# Statistical Verification of Learning-Based Cyber-Physical Systems

Mojtaba Zarei
mojtaba.zarei@duke.edu
Duke University
Durham, NC, USA

Yu Wang
yu.wang094@duke.edu
Duke University
Durham, NC, USA

Miroslav Pajic
miroslav.pajic@duke.edu
Duke University
Durham, NC, USA

## ABSTRACT

The use of Neural Network (NN)-based controllers has attracted significant attention in recent years. Yet, due to the complexity and non-linearity of such NN-based cyber-physical systems (CPS), existing verification techniques that employ exhaustive state-space search, face significant scalability challenges; this effectively limits their use for analysis of real-world CPS. In this work, we focus on the use of Statistical Model Checking (SMC) for verifying complex NN-controlled CPS. Using an SMC approach based on Clopper-Pearson confidence levels, we verify from samples specifications that are captured by Signal Temporal Logic (STL) formulas. Specifically, we consider three CPS benchmarks with varying levels of plant and controller complexity, as well as the type of considered STL properties – reachability property for a mountain car, safety property for a bipedal robot, and control performance of the closed-loop magnet levitation system. On these benchmarks, we show that SMC methods can be successfully used to provide high-assurance for learning-based CPS.

## CCS CONCEPTS

• **Computer systems organization → Robotic autonomy**; **Embedded and cyber-physical systems**.

## KEYWORDS

Cyber-physical systems verification, high-assurance learning-based control, Statistical Model Checking

**ACM Reference Format:**
Mojtaba Zarei, Yu Wang, and Miroslav Pajic. 2020. Statistical Verification of Learning-Based Cyber-Physical Systems. In *HSCC'20: ACM International Conference on Hybrid Systems: Computation and Control, 2020.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/1122445.1122456


*HSCC'20, 2020,*

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456

## 1 INTRODUCTION

Modern Cyber-Physical Systems (CPSs) are increasingly using Neural Network (NN)-based controllers. Yet, despite the tremendous promise that the use of such controllers would have on performance of CPS, providing assurance for such learning-based systems presents significant challenges. As a result, verifying learning-based CPS has attracted significant attention in recent years (e.g., [6, 7, 10, 19, 20, 23]).

The use of barrier functions is one of the common approaches to obtain safety guarantees for dynamical systems controlled by deep neural networks (DNNs) (e.g., [17, 21]). A disadvantage of this method is the reliance on an accurate system model. Furthermore, it can only deal with simple specifications, like safety and robustness, and does not easily support general temporal logic specifications.

Recent advancements in verification and reachability methods have resulted in new tools for verification of NN-based CPS [9]. For all these methods, in addition to the commonly exhibited scalability problems, there exist additional constraints imposed on the underlying NN structure; these constraints usually depend on the type of specifications supported by the employed verification or reachability tools. For example in [10], NN-controlled CPSs with sigmoid activation functions are verified based on a reachability analysis that is performed for NNs with only differentiable activation functions; this means the Rectified Linear Unit (ReLU), which is a common activation function for most NN-based controllers, cannot be supported. Satisfiability modulo theory (SMT) based methods or mixed-integer linear program (MILP) optimizer approaches, such as [8, 11, 19], commonly transform the considered DNN as an input to the SMT/MILP solvers; to achieve this, the piecewise-linear nature of the ReLUs is used to verify linear properties of the NN's output when the constraints on the inputs are linear. This, on the other hand, means they do not capture nonlinear properties of the considered physical plants.

Consequently, to avoid the aforementioned limitations, such as scalability and nonlinearity, we adopt a Statistical

Model Checking (SMC) approach to verify the desired specifications, formally defined in Signal Temporal Logic (STL) [15]. The main contribution of this works is to show the capability of SMC for verifying complex NN-controlled CPS. Building on an SMC approach based on Clopper-Pearson confidence levels, which we recently introduced in [22], we statistically verify STL specifications on three CPS benchmarks with varying levels of plant and controller complexity (including controllers with several layers containing several hundred of neurons per layer). Specifically, we verified a reachability property for a mountain car, safety property for a bipedal robot, and control performance of a closed-loop magnet levitation system, and showed how SMC can be used to provide high-assurance for learning-based CPS of realistic size. On the other hand, the bipedal robot verification problem is beyond the capability of the state-of-the-art verification tools, such as Verisig [10], due to the 400-neurons layers in the controlling neural network.

Compared to direct testing [23], our SMC approach provides guarantees on statistical accuracy of the results, with provable bounds on their significance levels. Unlike existing SMC methods that rely on sequential probability ratio tests [12, 16, 18], our method is based on Clopper-Pearson bounds, thus requiring no assumption on the "indifferent region".

This paper is organized as follows. After preliminaries in Section 2, Section 3 elaborates on the SMC method. In Section 4, the SMC method is employed on three CPS with learning-based controllers, before concluding in Section 5.

## 2 PRELIMINARIES

We denote the set of natural, rational, real numbers and non-negative real numbers by $\mathbb{N}$, $\mathbb{Q}$, $\mathbb{R}$ and $\mathbb{R}_{\geq 0}$, respectively. For $n \in \mathbb{N}$, let $[n] = \{1, \ldots, n\}$.

An STL formula is defined by

$$\varphi ::= f(\sigma) > 0 \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \mathcal{U}_{[t_1,t_2]} \varphi,$$

where $\sigma$ denotes an $n$-dimensional *signal*, $f : \mathbb{R}^n \to \mathbb{R}$ and $t_1, t_2 \in \mathbb{Q}$ with $t_2 > t_1 \geq 0$. Other temporal operators are defined as $\Diamond_{[t_1,t_2]}\varphi = \texttt{True}\mathcal{U}_{[t_1,t_2]}\varphi$ and $\Box_{[t_1,t_2]}\varphi = \neg(\Diamond_{[t_1,t_2]}\neg\varphi)$, where $\Diamond$ and $\Box$ stand for "finally" and "always", respectively.

The satisfaction of an STL formula on a given signal $\sigma : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is defined by

$$\begin{aligned}
&\sigma \models \mu && \Leftrightarrow f(\sigma(0)) > 0 \\
&\sigma \models \neg\varphi && \Leftrightarrow \sigma \not\models \varphi \\
&\sigma \models \varphi \wedge \psi && \Leftrightarrow \sigma \models \varphi \wedge \sigma \models \psi \\
&\sigma \models \varphi \mathcal{U}_{[t_1,t_2]} \psi && \Leftrightarrow \exists t \in [t_1, t_2] \text{ such that } \sigma^{(t)} \models \psi \\
& && \quad \wedge \forall t' < t, \sigma^{(t')} \models \varphi.
\end{aligned}$$

Here, $\sigma^{(t)}$ denotes the $t$-shift of $\sigma$, defined by $\sigma^{(t)}(t') = \sigma(t + t')$ for any $t' \in \mathbb{R}_{\geq 0}$.

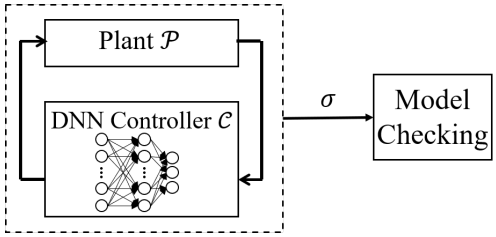



**Figure 1: SMC of NN-controlled CPS; $\sigma$ is the path of the controlled plant.**

## 3 OVERVIEW OF SMC METHOD

As illustrated in Figure 1, we consider a NN-controlled CPS $\mathcal{S} = \mathcal{P}||\mathcal{C}$, similarly to e.g., [10]; the system is derived by the (closed-loop) composition of a plant modeled as a hybrid system $\mathcal{P}$ and a controller $\mathcal{C}$ that is a (trained) neural network. In general, our goal is to check whether a given STL specification $\varphi$ holds on $\mathcal{S}$ for all possible paths/signals $\sigma$ of $\mathcal{S}$. However, when the initial state of $\mathcal{S}$ is uncertain, and the NN-based controller is of realistic size, the problem is currently out of reach of existing verification tools.

Consequently, as previously proposed for analysis of non-learning-based embedded systems [16], we draw the initial states of $\mathcal{S}$ from a probability distribution to model the uncertainty. This allows us to map the considered verification problem into reasoning whether the given STL specification $\varphi$ holds on $\mathcal{S}$ with probability greater than $p$ for a random initial state; this can be specified as

$$p_\varphi = \mathbf{Pr}(\sigma \models \varphi) > p \tag{1}$$

where $\sigma : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is a path/signal of $\mathcal{S}$ from a random initial state and $p \in [0, 1]$ is a probability threshold.

To statistically verify (1), we build on our SMC approach based on the Clopper-Pearson significance levels from [22]. Compared to the SMC methods based on sequential probability ratio test (SPRT), this approach requires no assumption on the indifference margin [5, 13]. For $i \in [N]$, let $\sigma_i$ be a sample path of the system $\mathcal{S}$ from an initial state drawn as an i.i.d. sample. For each $\sigma_i$, with a slight abuse of notation, let

$$\varphi(\sigma_i) = \begin{cases} 1, & \text{if } \varphi \text{ is true on } \sigma_i, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Then $T = \sum_{i \in [N]} \varphi(\sigma_i)$ should obey the binomial distribution $\text{Binom}(n, p_\varphi)$, and the average statistics $T/N$ is an unbiased estimator for $p_\varphi$. Intuitively, when $T/N < p$, it is more likely that $p_\varphi < p$; and the same holds for the other case. Therefore, we define the statistical assertion for problem (1) as

$$\mathcal{A}\big(\mathbf{Pr}(\sigma \models \varphi) < p\big) = \begin{cases} 1, & \text{if } T/N < p \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

**Algorithm 1** SMC of $\mathbf{Pr}(\sigma \models \varphi) < p$ on $\mathcal{S}$.

**Require:** CPS , desired significance level $\alpha_d$, batch size $B$.
1: $N \leftarrow 0, T \leftarrow 0$, initial significance level $\alpha_{\text{CP}} \leftarrow 1$
2: **while** $\alpha_{\text{CP}} > \alpha_d$ **do**
3: **for** $i \in [n]$ **do**
4: Draw $\sigma_{N+1}, \ldots \sigma_{N+B}$ from $S$ in $\mathcal{S}$.
5: $T \leftarrow T + \sum_{i=N+1}^{N+B} \varphi(\sigma_i)$; $N \leftarrow N + B$.
6: **end for**
7: Update $\mathcal{A}$ by (3) and $\alpha_{\text{CP}}$ by (4) and (5).
8: **end while**
9: **return** $\mathcal{A}$ and $\alpha_{\text{CP}}$.

This maps a formula $\mathbf{Pr}(\sigma \models \varphi) < p$ to 0 ("false") or 1 ("true"). From [22], the significance level of $\mathcal{A}$ – i.e., the probability that $\mathcal{A}(\mathbf{Pr}(\sigma \models \varphi) < p)$ disagrees with the truth value of $\mathbf{Pr}(\sigma \models \varphi) < p$, is given by the Clopper-Pearson (CP) bounds as follows

$$\alpha_{\text{CP}}(a, b \mid T, N) = 1 - \begin{cases} (1-a)^N - (1-b)^N, & \text{if } T = 0 \\ b^N - a^N, & \text{if } T = N \\ F_{\text{B}}(b \mid T+1, N-T) - F_{\text{B}}(a \mid T, N-T+1), & \text{else,} \end{cases} \tag{4}$$

where $F_{\text{B}}(\cdot \mid T_1, T_2)$ is the cumulative probability function of the beta distribution with the shape parameters $(T_1, T_2)$, and

$$[a, b] = \begin{cases} [0, p], & \text{if } T/N < p, \\ [p, 1], & \text{if } T/N > p. \end{cases} \tag{5}$$

Compared to the Hoeffding bounds, the CP bounds are tighter, as they are specialized for Bernoulli distributions.

For a desired significance level $\alpha_d > 0$, which captures the upper bound of the probability that the SMC algorithm returns a wrong answer, we design the following *sequential* SMC algorithm. Specifically, at each iteration, we draw $B$ new samples to compute the significance level using (4), until it becomes less than $\alpha_d$. This sequential approach is needed to exactly achieve the desired significance level $\alpha_d$ [5]. The detailed algorithm for verifying (1) on the NN-controlled CPS $\mathcal{S}$ is provided in Algorithm 1. Finally, from [22] the following holds.

Theorem 1 ([22]). *Algorithm 1 terminates with probability* 1 *and gives the correct statistical assertion with probability at least* $1 - \alpha_d$*, when* $p_\varphi \neq p$ *for* (1).

## 4 CASE STUDIES

We statistically verified three NN-controlled CPS models implemented in Simulink on a desktop with 16 GB RAM and Intel Xeon E-2176 CPU; specifically, a mountain car, the bipedal robot [3], and the magnetic levitation [4], differing in model complexity and type of employed controller, with the latter two models being built-in Matlab/Simulink models. All models and SMC source code are available at [1].

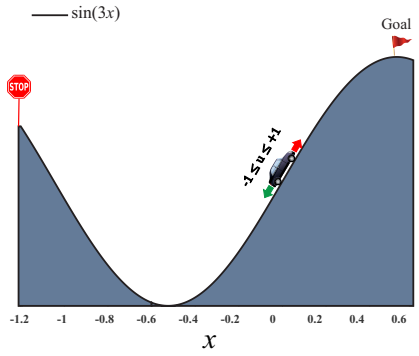



**Figure 2: Mountain Car.**

For each set of considered parameters and properties of interest, we executed SMC Algorithm 1 exactly 100 times; note that the number of samples used in each algorithm execution may vary, based on the model and the considered property. From each of these 100 algorithm runs, the majority vote on whether a property of interest is violated or satisfied captures the final SMC result, while the ratio of those 100 runs that align with the final SMC result is regarded as the accuracy (thus, the accuracy cannot go below 50%). For each case study, we report the accuracy (Acc.), average number of samples (Sam.) used in the SMC analysis, average SMC execution time (Time), and SMC results (Ans.); all these are reported for SMC with different significance levels $\alpha$.

### 4.1 Mountain Car

As illustrated in Figure 2, the dynamics of a mountain car is

$$\dot{x} = v, \qquad \dot{v} = Fu/m - mg\cos(3x) - B_f v$$

where $F = 0.2$, $m = 0.2$, $g = 9.81$, $B_f = 0.5$ denote the time step, force, mass, standard gravity, and friction factor, respectively. In addition, $x(t)$ and $v(t)$ are position and velocity of the car at time $t$ step, and $u(t)$ is the controller input. To meet the constraints at the beginning of simulation, the initial position and velocity were chosen from a normal distribution with mean parameters $\mu_x = 0$, and $\mu_v = 0$ and standard deviation parameters $\sigma_x = 0.1$ and $\sigma_v = 0.05$. Furthermore, the car's position, velocity, and input are constrained within $[-1.2, 0.6]$, $[-1, 1]$, and $[-1, 1]$, respectively.

We trained a controller using an actor-critic learning [14]. The actor's DNN structure contains two fully connected layers; the first has 25 neurons, and the ReLU activation function, and the second one consists of 15 neurons and tanh activation function. During learning, the employed control action reward was $-0.1u(t)^2$, penalizing larger control inputs to avoid a 'bang-bang' strategy. The reward on the

system states was $-0.03\mathcal{E}^2$, where $\mathcal{E} = 0.6 - x(t)$. Furthermore, a reward of 100 was added when the car reached its goal, and the total reward was the summation of the aforementioned rewards. Finally, we set the maximal simulation time to 10 seconds; if the car reaches the top before that, the simulation stops.

We considered the property that the car reaches the top of the right mountain ($x = 0.6$ at Fig. 2) within time $\delta$, when the car's initial position and velocity are selected from Gaussian distributions $\mathbf{N}_x(-0.3, 0.42^2)$ and $\mathbf{N}_v(0, 0.46^2)$, subjecting to the above constraints on the car's position and velocity, respectively; i.e.,

$$\mathbf{Pr}\big(\sigma \models \Diamond_{[0,\delta]}(x > 0.6)\big) > 1 - \varepsilon. \tag{6}$$

To statistically verify (6), we used Algorithm 1 with parameters $\varepsilon \in \{0.7, 0.4, 0.1\}$ and $\delta \in \{4, 7, 10\}$, under the desired significance levels $\alpha \in \{0.01, 0.05\}$; the obtained result are summarized in Table 1. As can be observed, the desired confidence level can be achieved with a relatively small number of samples (at most a few hundred samples for each setup). The estimated SMC accuracy complies with the desired significance levels, except for a small deviation for the boldface entry (which is caused by the fact that only 100 executions of Algorithm 1 were used).

Finally, Figure 3 shows whether STL property $\Diamond_{[0,\delta]}(x > 0.6)$ is satisfied or violated (green/red dots) for different initial states $(x(0), v(0))$ of the plant (i.e., car). The results show that (6) is true if the horizon $\delta = 10$ and false if $\delta = 7$. This means that the exact $\delta$ to make the satisfaction probability be 0.9 is somewhere between $[7, 10]$.

## 4.2 Bipedal Robot

The bipedal robot model [3] emulates human motions by a complex dynamical system of 5-links connected by revolute joints (2-links for each leg, and 1-link for the torso). As shown in Fig. 4a, each of the two identical legs is composed of the hip joint between the torso and thigh, knee joints between the thigh and shank, ankle joint between shank and foot, and a rigid body forms the torso. This joint structure has 5 Degrees of Freedom (DOF) for each leg and 1 DOF for waist or torso. The DOF for the waist is shared between legs. Also, the hip joint has 2-DOF, which allows its motion in the sagittal and the lateral plane. The employed (forward) kinematics for a geometric configuration of the robot determines the position and orientation of a foot with reference to the torso for the known values of the joint variables of the kinematic chain.

To support the SMC analysis and allow us to randomly reinitialize the initial state of the robot, we developed a simplified robot model based on the model's inverse kinematics (IK); the model captures the values of the joint variables, when the position and orientation of the feet are given. The simplified model has 25 joint variables, whose values depend

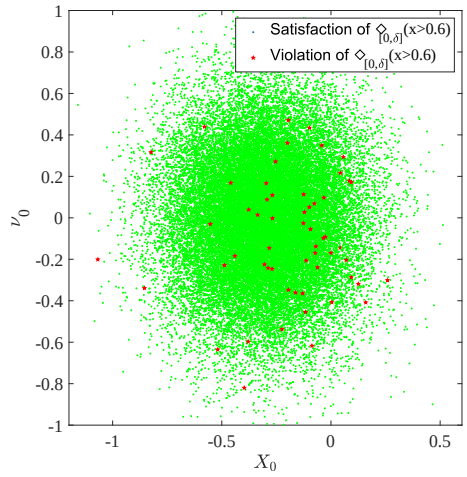



**Figure 3: Initial states of the mountain car resulting in satisfaction/violation of the property $\Diamond_{[0,\delta]}(x > 0.6)$.**

| $\delta$ | $1-\varepsilon$ | $\alpha$ | Acc. | Sam. | Time (s) | Ans. |
|---|---|---|---|---|---|---|
| 4 | 0.3 | 0.01 | 1.00 | 3.4e+01 | 3.8e-02 | True |
| 4 | 0.3 | 0.05 | 1.00 | 2.1e+01 | 4.2e-02 | True |
| 4 | 0.6 | 0.01 | 1.00 | 8.9e+00 | 8.4e-03 | True |
| 4 | 0.6 | 0.05 | 1.00 | 5.4e+00 | 4.6e-03 | True |
| 4 | 0.9 | 0.01 | 1.00 | 3.5e+00 | 2.7e-03 | True |
| 4 | 0.9 | 0.05 | 1.00 | 2.2e+00 | 1.7e-03 | True |
| 7 | 0.3 | 0.01 | **0.98** | 1.2e+02 | 1.5e-01 | False |
| 7 | 0.3 | 0.05 | 0.98 | 5.3e+01 | 6.8e-02 | False |
| 7 | 0.6 | 0.01 | 1.00 | 3.5e+01 | 4.4e-02 | True |
| 7 | 0.6 | 0.05 | 0.99 | 2.1e+01 | 2.6e-02 | True |
| 7 | 0.9 | 0.01 | 1.00 | 5.4e+00 | 5.6e-03 | True |
| 7 | 0.9 | 0.05 | 1.00 | 4.0e+00 | 4.3e-03 | True |
| 10 | 0.3 | 0.01 | 1.00 | 6.6e+00 | 6.5e-03 | False |
| 10 | 0.3 | 0.05 | 1.00 | 4.3e+00 | 3.8e-03 | False |
| 10 | 0.6 | 0.01 | 1.00 | 2.4e+01 | 2.7e-02 | False |
| 10 | 0.6 | 0.05 | 1.00 | 1.6e+01 | 1.8e-02 | False |
| 10 | 0.9 | 0.01 | 0.99 | 1.6e+02 | 2.0e-01 | True |
| 10 | 0.9 | 0.05 | 0.99 | 7.2e+01 | 8.8e-02 | True |

**Table 1: SMC results for the mountain car case study – the property of interest is captured by (6).**

on different randomly sampled initial positions and orientations of the feet. Specifically, as the Simulink model captures the 2D dynamics of the robot, the designed controller only controls the sagittal movement of the robot. Therefore, the initial state of the robot only contains the initial position and velocity in $x$-direction, while the initial position and velocity in $y$-direction are set to zero. Moreover, the heights of the torso $h$ is a constant value $h = 22.66$. Thus, the following dynamics based on the Zero Moment Point (ZMP) and the

Center of Mass (CoM) positions was used

$$\ddot{x}(t) = \frac{g}{h}(x(t) - p_x), \qquad \ddot{y}(t) = \frac{g}{h}(y(t) - p_y).$$

Furthermore, the robot's IK was used to calculate the joint space parameters – i.e., for each leg, it holds that

$$\begin{aligned} x_{foot} &= l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) \\ y_{foot} &= -l_1 \cos(\theta_1) - l_2 \cos(\theta_1 + \theta_2), \end{aligned} \tag{7}$$

From (7), we find the hip pitch ($\theta_1$) and knee pitch ($\theta_2$). The ankle pitch ($\theta_3$) is obtained by $\theta_3 = -(\theta_2 + \theta_1)$. Finally, the initial position and velocity for each legs are chosen from a normal distribution with mean parameters $\mu_{x_{foot}} = -0.05$, and $\mu_{v_{foot}} = -0.05$ and standard deviation parameters $\sigma_{x_{foot}} = 0.08$ and $\sigma_{v_{foot}} = 0.08$. And the feet are symmetrically positioned for stability.

We verified safety of the actor-critic learning-based controller from the Simulink Reinforcement Learning Toolbox [2]. This verification problem is beyond the capability of the state-of-the-art verification tools (e.g., Verisig [10]), due to the 400-neurons layers in the controlling neural network. The actor and critic's DNN structures are depicted in Figs. 4b and 4c, respectively. The actor structure contains three layers NN where the first and second layers are fully connected, and each of them has 400 neurons employing the ReLU activation function. On the other hand, the third layer is a fully-connected layer with six neurons, and its activation function is tanh. The critic is constructed by concatenating two parallel layers – action layer, whose inputs are the control outputs, and observation layer, whose inputs are the system states (i.e., plant outputs). The action space contains 6 torques (on the ankle, knee, and hip for each leg) within $[-3, 3]$ $(N \cdot m)$. The action layer contains two fully connected layers; the first one has 400 neurons employing the ReLU activation function, and the second one has 300 neurons. The concatenated layers are passed through a ReLU activation function to generate the output.

The following conditions were used to capture the safe properties of interest:

- $\eta_1$ : The lateral movement is greater than a threshold ($|y| > 0.5$),
- $\eta_2$ : The robot has fallen ($z < 0.1$),
- $\eta_3$ : The robot's roll, pitch, or yaw are greater than a threshold ($|\phi| > \frac{\pi}{4}$, $|\theta| > \frac{\pi}{4}$, $|\psi| > \frac{\pi}{4}$).

Specifically, we considered whether the robot reaches the desired goal position while the errors in angles and movement direction stay less than the predefined thresholds; the desired specification may be captured as

$$\mathbf{Pr}\Big(\Diamond_{[0,5]}(x > \delta) \wedge \Box_{[0,5]}\big(\neg\eta_1 \wedge \neg\eta_2 \wedge \neg\eta_3\big)\Big) > 1 - \varepsilon. \tag{8}$$

We statistically verified property (8) using Algorithm 1 with parameters $\varepsilon \in \{0.02, 0.12, 0.2\}$ and $\delta \in \{3.0, 2.4\}$, with the desired significance levels $\alpha \in \{0.01, 0.05\}$; the obtained results are summarized in Table 2. As can be seen, while the robot satisfies the safety conditions, its walking distance is greater than $\delta = 2.4$ with probability $1 - \varepsilon = 0.12$, but not greater than $\delta = 3.0$ with the same probability, showing that the 0.12 percentile is between $[2.4, 3.0]$.

| $\delta$ | $1-\varepsilon$ | $\alpha$ | Acc. | Sam. | Time (s) | Ans. |
|---|---|---|---|---|---|---|
| 2.4 | 0.02 | 0.01 | 1.00 | 7.4e+01 | 3.0e-01 | False |
| 2.4 | 0.02 | 0.05 | 0.99 | 4.4e+01 | 1.4e-01 | False |
| 2.4 | 0.12 | 0.01 | 1.00 | 4.2e+01 | 1.2e-01 | True |
| 2.4 | 0.12 | 0.05 | 1.00 | 2.1e+01 | 7.0e-02 | True |
| 2.4 | 0.20 | 0.01 | 1.00 | 1.3e+01 | 4.0e-02 | True |
| 2.4 | 0.20 | 0.05 | 1.00 | 6.7e+00 | 1.4e-02 | True |
| 3.0 | 0.02 | 0.01 | 1.00 | 1.1e+01 | 2.4e-02 | False |
| 3.0 | 0.02 | 0.05 | 1.00 | 6.5e+00 | 1.1e-02 | False |
| 3.0 | 0.12 | 0.01 | 1.00 | 1.4e+02 | 4.3e-01 | False |
| 3.0 | 0.12 | 0.05 | 0.98 | 7.0e+01 | 2.3e-01 | False |
| 3.0 | 0.20 | 0.01 | 1.00 | 1.6e+02 | 5.5e-01 | True |
| 3.0 | 0.20 | 0.05 | 0.98 | 1.0e+02 | 2.9e-01 | True |

**Table 2: SMC results for the bipedal robot case study – the property of interest is captured by (8).**

## 4.3 Magnet Levitation

The Magnet Levitation Simulink model [4] captures the control of the magnet in a power transformer, with dynamics

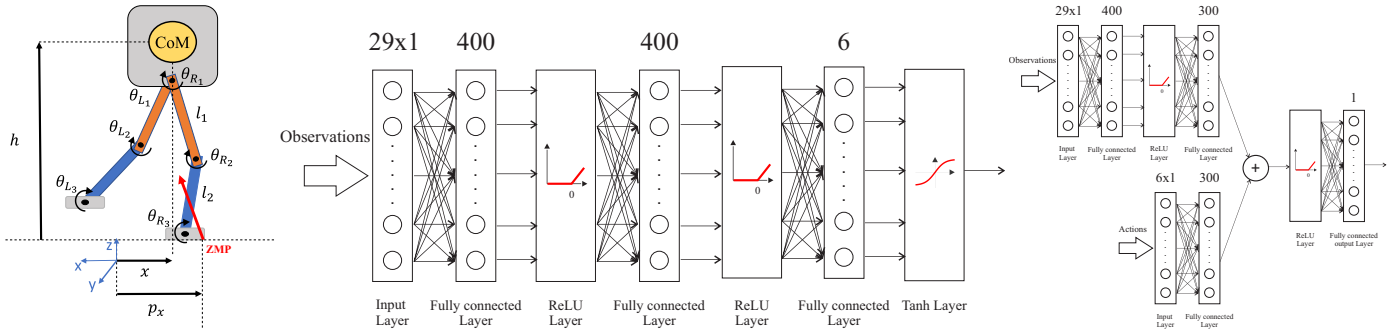
$$\ddot{y}(t) = -g + \frac{\alpha i^2(t)}{My(t)} - \frac{\beta}{M}\dot{y}(t), \tag{9}$$

Here, $y(t)$ is the distance of the magnet above the electromagnet, $i(t)$ is the current flowing in the electromagnet, $M$ is denoted the mass of the magnet, and $g$ is the gravitational constant. The parameter $\beta$ is a viscous friction coefficient, and $\alpha$ is a field strength constant.

The magnet (9) is controlled by the Nonlinear Auto Regressive Moving Average (NARMA-L2) NN-controller shown in Figure 5. The NN-controller has seven hidden layers, with five nodes per layer, as well as three delayed plant inputs and two delayed outputs. The activation functions are all tansig functions. Moreover, the sampling interval is $0.01s$. The training is carried out with 100 epochs with the Levenberg-Marquardt back-propagation method, and the maximum number of samples is set to 10000. Besides, the system's inputs are set within $[0.5,\ 4]$.

The NARMA-L2 NN-controller is evaluated using the Integral Absolute Error (IAE) of the magnet's $y$-position for random reference inputs. Formally, the desired specification for the IAE is expressed as

$$\mathbf{Pr}(\Box_{[0,5]}(IAE < \delta)\big) > 1 - \varepsilon, \tag{10}$$




(a) Robot geometry. (b) The actor's network structure. (c) The critic's network structure.

**Figure 4: The bipedal robot case study – the number above each layer captures the number of neurons in the layer.**

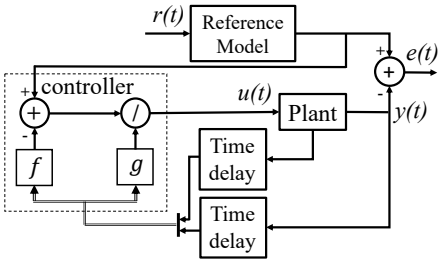



**Figure 5: NARMA-L2 controller for magnetic levitation, with seven hidden layers, each with five neurons.**

and the reference input of the NARMA-L2 NN-controller ($r(t)$ in Figure 5) is set to be a random step function whose amplitude obeys the normal distribution with mean $\mu_y = 2$ and standard deviation $\sigma_y = 0.7$, subjecting to the input constraint $[0.5,\ 4]$. The specification (10) is statistically verified using Algorithm 1, with parameters $\varepsilon \in \{0.5, 0.3, 0.1\}$ and $\delta \in \{1, 1.05, 1.1\}$, under the desired significance levels $\alpha \in \{0.01, 0.05\}$.

Summary of the obtained results is presented in Table 3. As can be observed, within 5 seconds., the IAE index [1] for the closed-loop system stays less than $\delta = 1.05$ with probability $1-\varepsilon = 0.5$, but not less than $\delta = 1.1$ with the same probability, showing that the 0.5 percentile is between $[1, 1.1]$.

| $\delta$ | $1-\varepsilon$ | $\alpha$ | Acc. | Sam. | Time (s) | Ans. |
|---|---|---|---|---|---|---|
| 1.00 | 0.50 | 0.01 | 1.00 | 1.6e+01 | 2.6e+00 | True |
| 1.00 | 0.50 | 0.05 | 1.00 | 1.1e+01 | 1.7e+00 | True |
| 1.00 | 0.70 | 0.01 | 1.00 | 6.6e+00 | 1.3e+00 | True |
| 1.00 | 0.70 | 0.05 | 1.00 | 4.9e+00 | 7.8e-01 | True |
| 1.00 | 0.90 | 0.01 | 1.00 | 3.6e+00 | 6.5e-01 | True |
| 1.00 | 0.90 | 0.05 | 1.00 | 2.5e+00 | 4.4e-01 | True |
| 1.05 | 0.50 | 0.01 | 1.00 | 3.6e+01 | 5.9e+00 | True |
| 1.05 | 0.50 | 0.05 | 0.99 | 1.9e+01 | 3.8e+00 | True |
| 1.05 | 0.70 | 0.01 | 1.00 | 9.7e+00 | 1.8e+00 | True |
| 1.05 | 0.70 | 0.05 | 1.00 | 6.3e+00 | 1.3e+00 | True |
| 1.05 | 0.90 | 0.01 | 1.00 | 4.6e+00 | 8.1e-01 | True |
| 1.05 | 0.90 | 0.05 | 1.00 | 3.1e+00 | 5.6e-01 | True |
| 1.10 | 0.50 | 0.01 | 1.00 | 3.3e+02 | 5.5e+01 | False |
| 1.10 | 0.50 | 0.05 | **0.92** | 1.2e+02 | 2.3e+01 | False |
| 1.10 | 0.70 | 0.01 | 1.00 | 5.6e+01 | 9.6e+00 | True |
| 1.10 | 0.70 | 0.05 | 0.99 | 2.9e+01 | 5.0e+00 | True |
| 1.10 | 0.90 | 0.01 | 1.00 | 8.9e+00 | 1.5e+00 | True |
| 1.10 | 0.90 | 0.05 | 1.00 | 6.6e+00 | 1.1e+00 | True |

**Table 3: SMC results for the magnetic levitation study – the property of interest is captured by** (10).

## 5 CONCLUSIONS

In this work, we have shown feasibility of statistical verification of learning-enabled CPSs for specifications captured using Signal Temporal Logic (STL) formulas. We have addressed the inherent scalability problems of the conventional methods based on the use of reachability analysis or SMT solvers; this is achieved by implementing a Statistical Model Checking (SMC) framework for Neural Networks (NN)-based CPS using Clopper-Pearson confidence levels. On three CPS real-world benchmarks (mountain car, bipedal robot, and magnetic levitation system) with varying levels of plant and controller complexity, as well as different types of considered STL properties, we showed that SMC methods can be used to reason about learning-based CPS of realistic-size.

## ACKNOWLEDGMENTS

This work is sponsored in part by the ONR under agreements N00014-17-1-2504 and N00014-20-1-2745, AFOSR under award number FA9550-19-1-0169, as well as the NSF CNS-1652544 award.

[1] Since IAE is monotone, we checked the values at the end of simulations.

# REFERENCES


[1] [n.d.]. CPS benchmarks. https://gitlab.oit.duke.edu/cpsl/smc_learning_enabled-cps Accessed: 2019-10-23.

[2] [n.d.]. Reinforcement Learning Toolbox (Bipedal Simulink model). https://www.mathworks.com/help/reinforcement-learning/ug/train-biped-robot-to-walk-using-ddpg-agent.html.

[3] R2019a. Bipedal robot. https://www.mathworks.com/help/reinforcement-learning/ug/train-biped-robot-to-walk-using-ddpg-agent.html The MathWorks.

[4] R2019a. Magnet levitation system. https://www.mathworks.com/help/sl3d/examples/magnetic-levitation-model.html The MathWorks.

[5] Gul Agha and Karl Palmskog. 2018. A survey of statistical model checking. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 28, 1 (2018), 1–39.

[6] Tommaso Dreossi, Daniel J Fremont, Shromona Ghosh, Edward Kim, Hadi Ravanbakhsh, Marcell Vazquez-Chanlatte, and Sanjit A Seshia. 2019. Verifai: A toolkit for the formal design and analysis of artificial intelligence-based systems. In *International Conference on Computer Aided Verification*. Springer, 432–442.

[7] Tommaso Dreossi, Shromona Ghosh, Alberto Sangiovanni-Vincentelli, and Sanjit A Seshia. 2017. Systematic testing of convolutional neural networks for autonomous driving. *arXiv preprint arXiv:1708.03309* (2017).

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

[9] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. 2017. Safety verification of deep neural networks. In *International Conference on Computer Aided Verification*. Springer, 3–29.

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

[11] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. 2017. Reluplex: An efficient SMT solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*. Springer, 97–117.

[12] Axel Legay, Benoît Delahaye, and Saddek Bensalem. 2010. Statistical model checking: An overview. In *International conference on runtime verification*. Springer, 122–135.

[13] Axel Legay and Mahesh Viswanathan. 2015. Statistical model checking: challenges and perspectives. *International Journal on Software Tools for Technology Transfer* 17, 4 (2015), 369.

[14] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[15] Oded Maler and Dejan Nickovic. 2004. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 152–166.

[16] Nima Roohi, Yu Wang, Matthew West, Geir E Dullerud, and Mahesh Viswanathan. 2017. Statistical verification of the Toyota powertrain control verification benchmark. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*. ACM, 65–70.

[17] Vicenc Rubies Royo, David Fridovich-Keil, Sylvia Herbert, and Claire J Tomlin. 2018. Classification-based approximate reachability with guarantees applied to safe trajectory tracking. *arXiv preprint arXiv:1803.03237* (2018).

[18] Koushik Sen, Mahesh Viswanathan, and Gul Agha. 2004. Statistical model checking of black-box probabilistic systems. In *International Conference on Computer Aided Verification*. Springer, 202–215.

[19] Xiaowu Sun, Haitham Khedr, and Yasser Shoukry. 2019. Formal verification of neural network controlled autonomous systems. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*. 147–156.

[20] Hoang-Dung Tran, Diago Manzanas Lopez, Patrick Musau, Xiaodong Yang, Luan Viet Nguyen, Weiming Xiang, and Taylor T Johnson. 2019. Star-Based Reachability Analysis of Deep Neural Networks. In *International Symposium on Formal Methods*. Springer, 670–686.

[21] Cumhur Erkan Tuncali, James Kapinski, Hisahiro Ito, and Jyotirmoy V Deshmukh. 2018. Reasoning about safety of learning-enabled components in autonomous cyber-physical systems. In *Proceedings of the 55th Annual Design Automation Conference*. 1–6.

[22] Yu Wang, Mojtaba Zarei, Borzoo Bonakdarpour, and Miroslav Pajic. 2019. Statistical Verification of Hyperproperties for Cyber-Physical Systems. *ACM Transactions on Embedded Computing Systems (TECS)* 18, 5 (2019), 1–23.

[23] Shakiba Yaghoubi and Georgios Fainekos. 2019. Gray-box adversarial testing for control systems with machine learning components. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*. 179–184.