space. If trajectories are missing from areas where the specification is likely to be violated, both the neural and symbolic approximations in those regions risk being inaccurate. Consequently, the optimal control problem may produce a spurious falsifying trajectory that does not correspond to a true violation of the SUT.

Avoiding overfitting presents an additional challenge when training the neural ODE. Standard approaches rely on cross-validated early stopping by collecting additional data into a validation set. In our setting, however, the number of experiments is directly linked to the algorithm's performance metric, rendering such early stopping strategies potentially counterproductive.

A benefit of neural ODEs is that they approximate a smooth vector field which enables local extrapolation of the learned dynamics into neighboring areas of the covered state space. When spurious counterexamples are found to the falsification problem, they are added to the training data, expanding the part of state space in which the surrogate dynamics are accurate after further learning.

6.2 Symbolic Distillation

The complexity of a symbolic regression candidate is a heuristic measure of how many terms and operators are included. As shown in Figure 3, the SR loss is generally reduced by increasing the complexity of an expression. However, there exists a discrepancy between the minimal derivative fitness of a candidate and the trajectory error for said candidate. In the figure, the ideal candidate from SR would be of complexity 30, but that same candidate is outperformed by a less complex candidate, namely 23, when simulating a full trajectory instead and comparing it with data from $\mathcal{D}$. For symbolic candidate selection in the neural network benchmark, we find that trajectory fitness improves with higher complexities before flattening out around complexity 25. Although higher complexities become costlier to evaluate in the SNOPT solver, we limit the candidate selection strategy to the one described.

Although the distillation step in general introduces additional approximation errors to the surrogate, we find that the distilled symbolic surrogate accurately represents its neural counterpart in the regions relevant to falsification, as shown in Figure 4. Local deviations occur but tend to have minor qualitative impact on the trajectory over the entire time interval.

6.3 Falsification

SNOPT is a local optimizer that seeks to minimize the objective function. In the context of falsification, however, a solution may still be useful even if it is suboptimal with respect to robustness minimization. As a result, solutions that do not fully minimize the objective can still end up falsifying the SUT. In cases where the optimal control problem fails to find a locally optimal solution, the resulting suboptimal trajectory may still falsify both the SUT and the surrogate. We discard such incidental successes for counterexample purposes, as they arise from serendipitous falsifications, rather than from the intended operation of our method. Nonetheless, we add these serendipitous trajectories resulting from performing experiment on the SUT to the training data.

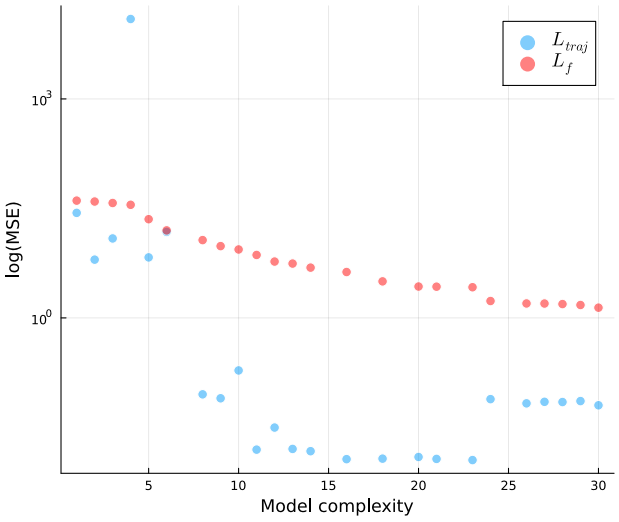


Figure 3: Blue points indicate the MSE trajectory loss L_{traj}, computed by simulating each candidate model along trajectories in $\mathcal{D}$. Red points represent the MSE loss in dynamics. Notably, a candidate model that minimizes the dynamics error in the sampled points does not necessarily achieve minimal error when used in trajectory simulation.

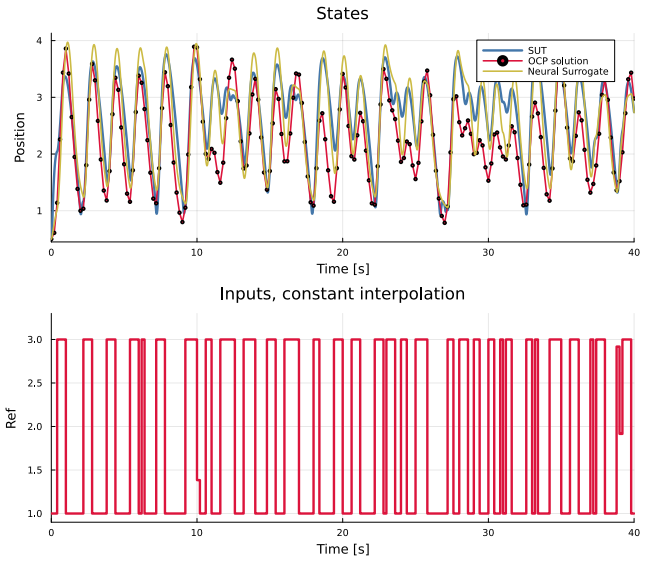


Figure 4: Synthesized counterexample in the agnostic parameterization case, along with the corresponding SUT trajectory, symbolic and neural surrogates. The resulting SUT robustness is -2.014 and the robustness from the optimal control problem is -2.13.

Table 3 shows the resulting aggregate performance metrics over repeated executions, each corresponding to one benchmark problem and specification. We repeat the falsification 10 times on each

specification, to reduce the influence of stochasticity and randomness in the algorithm. In the unconstrained case, the results show that our approach requires roughly as many or fewer SUT experiments compared to other tools to find counterexamples on the selected benchmarks. Across repeated runs, both NN and $\text{NN}_{\beta=0.04}$ are consistently falsified within at most four iterations of the algorithm. When a counterexample is not found from solving the optimal control problem during the first iteration of Algorithm 1, it typically results from SNOPT not producing a solution within the default termination criteria. These failures are likely due to the highly nonlinear and coupled nature of the dynamics and STL constraints. The resulting counterexamples pursue an input signal whose reference value changes rapidly, as illustrated in Figure 5. This behavior is anticipated, since repeatedly shifting the reference position causes the system to violate the requirement that the actual position must settle within two seconds after diverging from the reference, thereby reducing robustness.

In the parameterization-compliant setting, our approach consistently requires significantly fewer or equally many experiments as other tools for instance 2, performing especially well compared to tools that do not model system dynamics. For instance 1, only FReaK requires fewer experiments in the NN-specification whereas consistent falsification within 200 experiments remains difficult for the remaining specifications.

7 CONCLUSIONS

We have presented a novel falsification framework that integrates neural surrogate modeling, symbolic regression and optimal control to synthesize falsifying counterexamples. The method operates by iteratively training a neural ODE to approximate the SUT dynamics and distilling a symbolic representation suitable for optimal control synthesis. Having access to this symbolic approximation of the SUT enables the use of a sparse nonlinear optimizer (SNOPT) to solve an optimal control problem, computing least-robust trajectories. Across three benchmark specifications from the ARCH-COMP 2024 competition, the method is consistently able to find counterexamples to all specifications with roughly equal or fewer experiments than the tools in the competition. While the overall iterative surrogate refinement loop is conceptually similar to recent work [2, 22], our approach differs fundamentally in how the system dynamics are modeled and how the falsifying counterexamples are computed. Modeling the state variables explicitly as a neural ODE enables learning of nonlinear dynamics while supporting the incorporation of a priori knowledge of the dynamics, increasing learning speed and generalization beyond training data. Our framework does not necessitate a parameterization of the input signal, which typically requires expert insight. Instead, it treats the discretized input signal as a finite set of decision variables to be optimized.

Future work includes extending the framework to support hybrid systems, with both switching conditions and mixed discrete-continuous states.

Table 3: Performance comparison between ARCH-COMP 2024 results and our framework. Metrics (aggregated over repeated experiments per specification) are: successful SUT falsifications (FR) out of 10, mean $\bar{S}$, and median $\tilde{S}$ experiments to find a counterexample. For our tool, shown top values correspond to freely constructed input signals (budget 10) while bottom values are for specification-compliant parameterization as described in [15] (budget 200).

	Instance 1			Instance 2		
Framework	FR	$\bar{S}$	$\tilde{S}$	FR	$\bar{S}$	$\tilde{S}$
			NN			
Our Approach[2]	10	3.0	4.0	10	**3.0**	**4.0**
	10	12.1	8.5	10	**4.8**	**4.0**
UR	10	38.6	27.5	10	277.2	158.5
ARIsTEO	5	133.4	108.0	10	119.2	91.0
ATheNA	9	386.0	276.0	10	49.7	47.5
EXAM-Net	10	47.2	47.0	10	47.2	47.0
FORESEE	10	122.8	97.5	10	122.8	97.5
FReaK	10	**2.0**	**2.0**	10	37.2	25.0
Moonlight	10	26.8	22.0	10	48.0	32.0
OD	10	53.0	39.0	10	53.0	39.0
ψ–TaLiRo	10	366.9	236.0	10	75.9	83.0
			$\text{NN}_{\beta=0.04}$			
Our Approach[2]	10	**2.4**	**2.0**	10	**2.4**	**2.0**
	1	37.0	37.0	10	**6.2**	**8.0**
UR						
ARIsTEO	0	-	-	1	55.0	55.0
ATheNA	0	-	-	2	503.0	503.0
EXAM-Net						
FORESEE	6	963.7	1019.0	6	963.7	1019.0
FReaK	10	30.9	33.0	10	534.0	472.0
Moonlight						
OD	4	591.0	690.0	4	591.0	690.0
ψ–TaLiRo	0	-	-	0	-	-
			NN_x			
Our Approach[2]	6	4.8	3.0	6	4.8	3.0
	0	-	-	10	149.7	87.5
UR				8	457.1	380.5
ARIsTEO	0	-	-	7	11.4	6.0
ATheNA	0	-	-	9	182.6	178.0
EXAM-Net	0	-	-	0	-	-
FORESEE						
FReaK	10	**192.3**	**165.5**	10	165.3	**65.5**
Moonlight	9	142.7	128.0	0	-	-
OD	0	-	-	0	-	-
ψ–TaLiRo	0	-	-	10	**143.7**	135.0

[2]Top values show the parameterization-agnostic results while the bottom values show the ARCH-COMP specification-compliant result.

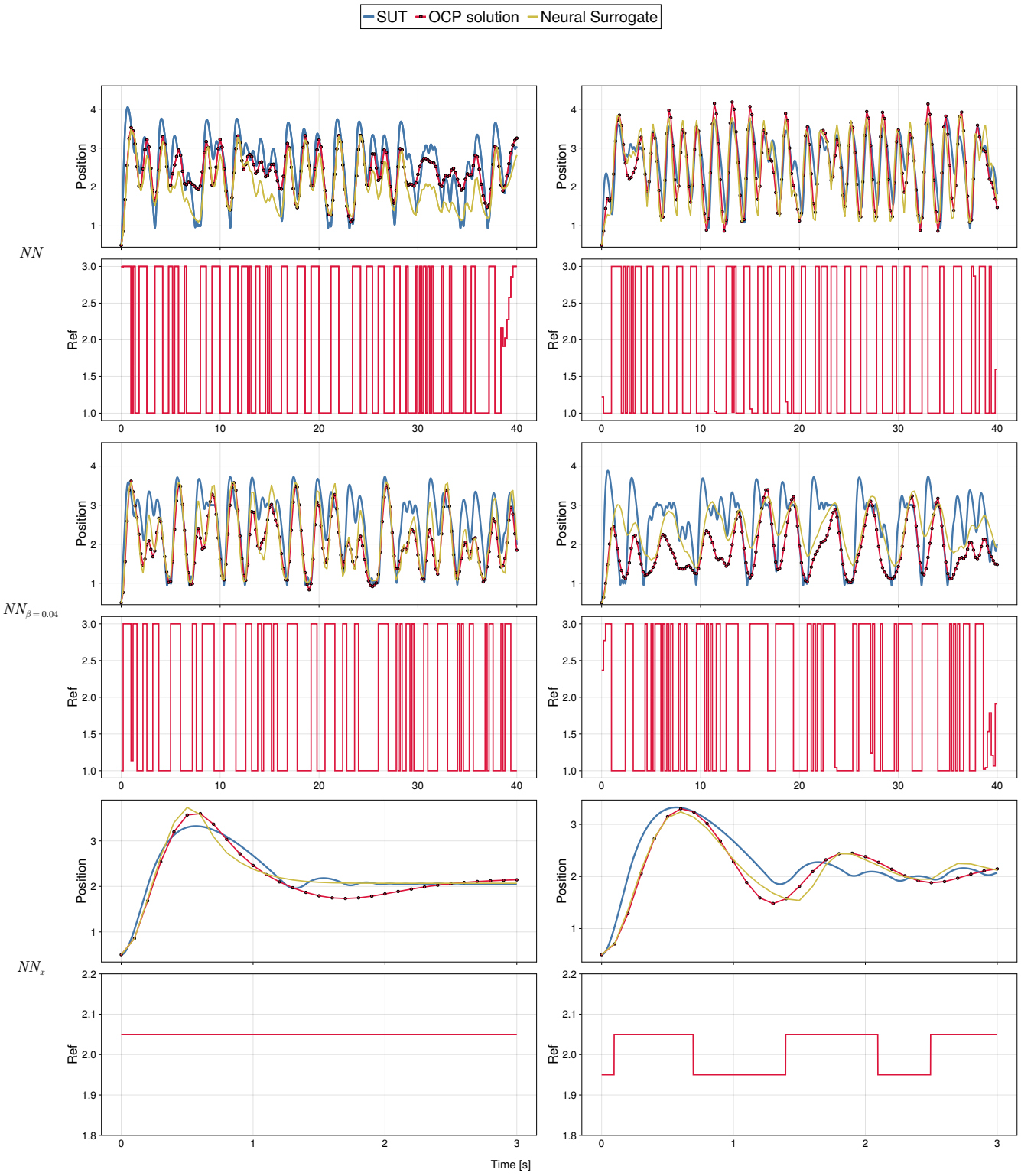


Figure 5: Results from two independent falsification executions per specification, using our framework in a parameterization-agnostic setting. The SUT (blue) is executed using the candidate counterexample from the optimal control problem (red). The neural surrogate (yellow) is plotted for comparison. All results show a falsifying trace except the bottom leftmost. For specifications NN and $NN_{\beta=0.04}$, the violation is achieved by frequently altering the reference value, preventing convergence of the position to the reference.

References

[1] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of model checking*. MIT press.

[2] Stanley Bak, Sergiy Bogomolov, Abdelrahman Hekal, Niklas Kochdumper, Ethan Lew, Andrew Mata, and Amir Rahmati. 2024. Falsification using reachability of surrogate koopman models. In *Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control*. 1–13.

[3] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. 2017. Julia: A fresh approach to numerical computing. *SIAM review* 59, 1 (2017), 65–98.

[4] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. 2016. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences* 113, 15 (2016), 3932–3937.

[5] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. 2018. Neural ordinary differential equations. *Advances in neural information processing systems* 31 (2018).

[6] Miles Cranmer. 2023. Interpretable Machine Learning for Science with PySR and SymbolicRegression.jl. doi:10.48550/arXiv.2305.01582 arXiv:2305.01582 [astro-ph, physics:physics].

[7] Jyotirmoy Deshmukh, Marko Horvat, Xiaoqing Jin, Rupak Majumdar, and Vinayak S Prabhu. 2017. Testing cyber-physical systems through bayesian optimization. *ACM Transactions on Embedded Computing Systems (TECS)* 16, 5s (2017), 1–18.

[8] Alexandre Donzé. 2010. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *International Conference on Computer Aided Verification*. Springer, 167–170.

[9] Jean-Christophe Filliâtre. 2011. Deductive software verification. *International Journal on Software Tools for Technology Transfer* 13, 5 (2011), 397–403.

[10] Federico Formica, Tony Fan, and Claudio Menghi. 2023. Search-based software testing driven by automatically generated and manually defined fitness functions. *ACM Transactions on Software Engineering and Methodology* 33, 2 (2023), 1–37.

[11] Philip E. Gill, Walter Murray, and Michael A. Saunders. 2002. SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Journal on Optimization* 12, 4 (2002), 979–1006.

[12] Yann Gilpin, Vince Kurtz, and Hai Lin. 2020. A smooth robustness measure of signal temporal logic for symbolic control. *IEEE Control Systems Letters* 5, 1 (2020), 241–246.

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

[14] Matthew Kelly. 2017. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM review* 59, 4 (2017), 849–904.

[15] Tanmay Khandait, Federico Formica, Paolo Arcaini, Surdeep Chotaliya, Georgios Fainekos, Abdelrahman Hekal, Atanu Kundu, Ethan Lew, Michele Loreti, Claudio Menghi, et al. 2024. Arch-comp 2024 category report: Falsification. In *International Workshop on Applied Verification for Continuous and Hybrid Systems*. EasyChair, 122–144.

[16] Atanu Kundu, Sauvik Gon, and Rajarshi Ray. 2024. Data-driven falsification of cyber-physical systems. In *Proceedings of the 17th Innovations in Software Engineering Conference*. 1–5.

[17] Vincent Kurtz and Hai Lin. 2022. Mixed-integer programming for signal temporal logic with fewer binary variables. *IEEE Control Systems Letters* 6 (2022), 2635–2640.

[18] William La Cava, Bogdan Burlacu, Marco Virgolin, Michael Kommenda, Patryk Orzechowski, Fabrício Olivetti de França, Ying Jin, and Jason H Moore. 2021. Contemporary symbolic regression methods and their relative performance. *Advances in neural information processing systems* 2021, DB1 (2021), 1.

[19] Karen Leung, Nikos Aréchiga, and Marco Pavone. 2023. Backpropagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods. *The International Journal of Robotics Research* 42, 6 (2023), 356–370.

[20] Nour Makke and Sanjay Chawla. 2024. Interpretable scientific discovery with symbolic regression: a review. *Artificial Intelligence Review* 57, 1 (2024), 2.

[21] Oded Maler and Dejan Nickovic. 2004. Monitoring Temporal Properties of Continuous Signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, Yassine Lakhnech and Sergio Yovine (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 152–166.

[22] Claudio Menghi, Shiva Nejati, Lionel Briand, and Yago Isasi Parache. 2020. Approximation-refinement testing of compute-intensive cyber-physical models: An approach based on system identification. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 372–384.

[23] Laura Nenzi, Ezio Bartocci, Luca Bortolussi, Simone Silvetti, and Michele Loreti. 2023. MoonLight: a lightweight tool for monitoring spatio-temporal properties. *International Journal on Software Tools for Technology Transfer* 25, 4 (2023), 503–517.

[24] Yash Vardhan Pant, Houssam Abbas, and Rahul Mangharam. 2017. Smooth operator: Control using the smooth robustness of temporal logic. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 1235–1240.

[25] Jarkko Peltomäki, Frankie Spencer, and Ivan Porres. 2022. Wasserstein generative adversarial networks for online test generation for cyber physical systems. In *Proceedings of the 15th Workshop on Search-Based Software Testing*. 1–5.

[26] Lev Semenovich Pontryagin. 2018. *Mathematical theory of optimal processes*. Routledge.

[27] Zahra Ramezani, Kenan Šehić, Luigi Nardi, and Knut Åkesson. 2025. Falsification of cyber-physical systems using Bayesian optimization. *ACM Transactions on Embedded Computing Systems* 24, 3 (2025), 1–23.

[28] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. 2015. Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE* 104, 1 (2015), 148–175.

[29] Russ Tedrake and the Drake Development Team. 2019. Drake: Model-based design and verification for robotics. https://drake.mit.edu

[30] The MathWorks, Inc. 2025. Design NARMA-L2 Neural Controller in Simulink. https://www.mathworks.com/help/deeplearning/ug/design-narma-l2-neural-controller-in-simulink.html. Accessed: 2025-11-17.

[31] Quinn Thibeault, Jacob Anderson, Aniruddh Chandratre, Giulia Pedrielli, and Georgios Fainekos. 2021. Psy-taliro: A python toolbox for search-based test generation for cyber-physical systems. In *International Conference on Formal Methods for Industrial Critical Systems*. Springer, 223–231.

[32] Evren Mert Turan and Johannes Jäschke. 2021. Multiple shooting for training neural differential equations on time series. *IEEE Control Systems Letters* 6 (2021), 1897–1902.

[33] Masaki Waga. 2020. Falsification of cyber-physical systems with robustness-guided black-box checking. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*. 1–13.

[34] Zhenya Zhang, Deyun Lyu, Paolo Arcaini, Lei Ma, Ichiro Hasuo, and Jianjun Zhao. 2021. Effective hybrid system falsification using Monte Carlo tree search guided by QB-robustness. In *International Conference on Computer Aided Verification*. Springer, 595–618.