Statistical Hypothesis Testing of Controller Implementations Under Timing Uncertainties

Authors: Bineet Ghosh, Clara Hobbs , Shengjie Xu , Parasara Sridhar Duggirala , James H. Anderson , P. S. Thiagarajan , Samarjit Chakraborty

Publication presentation by Mahdis Rahmani

Context of this research problem

***Abstract*—Software in autonomous systems, owing to performance requirements, is deployed on heterogeneous hardware comprising task specific accelerators, graphical processing units, and multicore processors. But performing timing analysis for safety critical control software tasks with such heterogeneous hardware is becoming increasingly challenging. Consequently, a number of recent papers have addressed the problem of *stability analysis* of feedback control loops in the presence of timing uncertainties (*cf.*, deadline misses).**

Summary: Running multiple jobs on the processor can lead to some deadline misses that Threats system's stability - addressed by cited papers - (and also safety - addressed by this paper)

Safe vs. Stable

"Safe" refers to the property that a system will not violate any specified safety constraints or cause harm to the environment or humans.

On the other hand, "stable" refers to the property that a system will converge to a steady state or oscillate within a bounded range of values.

A system can be safe but not stable, or stable but not safe, or both safe and stable.

	Safe	Unsafe
Stable	**System:** a thermostat that controls the temperature of a room **Safe**: it does not pose any harm to humans or the environment (no risk of collision) **Stable**: it maintains the temperature within a bounded programmed range without oscillating or diverging from the desired value	**System:** a robot arm that is programmed to move a heavy object from one location to another **Unsafe:** arm can collide with obstacles or humans if they enter its workspace, causing harm or damage **Stable:** arm can still collide with obstacles or humans if they enter its workspace, causing harm or damage (the range of movement is bounded)
Unstable	**System:** a thermostat that controls the temperature of a room, however the range of temperature is not bounded (just pad the value based on the sensor input) **Safe**: does not pose an immediate danger to the occupants or the environment **Unstable**: it can diverge from the desired value, if the thermostat is faulty or the temperature sensor is inaccurate	**System:** a drone that is programmed to fly autonomously to a destination **Unsafe:** the drone can collide with obstacles or humans if the sensors or control system fail **Unstable:** drone can oscillate or diverge from the desired trajectory if the control system is not properly tuned or if there are disturbances in the environment

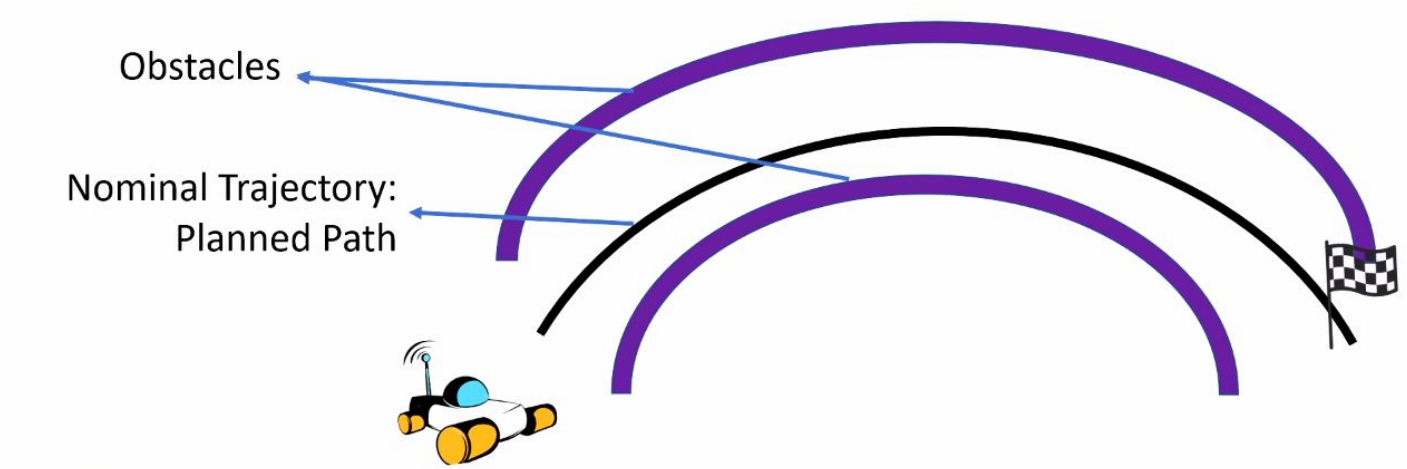
Issue that's being researched

In this paper, we address a different class of safety properties, *viz.*, whether the system trajectory deviates too much from the nominal trajectory, with the latter computed for the *ideal* timing behavior. Verifying such *quantitative* safety properties involves performing a reachability analysis that is computationally intractable, or is too conservative.

Summary: Quantitative safety properties (deviation from the nominal or ideal trajectory where there are no deadline misses) under deadline misses is what is being researched but reachability analysis methods are too conservative or computationally intractable

Quantitative Safety (in our case): Robot Maneuvers

Robot trying to reach its destination, avoiding obstacles.



[This is a slide from a video presentation of this paper by Dr. Ghosh]

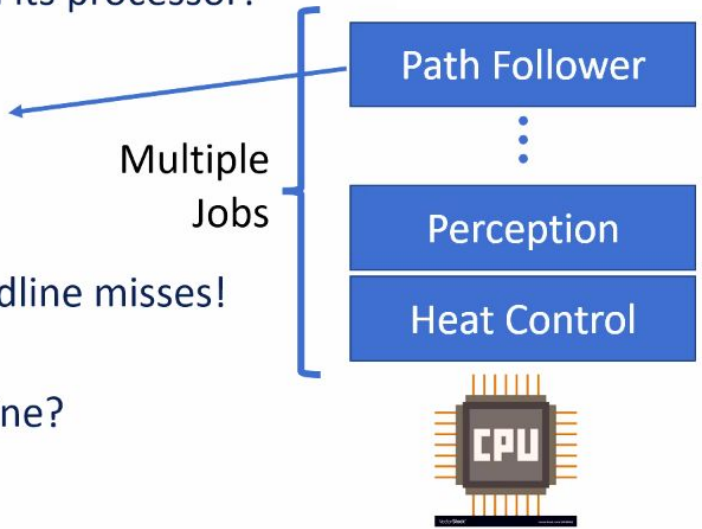
Quantitative Safety: Robot Maneuvers

But: The robot is running multiple jobs on its processor!

Responsible for moving the robot along the planned trajectory.

All jobs cannot always be scheduled—deadline misses!

What if the path follower misses its deadline?

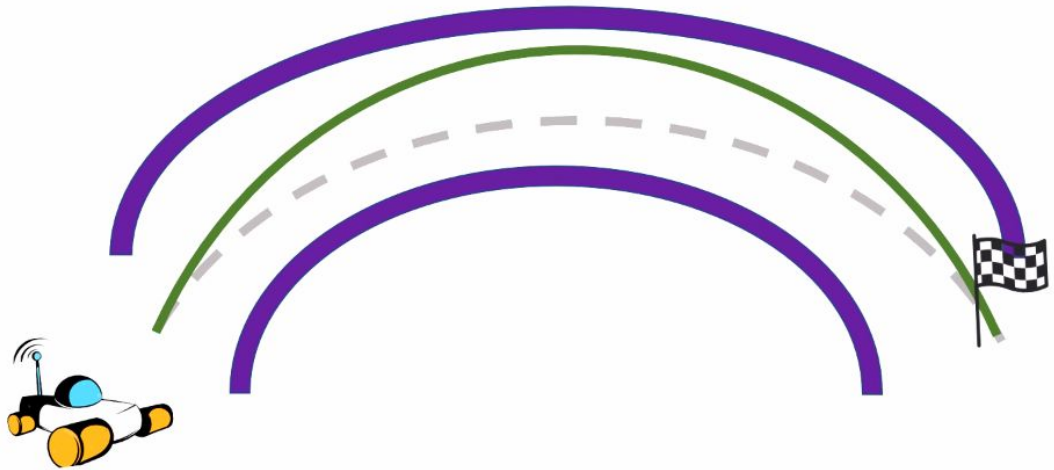


[This is a slide from a video presentation of this paper by Dr. Ghosh]

Quantitative Safety: Robot Maneuvers

What if the path follower misses some deadlines?

The trajectory can deviate from the nominal trajectory!

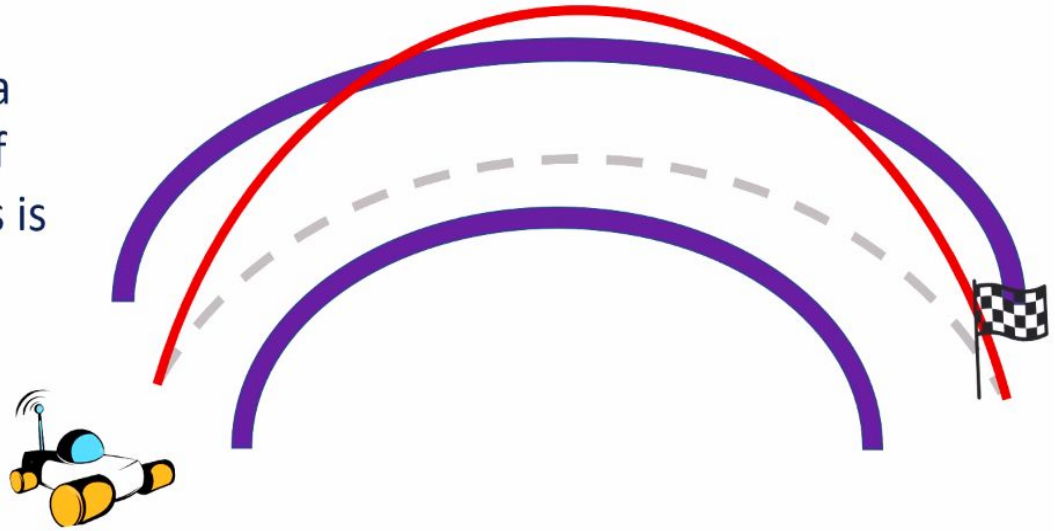


[This is a slide from a video presentation of this paper by Dr. Ghosh]

Quantitative Safety: Robot Maneuvers

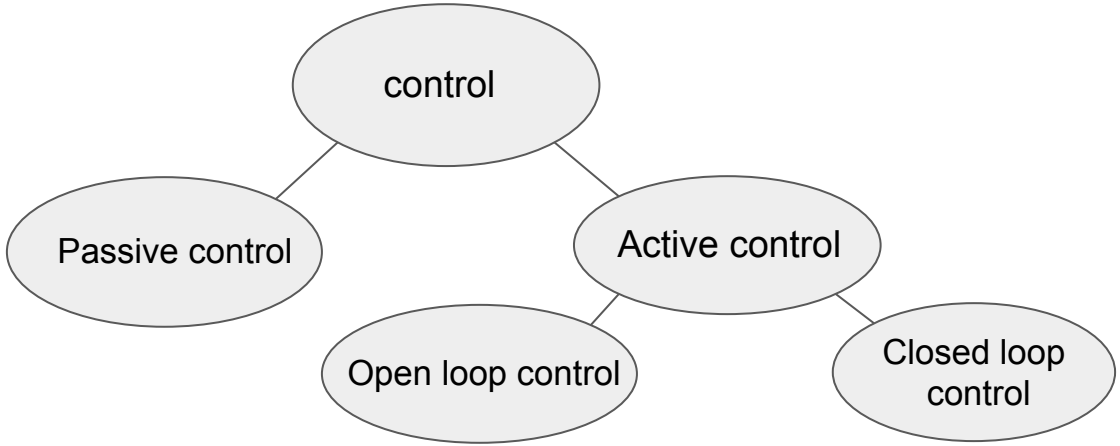
In Conclusion: Not all *patterns* of deadline misses are ***safe***!

Goal: Detect if a given *pattern* of deadline misses is safe!



[This is a slide from a video presentation of this paper by Dr. Ghosh]

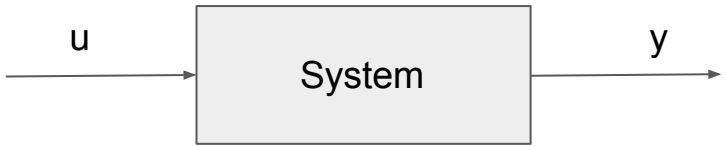
Types of control



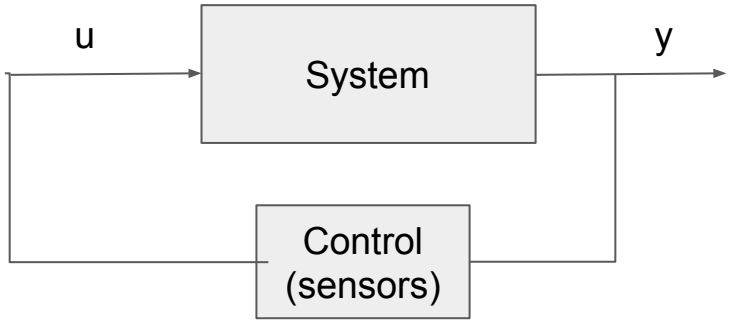
Example: Passive control: streamlined tail sections of 18-wheelers to reduce drag (no energy expenditure)
Active control: (constantly pumping energy into system) - controlling an inverted pendulum

Active control systems

Open loop control system:

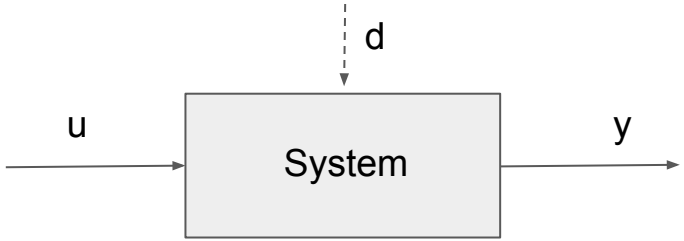


Closed loop control System: (feedback control)

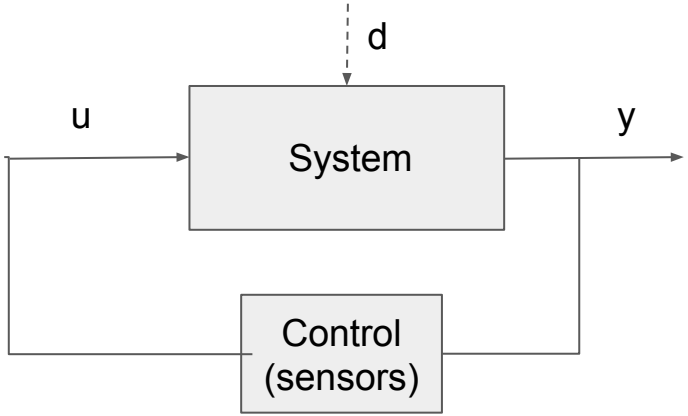


Why feedback?

Open loop control system:



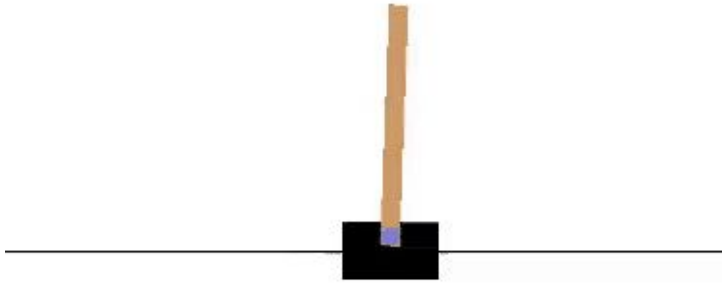
Closed loop control System: (feedback control)



1. Uncertainty about modeling of the system
2. Instability
3. Disturbances (d)
4. Efficiency

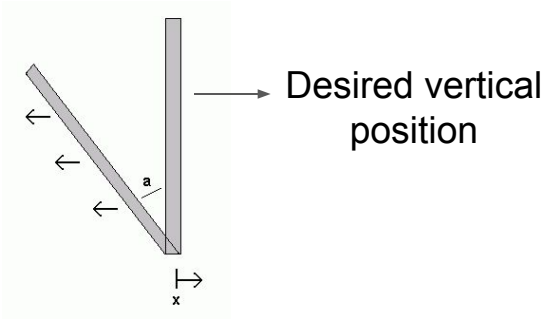
Inverted pendulum example

Problem statement: Holding an inverted pendulum in a vertical position



Open loop approach: pumping the pendulum up and down at a high enough frequency

Closed loop approach: calculating the angle a that the inverted pendulum makes with the vertical axis and pass an input into the system based on the angle a that we receive as the input from sensors (e.g. eyes or cameras)



F1Tenth Example

F1TENTH Autonomous Racing Community

F1TENTH is an open-source evaluation environment for continuous control and reinforcement learning.

186 followers https://f1tenth.org/

Follow

Pinned

f1tenth_gym Public
This is the repository of the F1TENTH Gym environment.
Python 94 68

f1tenth_gym_ros Public
Containerized ROS communication bridge for F1TENTH gym environment.
Python 93 63

f1tenth_system Public
Drivers and system level code for the F1TENTH vehicles
Python 51 54

AutonomousRacing_Literature Public
Forked from JohannesBetz/AutonomousRacing_Literature
Autonomous Literature Overview
TeX 7

f1tenth_racetracks Public
This repository contains maps from over 20 real race tracks (mainly F1 and DTM) downscaled for the usage in the F1TENTH Gym and F1TENTH Simulator.
Python 24 15

vesc Public
Repository for the VESC Controller (ROS1 and ROS2)
C++ 29 47

People

Top languages

Python C++ TeX Dockerfile

Report abuse

Trajectories of a lane-following controller for an F1Tenth model car

Feedback: car's steering angle and velocity is computed

Samples: 100 trajectories where the control task missed it's deadline is shown in the figure

trajectory shown in red deviates too far from the nominal trajectory potentially resulting in a collision with an obstacle near x = 4

This example demonstrates why it's so important to have a guaranteed estimation of the safety envelop under deadline misses

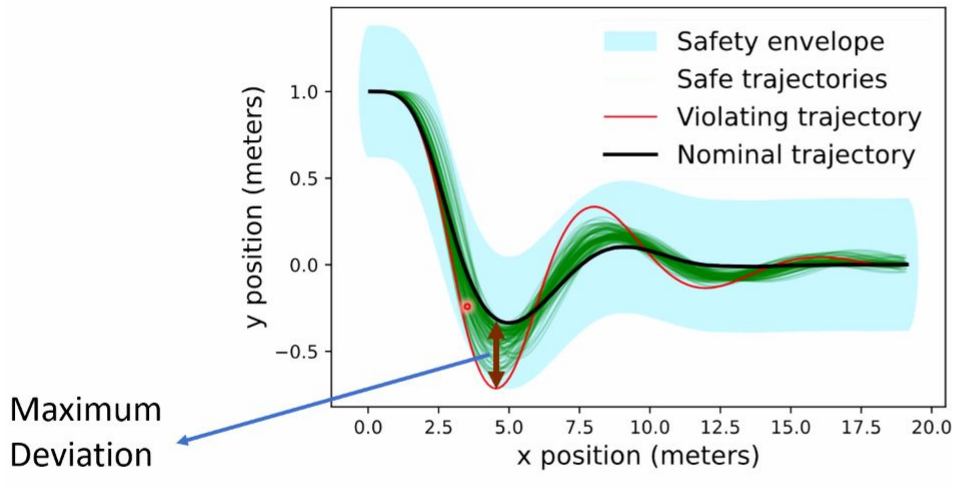


Fig. 1. Deviation in the path of an F1Tenth car due to timing uncertainty.

Problem statement

Given an ideal system behavior (when the control task always meets its deadline), and a pattern of deadline hits and misses (strings of ones and zeros), we want to estimate the maximum possible deviation from the ideal behavior in the presence of the allowed patterns of deadline misses over a time horizon of length H.

Our Setup

- timing behaviors of interest are specified as patterns of hits (the deadline is met) and misses (deadline is not met) and are assumed to constitute a regular language over the alphabet {hit, miss} (or {1, 0}).

- a uniform distribution over these strings of length H (the time horizon of interest). This enables us to implement an efficient sampling method based on the Recursive RGA algorithm.

Our Setup

We are given:

- a set of initial states of the system
- a mathematical model of its (discrete time) dynamics
- a regular language L of strings of length H over the alphabet {hit, miss}

Our goal:

estimate an upper bound dub on the deviation of the trajectory induced by any string in L from the nominal trajectory induced by the string consisting of only hits (the ideal timing behavior)

Computing Deviation: A Naïve Approach

- Given a pattern of deadline misses.
- Compute the maximum deviation up-to a bounded time H.
- **Naïve Approach:** Requires computing deviation of 2^H many trajectories!
- **Instead:** Compute an over-approximation of the maximum deviation.

[This is a slide from a video presentation of this paper by Dr. Ghosh]

Computing Deviation: Other Approaches

- Requires computing ***reachable sets***.

- **Disadvantages:**
  - Computationally slower (generally).
  - The computed bounds on the maximum deviation are not tight (generally).

[This is a slide from a video presentation of this paper by Dr. Ghosh]

Justification and Contributions of this research paper

To alleviate these problems we propose to provide statistical guarantees over behavior of control systems with timing uncertainties. More specifically, we present a Bayesian hypothesis testing method based on Jeffreys's Bayes factor test that estimates deviations from a nominal or ideal behavior. We show that our analysis can provide, with high confidence, tighter estimates of the deviation from nominal behavior than using known reachability based methods. We also illustrate the scalability of our techniques by obtaining bounds in cases where reachability analysis fails to converge, thereby establishing the former's practicality.

Summary: Estimating an upper bound with Bayesian Hypothesis testing method which is tighter and has a high confidence compared to other methods, more practical

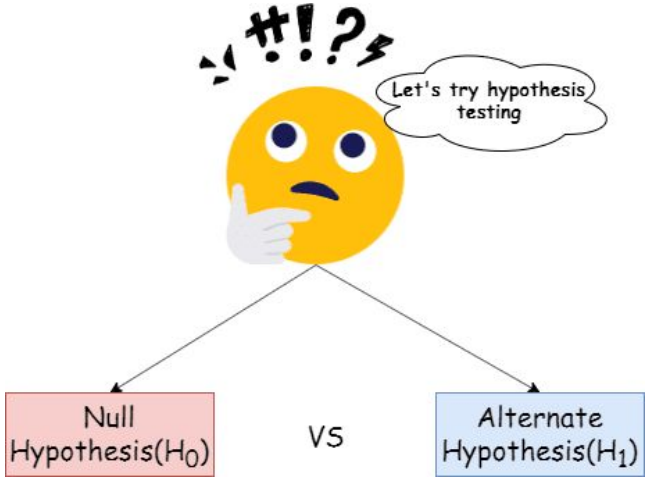
Problem statement

Given an ideal system behavior (when the control task always meets its deadline), and a pattern of deadline hits and misses, we want to estimate the maximum possible deviation from the ideal behavior in the presence of the allowed patterns of deadline misses over a time horizon of length H.

Our contribution is a statistical hypothesis testing (SHT) framework and we use the Jeffreys's Bayes factor test instead of reachability analysis of the states visited by the trajectories of the closed-loop system in the time interval [0, H]

Hypothesis testing in general

Given two competing hypotheses (claims) and some relevant data (samples), Hypothesis testing is a statistical method used to determine if there is enough evidence (by conducting a test on data / samples) to reject or accept a hypothesis about a population parameter.



Hypothesis testing in general

The process of hypothesis testing involves the following steps:

1. Formulating hypotheses: Define the null hypothesis (H0) and the alternative hypothesis (Ha). These two should be mathematical opposites since (Ha) is contradicting (H0).

2. Collecting data

3. Assesing the fitness of the model against the collected data

Hypothesis testing in general

We have a prior belief (mayble considering the results of a recent survey) and based on a few experiments (collecting data and testing the data), we update our belief system.

The main challenge is, HOW SIGNIFICANT should the result of the statistical test be, in order for us to change our prior belief and update it to our posterior belief.

Bayes factor

Proposed SHT framework

Goal: test if a given dub is an upper bound for the maximum deviation

null hypothesis H0: with at most probability c, a randomly chosen trajectory will have a deviation bounded by dub

alternative hypothesis H1: with at least probability c, a randomly chosen run will have a deviation that is bounded by dub.

$$H_0 : \mathcal{P}rob\left[\mathcal{T}, x[0], \tau_{nom}, \mathbf{d}_{ub}\right] < c$$
$$H_1 : \mathcal{P}rob\left[\mathcal{T}, x[0], \tau_{nom}, \mathbf{d}_{ub}\right] \geq c$$

$c \in (0,1)$ → representing the strength (probability) with which the user wishes to assert that dub is an upper bound.

Proceeding the statistical test (verifying the guessed dub)

We're using Jeffreys's Bayes factor test to decide between these two hypotheses

The *Bayes Factor* is the ratio of the above two probabilities:

$$\frac{\mathbf{Pr}\left[\forall \tau \in X : dev(\tau, \tau_{nom}) \leq \mathbf{d}_{ub} \mid H_1\right]}{\mathbf{Pr}\left[\forall \tau \in X : dev(\tau, \tau_{nom}) \leq \mathbf{d}_{ub} \mid H_0\right]}$$

Bayes Factor is the strength of evidence favoring alternative hypothesis over null hypothesis.

We next draw K samples $X = \{\tau_1, \tau_2, \ldots, \tau_K\}$ according to the distribution assumed over the set of executions.

Calculating Bayes Factor

The probability that $\mathbf{d}_{ub}$ is valid, given the null hypothesis, is

$$\mathbf{Pr}\left[\forall \tau \in X : dev(\tau, \tau_{nom}) \leq \mathbf{d}_{ub} \mid H_0\right] = \int_0^c q^K \, dq. \quad (6)$$

Similarly, the probability that $\mathbf{d}_{ub}$ is valid, given the alternative hypothesis, is

$$\mathbf{Pr}\left[\forall \tau \in X : dev(\tau, \tau_{nom}) \leq \mathbf{d}_{ub} \mid H_1\right] = \int_c^1 q^K \, dq. \quad (7)$$

The *Bayes Factor* is the ratio of the above two probabilities:

$$\frac{\mathbf{Pr}\left[\forall \tau \in X : dev(\tau, \tau_{nom}) \leq \mathbf{d}_{ub} \mid H_1\right]}{\mathbf{Pr}\left[\forall \tau \in X : dev(\tau, \tau_{nom}) \leq \mathbf{d}_{ub} \mid H_0\right]} = \frac{1 - c^{K+1}}{c^{K+1}} \quad (8)$$

Why not any other method?

1. In sequential hypothesis testing (-this allows for early stopping-), unlike our method, the number of required samples cannot be fixed.

2. Our method imposes no restriction on the distribution

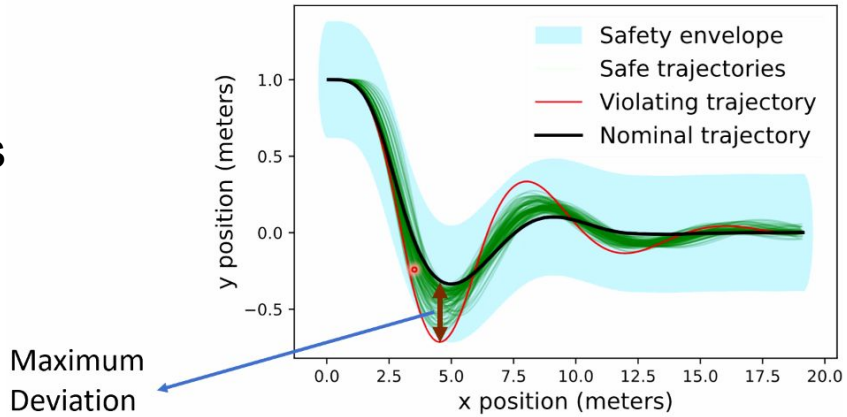
3. Multiple counterexamples can be encountered during random sampling in a round of sequential hypothesis testing, allowing known counterexamples to be explored. In our setting a counterexample represent a violation of a safety property estimate and accordingly Jeffreys's Bayes factor does not allow such known counter examples

An important consequence of our test

When the samples we have drawn do not support the alternative hypothesis, they will contain a counterexample with a deviation that exceeds the current value of dub.

=> We found a trajectory that voilated the dub.

This will lead to the next iteration of hypothesis testing based on a new, larger dub.



Our method: a counterexample guided refinement strategy

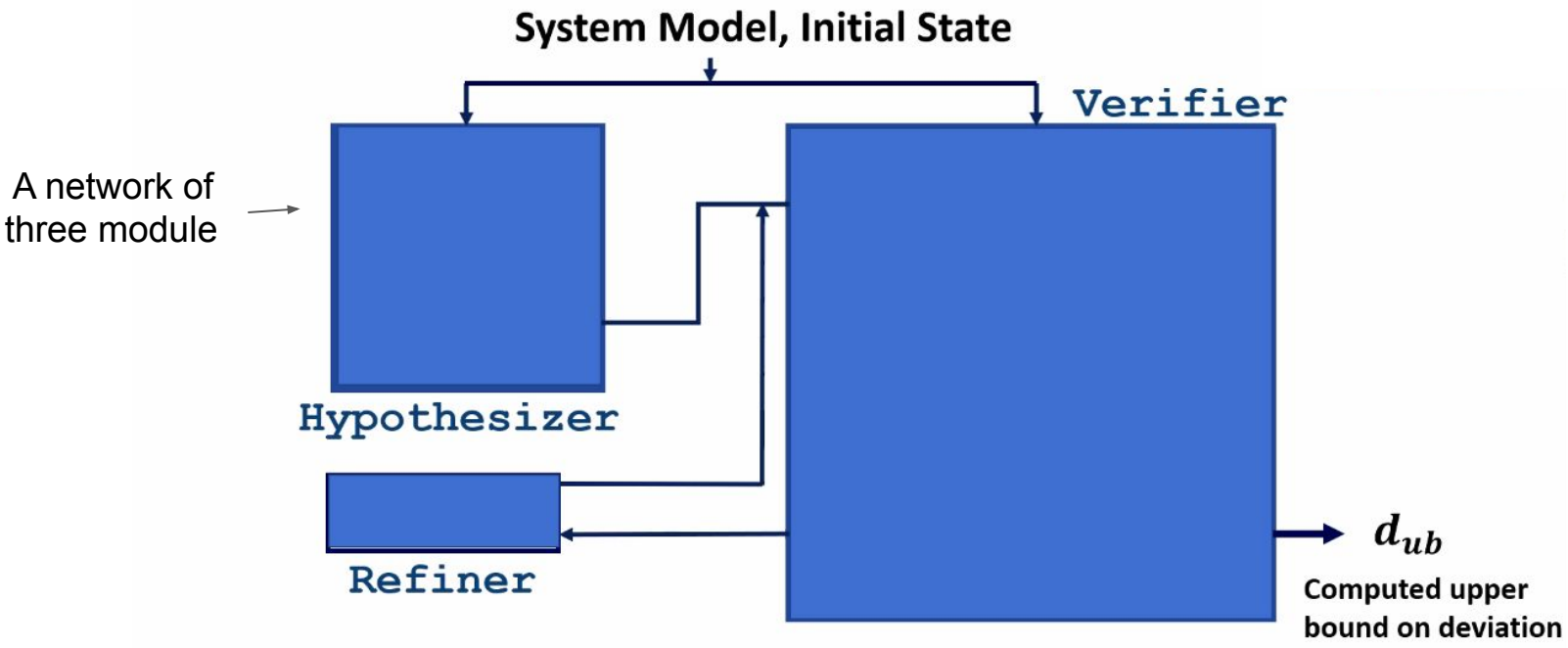
Therefore, we eventually accept the alternative hypothesis.

Perks:

1. the probabilistic guarantee c

2. bounding the type I (False positive) error rate, (i.e., the probability of inferring the alternative hypothesis when in fact the null hypothesis holds and we had to strech dub) while type II errors are not relevant in our setting

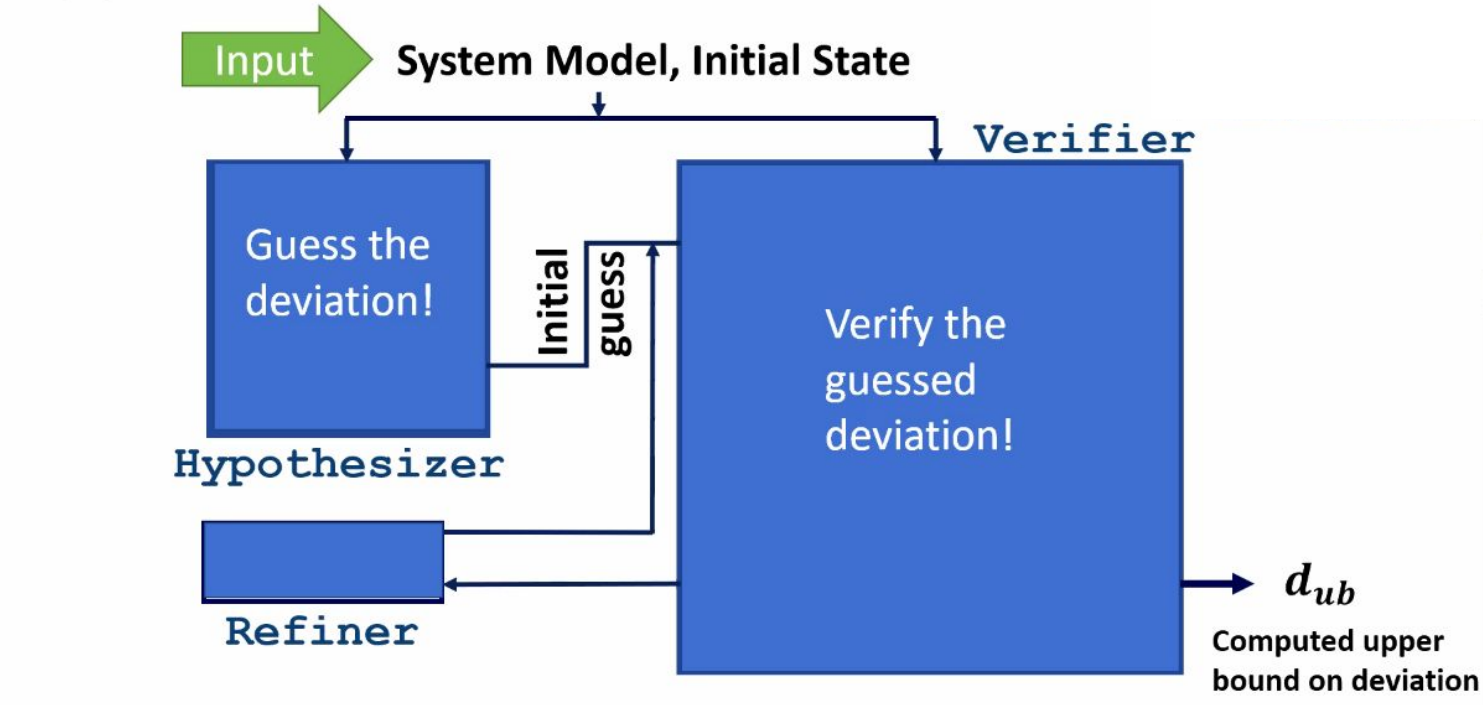
Approach Overview



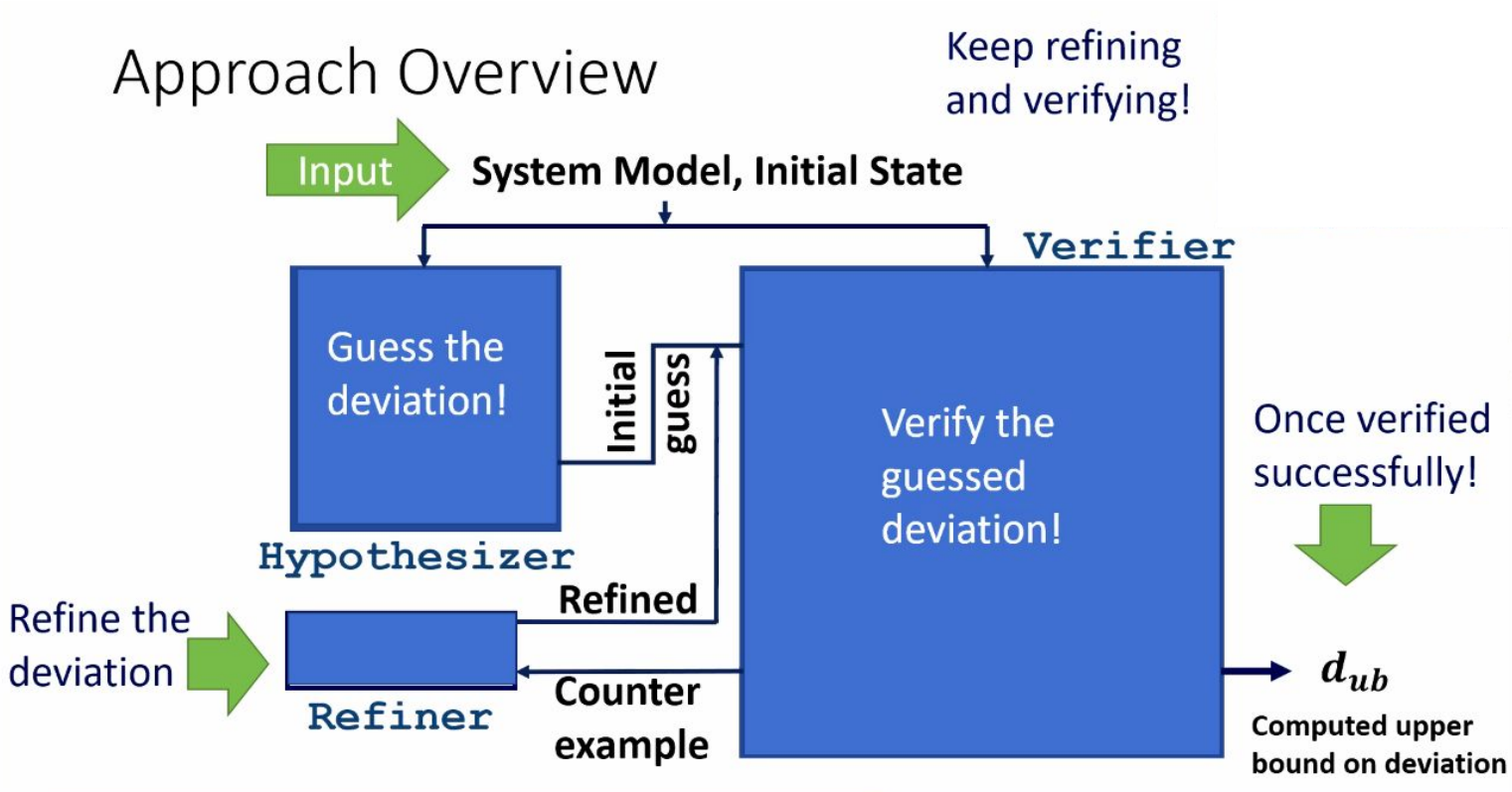
The inputs to our algorithm that estimates dub are a DFA that models the behavior of scheduler, the initial set of plant states x[0], a time horizon H, and the nominal trajectory

[This is a slide from a video presentation of this paper by Dr. Ghosh]

Approach Overview

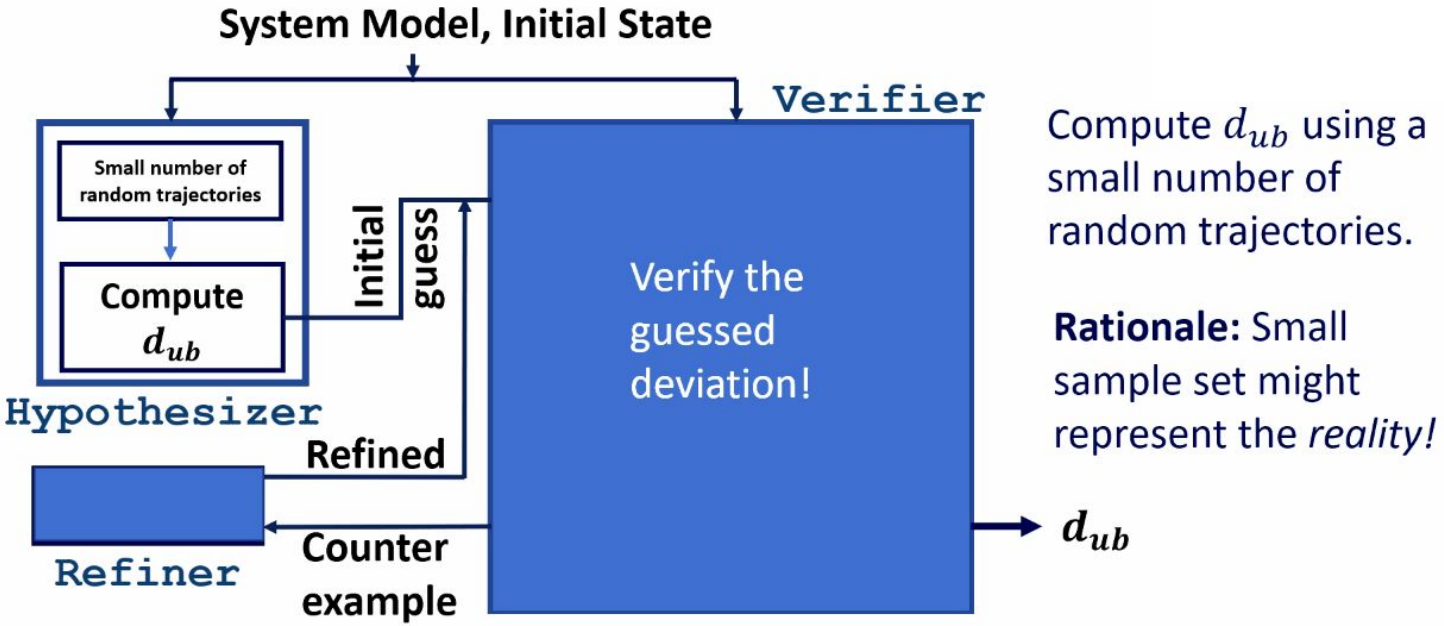


[This is a slide from a video presentation of this paper by Dr. Ghosh]



[This is a slide from a video presentation of this paper by Dr. Ghosh]

Approach Overview: Hypothesizer



Compute d_{ub} using a small number of random trajectories.

Rationale: Small sample set might represent the *reality!*

[This is a slide from a video presentation of this paper by Dr. Ghosh]

System Modeling: State-Space Equation

$$x[t+1] = Ax[t] + Bu[t]$$

$A \in \mathbb{R}^{n \times n}$,

$B \in \mathbb{R}^{n \times p}$,

State vector @ time = t + 1

State vector @ time = t

Feedback Input vector @ time = t

$$u[t] = Kx[t-1]$$

$K \in \mathbb{R}^{p \times n}$

Feedback Input vector @ time = t

State vector @ time = t - 1

$x[t + 1] = Ax[t] + BKx[t-1]$ → Using an augmented state space →

$$z[t] = \begin{bmatrix} x[t]^T & u_a[t-1]^T \end{bmatrix}^T$$

System Modeling: Justifying using the augmented form

$$z[t] = \begin{bmatrix} x[t]^T & u_a[t-1]^T \end{bmatrix}^T$$

Giving the following model:

$$z[t+1] = \begin{bmatrix} A & B \\ K_x & K_u \end{bmatrix} z[t]$$

$K_u \in \mathbb{R}^{p \times p}$

$K_x \in \mathbb{R}^{p \times n}$

providing feedback from each of the vectors x and u

This augmented form permits standard controller design techniques such as linear quadratic regulator (LQR). Further, it allows the plant and controller to be represented as a single dynamics matrix.

a mathematical representation of the behavior of a system

a control algorithm used to optimize the performance of a system, minimizes a cost function

Modeling the system behavior under a sequence of deadline hits and misses

- a sample of the system state at step t − 1 is used to compute the control input at time t.

- A software job is released when x[t − 1] is read, and has its deadline as when x[t] is to be read. If the job completes on time, the control input is computed. If the job misses its deadline, several different actions can be taken, both for generating the missing control input and for handling the task that has missed its deadline

Transducer automaton

We specify the behavior of the scheduler as an automaton that maps the allowed patterns of hits and misses to the accompanying plant dynamics and control inputs.

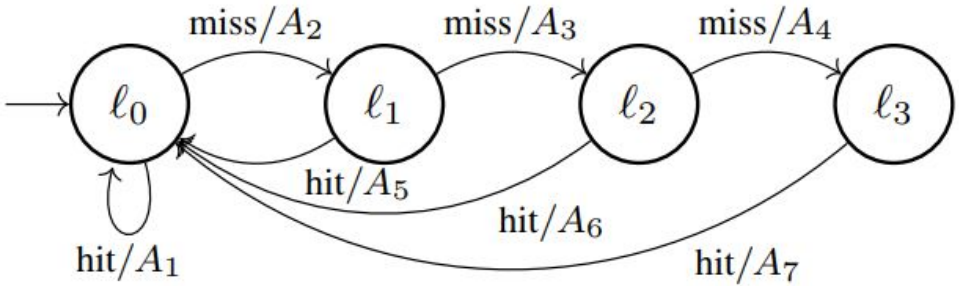
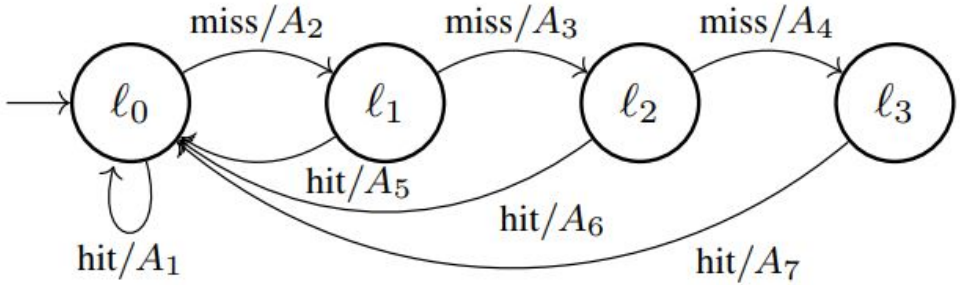


Fig. 3. Transducer automaton capturing 3 maximum consecutive misses.

Transducer automaton



$$\mathcal{T} = \langle L, \mathcal{A}, T, \mu, \ell_{\text{int}} \rangle$$

- $L = \{\ell_1, \ell_2, \ldots, \ell_m\}$: Set of automaton states.
- $\mathcal{A}$: Set of scheduler actions. $\mathcal{A} = \{\text{hit}, \text{miss}\}$
- T: The transition function, where $T : L \times \mathcal{A} \to L$. Let $\mathbb{T} = \left\{ T(\ell_i, a) = \ell_j \;\middle|\; \ell_i, \ell_j \in L, a \in \mathcal{A} \right\}$ denote the set of all transitions of the automaton.
- μ: Associates a dynamics matrix with a transition. Formally, $\mu : \mathbb{T} \to \mathbb{R}^{n \times n}$, where n is the dimension of the system under consideration.
- $\ell_{\text{int}} \in L$: Initial state of the automaton.

Policies for handling deadline misses

Control input:

Zero: a control input of 0 is applied

Hold: the current control input is used again until a new one can be computed

Treat the job that has missed its deadline:

Kill: the job is killed as soon as its deadline is passed

Skip-Next: job is allowed to run to completion past its deadline, but no new job instance may be released until this happens

Combining the strategies

Gives us four strategies:

- Zero & Kill
- Hold & Kill
- Zero & Skip-Next
- Hold & Skip-Next

Sampling Random Trajectories

- Plant states x[t] at time t is subsets of the metric space $(\mathbb{R}^n, dis)$
- Here we use the Euclidean distance for dis

  (This metric applies only to the plant state, not the augmented state vector used by a transducer automaton)

A possible behavior of the system is defined as a *run* consisting of an alternating sequence of locations and actions:

$$\tau = \{\ell_1, a_1, \ell_2, \ldots, a_{H-1}, \ell_H\}$$

where $\ell_1 = \ell_{\text{int}}$, $a_i \in \mathcal{A}$, and H is the time bound. Let the set of all possible runs be $\bar{\tau}$.

Defining the dub

Next, the evolution of a run $\tau = \{\ell_1, a_1, \ell_2, \ldots, a_{H-1}, \ell_H\}$, with an initial set $x[0] \subset \mathbb{R}^n$ is denoted as $evol(\tau)$, given by

$$evol(\tau) = \big\{x_0, x[1] = A_1 x[0], x[2] = A_2 x[1], \ldots, x[H] = A_H x[H-1]\big\}.$$

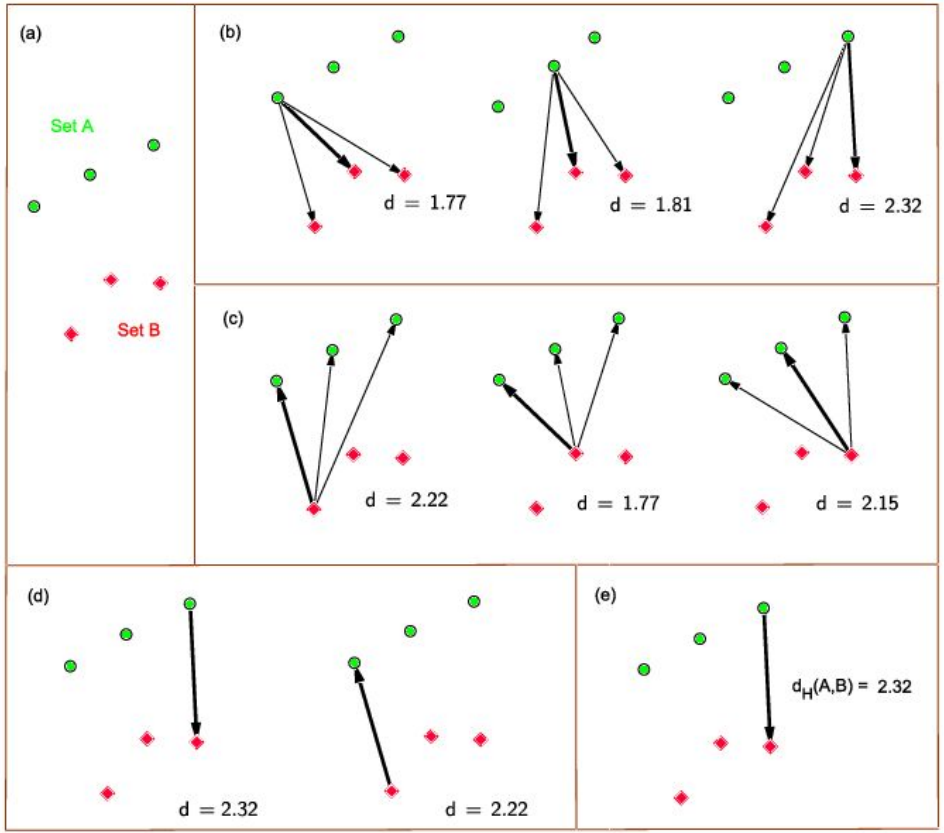
Here, $A_t = \mu(a_t)$ and $x[t]$ are the plant states reached at time step t. Given evolution of a run $evol(\tau)$, let

$$evol(\tau)[t] = x[t], \qquad \text{for } 1 \leq t \leq H.$$

We now define the distance between two sets $S, R \subset \mathbb{R}^n$ using the standard Hausdorff distance, which we denote as

$$\Delta(S, R) = \max\Big\{\sup_{s \in S} \inf_{r \in R} dis(s, r), \ \sup_{r \in R} \inf_{s \in S} dis(s, r)\Big\}.$$

Intuition behind using Huasdorff distance



We now define the distance between two sets $S, R \subset \mathbb{R}^n$ using the standard Hausdorff distance, which we denote as

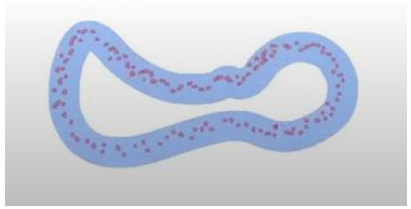
$$\Delta(S,R) = \max\left\{\sup_{s\in S}\inf_{r\in R} dis(s,r),\ \sup_{r\in R}\inf_{s\in S} dis(s,r)\right\}.$$

Intuition behind using Huasdorff distance

inf(How much to blow up A to contain B)

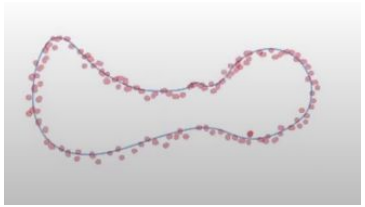
Set A

inf(How much to blow up B to contain A)



Set B

$$\Delta(S,R) = \max\left\{\sup_{s\in S}\inf_{r\in R} dis(s,r),\ \sup_{r\in R}\inf_{s\in S} dis(s,r)\right\}.$$



The intuition behind using this metric

Hausdorff Distance: How similar are two sets?

HAUSDORFF

Given sets $A, B \subseteq \mathbb{R}^n$ and a rational number $t \in \mathbb{Q}$.

$$\vec{d}_H(A, B) := \sup_{a \in A} \inf_{b \in B} \|a - b\|$$

Furthest point $a \in A$ from B.

How much to blow up B to contain A?

Uniformly and in all directions

$$d_H(A, B) := \max\{\vec{d}_H(A, B), \vec{d}_H(B, A)\}$$

Symmetry

Defining dub

Given two runs $\tau_1, \tau_2 \in \bar{\tau}$, we define deviation between the two runs as the maximum Hausdorff distance between the evolution of the two runs. Formally:

Definition 2 (Deviation): The deviation between two runs τ_1 and τ_2 is given by:

$$dev(\tau_1, \tau_2) = \max_{1 \leq t \leq H} \left\{ \Delta\big(evol(\tau_1)[t], evol(\tau_2)[t]\big) \right\} \quad (2)$$

Finally, as mentioned in the introduction, we assume a probability distribution $\mathcal{D}$ over the set of runs $\bar{\tau}$. Accordingly, by a random run we shall mean a run drawn from $\bar{\tau}$ according to $\mathcal{D}$. In the present setting, $\mathcal{D}$ is the uniform distribution. However, our analysis method is applicable to any distribution $\mathcal{D}$, provided one can effectively draw samples from $\mathcal{D}$.

Defining dub

Problem 1: Given a transducer automaton $\mathcal{T}$, an initial set of plant states $x[0] \subset \mathbb{R}^n$, and a nominal run $\tau_{nom} \in \bar{\tau}$, compute the maximum deviation $\mathbf{d}_{max}$, where:

$$\mathbf{d}_{max} = \max\{ dev(\tau, \tau_{nom}) \mid \tau \in \bar{\tau} \}. \quad (3)$$

Two deterministic approaches:

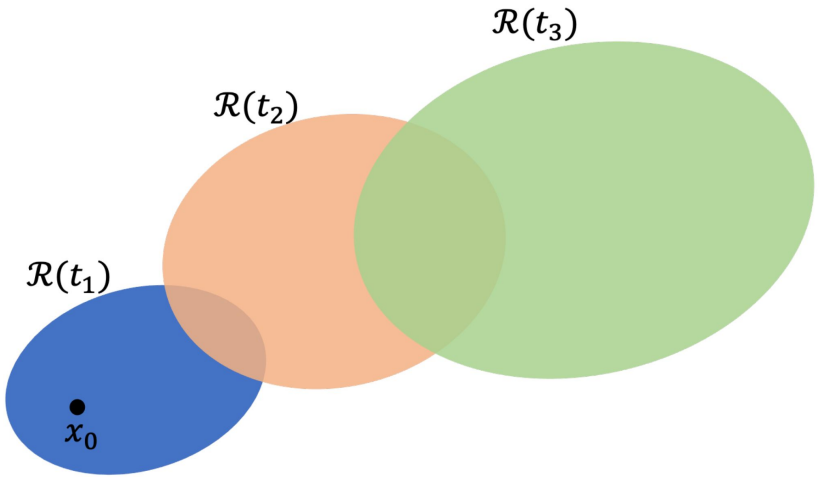
1. Our method: SHT
2. RS (Reachable Set)

RS method:

1. Fixing a small number of time steps m
2. Given x[0] as an initial set, the algorithm proceeds iteratively, computing the reachable sets for each successive span of m sampling periods
3. All trajectories of length m starting from the corners of the initial set x[0] are computed. We store the minimum bounding box of all such trajectories at each time step, yielding our first m over-approximated reachable sets.
4. At the end of each iteration, we group the runs by their final locations in the automaton, and compute a bounding box for each location

RS method:

5. Using these boxes and their corresponding locations as initial conditions, we compute the over-approximated reachable sets for the following m time steps. This procedure is iterated as many times as required to span the time horizon H (i.e., ⌈H/m⌉ iterations)



Our method: SHT

We have a network of 3 modules:

A. **`Hypothesizer`***: Guessing an upper-bound on deviation*

To guess an initial upper bound, we observed that a small set of randomly chosen samples can sometimes provide a reasonably good representation of the actual distribution. Thus our initial guess consists of the following steps, with R and ϵ being parameters supplied by the user:

1) Let $\mathcal{S} = \{\tau_1, \tau_2, \ldots, \tau_R\}$ be a set of randomly generated runs sampled according to the given distribution over the set of strings of length H specified by the DFA.
2) Let $\mathbf{d}'_{ub} = \max_{\tau \in \mathcal{S}} \{dev(\tau, \tau_{nom})\}$.
3) Return $\mathbf{d}_{ub} = \mathbf{d}'_{ub} + \epsilon$.

We will refer to this heuristic as $\texttt{SmallSample}(\cdot)_{[R,\epsilon]}$.

Verifier and Refiner:

Algorithm 1: Computing upper bound on the deviation as defined in Eq. (3)

input : A transducer automaton $\mathcal{T}$, initial set $x[0]$, nominal run τ_{nom}, time bound H

output: Compute an upper bound $\mathbf{d}_{ub}$ for $\mathbf{d}_{max}$

/* we assume parameters R, ϵ, B and c are provided by the user. */

1 $\mathbf{d}_{ub} \leftarrow \texttt{SmallSample}(\mathcal{T}, x[0], \tau_{nom})_{[R,\epsilon]}$; // initial guess

2 $H_0 \leftarrow \mathcal{P}rob\left[\mathcal{T}, x[0], \tau_{nom}, \mathbf{d}_{ub}\right] < c$; // form H_0 using Eq. (4)

3 $H_1 \leftarrow \mathcal{P}rob\left[\mathcal{T}, x[0], \tau_{nom}, \mathbf{d}_{ub}\right] \geq c$; // form H_1 using Eq. (5)

4 $res \leftarrow \texttt{Verifier}_B(H_0, H_1)$; // perform statistical verification

5 **if** res = `True` **then**

6 **return** $\mathbf{d}_{ub}$;

7 **while** `True` **do**

8 $\mathbf{d}_{ub} \leftarrow \texttt{Refiner}(res)_{\epsilon}$; // refine using the counter example

9 $H_0 \leftarrow \mathcal{P}rob\left[\mathcal{T}, x[0], \tau_{nom}, \mathbf{d}_{ub}\right] < c$; // refine H_0 with new $\mathbf{d}_{ub}$

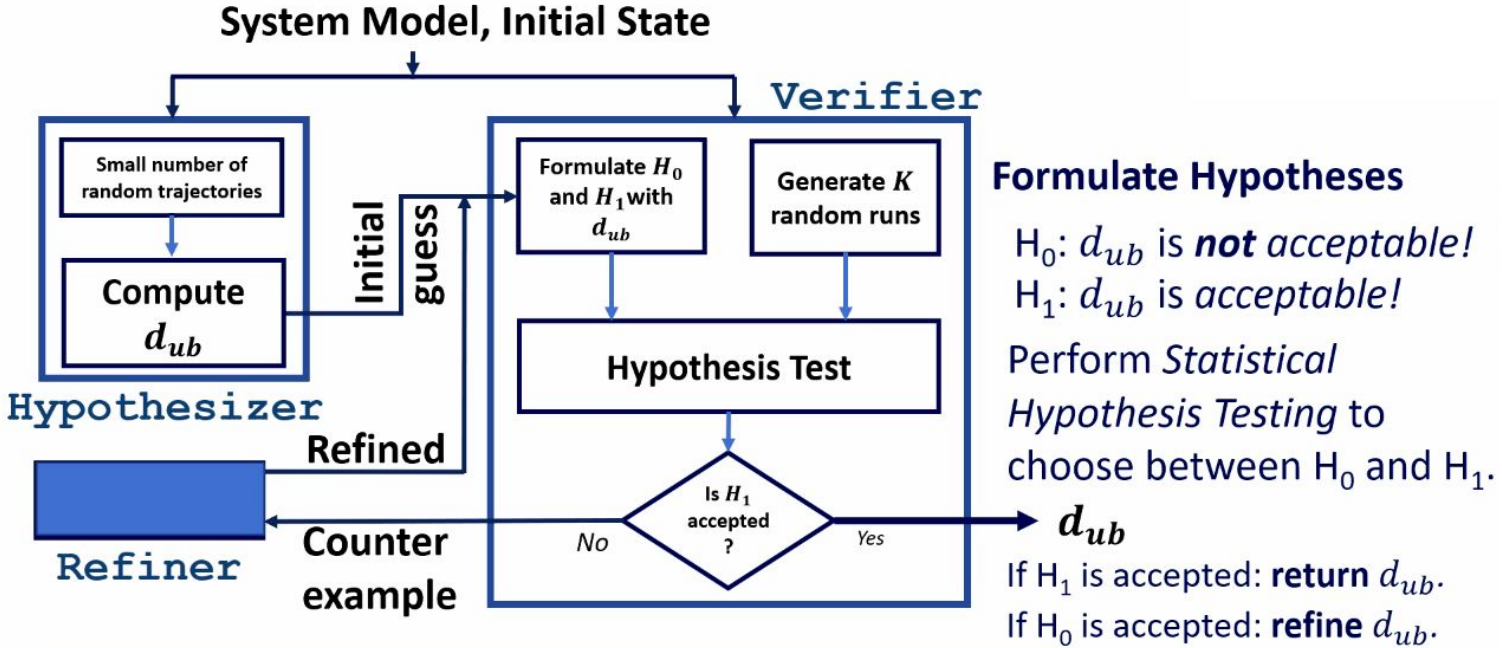
10 $H_1 \leftarrow \mathcal{P}rob\left[\mathcal{T}, x[0], \tau_{nom}, \mathbf{d}_{ub}\right] \geq c$; // refine H_1 with new $\mathbf{d}_{ub}$

11 $res \leftarrow \texttt{Verifier}_B(H_0, H_1)$; // re-perform statistical verification

12 **if** res = `True` **then**

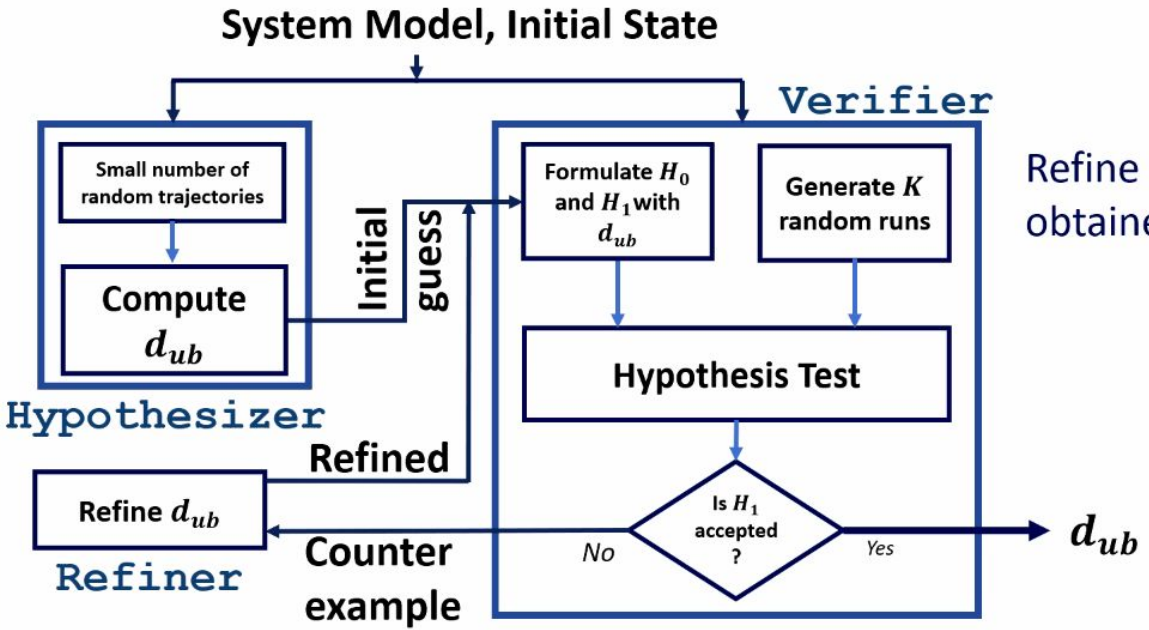
13 **return** $\mathbf{d}_{ub}$;

Approach Overview: Verifier



[This is a slide from a video presentation of this paper by Dr. Ghosh]

Approach Overview: Refiner



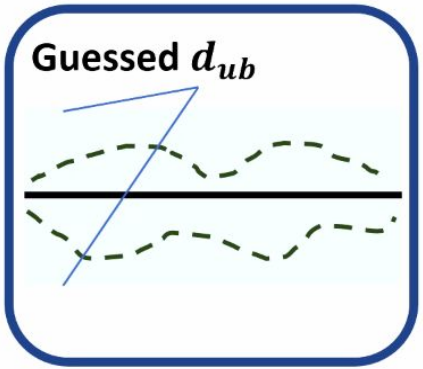
Refine d_{ub} using the obtained counter example.

Pads the value of dub by a slack with respect to the counterexample.

This will initiate the next round of hypothesis testing

[This is a slide from a video presentation of this paper by Dr. Ghosh]

Approach Overview: Steps



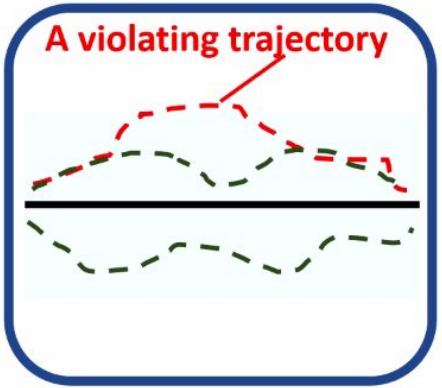
Step 1: Guess the deviation bound

Hypothesizer: *Generate few random trajectories and compute the maximum deviation.*

Black: Nominal Trajectory.
Green: Random Trajectories.
Light Blue: d_{ub}(Guessed deviation bound)

[This is a slide from a video presentation of this paper by Dr. Ghosh]

Approach Overview: Steps



Step 2: Statistically verify the guessed bound

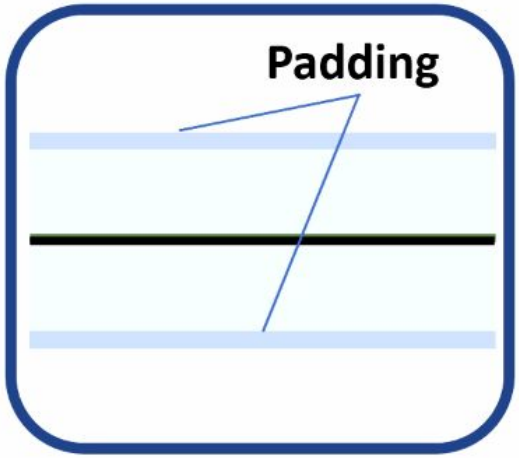
Verifier: *Verify d_{ub} by generating K random trajectories.*

K is computed using Jefferey's Bayes Factor based method.

If a *violating trajectory* is found (counter example), use it to refine d_{ub} (and re-verify)!

[This is a slide from a video presentation of this paper by Dr. Ghosh]

Approach Overview: Steps

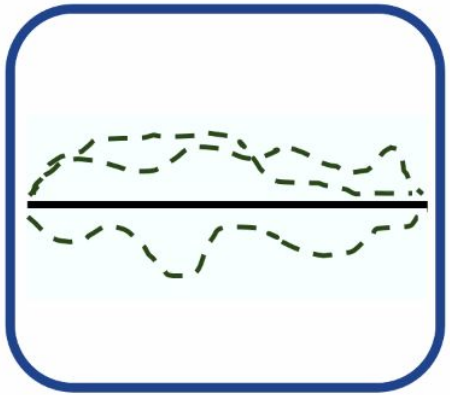


Step 3: Refine the guessed bound

Refiner: *Pads the deviation bound obtained from the counterexample with slack ϵ.*

[This is a slide from a video presentation of this paper by Dr. Ghosh]

Approach Overview: Steps



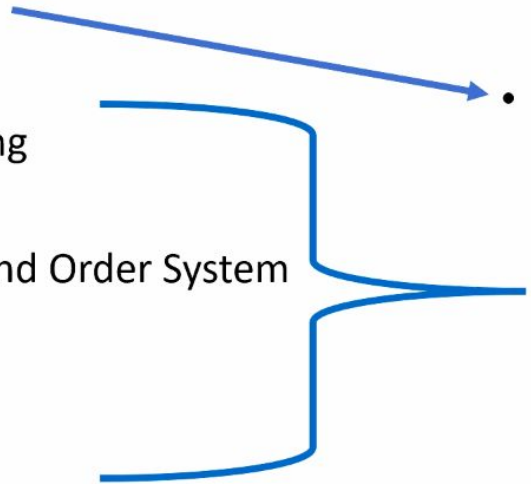
Step 4: Statistically re-verify the guessed bound

Step 5: Return the accepted bound

[This is a slide from a video presentation of this paper by Dr. Ghosh]

Case Studies: Comparison with Benchmark Approaches

- RC Network
- Electric Steering
- Unstable Second Order System
- F1 Tenth



- Comparable upper bounds, and computation time.
  - Computed tighter bounds on the deviation.
  - In less computation time.

[This is a slide from a video presentation of this paper by Dr. Ghosh]

Comparing results

TABLE I
RESULTS COMPARING OUR PROPOSED STATISTICAL METHOD TO THE **RS** METHOD ON FOUR EXAMPLES

Example	Statistic	*Hold&Kill*	*Zero&Kill*	*Hold&Skip-Next*	*Zero&Skip-Next*
RC network	$\mathbf{d}_{ub}$ (Alg. 1)	1.898 (0)	1.898 (0)	1.819 (0)	1.621 (0)
	$\mathbf{d}_{ub}$ (**RS**)	1.90	1.90	1.90	1.90
	Time Taken (Alg. 1)	2.3 s	2.03 s	3 s	2.53 s
	Time Taken (**RS**)	0.68 s	0.78 s	2.27 s	2.37 s
Electric Steering	$\mathbf{d}_{ub}$ (Alg. 1)	3.809 (0)	7.835 (0.15)	4.65 (0)	7.207 (0.203)
	$\mathbf{d}_{ub}$ (**RS**)	12.37	13.63	12.38	13.75
	Time Taken (Alg. 1)	1.68 s	5.29 s	3.26 s	7.03 s
	Time Taken (**RS**)	30.8 s	685 s	2845 s	2864 s
Unstable second-order	$\mathbf{d}_{ub}$ (Alg. 1)	3.552 (0)	11.124 (0.71)	7.065 (0)	9.784 (0.49)
	$\mathbf{d}_{ub}$ (**RS**)	*timed out*	*timed out*	*timed out*	*timed out*
	Time Taken (Alg. 1)	2.28 s	5.55 s	2.17 s	4.05 s
	Time Taken (**RS**)	*timed out*	*timed out*	*timed out*	*timed out*
F1Tenth	$\mathbf{d}_{ub}$ (Alg. 1)	8.763 (0)	15.928 (0.55)	14.2 (0.65)	14.02 (0.5)
	$\mathbf{d}_{ub}$ (**RS**)	6.01	*timed out*	*timed out*	*timed out*
	Time Taken (Alg. 1)	2.02 s	5 s	7.19 s	6.08 s
	Time Taken (**RS**)	1569 s	*timed out*	*timed out*	*timed out*

RC Network

A RC network is a type of electronic circuit that consists of a resistor (R) and a capacitor (C) connected in series or in parallel.

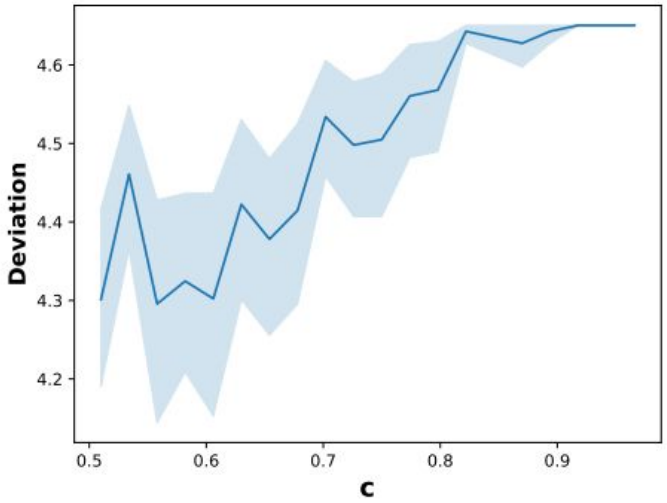
RC networks are commonly used in electronic filters, oscillators, and timing circuits. They can be used to control the frequency response of a circuit, or to create time delays and pulse shaping.

Example	Statistic	*Hold&Kill*	*Zero&Kill*	*Hold&Skip-Next*	*Zero&Skip-Next*
RC network	$\mathbf{d}_{ub}$ (Alg. 1)	1.898 (0)	1.898 (0)	1.819 (0)	1.621 (0)
	$\mathbf{d}_{ub}$ (**RS**)	1.90	1.90	1.90	1.90
	Time Taken (Alg. 1)	2.3 s	2.03 s	3 s	2.53 s
	Time Taken (**RS**)	0.68 s	0.78 s	2.27 s	2.37 s

Electric steering:

Example	Statistic	*Hold&Kill*	*Zero&Kill*	*Hold&Skip-Next*	*Zero&Skip-Next*
Electric Steering	$\mathbf{d}_{ub}$ (Alg. 1)	3.809 (0)	7.835 (0.15)	4.65 (0)	7.207 (0.203)
	$\mathbf{d}_{ub}$ (**RS**)	12.37	13.63	12.38	13.75
	Time Taken (Alg. 1)	1.68 s	5.29 s	3.26 s	7.03 s
	Time Taken (**RS**)	30.8 s	685 s	2845 s	2864 s

Mean values of dub (navy) and 95% confidence interval (light blue) with varying probabilistic guarantee c on the computed bound

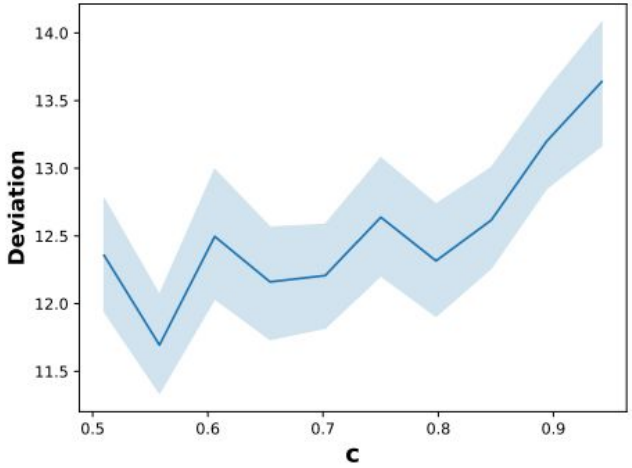


(a) Electric steering

F1Tenth

Example	Statistic	*Hold&Kill*	*Zero&Kill*	*Hold&Skip-Next*	*Zero&Skip-Next*
F1Tenth	$\mathbf{d}_{ub}$ (Alg. 1)	8.763 (0)	15.928 (0.55)	14.2 (0.65)	14.02 (0.5)
	$\mathbf{d}_{ub}$ (**RS**)	6.01	*timed out*	*timed out*	*timed out*
	Time Taken (Alg. 1)	2.02 s	5 s	7.19 s	6.08 s
	Time Taken (**RS**)	1569 s	*timed out*	*timed out*	*timed out*

Mean values of dub (navy) and 95% confidence interval (light blue) with varying probabilistic guarantee c on the computed bound

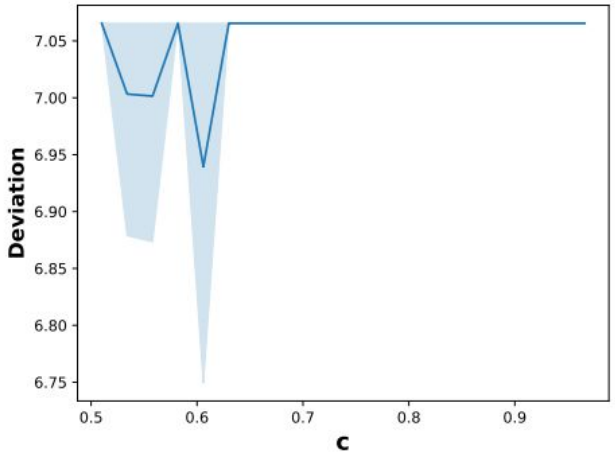


(c) F1Tenth

Unstable second-order

Example	Statistic	*Hold&Kill*	*Zero&Kill*	*Hold&Skip-Next*	*Zero&Skip-Next*
Unstable second-order	$\mathbf{d}_{ub}$ (Alg. 1)	3.552 (0)	11.124 (0.71)	7.065 (0)	9.784 (0.49)
	$\mathbf{d}_{ub}$ (**RS**)	*timed out*	*timed out*	*timed out*	*timed out*
	Time Taken (Alg. 1)	2.28 s	5.55 s	2.17 s	4.05 s
	Time Taken (**RS**)	*timed out*	*timed out*	*timed out*	*timed out*

Mean values of dub (navy) and 95% confidence interval (light blue) with varying probabilistic guarantee c on the computed bound



(b) Unstable second-order

Future work

- Complicated deadline miss patterns
- High dimensional and non-linear systems as opposed to low dimensional linear systems
- A richer set of quantitative properties that autonomous systems are often required to satisfy (e.g., the system must avoid certain regions but must also visit some other locations with a specified frequency)

  Example: delivery drone