Synthesis of Controllers for Continuous Blackbox Systems *

Benedikt Maderbacher[0000−0002−5834−352X]1, Felix Windisch[0009−0001−8174−9907]1, Alberto Larrauri[0000−0002−5935−4917]2, and Roderick Bloem[0000−0002−1411−5744]1

[1] Graz University of Technology, Graz, Austria
{benedikt.maderbacher,roderick.bloem}@iaik.tugraz.at
felix.windisch@tugraz.at
[2] University of Oxford, Oxford, UK
alberto.larrauri@cs.ox.ac.uk

Abstract. Feed-forward controllers compute control outputs to adapt to changes in environmental parameters in a cyber-physical system. When synthesizing control functions it can be difficult to give an analytical description of the controlled plant or to decide the expected control output ahead of time. These systems must, however, adhere to strict safety requirements, which makes it hard to write correct controllers. In this paper, we propose a novel blackbox synthesis approach to construct a continuous control function while dynamically sampling a limited number of test cases. The controller is guaranteed to be correct for a given Lipschitz bound. It can be adapted to work for increasingly conservative estimates of the bound based on observed behavior, iteratively providing increasing confidence in its correctness. Our algorithm employs a linear interpolation model, based on a Delaunay triangulation, to identify candidate control functions. It then generates additional test cases to either confirm a candidate or to improve the model. We evaluate our approach on random benchmarks and CPS examples to show its effectiveness.

1 Introduction

In this paper, we focus on the synthesis of a type of feed-forward control functions that interact with a blackbox environment that can only be sampled but have strong requirements to be correct. A feed-forward controller measures an environmental parameter, such as temperature, and produces control outputs to ensure the system behaves correctly. This architecture is well suited to handle sudden large changes in the environment, because it can preemptively change its control outputs. In practice feed-forward and feedback controllers are often used together where the feed-forward controller handles large disturbances with

* We would like to thank Dejan Ničković for helpful discussions on synthesis for blackbox systems. This work has received funding from the Austrian research promotion agency FFG under project FATE (№ 894789). This work was supported partially by UKRI EP/X024431/1.

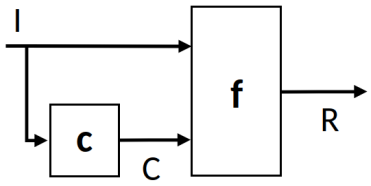


Fig. 1: Schematic

measurable causes and the feedback controller handles smaller disturbances. This paper focuses on synthesizing feed-forward controllers that control a black box system.

Figure 1 show the high-level structure of the synthesis problem. The system and its specification is modeled as function f that takes as inputs the uncontrollable environment input I (e.g. temperature) and the controllable input C (e.g. valve setting) that is controlled by the controller c. The output of f is a real-valued *robustness score* quantifying how robustly the plant satisfies the specification, where a positive (negative) value signifies a satisfied specification (a violated specification, resp.). We want to automatically synthesize a controller c for cyber-physical systems (CPS) where both the uncontrollable (I) and controllable inputs C are real-valued. The controllable inputs C have to be chosen such that for every uncontrollable input I the system satisfies the specification.

We consider a setting where the system is only available as a blackbox function. Cyber-physical systems often depend on a physical environment for which no accurate model is available. Even for applications where source code is available, such as analog systems, analytical reasoning about the system is often impossible. Synthesis is performed in a test environment (a laboratory or simulator) where both the controllable and uncontrollable inputs can be selected for a simulation. This setup makes it impossible to use traditional synthesis methods that require a white-box model of the environment. When a white-box model is available traditional methods such as reactive synthesis [5] or program synthesis [3, 18] can be used. Other methods use a set of input-output examples [21] which requires the user to know the desired output.

As a further complication, the evaluation of a cyber-physical system may be very time-consuming. For instance, executing a robotic system in physical environments can take a long time. Performing gate-level hardware simulations is also much slower than high-level software models. For example, a single simulation using the circuit simulator in our industrial case study takes between one and two minutes. We thus have a setting in which the behavior of the system can only be approximated by testing it, and the number of test runs has to be minimized as much as possible for the procedure to be time efficient.

Formal correctness guarantees are not available when using blackbox optimization and machine learning methods that can be used to handle blackbox systems, these also require a large number of simulations. We use robustness measures [10] along with the assumption of Lipschitz continuity to show formal correctness in

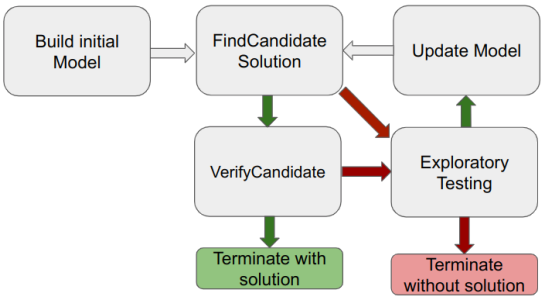


Fig. 2: Blackbox synthesis overview.

a blackbox setting. The assumption of Lipschitz continuity is well established in the control community [30] and also applied in the verification of continuous systems [17]. Lipschitz continuity together with a known Lipschitz constant means that for every sample that satisfies the specification we can conclude that all inputs within a certain distance around the sample are guaranteed to satisfy the specification without having to sample them. This makes it possible to verify a function over a continuous state space using a finite number of samples.

Note that if the plant is discontinuous in arbitrary ways, we cannot draw conclusions from any finite number of samples. We thus adopt the assumption that the plant is Lipschitz continuous with constant k.

Our Approach. We present a novel approach that combines blackbox testing with functional synthesis to find control functions for blackbox environments. Our algorithm uses a linear interpolation model of the CPS based on a Delaunay triangulation of samples of the CPS. To build an initial model, we run a small number of random tests consisting of uncontrollable and controllable inputs.

We then proceed in three steps. (See Fig. 2 for an overview.) First, we construct a candidate control function that is correct according to the model by performing a search for a path starting at the minimum input value and ending at the maximum input value that goes only through vertices with positive values. Second, if we find a candidate that is correct according to the model, we run additional tests with the aim to falsify or verify it (for the given Lipschitz constant). The testing effort is guided by an uncertainty metric that quantifies the likelihood that the model is wrong at any given point by comparing the model's prediction with the upper and lower bounds provided by the Lipschitz constant. If the candidate is falsified by finding an input for which it does not meet the specification, we use the executed tests to improve the model and identify a different candidate based on the new model. Third, if we cannot find a candidate, we perform exploratory testing to reduce the uncertainty of the overall model. This helps to identify promising regions for candidate functions that might otherwise be overlooked.

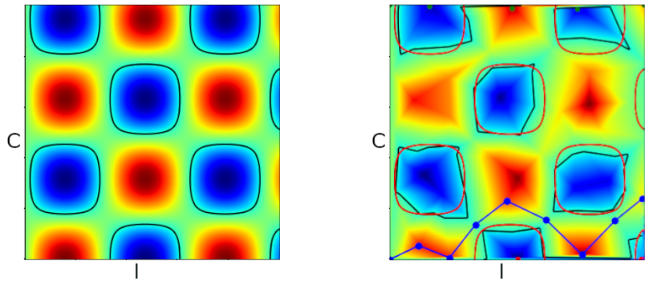


Fig. 3: Blackbox function of running example (left). Final model and solution (right).

The tests that we run during exploration and validation lead to an iteratively-improving model. If a solution with a good enough robustness exists, the model will eventually be good enough to find it. If no solution exists the model improves until it can guarantee that no solution exists (Proposition 8).

The Lipschitz constant may be known, or it can be estimated using the samples obtained during the synthesis of the controller. In the latter case, we can estimate the actual value of k using one of several known means such as the Strongin estimator. This allows us to generate a controller for an arbitrarily conservative estimate of k, or even a sequence of controllers that is guaranteed to be correct for increasingly conservative estimates.

We use the simple synthetic function $f(i, c) = \sin(i) * \cos(c) + 0.2$ as a running example. Fig. 3(left) shows the black box function f, where the function values are shown as a color gradient between red for high values (correct behavior) and blue for low values (incorrect behavior). The zero-level curves are shown in black. On the right, we show the interpolation model including the solution in blue, the zero-level lines of the model in black, and the zero-level lines of f in red. The right part shows the constructed model and the found solution.

Contributions. Our main results can be summarized as follows.

- We provide a novel approach to synthesize continuous control functions for blackbox CPSs that uses guided testing and counterexample-guided model improvement to find promising solutions.
- We provide an uncertainty measure to estimate the quality of a model that improves iteratively.
- We evaluate our method on multiple case studies and demonstrate that it can find provably correct control functions using a small number of test cases.

2 Related Work

2.1 Simulation based Validation

Simulation-based validation [7, 8, 16] or falsification testing can be used to falsify cyber-physical systems. Given a blackbox function representing a plant, the goal

is to find a controller input that violates a given formal specification. These robustness specifications can be used to guide an optimizer to find inputs that violate the specification [11, 13, 25].

These methods focus on testing and falsifying a plant in combination with a given controller. We use similar robustness specifications to quantify the satisfaction of a controller. In contrast to these falsification approaches, we consider a synthesis problem. We are not given the controller, but we try to find one while executing tests. The given Lipschitz bound also allows us to verify a system by obtaining a test suite that guarantees correctness.

2.2 Min-Max Blackbox Optimization

A variation of the blackbox synthesis problem can be formulated as a multi-objective optimization problem. These are often referred to as min-max blackbox optimization or robust blackbox optimization [2, 12, 22, 24]. Given a function with two sets of parameters, in blackbox optimization, we want to find a value for the first parameter that maximizes the minimal output of the function over all possible values of the second parameter. This approach could be used to perform a parameter synthesis version of our problem: it can find a *fixed* controller output that maximizes the worst plant output for any of the uncontrollable input values. If this output is positive, the controller output satisfies the specification for any uncontrollable input.

We propose a more general approach in which the controller output is not fixed but rather depends on its input. Of course, in the case that the structure of the control function is known, we can use blackbox optimization to search for a set of template parameters. However, often such a template is not available. Furthermore, blackbox optimization methods provide no guarantee that the found minimum is correct or greater than zero (especially for functions that are not convex). Our method, on the other hand, can prove that the proposed solution is correct and uses fewer samples as we are not interested in an optimal solution but only in one that satisfies our specification.

2.3 Models for Blackbox systems

Models are widely used in blackbox optimization settings, or even in situations where the objective function is known but too complex, so performing optimization on a simpler approximate model is more desirable than doing so on the original function.

A popular approach to blackbox optimization is Bayesian Optimization, which relies on statistical models. These models provide an estimate of the objective function at each point together with a measure of uncertainty, given by the standard deviations of some underlying probability distributions. This measure of uncertainty is useful for balancing the exploration of low information areas and exploitation of the information accrued so far. Out of these models, Gaussian Processes are a popular choice across multiple domains of application [27]. In particular, they have been used in the verification of cyber-physical systems [9,34].

Another important family of models is those obtained via interpolation of different kinds of basis functions. It is possible to equip such models with tailored uncertainty measures and use them in blackbox optimization, achieving similar results to Bayesian optimization [4].

In our proposed method we opt for piece-wise linear interpolations as models. These are simpler than the models described above, and hence it is easier to perform tasks like finding level curves on them. This can be done exactly and efficiently on our models, whereas one would need to resort to inexact and more inefficient numerical methods to find level curves on Gaussian Processes or other more involved kinds of interpolations. A potential drawback of our simpler models is its lower precision, but whereas in traditional blackbox optimization precision plays a more important role (i.e., one needs to distinguish between potentially close and high values when looking for a global maximum), we only need to ensure that the output of the plant remains positive, which often allows us to use less precise models.

3 Preliminaries

3.1 Formal Problem Statement

We consider blackbox functions $f : I \times C \to \mathbb{R}$ with two types of inputs: the uncontrollable input I and the controllable inputs C. We assume that I and C are box-shaped real-valued domains with dimensions 1 and n. We assume without loss of generality that every dimension has a lower bound of min and an upper bound of max. The input domains are thus $I = [min, max]$ and $C = [min, max]^n$. We use s^I to refer to the uncontrollable value and s^C to the controllable values of s from $I \times C$. The controllable inputs to the plant are computed by a control function $c \in (I \to C)$, a continuous function that selects a controllable input for each uncontrollable input.

The goal of our approach is to find a control function c for a given black box plant and specification f that always satisfies the specification, i.e., $\forall i \in I : f(i, c(i)) \geq 0$. This should be done by collecting as few samples as possible. We define a sample s as a point in the domain $I \times C$ for which we evaluate the blackbox function f and we refer to the set of all collected samples as $S \subset I \times C$.

Many systems do not have a single output where positive values correspond to satisfying the specification. In that case the expected behavior of the system is provided in the form of a quantitative specification. This evaluates the quality of the produced output and assigns it a real-valued score. Positive values indicate that the specification is satisfied whereas negative values signify a violated specification. Similar robustness measures are often used for temporal specifications [10, 14, 23]. This quantitative specification is integrated into the blackbox function $f/$

3.2 Lipschitz Continuity

We assume the blackbox function f consisting of plant and specification is Lipschitz continuous with constant k, that is, $\forall x_1, x_2 \in I \times C : |f(x_1) - f(x_2)| \leq k\|x_1 - x_2\|$ where $\|\cdot\|$ refers to the Euclidean norm.

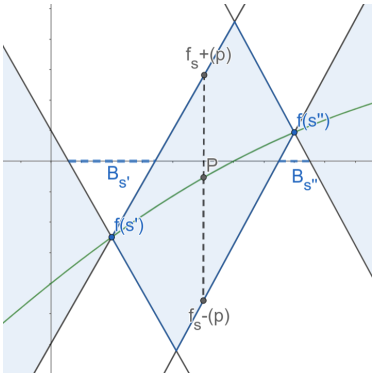


Fig. 4: Lipschitz bound and derived quantities.

Given a set of samples $S \in I \times C$, we can compute the maximum and minimum value that f may take at a point p in $I \times C$ as $\hat{f}_S^+(p) = \min_{s \in S} f(s) + k \, \|s - p\|$ and $\hat{f}_S^-(p) = \max_{s \in S} f(s) - k \, \|s - p\|$. Lipschitz continuity provides such a guarantee for the neighborhood of any positive or negative samples. We define the *Lipschitz ball* around a sample s with radius $|f(s)|/k$ as $B_k(s) = \{p \in I \times C \mid \|p - s\| \leq \frac{|f(s)|}{k}\}$. For all $p \in B_k(s)$ for a sample s, Lipschitz continuity with a Lipschitz constant less than or equal to k guarantees that $f(p) \geq 0$ if $f(s) \geq 0$ and that $f(p) \leq 0$ if $f(s) \leq 0$.

Figure 4 depicts these concepts using a function f with the one-dimensional input s on the x-axis and the function value $f(s)$ on the y-axis. Based on the samples s' and s'', the function f is guaranteed to lie inside the shaded area created by the cones around the samples. The balls (intervals in the one-dimensional case) around each s are denoted by $B(s)$. For one point p the figure shows $\hat{f}_S^+(p)$ and $\hat{f}_S^-(p)$.

If the Lipschitz constant is not known, it can be approximated using the collected samples. A classic approach is the Strongin estimator [29], defined as $\hat{k} = r \cdot \max_{i \neq j} |f(s_i) - f(s_j)| / \|s_i - s_j\|$ for $r > 1$. There are also more precise stochastic estimations [15, 19, 33].

3.3 Delaunay Triangulation and Interpolation

Let $S \subseteq I \times C$ be a finite set of points. A *Delaunay triangulation* $\mathcal{T}$ of S is a subdivision of the convex hull of S into d-simplices (d-dimensional generalizations of triangles) that are "compact" in the sense that no point lies strictly within the circum-hypersphere (the generalization of the circumcircle of a triangle) of any simplex T.

Given a function $f : I \times C \to \mathbb{R}$, a set of samples $\{(s_1, f(s_1)), \ldots, (s_n, f(s_n))\}$, and a triangulation $\mathcal{T}$ of the set $S = \{s_1, \ldots, s_n\}$, the piece-wise linear interpolation of f given by $\mathcal{T}$ is the map $m_{\mathcal{T}}$ defined as follows: The domain of $m_{\mathcal{T}}$ is the convex hull of S. Given a simplex $T \in \mathcal{T}$ with vertices $s_{r_1}, \ldots, s_{r_d}$ and

a point $x \in T$, there are unique $\lambda_1, \dots \lambda_d \in [0,1]$ satisfying $\sum_{t=1}^{d} \lambda_t = 1$ and $x = \sum_{t=1}^{d} \lambda_t s_{r_t}$ where the second sum is between vectors in the Euclidean space. Then the value $m_{\mathcal{T}}(x)$ is defined as $\sum_{t=1}^{d} \lambda_t f(s_{r_t})$.

Delaunay triangulations are well-suited for the interpolation of spatial data [32], as they avoid long thin triangles and induce a neighborhood structure on the vertices that relates well to their distances on the space [28].

4 Blackbox Synthesis Method

4.1 Exploration and Model

Our method uses samples collected from simulations to identify new candidate solutions. We use $m_{\mathcal{T}} : I \times C \to \mathbb{R}$ to denote the value predicted by the linear approximation model for an arbitrary point in the domain. In combination with a known Lipschitz bound a precise model of f can be obtained.

Proposition 1. *Suppose f's Lipschitz constant is $k \geq 0$, and that the maximum length of an edge inside $\mathcal{T}$ is d. Then $|m_{\mathcal{T}}(x) - f(x)| \leq kd$ for all $x \in I \times C$.*

Proof. Consider a simplex T in $\mathcal{T}$ with vertices $v_1, ..., v_m$. For any point $x \in T$ it holds that

$$\min\{f(v_h) \mid 1 \leq h \leq m\} \leq m_{\mathcal{T}}(x) \leq \max\{f(v_h) \mid 1 \leq h \leq m\}.$$

The maximal distance between any vertex of T to any point inside T is bounded by the length of the longest edge d.

As such, $\forall v \in \{v_1, ..., v_m\} \forall x \in \mathcal{T} : f(v) - kd < f(x) < f(v) + kd$. Symmetrically, this also implies $\forall v \in \{v_1, ..., v_m\} \forall x \in \mathcal{T} : f(x) - kd < f(v) < f(x) + kd$. As $m_{\mathcal{T}}(x)$ is a weighted average of $f(v_i)$ for $i \in \{1, ..., m\}$, $m_{\mathcal{T}}(x)$ is contained in the interval $[f(x) - kd, f(x) + kd]$, which directly implies $|m_{\mathcal{T}}(x) - f(x)| \leq kd$.

A corollary is the following:

Proposition 2. *Let k be f's Lipschitz constant, and let d be the smallest edge length in the triangulation $\mathcal{T}$. Suppose that $kd \leq \epsilon$ for some $\epsilon > 0$. Then the following hold:*

- *If $c : I \to C$ is such that $m_{\mathcal{T}}(x, c(x)) \geq \epsilon$ for all x, then c is a solution for f.*
- *If $c : I \to C$ is such that $f(x, c(x)) \geq \epsilon$, then c is a solution for $m_{\mathcal{T}}$.*

To guide the selection of samples we define a heuristic to select interesting points. An *uncertainty estimator* assigns a non-negative value to each point in the domain $I \times C$, where zero means that the model is correct at this point and a higher score indicates less confidence in the predicted value. We provide two uncertainty estimates, each using the Lipschitz property. The first estimate, $\hat{u}$, compares the upper and lower bounds: $\hat{u}_S(p) = \hat{f}_S^+(p) - \hat{f}_S^-(p)$. This assigns the highest uncertainty to points with the largest range of possible values.

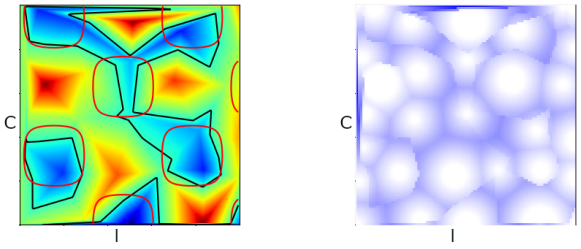


Fig. 5: Delaunay interpolation model for the running example (left). Uncertainty scores $\hat{u}$ for the same model (right), white denotes zero uncertainty and the darker the color the higher the uncertainty.

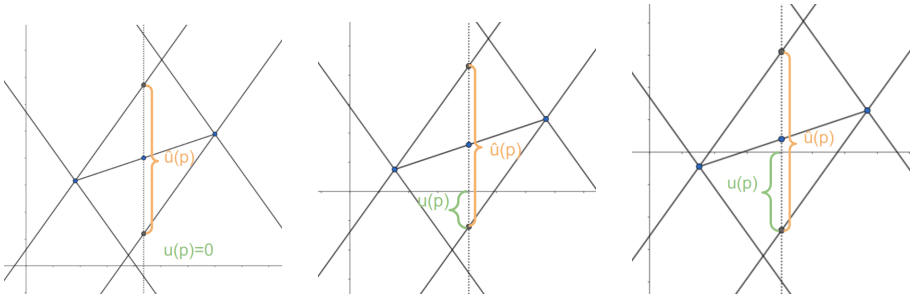


Fig. 6: Uncertainty estimation.

We propose a novel uncertainty estimator u that takes into account that we are looking for a model that correctly predicts the sign of a function, but might be less certain about the exact value.

$$u_S(p) = \begin{cases} \max(\hat{f}_S^+(p), 0) & \text{if } m_{\mathcal{T}}(p) \leq 0, \\ \max(-\hat{f}_S^-(p), 0) & \text{otherwise.} \end{cases}$$

Figure 6 shows three cases that are assigned the same uncertainty score using $\hat{u}$, but have different uncertainty scores using u based on how far the prediction is from zero. Figure 5 shows the Delaunay interpolation model of an early iteration and the values of the uncertainty measure u for the same model.

To improve the global quality of our model we use exploratory tests. These help to find new candidates in under-explored areas. Algorithm 1 select points using three heuristics. Two of them are based on the uncertainty estimators u and $\hat{u}$, for which we use the shorthand $\hat{u}_V(x)$ and $u_V(x)$ where the set V is implicitly selected as the vertices of the simplex in $\mathcal{T}$ that contains x. The third heuristic eventually selects all points on a grid with spacing $l = \epsilon/(2k\sqrt{n+1})$.

Proposition 3. *Let k be f's Lipschitz constant, and let d be the longest edge length in the triangulation $\mathcal{T}$. Exploratory testing will eventually result in a model such that $k \cdot d \leq \epsilon$ for some $\epsilon > 0$.*

Algorithm 1 Exploratory Testing

1: **function** ExploratoryTesting($f, m_{\mathcal{T}}$)
2: **for** *number of explorations* **do**
3: $x_1 \leftarrow \arg\max_{x \in I \times C} \hat{u}_V(x)$
4: $x_2 \leftarrow \arg\max_{x \in I \times C} u_V(x)$
5: $x_3 \leftarrow (s_0 l, \ldots, s_n l)$ for unused $s_0 \ldots s_n \in \{0, 1, \ldots, max/l\}$ ▷ Terminate if the grid is complete.
6: $m_{\mathcal{T}} \leftarrow$ DelaunayTriangulation($S \cup \{x_1, x_2, x_3\}$)

Every call to exploratory testing includes one sample from a grid with spacing $l = \epsilon/(2k\sqrt{n+1})$, where n is the number of controllable parameters. The factor $\sqrt{n+1}$ is the length of the diagonal in such a grid cell. The domain is bounded so this grid contains only a finite number of points. After all the points from this grid have been sampled, any Delaunay triangulation contains no edges longer than ϵ/k.

4.2 Searching for Candidate Solutions

The candidate search procedure takes a model in the form of a triangulation graph $\mathcal{T}$ and its associated interpolation function $m_{\mathcal{T}}$. We want to identify candidate solutions c for the controller. The main criterion for a candidate controller is that it satisfies the specification according to our model. This leaves us with a very large and unstructured search space. To make the problem more tractable we limit the candidates to piece-wise linear functions. This is a reasonable restriction as with enough segments we can represent a solution to every realistic problem for which a solution exists.

Most models will allow for multiple solutions. In principle, any of them would serve as a candidate solution. However, candidates with a higher minimal value seem to result in faster overall convergence. We rank candidates c by the lowest point according to the interpolation model i.e. $\min_{i \in I}(m_{\mathcal{T}}(i, c(i)))$.

We present two algorithms to search for candidates based on the triangulation model. The first uses the bottleneck shortest path algorithm along the edges of the triangulation to find candidates with high minimal value. This works well in practice, but it might miss some candidates. For completeness, we also present a second algorithm which is guaranteed to find a candidate if one exists.

Using Bottleneck Shortest Paths A solution corresponds to a path that connects a point at the lower bound of the input domain to another point at the upper bound of the input domain. Instead of searching all possible piece-wise linear functions, we limit the search to only segments that are part of the Delaunay triangulation graph. This allows us to use standard algorithms for finite graphs, but it might miss some candidates. The limited number of considered paths is alleviated by improving the model which increases the number of edges.

Algorithm 2 shows how to translate our synthesis problem into a bottleneck shortest path problem on a directed subgraph of the Delaunay triangulation

Algorithm 2 Path search for finding candidates.

function FINDCANDIDATESOLUTION($m_{\mathcal{T}}$)
  $S, E \leftarrow$ vertices and edges from $\mathcal{T}$
  $E \leftarrow \{((s_1^I, s_1^C), (s_2^I, s_2^C)) \in E \mid s_1^I < s_2^I\}$
  $E' \leftarrow \{(s_1, s_2) \in E \mid m_{\mathcal{T}}(s_1) > 0 \wedge m_{\mathcal{T}}(s_2) > 0\}$
  $S' \leftarrow \{s \in S \mid m_{\mathcal{T}}(s) > 0\}$
  $S'' \leftarrow S' \cup \{start, end\}$
  $E'' \leftarrow E' \cup \{(start, s) \mid s^I = min\} \cup \{(s, end) \mid s^I = max\}$
  $w(s_1, s_2) = \begin{cases} \min(m_{\mathcal{T}}(s_1), m_{\mathcal{T}}(s_2)) & (s_1, s_2) \in E' \\ 0 & \text{otherwise} \end{cases}$
  return BOTTLENECKSHORTESTPATH((S'', E''), $start$,end, w)

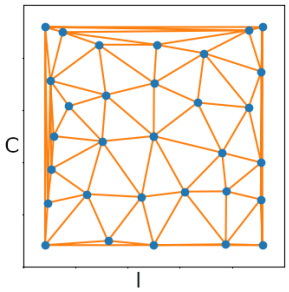


Fig. 7: Delaunay graph.

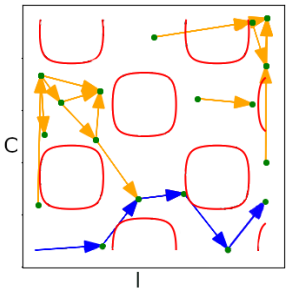


Fig. 8: Directed positive subgraph with candidate.

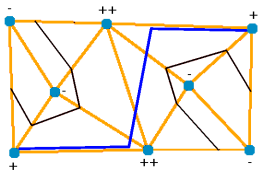


Fig. 9: Configuration where Algorithm 2 does not find an existing candidate solution.

consisting of the edges that are positive and in the direction of increasing I. We add *source* and *sink* nodes and edges to allow any vertices with $s^I = min$ as start points and vertices with $s^I = max$ as end points. The bottleneck shortest path on the graph from *source* to *sink* is the path where the minimal edge weight is maximized. This is efficiently computable using a variant of Dijkstra's algorithm or more specialized implementations. See [20] for an overview. Figure 7 shows the graph resulting from the triangulation of the running example and Figure 8 the preprocessed graph including the bottleneck shortest path.

Proposition 4 (Soundness of the candidate search). *Any candidate found by Algorithm 2 is a solution to the synthesis problem on the model.*

The search is performed on a graph with only positive edges that point in the positive I direction. A candidate solution is a continuous path from the minimum input to the maximum input. Any found path is thus a valid candidate solution.

Search algorithm with completeness guarantee. In some cases, searching only along the edges of the triangulation might result in no candidate solution being found, even when one exists (an example of this situation can be seen in Figure 9).

To solve this problem we add some auxiliary vertices and edges to the graph of the triangulation and search for a path in this extended graph. However, in this new graph vertical edges are included, so a left-to-right path in the graph does not necessarily correspond to a candidate solution c. Hence, one needs to be careful and consider only paths whose vertical segments can be "perturbed" slightly and remain non-negative. We give an outline of the procedure below.

We consider a triangulation $\mathcal{T}$ of $I \times C$. Recall that $\mathcal{T}$ is a simplicial complex. A facet $F \in \mathcal{T}$ is a d-dimensional simplex in the complex. Similarly to Algorithm 2, we build a graph $G_{\mathcal{T}}$ from $\mathcal{T}$ in such a way that paths in the graph correspond to solutions in $m_{\mathcal{T}}$. Initially, the vertex set V corresponds to the vertices $v \in \mathcal{T}$ with $m_{\mathcal{T}}(v) \geq 0$. Afterwards, for each edge $e \in \mathcal{T}$ that has at least one endpoint with a non-zero value, we consider the only point (if any) $v_e \in e$ that achieves $m_{\mathcal{T}}(v_e) = 0$. We add to V all those new points of the zero-level curve of $m_{\mathcal{T}}$. At this stage we consider for each $v \in V$ the vertical hyper-plane given by $h_v = \{x \in I \times C \mid x^I = v^I\}$. Given a non-vertical edge $e \in \mathcal{T}$, we add to V all the intersections of e with the hyperplanes h_v where the value of $m_{\mathcal{T}}$ is non-negative. Now we define a set E of directed edges on V, given by $(u, v) \in E$ if $u^I < v^I$ and u, v share a common facet in $\mathcal{T}$. The final graph $G_{\mathcal{T}}$ has vertex set $V(G_{\mathcal{T}}) = V \cup E$ (that is, we have a new vertex representing each edge in E), and its set of directed edges is given by

$$\begin{aligned} E(G_{\mathcal{T}}) = & E \cup \{(v, (u_1, u_2)) \in V \times E \mid v = u_1\} \\ & \cup \{((u_1, u_2), v) \in E \times V \mid v = u_2\} \\ & \cup \{((v_1, v_2), (u_1, u_2)) \in E \times E \mid v_1^I = u_1^I, \text{ and} \\ & v_1, v_2, u_1, u_2 \text{ share a facet in } \mathcal{T}\}. \end{aligned}$$

In summary $G_{\mathcal{T}}$ has two types of vertices: points $x \in I \times C$ and ones that correspond to "segments" $(x, y) \in (I \times C)^2$. Two points are joined by a directed edge when they lie on a shared facet and the I-coordinate of the first is smaller than the I-coordinate of the second. Vertices representing segments are joined to both their endpoints via directed edges. Finally, two vertices that represent segments are joined by an edge if the left endpoints of the segments have the same I-coordinates and the endpoints of the segments lie in some common facet.

A left-to-right path in $G_{\mathcal{T}}$ is just a directed path $v_0, \ldots, v_k$ in the graph where $v_0, v_k \in I \times C$ and $v_0^I = \mathit{min}$, $v_k^I = \mathit{max}$. The **extended search algorithm** builds the graph $G_{\mathcal{T}}$ and searches for a left-to-right path inside. The following two results state that such a path exists if and only if there is some candidate solution for $m_{\mathcal{T}}$.

Proposition 5 (Soundness of the extended search algorithm). *Suppose that there is a left-to-right path in $G_{\mathcal{T}}$. Then there is a continuous map $c : I \to C$ with $m_{\mathcal{T}}(i, c(i)) \geq 0$ for all $i \in I$.*

It is illustrative to see how a left-to-right path $v_0, \ldots, v_k$ in $G_{\mathcal{T}}$ corresponds to a solution $c : I \to C$ in last proposition. The idea is that from this path we obtain a sequence of points $x_0, \ldots, x_k \in I \times C$ whose I-coordinates are increasing.

Algorithm 3 Verification of a candidate solution.

```
1: function VerifyCandidate(c,f,m)
2:     edges ← edges of the candidate c
3:     while edges ≠ ∅ do
4:         e ← get and remove the edge with the maximum u from edges.
5:         v1, v2 ← vertices of e
6:         if ∃p ∈ e : u_{v1,v2}(p) > 0 then        ▷ equivalent to e ⊄ B_k(v1) ∪ B_k(v2)
7:             z ← arg max_{z∈e} u_{v1,v2}(z)
8:             S ← S ∪ {z}
9:             if f(z) < ϵ then                      ▷ Evaluate blackbox function
10:                return False, S
11:            edges ← edges ∪ {(v1, z), (z, v2)}
12:    return True, S
```

Then the curve $p(i) = (i, c(i))$ is just the union of the segments $[x_h, x_{h+1}]$. If the vertex v_h is just a point in $I \times C$, then we set $x_h = v_h$. Alternatively, suppose that $v_h, v_{h+1}, \ldots, v_{h+\ell}$ is a maximal sub-path where all vertices represent segments. I.e., for each $j = 0, \ldots, \ell$, $v_{h+j} = (u_j, w_j) \in (I \times C)^2$. Observe that by the definition of $G_{\mathcal{T}}$, u_j^I remains constant for all j, and $w_j^I > u_j^I$. Then for each $j = 0, \ldots, \ell$ we pick x_{h+j} as an arbitrary point in the interval $[u_j, w_j]$ in such a way that the I-coordinates of the points are increasing. This allows us to complete the choice of the points $x_0, \ldots, x_k$ and define the curve $(i, c(i))$.

Proposition 6 (Completeness of the extended search algorithm). *Suppose there is a continuous map $c : I \to C$ with $m_{\mathcal{T}}(i, c(i)) \geq 0$ for all $i \in I$. Then there is a left-to-right path in $G_{\mathcal{T}}$.*

4.3 Verification of Candidate Solutions

When a candidate solution is found we need to verify whether $\forall i \in I : f(i, c(i)) \geq \epsilon$ by sampling new values and using previous samples of f. This is done via a guided testing procedure shown in Algorithm 3. It aims to verify a candidate solution by finding samples S such that the Lipschitz balls induced by the samples completely cover the candidate.

A candidate solution consists of a set of edges. The verification is done one edge at a time and either confirms that it is covered by the Lipschitz balls induced by its vertices or adds a new sample on the edge. The new sample might be negative which falsifies the candidate, or it is used to split the edge into two parts that need to be verified.

The edges are verified in the order of their uncertainty u, based on their vertices. We define the uncertainty of an edge $e = (v_1, v_2)$ as $u(e) = \max_{x \in e} u_{\{v_1, v_2\}}(x)$. Selecting edges in this order makes it more likely to quickly reject spurious candidates.

If an edge e is not covered, the point on the edge with the highest uncertainty is added as a new sample. This either falsifies the candidate or splits the edge

into two smaller edges to be tested. The algorithm rejects all candidates with values less than ϵ, as the verification might otherwise get stuck for a candidate that asymptotically approaches zero.

Proposition 7 (Soundness and ϵ-completeness of the verification). *Given a Lipschitz constant k and an $\epsilon > 0$, for all candidates c, the verification will terminate with either a counterexample $p \in I$ such that $f(p, c(p)) < \epsilon$, or by showing that $f(i, c(i)) \geq 0$ for all $i \in I$.*

Proof. In case a candidate is verified every point in the candidate solution is included in a Lipschitz-ball of a positive sample. If a counter-example is found this is witnessed by a point on the candidate with a function value less than ϵ. To show termination we observe that the highest uncertainty of any edge is decreasing in every iteration of the verification loop and it is bounded below by 0. Given an edge $e = (v_1, v_2)$ and a point $z \in e$ with $f(z) \geq \epsilon$ and $u_{\{v_1,v_2\}}(z) > 0$. We have that $\|v_i - z\| \geq \epsilon/k$ for $i \in \{1, 2\}$, because any point closer to one of the vertices has a uncertainty of 0. Splitting e into two edges $e_1 = (v_1, z)$ and $e_2 = (z, v_2)$ results in the uncertainties $u_{\{v_1,z\}}(e_1)$ and $u_{\{z,v_2\}}(e_2)$ which are both less than $u_{\{v_1,v_2\}}(e) - \epsilon$. Every iteration of the loop reduces the highest uncertainty of all edges by at least ϵ, or it decreases the finite number of edges that share the highest uncertainty. Therefore, the verification loop terminates with a correct result.

4.4 Blackbox Synthesis

Based on the components defined in the previous sections we can define our method for blackbox function synthesis as shown in Algorithm 4. The main loop starts by performing some guided exploration tests as described in Section 4.1 and constructs a triangulation model from all samples. Then, one of the search algorithms from Section 4.2 is used to search for a candidate solution. If no candidate can be found, more exploration tests are performed. Once a candidate has been found, verification as described in Section 4.3 is attempted. When a counter-example is found, it and all other samples collected during verification are incorporated into the model. The algorithm terminates once a candidate has been verified successfully.

Proposition 8 (Soundness and ϵ-completeness of the synthesis algorithm). *For a known Lipschitz constant k and an $\epsilon > 0$. Given there exists a $c' \in I \to C$ with $f(i, c'(i)) \geq 2\epsilon$ for all $i \in I$. The Algorithm 4 finds a control function $c \in I \to C$ such that $f(i, c(i)) \geq 0$ for all $i \in I$.*

Proof. As long as no solution has been found, the algorithm performs repeated calls to ExploratoryTesting, because all functions called in Algorithm 4 terminate. ExploratoryTesting performs a finite number of simulations. DelaunayTriangulations and the path search in FindCandidateSolution are graph algorithms known to terminate. The termination of FindCandidateSolution is shown in Proposition 7. Given that we perform arbitrarily many calls

Algorithm 4 Blackbox Functional Synthesis

```
1: function BlackBoxSynthesis(f)
2:     Using blackbox function f, set of samples S, model m, and candidate solution c.
3:     loop
4:         S ← S ∪ ExploratoryTesting(f, m)
5:         m ← DelaunayTriangulation(S)
6:         c ← FindCandidateSolution(m)
7:         if c = null then
8:             continue
9:         Correct, S' ← VerifyCandidate(c, f)
10:        if Correct then
11:            return c
12:        else
13:            S ← S ∪ S'
```

to ExploratoryTesting eventually all grid points are sampled. So the length d of the longest edge in the triangulation satisfies $kd \leq \epsilon$ (Proposition 3).

Proposition 2 shows for the given c with $\forall i : f(i, c(i)) \geq 2\epsilon$, there is a c' in the model such that $m_{\mathcal{T}}(i, c'(i)) \geq \epsilon$. By Proposition 6 the search finds a candidate solution c such that $m_{\mathcal{T}}(i, c(i)) \geq \epsilon$. By Proposition 7 the verification is guaranteed to succeed for the candidate c as no valid counterexample exists. As such, VerifyCandidate will show that $f(i, c(i)) \geq 0$ for all $i \in I$.

The output upon termination of the algorithm depends on the quality of possible solutions in the blackbox function. We distinguish three cases based on the bottleneck value $b := \max_c(min_{i \in I} f(i, c(i))$:

$b > 2\epsilon$	The algorithm is guaranteed to find a solution c such that $\forall i f(i, c(i)) > 0$.
$0 < b < 2\epsilon$	The algorithm may find a valid solution c such that $\forall i f(i, c(i)) > 0$ or it will terminate without a solution.
$b < 0$	The algorithm is guaranteed to terminate without a solution.

The ϵ-completeness guarantees rely on an extensive exploration. Even though a negative result can only be shown for very small problems, this property shows that our algorithm is guaranteed to progress towards a solution.

4.5 Dynamic Lipschitz Bound Estimation

The previous section describes a sound and ϵ-complete synthesis algorithm for the case where the Lipschitz bound for the blackbox function is known. For some systems only the fact that they are Lipschitz continuous is know without also knowing the value of the Lipschitz constant. In that case no soundness or completeness guarantees are possible. However, Lipschitz bound estimation can be incorporated into our method by computing an estimate based on the collected samples after a solution has been found. The algorithm can then be restarted with a more precise or conservative Lipschitz bound and using all previous samples to

Algorithm 5 Synthesis with dynamic Lipschitz bound estimation.

```
1: S ← ExploratoryTesting(f)                    ▷ Initialization with some random samples
2: select r ≥ 1.0
3: loop
4:     k ← LipschitzEstimate(S, r)
5:     c, S ← BlackBoxSynthesis(f, S, k)
6:     k' ← LipschitzEstimate(S, r)
7:     if k' ≤ k then                                  ▷ estimate did not change
8:         if c is null then
9:             yield (null, r)                  ▷ Return null and confidence and continue.
10:            increase r
11:        else
12:            yield (c, r)                ▷ Return solution and confidence and continue.
13:            increase r
```

create the initial model. This way solutions for ever-improving Lipschitz bound estimates can be obtained. Algorithm 5 describes how to combine our method with Strongin's method for Lipschitz estimation. First a small number of random samples are used to compute a first estimate for the Lipschitz bound. Then the blackbox synthesis algorithm is run with the estimated Lipschitz bounds and the existing samples are used to build the initial model. If the Lipschitz bound estimate did change due to the samples collected during synthesis the synthesis algorithm is rerun with the updated Lipschitz bound. Otherwise a result is reported together with the confidence parameter for which the result is correct. The loop can then either be terminated if the confidence is good enough or repeated with a more conservative Lipschitz estimation. To improve the confidence in a found solution the candidate verification algorithm can also be run with more conservative estimates computed only from samples close to the candidate. This is justified, because rapid changes that are far away from the solution do not affect the validity of that solution.

5 Evaluation

We implemented a prototype implementation of our approach in Python. The implementation uses QHull[3] to compute Delaunay triangulations. SciPy [31] is used for its graph implementations. Our implementation uses the incomplete candidate search described in Algorithm 2 with a known Lipschitz constant. For exploratory testing one grid point and two of each uncertainty sample (u and $\hat{u}$) are added. Our prototype implementation and experiments are available at https://doi.org/10.5281/zenodo.13880504.

The focus of our algorithm is to find solutions with few evaluations of the blackbox function. We perform our evaluation on randomly generated problem instances, six CPS examples, and an industrial case study. To provide a baseline

[3] http://www.qhull.org/

for our implementation we compare with a simple DFS search and a state-of-the-art blackbox min-max optimization tool [1, 2]. Blackbox optimization has some differences compared to our approach, it requires a template and tries to maximize the function value, but does not provide a correctness guarantee. To obtain a comparable setting we use a piece-wise linear function defined by four points as the template, combine it with our verification function, and report the number of samples where the first verifiably correct function was found.

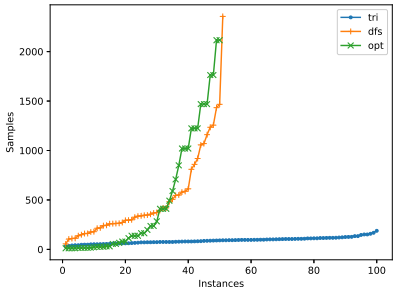
5.1 Random Perlin Noise Functions

Perlin noise [26] is a type of random gradient noise that includes a smoothing step. By varying the scale parameter, random functions with specific Lipschitz constants may be sampled for arbitrary dimensions. Instances of Perlin noise also tend to include features like dead ends and low valleys that make the problem more difficult. Setting f to an instance of Perlin noise can then supply us with an infinite number of random test cases. Not all instances of Perlin noise, particularly in low dimensions contain a valid solution c. By adding a constant value $o \in \mathbb{R}^+$ to the Perlin noise, the difficulty of the problem can be varied. We use three configurations with a different number of controllables for Perlin noise generation and generate 100 random instances of each. The results for the three evaluated methods are shown as cactus plots in Figure 10. Our method consistently finds solutions with the lowest amount of samples. In the case of one controllable about half the instances are only solved by our method.

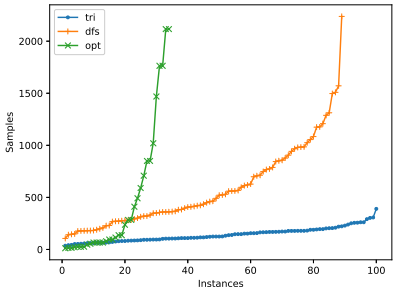
5.2 Cyber-Physical Systems

We also evaluated our approach on CPS problems three of which are illustrated in Figure 11. The first problem is a robot arm controller where the gripper at the top of the arm has to remain in a target zone and the arm must avoid collisions with obstacles. The input is the position of the target and the controllables are the two joints and the length of the upper segment. The robustness specification is based on the distance to the obstacles and to the target. The next set of problems are motion planning tasks where one or two agents have to avoid moving obstacles and each other. The controller has to select a position for each agent based on the current time. Another problem is a controller for the lights in an office space with four desks, four lamps, and windows. Based on the sunlight intensity the lamps have to be dimmed such that the amount of light at each desk is within a given tolerance.

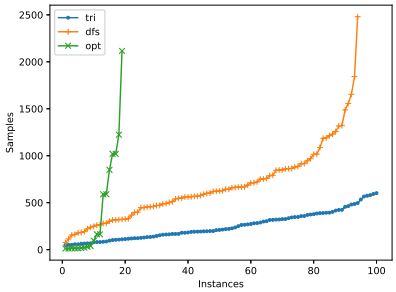
Our experiments show that our triangulation-based method outperforms the two other approaches on most benchmarks. The optimization-based method performs well on some of the simpler benchmarks where it is slightly better than our method. Our method can solve the harder examples using only a few hundred samples whereas the other methods exceeded the maximum of 10000 samples.



(a) 1 controllable



(b) 2 controllables

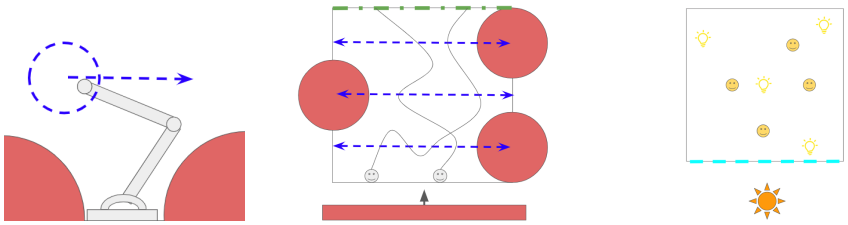


(c) 3 controllables

Fig. 10: Cactus plots for solving random instances of Perlin noise problems. It shows how many instances can be solved with a certain amount of samples. The number of simulations is capped at 2500.

5.3 Industrial Case Study

We applied our method to an industrial bandgap device [6] provided by an industry partner. It is used to provide a reference for voltage and current that is as close as possible to a target value. Figure 12 shows a schematic of the case study. The device consists of two main parts an analog bandgap and a digital trimmer. The analog part produces intermediate current and voltage from a power supply. The trimmer is then used to adjust these to be as close as possible to their respective reference values. Both components show different behavior based on the environment temperature. To compensate for these effects a small controller is used to adjust the trimmer based on the current temperature. In this case study we synthesize this temperature-trimming controller. The relationship between temperature, trimming parameters, and output voltage and current is too complex to be analyzed as a whitebox system. Therefore, we treat it as blackbox system on which simulations can be performed. The blackbox function for our method is marked by a grey box, it consists of the analog bandgap, the trimmer and the specification. This compares the output current and voltages to their reference



(a) Robot arm control with red obstacles and blue moving target zone.

(b) Multi agent motion planning.

(c) Light control with 4 desks, 4 lamps, and windows on the south side.

Fig. 11: Illustrations of selected CPS problems.

values and an allowed tolerance. The robustness value shows the distance to the tolerance boundary with positive values inside the tolerance and negative values outside. For every call to the blackbox function, nine simulations are performed using different supply parameters to obtain the worst-case robustness over variation in the supply.

The experiments are performed on a machine-learned surrogate model of the system that was trained from a highly accurate analog electronic circuit simulator model. One evaluation of the blackbox function takes about one second on the ML model and 10 minutes on the circuit simulator. Table 1 includes the median results of our experiments for five variants of the bandgap device. Out of the 25 runs (5 random seeds for each of the 5 variants) performed for the case study the optimization approach was unable find any solution, the DFS method succeeded 4 times with a best case performance of 632 samples. Our triangulation method succeeded on every run. We can conclude that our triangulation-based approach is the only one able to consistently find a solution in less than 1000 simulations.

Due to the long simulation times, this use case greatly benefits from our method which uses more resources to process the samples but can find verified solutions with a small number of samples.

6 Conclusion

We have presented a novel blackbox synthesis approach to construct a continuous control function for CPS. Our approach works for systems that have to be treated as black boxes because code is either not available or cannot easily be explored analytically. We focus on systems that are expensive to execute, as may be the case for physical or analog systems. Thus, we attempt to minimize the number of executions of the system needed to synthesize a correct controller.

Our algorithm employs a linear interpolation model based on a Delaunay triangulation to identify candidate control functions. It then generates additional test cases to either confirm a candidate or to improve the model. In case good

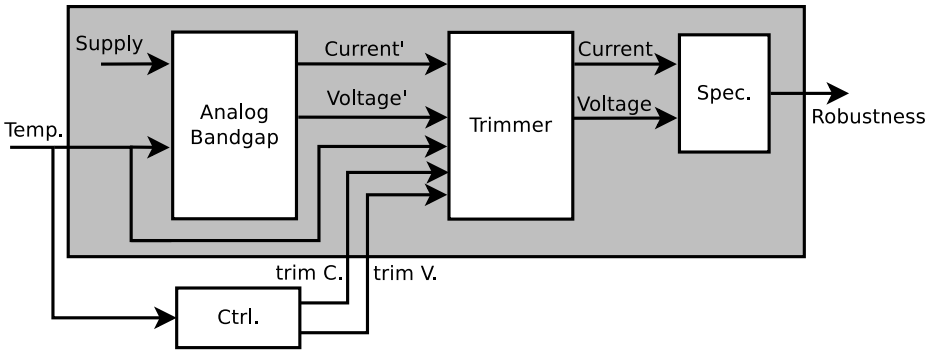


Fig. 12: Schematic of the industrial bandgap case study.

Table 1: Samples required to solve the synthesis problems. DFS is the greedy baseline, OPT is the min-max optimizer from [2], and TRI is the triangulation-based approach presented in this paper. The table reports the median of 5 runs.

Problem	DFS	OPT	**TRI**
Robot arm control	717	— (>2500)	**310**
Multi-agent multi-obstacle motion planning	— (>2500)	— (>2500)	**295**
Multi-obstacle motion planning	— (>2500)	1020	**50**
Multi-agent single obstacle motion planning	— (>2500)	— (>2500)	**346**
Light control	1578	**26**	68
Beach line	— (>2500)	— (>2500)	**34**
Bandgap temperature trimming v1	— (>1000)	— (>1000)	**159**
Bandgap temperature trimming v2	— (>1000)	— (>1000)	**268**
Bandgap temperature trimming v3	— (>1000)	— (>1000)	**152**
Bandgap temperature trimming v4	— (>1000)	— (>1000)	**147**
Bandgap temperature trimming v5	— (>1000)	— (>1000)	**99**

candidates cannot be found, exploratory testing yields more global information about the CPS.

Our algorithm constructs a controller that guarantees correctness for a given Lipschitz bound on the system. This bound may be either known or may be learned during synthesis with a given confidence. Thus, given enough time, we can construct controllers that are correct with arbitrary confidence. The Lipschitz bound limits the amount of testing that needs to be done. The algorithm is guaranteed to terminate with a solution if there is a control function with a quality of at least 2ϵ. We have shown the effectiveness of our approach on random benchmarks and CPS examples.

References

1. Akimoto, Y.: Saddle point optimization with approximate minimization oracle. In: Chicano, F., Krawiec, K. (eds.) GECCO '21: Genetic and Evolutionary Computation Conference, Lille, France, July 10-14, 2021. pp. 493–501. ACM (2021). https://doi.org/10.1145/3449639.3459266
2. Akimoto, Y., Miyauchi, Y., Maki, A.: Saddle point optimization with approximate minimization oracle and its application to robust berthing control. ACM Trans. Evol. Learn. Optim. **2**(1), 2:1–2:32 (2022). https://doi.org/10.1145/3510425
3. Alur, R., Bodík, R., Juniwal, G., Martin, M.M.K., Raghothaman, M., Seshia, S.A., Singh, R., Solar-Lezama, A., Torlak, E., Udupa, A.: Syntax-guided synthesis. In: Formal Methods in Computer-Aided Design, FMCAD 2013, Portland, OR, USA, October 20-23, 2013. pp. 1–8. IEEE (2013)
4. Bemporad, A.: Global optimization via inverse distance weighting and radial basis functions. Computational Optimization and Applications **77**(2), 571–595 (2020). https://doi.org/10.1007/S10589-020-00215-W
5. Bloem, R., Chatterjee, K., Jobstmann, B.: Graph games and reactive synthesis. In: Clarke, E.M., Henzinger, T.A., Veith, H., Bloem, R. (eds.) Handbook of Model Checking, pp. 921–962. Springer (2018). https://doi.org/10.1007/978-3-319-10575-8_27
6. Bloem, R., Larrauri, A., Lengfeldner, R., Mateis, C., Nickovic, D., Ziegler, B.: Industry paper: Surrogate models for testing analog designs under limited budget - a bandgap case study. In: International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS 2022, Shanghai, China, October 7-14, 2022. pp. 21–24. IEEE (2022). https://doi.org/10.1109/CODES-ISSS55005.2022.00016
7. Corso, A., Moss, R.J., Koren, M., Lee, R., Kochenderfer, M.J.: A survey of algorithms for black-box safety validation of cyber-physical systems. J. Artif. Intell. Res. **72**, 377–428 (2021). https://doi.org/10.1613/JAIR.1.12716
8. Dang, T., Donzé, A., Maler, O., Shalev, N.: Sensitive state-space exploration. In: CDC. pp. 4049–4054. IEEE (2008)
9. Deshmukh, J., Horvat, M., Jin, X., Majumdar, R., Prabhu, V.S.: Testing cyber-physical systems through bayesian optimization. ACM Transactions on Embedded Computing Systems (TECS) **16**(5s), 1–18 (2017). https://doi.org/10.1145/3126521
10. Donzé, A., Maler, O.: Robust satisfaction of temporal logic over real-valued signals. In: Chatterjee, K., Henzinger, T.A. (eds.) Formal Modeling and Analysis of Timed Systems - 8th International Conference, FORMATS 2010, Klosterneuburg, Austria, September 8-10, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6246, pp. 92–106. Springer (2010). https://doi.org/10.1007/978-3-642-15297-9_9
11. Eddeland, J.L., Donzé, A., Åkesson, K.: Multi-requirement testing using focused falsification. In: Proc. of HSCC '22: 25th ACM International Conference on Hybrid Systems: Computation and Control. pp. 4:1–4:11. ACM (2022). https://doi.org/10.1145/3501710
12. Edo, H., Miyauchi, Y., Maki, A., Akimoto, Y.: Trade-off between robustness and worst-case performance in min-max optimization. In: GECCO. pp. 1339–1347. ACM (2023). https://doi.org/10.1145/3583131.3590362
13. Ernst, G., Sedwards, S., Zhang, Z., Hasuo, I.: Falsification of hybrid systems using adaptive probabilistic search. ACM Trans. Model. Comput. Simul. **31**(3), 18:1–18:22 (2021). https://doi.org/10.1145/3459605
14. Fainekos, G.E., Pappas, G.J.: Robustness of temporal logic specifications for continuous-time signals. Theor. Comput. Sci. **410**(42), 4262–4291 (2009). https://doi.org/10.1016/J.TCS.2009.06.021

15. Fazlyab, M., Robey, A., Hassani, H., Morari, M., Pappas, G.J.: Efficient and accurate estimation of lipschitz constants for deep neural networks. In: NeurIPS. pp. 11423–11434 (2019)
16. Girard, A., Pappas, G.J.: Verification using simulation. In: HSCC. Lecture Notes in Computer Science, vol. 3927, pp. 272–286. Springer (2006). https://doi.org/10.1007/11730637_22
17. Gruenbacher, S.A., Lechner, M., Hasani, R.M., Rus, D., Henzinger, T.A., Smolka, S.A., Grosu, R.: Gotube: Scalable statistical verification of continuous-depth models. In: Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022. pp. 6755–6764. AAAI Press (2022). https://doi.org/10.1609/AAAI.V36I6.20631
18. Gulwani, S., Polozov, O., Singh, R., et al.: Program synthesis. Foundations and Trends® in Programming Languages **4**(1-2), 1–119 (2017)
19. Huang, J.W., Roberts, S.J., Calliess, J.P.: On the sample complexity of lipschitz constant estimation. Transactions on Machine Learning Research (2023)
20. Kaible, V., Peinhardt, M.A.: On the bottleneck shortest path problem. ZIB-Report (05 2006)
21. Kitzelmann, E.: Inductive programming: A survey of program synthesis techniques. In: Schmid, U., Kitzelmann, E., Plasmeijer, R. (eds.) Approaches and Applications of Inductive Programming, Third International Workshop, AAIP 2009, Edinburgh, UK, September 4, 2009. Revised Papers. Lecture Notes in Computer Science, vol. 5812, pp. 50–73. Springer (2009). https://doi.org/10.1007/978-3-642-11931-6_3
22. Liu, S., Lu, S., Chen, X., Feng, Y., Xu, K., Al-Dujaili, A., Hong, M., O'Reilly, U.: Min-max optimization without gradients: Convergence and applications to black-box evasion and poisoning attacks. In: ICML. Proceedings of Machine Learning Research, vol. 119, pp. 6282–6293. PMLR (2020)
23. Maler, O., Nickovic, D.: Monitoring properties of analog and mixed-signal circuits. Int. J. Softw. Tools Technol. Transf. **15**(3), 247–268 (2013). https://doi.org/10.1007/S10009-012-0247-9
24. Marzat, J., Walter, E., Piet-Lahanier, H.: Worst-case global optimization of black-box functions through kriging and relaxation. J. Glob. Optim. **55**(4), 707–727 (2013). https://doi.org/10.1007/S10898-012-9899-Y
25. Nghiem, T., Sankaranarayanan, S., Fainekos, G., Ivancic, F., Gupta, A., Pappas, G.J.: Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems. In: Proc. of HSCC 2010: the 13th ACM International Conference on Hybrid Systems: Computation and Control. pp. 211–220. ACM (2010). https://doi.org/10.1145/1755952.1755983
26. Perlin, K.: An image synthesizer. ACM SIGGRAPH Computer Graphics **19**(3), 287–296 (Jul 1985). https://doi.org/10.1145/325165.325247
27. Seeger, M.: Gaussian processes for machine learning. International journal of neural systems **14**(02), 69–106 (2004). https://doi.org/10.1142/S0129065704001899
28. Sibson, R.: Locally equiangular triangulations. The Computer Journal **21**(3), 243–245 (01 1978). https://doi.org/10.1093/comjnl/21.3.243
29. Strongin, R.: On the convergence of an algorithm for finding a global extremum. Eng. Cybernetics **11**, 549–555 (1973)
30. Teel, A., Praly, L.: Tools for semiglobal stabilization by partial state and output feedback. Siam J. Control Optim. (1995)

31. Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., SciPy 1.0 Contributors: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods **17**, 261–272 (2020). https://doi.org/10.1038/s41592-019-0686-2
32. Watson, D., Philip, G.: Systematic triangulations. Computer vision, graphics, and image processing **26**(2), 217–223 (1984). https://doi.org/10.1016/0734-189X(84)90184-1
33. Wood, G.R., Zhang, B.P.: Estimation of the lipschitz constant of a function. J. Glob. Optim. **8**(1), 91–103 (1996). https://doi.org/10.1007/BF00229304
34. Zhang, Z., Arcaini, P.: Gaussian process-based confidence estimation for hybrid system falsification. In: International Symposium on Formal Methods. pp. 330–348. Springer (2021). https://doi.org/10.1007/978-3-030-90870-6_18