Smooth Operator: Control Using the Smooth Robustness of Temporal Logic

Yash Vardhan Pant*, Houssam Abbas*, Rahul Mangharam

***Abstract*— Cyber-Physical Systems must withstand a wide range of errors, from bugs in their software to attacks on their physical sensors. Given a formal specification of their desired behavior in Metric Temporal Logic (MTL), the robust semantics of the specification provides a notion of *system robustness* that can be calculated directly on the output behavior of the system, without explicit reference to the various sources or models of the errors. The robustness of the MTL specification has been used both to verify the system offline (via robustness minimization) and to control the system online (to maximize its robustness over some horizon). Unfortunately, the robustness objective function is difficult to work with: it is recursively defined, non-convex and non-differentiable. In this paper, we propose smooth approximations of the robustness. Such approximations are differentiable, thus enabling us to use powerful off-the-shelf gradient descent algorithms for optimizing it. By using them we can also offer guarantees on the performance of the optimization in terms of convergence to minima. We show that the approximation error is bounded to any desired level, and that the approximation can be tuned to the specification. We demonstrate the use of the smooth robustness to control two quad-rotors in an autonomous air traffic control scenario, and for temperature control of a building for comfort.**

I. Controlling for robustness

The errors in Cyber Physical Systems (CPS) can affect both the cyber components (e.g., software bugs) and physical components (e.g., sensor failures and attacks) of a system. Under certain error models, like a bounded disturbance on a sensor reading, a CPS can be designed to be robust to that source of error. In general, however, unforeseen issues will occur. To deal with unforeseen problems, at design time, the system must be verified to be *robust* : i.e., not only does it satisfy its design specifications under the known error models, it must satisfy them robustly. Similarly, at runtime, the system's controller must make decisions that maximize this satisfaction margin, or *robustness*. This can give a margin of maneuvarability to the system during which it addresses the unforeseen problem. Since these problems are, by definition, unforeseen and unmodeled and only detected by their effect on the output, the notion of robustness must be computable using only the output behavior of the system.

Example 1: Air-Traffic Control (ATC) coordinates landing arrivals at an airport. ATCs have very complex rules to ensure that all airplanes, of different sizes and speeds, approach the airport and land safely, *with sufficient margin to other airplanes* to accommodate emergencies and wind gusts. Fig. 1 depicts the airspace of Chicago O'Hare (ORD), the third busiest airport in the U.S. The arrival airspace is divided into 3 zones with different, hierarchical, altitude floors and ceilings. It also shows holding zones, landing approaches, and allowable trajectories for the landing. The following is a subset of the rules that apply for incoming air-crafts.

1) When an aircraft enters one of the zones (indicated by the numbers 1,2,3 in Fig. 1), it must stay between that zone's altitude floor and ceiling.
2) If an aircraft approaches from the West, it must follow one of the trajectories numbered 4 or 5.
3) If the air-space is too busy, an aircraft must maintain a holding pattern in either holding zones 6 or 7, until some maximum amount of time expires.
4) A minimum separation must be maintained between aircrafts.

How do we ensure that the ATC system satisfies these complex rules *robustly*?

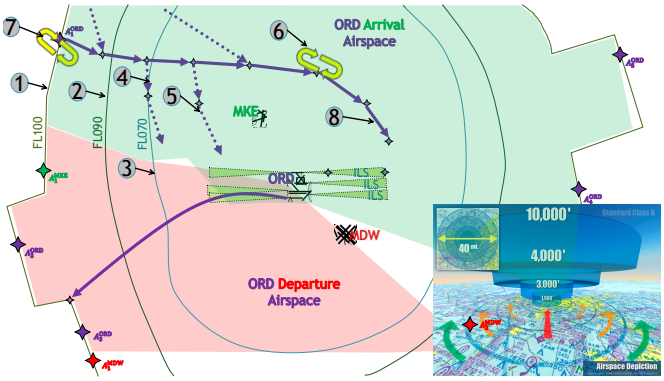

Fig. 1: A simplified depiction of Chicago O'Hare (ORD) airport and its airspace (40 miles radius). The numbers indicate when and where different landing rules apply. Sub-figure on the right shows the altitude rules, see text. Figure courtesy of Max Z. Li, University of Pennsylvania.

A. The need for temporal logic

The above requirements go beyond traditional control objectives like stability, tracking, quadratic cost optimization and reach-while-avoid for which we have well-developed theory. While these requirements can be directly encoded from natural language into a Mixed Integer Program (MIP), such direct encoding can easily have thousands of variables, as will be seen in the experiments. Thus it is error-prone and checking that it corresponds to the designer's intent is near impossible. On the other hand, such control requirements are easily and succinctly expressed in Metric Temporal

*The authors contributed equally.
Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, USA.
{yashpant, habbas, rahulm}@seas.upenn.edu
This work was supported by STARnet a Semiconductor Research Corporation program sponsored by MARCO and DARPA, NSF MRI-0923518 and the US Department of Transportation University Transportation Center Program.

Logic (MTL) [15], [20]. MTL is a formal language for expressing reactive requirements with constraints on their time of occurence and sequencing, such as those of the ATC. The advantage of first expressing them as MTL formulas is that such formulas are more succinct and legible, and less error-prone, than the corresponding MIP. In this sense, MTL bridges the gap between the ease of use of natural language and the rigor of mathematical formulation.

Example1, continued. *The ATC rules can be formalized in Metric Temporal Logic (MTL). Rule 1 can be formalized as follows ($\Box$ means 'Always', q is an aircraft and q_z is its altitude).*

$$\Box(q \in Zone1 \implies q_z \leq \text{Ceiling1} \wedge q_z \geq \text{Floor1}) \quad (1)$$

Rule 3 can be formalized as follows.

$$\Box(Busy \implies \Diamond_{[t_1,t_2]}(q \in \text{Holding-6} \vee q \in \text{Holding-7}) \mathcal{U}_{[0,\text{MaxHolding}]} \neg Busy) \quad (2)$$

This says that Always ($\Box$), if airport is Busy, then Eventually ($\Diamond$), sometime between times t_1 and t_2, the plane goes into a holding area. It stays there Until the airport is not ($\neg$) busy, or the timer expires at time MaxHolding.

By maximizing the robusness of these MTL specifications, the ATC can automatically find landing patterns that leave room for maneuvering in case of emergencies. Any unforeseen disturbance smaller than a known bounded size will not violate the rules, and will not lead to an unsafe situation.

Once the requirements are expressed as an MTL formula, there are broadly two ways of designing a controller that satisfies the formula (fulfills the requirements). The first method automatically creates a MIP from the semantics of the MTL formula and solves the MIP to yield a satisfying control sequence [23], [24]. See Related Work and the Experiments. The second method, upon which we build in this work, uses the *robustness of MTL specifications* [12], [8] . Robustness is a rigorous notion that has been used successfully for the testing and verification of automotive systems [11], [9], medical devices [26], and general CPS. Given a specification φ written in Metric Temporal Logic (MTL) and a system execution $\mathbf{x}$, the robustness $\rho_\varphi(\mathbf{x})$ of the spec relative to $\mathbf{x}$ measures two things: its sign tells whether $\mathbf{x}$ satisfies the spec ($\rho_\varphi(\mathbf{x}) > 0$) or falsifies it (i.e., violates it, $\rho_\varphi(\mathbf{x}) < 0$). Its magnitude $|\rho_\varphi(\mathbf{x})|$ measures how *robustly* the spec is satisfied or falsified. Namely, any perturbation to $\mathbf{x}$ of size less than $|\rho_\varphi(\mathbf{x})|$ will not cause its truth value to change relative to φ. Thus, the control algorithm can *maximize* the robustness over all possible control actions to determine the next control input.

Unfortunately, the robustness function ρ_φ is hard to work with. In particular, it is non-convex, so optimization cannot be guaranteed to yield global optima. And it is non-differentiable, so we have to resort to heuristics or costly non-smooth optimizers. This makes its optimization a challenge - indeed, most existing approaches treat it as a black box and apply heuristics to its optimization (see Related Work below). These heuristics provide little to no guarantees, have too many user-set parameters, and don't have rigorous termination criteria. On the other hand, *gradient descent optimization* algorithms typically offer convergence guarantees to the function's (local) minima, have known convergence rates for certain function classes, and have been optimized so they outperform heuristics that don't have access to the objective's gradient information. The existence of a gradient also allows us to do *local* search for falsifying trajectories, which is necessary for corner case bugs or dangerous situations. Moreover, gradient descent algorithms usually have a fewer number of parameters to be set, and important issues like step-size selection are rigorously addressed.

Contributions. In this paper, we present smooth (infinitely differentiable) approximations to the robustness function of arbitrary MTL formulae. We show that the smooth approximation is always within a user-defined error of the true robustness, and illustrate the result experimentally. This allows us to run powerful and rigorous off-the-shelf gradient descent optimizers. Through multiple examples, we show that our method is faster and performs better than heuristics like Simulated Annealing optimizing the original, non-differentiable robustness and also better than the MIP-based approaches used in the tool BluSTL [23] and subsequent work. We demonstrate the results on a case study for an autonomous airport traffic controller for two quad-rotors, where the MIP-based approach fails to yield a satisfying controller. While we do not tackle the non-convexity issue directly, having an inexpensive gradient optimizer makes it possible to run an efficient multi-start optimization, increasing the chances of approaching the global optimum.

Related work. Current approaches to optimizing the robustness fall into four categories: the use of heuristics like Simulated Annealing and RRTs [19], [2], [25], [9], [6], non-smooth optimization [3], Mixed Integer Linear Programming (MILP) [23], and iterative approximations [1], [4]. Black-box heuristics are the most commonly used approach: for example, Simulated Annealing [19], cross-entropy [25] and RRTs [9]. The clear advantage of these methods is that they do not require any special form of the objective function: they simply need to evaluate it at various points of the search space, and use its value as feedback to decide on the next point to try. A significant shortcoming is that, unlike gradient descent optimization, they offer little to no guarantees of convergence to local minima, and their convergence rates are often not known. They also use many 'magical' parameters that are heuristically set and may affect the results significantly, thus requiring more user interaction than desired. Because the robustness is non-smooth, the work in [3] developed an algorithm that decreases the objective function along its sub-gradient. This involved a series of conservative approximations, and was restricted to the case of safety formulae. In [23], the authors encoded the MTL formula as a set of linear and boolean constraints (when the dynamical system is linear), and used Gurobi to solve them. MILPs are NP-complete, non-convex, and do not scale well with the number of variables. The sophisticated heuristics used to mitigate this make it hard to characterize their runtimes, which is important in control - see examples in [23] and this paper. In general, MILP solvers can only guarantee achieving

local optima. Note also that [23] requires all constraints to be linear, so all atomic propositions must involve half-spaces ($p : a'x \leq b$), which is not a restriction we need. The work closest to ours is [1], [4]. There, the authors considered safety formulas, for which the robustness reduces to the minimum distance between $\mathbf{x}$ and the unsafe set U. By sub-optimally focusing on one point on the trajectory $\mathbf{x}$, they replaced the objective by a differentiable indicator function for U and solved the resulting problem with gradient descent.

By computing fast smooth approximations of robustness, we circumvent most of the above issues and get closer to real-time control by robustness maximization.

II. Robustness of MTL formulae

Consider a discrete-time dynamical system $\mathcal{H}$ given by

$$x_{t+1} = f(x_t, u_t) \tag{3}$$

where $x \in X \subset \mathbb{R}^n$ is the state of the system and $u \in U \subset \mathbb{R}^m$ is its control input. The system's initial state x_0 takes value from some initial set $X_0 \subset \mathbb{R}^n$. Given an initial state x_0 and a control input sequence $\mathbf{u} = (u_0, u_1, \ldots), u_t \in U$, a trajectory of the system is the unique sequence of states $\mathbf{x} = (x_0, x_1, \ldots)$ s.t. for all t, x_t is in X and it obeys (3) at every time step. We will use $\mathbb{T}$ to abbreviate the time domain $\{0, 1, 2, \ldots\}$. All temporal intervals that appear in this paper are discrete-time, e.g. $[a, b]$ means $[a, b] \cap \mathbb{T}$. For an interval I, we write $t + I = \{t + a \mid a \in I\}$. The set of subsets of a set S is denoted $\mathcal{P}(S)$. The signal space $X^{\mathbb{T}}$ is the set of all signals $\mathbf{x} : \mathbb{T} \to X$. The max operator is written $\sqcup$ and min is written $\sqcap$.

A. Metric Temporal Logic

The controller of $\mathcal{H}$ is designed to make the closed loop system (3) satisfy a specification expressed in Metric Temporal Logic (MTL) [20]. MTL allows one to formally express complex reactive specifications, beyond stability, trajectory tracking and the like. See examples (1) and (2).

Formally, let AP be a set of atomic propositions, which can be thought of as point-wise constraints on the state of the system. An MTL formula φ is built recursively from the atomic propositions using the following grammar:

$$\varphi := \top | p | \neg\varphi | \varphi_1 \vee \varphi_2 | \varphi_1 \wedge \varphi_2 | \varphi_1 \mathcal{U}_I \varphi_2$$

where $I \subset \mathbb{R}$ is a time interval. Here, $\top$ is the Boolean True, p is an atomic proposition, $\neg$ is Boolean negation, $\vee$ and $\wedge$ are the Boolean OR and AND operators, respectively, and $\mathcal{U}$ is the Until temporal operator. Informally, $\varphi_1 \mathcal{U}_I \varphi_2$ means that φ_1 must hold *until* φ_2 holds, and that the hand-over from φ_1 to φ_2 must happen sometime during the interval I. The implication ($\implies$), Always ($\Box$) and Eventually ($\Diamond$) operators can be derived using the above operators.

Formally, the *pointwise semantics* of an MTL formula define what it means for a system trajectory $\mathbf{x}$ to satisfy the formula φ. Let $\mathcal{O} : AP \to \mathcal{P}(X)$ be an *observation* map for the atomic propositions. The boolean truth value of a formula φ w.r.t. the trajectory $\mathbf{x}$ at time t is defined recursively.

Definition 2.1 (MTL semantics):

$$\begin{aligned}
(\mathbf{x}, t) &\models \top \quad \Leftrightarrow \quad \top \\
\forall p \in AP, (\mathbf{x}, t) &\models_{\mathcal{O}} p \quad \Leftrightarrow \quad x_t \in \mathcal{O}(p) \\
(\mathbf{x}, t) &\models_{\mathcal{O}} \neg\varphi \quad \Leftrightarrow \quad \neg(\mathbf{x}, t) \models_{\mathcal{O}} \varphi \\
(\mathbf{x}, t) &\models_{\mathcal{O}} \varphi_1 \vee \varphi_2 \quad \Leftrightarrow \quad (\mathbf{x}, t) \models_{\mathcal{O}} \varphi_1 \vee (\mathbf{x}, t) \models_{\mathcal{O}} \varphi_2 \\
(\mathbf{x}, t) &\models_{\mathcal{O}} \varphi_1 \wedge \varphi_2 \quad \Leftrightarrow \quad (\mathbf{x}, t) \models_{\mathcal{O}} \varphi_1 \wedge (\mathbf{x}, t) \models_{\mathcal{O}} \varphi_2 \\
(\mathbf{x}, t) &\models_{\mathcal{O}} \varphi_1 \mathcal{U}_I \varphi_2 \quad \Leftrightarrow \quad \exists t' \in t + I . (\mathbf{x}, t') \models_{\mathcal{O}} \varphi_2 \\
&\qquad\qquad \wedge \forall t'' \in (t, t'), \ (\mathbf{x}, t'') \models_{\mathcal{O}} \varphi_1
\end{aligned}$$

As $\mathcal{O}$ is fixed in this paper, we drop it from the notation. We say $\mathbf{x}$ satisfies φ if $(\mathbf{x}, 0) \models \varphi$. All formulas that appear in this paper have bounded temporal intervals: $0 \leq \inf I < \sup I < +\infty$. To evaluate whether such a formula φ holds on a given trajectory, only a finite-length prefix of that trajectory is needed. Its length can be upper-bounded by the *horizon* of φ, $hrz(\varphi)$, calculable as shown in [7]. For example, the horizon of $\Diamond_{[2,4]} p$ is 6.

B. Robust semantics of MTL

Designing a controller that satisfies the MTL formula φ[1] is not always enough. In a dynamic environment, where the system must react to new unforeseen events, it is useful to have a margin of maneuverability. That is, it is useful to control the system such that we *maximize* our degree of satisfaction of the formula. When unforeseen events occur, the system can react to them without violating the formula. This degree of satisfaction can be formally defined and computed using the robust semantics of MTL. Given a point $x \in X$ and a set $A \subset X$, $\mathbf{dist}(x, A) := \inf_{a \in \overline{A}} |x - a|_2$ is the distance from x to the closure $\overline{A}$ of A.

Definition 2.2 (Robustness[10]): The *robustness* of φ relative to $\mathbf{x}$ at time t is recursively defined as

$$\begin{aligned}
\rho_\top(\mathbf{x}, t) &= +\infty \\
\forall p \in AP, \ \rho_p(\mathbf{x}, t) &= \begin{cases} \mathbf{dist}(x_t, X \setminus \mathcal{O}(p)), & \text{if } x_t \in \mathcal{O}(p) \\ -\mathbf{dist}(x_t, \mathcal{O}(p)), & \text{if } x_t \notin \mathcal{O}(p) \end{cases} \\
\rho_{\neg\varphi}(\mathbf{x}, t) &= -\rho_\varphi(\mathbf{x}, t) \\
\rho_{\varphi_1 \vee \varphi_2}(\mathbf{x}, t) &= \rho_{\varphi_1}(\mathbf{x}, t) \sqcup \rho_{\varphi_2}(\mathbf{x}, t) \\
\rho_{\varphi_1 \wedge \varphi_2}(\mathbf{x}, t) &= \rho_{\varphi_1}(\mathbf{x}, t) \sqcap \rho_{\varphi_2}(\mathbf{x}, t) \\
\rho_{\varphi_1 \mathcal{U}_I \varphi_2}(\mathbf{x}, t) &= \sqcup_{t' \in t + \mathbb{T} I} \Big(\rho_{\varphi_2}(\mathbf{x}, t') \bigsqcap \\
&\qquad \sqcap_{t'' \in [t, t')} \rho_{\varphi_1}(\mathbf{x}, t'') \Big)
\end{aligned}$$

When $t = 0$, we write $\rho_\varphi(\mathbf{x})$ instead of $\rho_\varphi(\mathbf{x}, 0)$.
The robustness is a real-valued function of $\mathbf{x}$ with the following important property.

Theorem 2.1: [10] For any $\mathbf{x} \in X^{\mathbb{T}}$ and MTL formula φ, if $\rho_\varphi(\mathbf{x}, t) < 0$ then $\mathbf{x}$ falsifies the spec φ at time t, and if $\rho_\varphi(\mathbf{x}, t) > 0$ then $\mathbf{x}$ satisfies φ at t. The case $\rho_\varphi(\mathbf{x}, t) = 0$ is inconclusive.

Thus, we can compute control inputs by maximizing the robustness over the set of finite input sequences of a certain length. The obtained sequence $\mathbf{u}^*$ is valid if $\rho_\varphi(\mathbf{x}, t)$ is positive, where $\mathbf{x}$ and $\mathbf{u}^*$ obey (3). The larger $\rho_\varphi(\mathbf{x}, t)$, the

[1]Strictly speaking, a controller such that the closed-loop satisfies the formula.

more robust is the behavior of the system: intuitively, $\mathbf{x}$ can be disturbed and ρ_φ might decrease but not go negative.

III. Smooth approximation

In this paper, we seek to design control inputs such that the closed-loop system satisfies an MTL specification. Formally, the goal is to solve the following problem. Let φ be an MTL formula with horizon N. We aim to solve the following control problem P_ρ.

$$P_\rho : \max_{\mathbf{u}} \rho_\varphi(\mathbf{x}) - \gamma \sum_{k=0}^{N-1} l(x_{k+1}, u_k) \tag{4a}$$

$$\text{s.t. } x_{k+1} = f(x_k, u_k),\ \forall k = 0, \dots, N-1 \tag{4b}$$

$$x_k \in X,\ \forall k = 0, \dots, N \tag{4c}$$

$$u_k \in U,\ \forall k = 0, \dots, N-1 \tag{4d}$$

$$\delta \rho_\varphi(\mathbf{x}) \geq \delta \epsilon_{\min} \tag{4e}$$

Here, $\mathbf{u} = (u_0, \dots, u_{N-1})$, $l(x_{k+1}, u_k)$ is a control cost, e.g. the LQR cost $x_k' Q x_k + u_k' R u_k$, and $\gamma \geq 0$ is a trade-off weight. The scalar $\epsilon_{\min} \geq 0$ is a desired minimum robustness. If $\delta = 0$, then this constraint is effectively removed, while $\delta = 1$ enforces the constraint.

A. The need for smoothing

To apply gradient descent methods, we require a differentiable objective function. Our objective function, ρ_φ, is non-differentiable, because it uses the distance, max, and min functions, all of which are non-differentiable. One may note that these functions are all differentiable almost everywhere (a.e.) on their domain. That is, the set of points in their domain where they are non-differentiable has measure 0 in $\mathbb{R}^n$. Therefore, by measure additivity, the composite function ρ_φ is itself differentiable almost everywhere. Thus, one may be tempted to 'ignore' the singularities (points of non-differentiability), and apply gradient descent to ρ_φ anyway. The rationale for doing so is that sets of measure 0 are unlikely to be visited by gradient descent, and thus don't matter. However, as we show in the next example, the lines of singularity (along which the objective is non-differentiable) can be precisely the lines along which the objective increases the fastest. See also [5]. Thus they are consistently visited by gradient descent, after which it fails to converge because of the lack of a gradient.

Example 2: A simple example illustrates how gradient descent gets stuck at singularities. We use the optimization algorithm Sequential Quadratic Programming (SQP) [22] to maximize the robustness of $\varphi = \neg(x \in U)$, where $U = [-1, 1]^2$ is the unsafe red square in Fig. 2. In this case, ρ_φ is simply $\mathbf{dist}(x_0, U)$, the distance of the first trajectory point to the set. The search space is $[-2.5, 2.5]^2$ (big grey square in Fig. 2). The most robust points are the corners of the grey square, such as $x^* = [2.5, 2.5]$ (green '+' in figure), being furthest from the unsafe set. We initialize the SQP at $x_0 = [0, 0]$. SQP generates iterates (blue circles) *on the line of singularity* connecting $[1, 1]$ to x^* and ultimately gets stuck at $x = [1, 1]$. That's because along the line, the gradient does not exist and attempts by SQP to approximate it numerically fail, prompting it to generate smaller and smaller step-sizes for the approximation. Ultimately, SQP aborts due to the step-size being too small, and concludes it is at a local minimum.

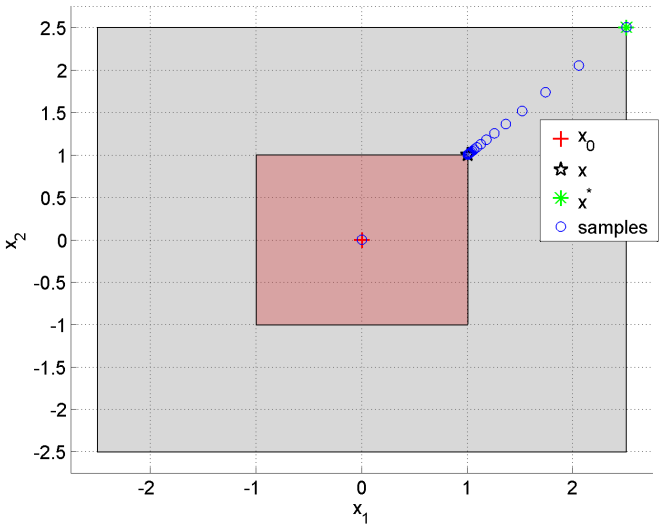

Fig. 2: Iterates of SQP for Example 2. Colors in online version.

B. Approximating the distance function

To create a smooth approximation to ρ_φ, we use smooth approximations to each of its non-differentiable components: the set-distance, min, and max functions.

Recall that for a set $U \subset \mathbb{R}^n$, $\mathbf{dist}(x, U) = \inf_{a \in \overline{U}} |x - a|_2$, where $|\cdot|_2$ is the Euclidian norm. This function is globally Lipschitz with Lipschitz constant 1 and therefore differentiable almost everywhere (Rademacher's theorem), and has a second derivative almost everywhere if U is convex (Alexandrov's theorem) [17].

It is well-known that if an a.e.-differentiable function is convolved with a smooth kernel, the output function no longer has those singularities. We leverage this to provide a smooth approximation of the distance function by expanding it over a basis of orthonormal Meyer wavelets. Specifically, the 1-D Meyer wavelet function is given in the frequency domain by ($i = \sqrt{-1}$):

$$\widehat{\psi}(\omega) = \frac{1}{\sqrt{2\pi}} \begin{cases} \sin(\frac{\pi}{2}\nu(\frac{3|\omega|}{2\pi} - 1))e^{i\omega/2} & 2\pi/3 \leq |\omega| \leq 4\pi/3 \\ \cos(\frac{\pi}{2}\nu(\frac{3|\omega|}{4\pi} - 1))e^{i\omega/2}, & 4\pi/3 \leq |\omega| \leq 8\pi/3 \\ 0, & \text{otherwise} \end{cases}$$

where $\nu(x) = 0$ if $x \leq 0$, 1 if $x \geq 1$, and equals x if $0 \leq x \leq 1$. The time-domain expression for this wavelet is given in [28]. To obtain an n-D wavelet, we use the tensor product construction [18]. Let E be the set of vertices of the unit hypercube $[0, 1]^n$. For every $e = (e_1, e_2, \dots, e_n) \in E$ and $x = (x_1, \dots, x_n) \in \mathbb{R}^n$, define $\Psi^e : \mathbb{R}^n \to \mathbb{R}$

$$\Psi^e(x) = \psi^{e_1}(x_1) \dots \psi^{e_n}(x_n)$$

Given $k \in \mathbb{Z}$ and $j \in \mathbb{Z}^n$, a *dyadic cube* in $\mathbb{R}^n$ is a set of the form $I = 2^{-k}(j + [0, 1]^n)$. Let D be the set of all dyadic cubes in $\mathbb{R}^n$ obtained by varying k over $\mathbb{Z}$ and j over $\mathbb{Z}^n$. Then $\{\Psi_I^e, e \in E, I \in D\}$ is an orthonormal basis for

$L_2(\mathbb{R}^n)$ (because the Meyer wavelet itself is orthonormal). Then every function in $L_2(\mathbb{R}^n)$ has an expansion

$$f(x) = \sum_{I\in D}\sum_{e\in E} c_I^e \Psi_I^e(x),\ c_I^e := \langle f, \Psi_I^e\rangle$$

with $\langle h, g\rangle := \int_{\mathbb{R}^n} f(x)g(x)dx$. The desired approximation is obtained by truncating this expansion after a finite number of terms, i.e., by using a *finite* set $D' \subsetneq D$

$$\mathbf{dist}(x, U) \approx \widetilde{\mathbf{dist}}(x, U) := \sum_{I\in D'}\sum_{e\in E} c_I^e \Psi_I^e(x) \quad (5)$$

The coefficients $c_I^e := \langle \mathbf{dist}(\cdot, U), \Psi_I^e\rangle$ are calculated offline and stored in a lookup table for online usage. Using more coefficients yields a better approximation.

C. Smooth max and min

We use the following standard smooth approximations of m-ary max and min. Let $k \geq 1$.

$$\widetilde{\max}_k(a_1, \ldots, a_m) := \frac{1}{k}\ln(e^{ka_1} + \ldots + e^{ka_m}) \quad (6)$$

$$\widetilde{\min}_k(a_1, \ldots, a_m) := -\widetilde{\max}(-a_1, \ldots, -a_m) \quad (7)$$

Suppose $k = 1$ and that a_1 is the largest number. Then e^{a_1} is even larger than the other e^{a_i}'s, and dominates the sum. Thus $\widetilde{\max}_1(\mathbf{a}) \cong \ln e^{a_1} = a_1 = \max(\mathbf{a})$. If a_1 is not significantly larger than the rest, the sum is not well-approximated by e^{a_1} alone. To counter this, the scaling factor k is used: it amplifies the differences between the numbers. It holds that for any set of m reals,

$$0 \leq \widetilde{\max}_k(a_1, \ldots, a_m) - \max(a_1, \ldots, a_m) \leq \ln(m)/k \quad (8)$$

$$0 \leq \min(a_1, \ldots, a_m) - \widetilde{\min}_k(a_1, \ldots, a_m) \leq \ln(m)/k \quad (9)$$

Indeed, the error of smooth max can be bounded as follows. Assume a_1 is the largest number, then

$$\begin{aligned}\varepsilon_M &:= \widetilde{\max}_k(\mathbf{a}) - a_1 = \frac{\ln(\sum_i e^{ka_i}) - ka_1}{k}\\ &= k^{-1}\ln\left(\frac{\sum_i e^{ka_i}}{e^{ka_1}}\right) \leq k^{-1}\ln\left(\frac{me^{ka_1}}{e^{ka_1}}\right)\\ &= \frac{\ln m}{k}\end{aligned}$$

It is also clear from what preceded that $\varepsilon_M \geq 0$. The maximum error is achieved when all the a_i's are equal.

D. Overall approximation

Putting the pieces together, we obtain the approximation error for the robustness of any MTL formula. Given a set U, let $\widetilde{\mathbf{dist}}_\varepsilon(\cdot, U)$ be an ε-approximation of $\mathbf{dist}(\cdot, U)$, i.e. for all x in their common domain, $|\mathbf{dist}(x, U) - \widetilde{\mathbf{dist}}_\varepsilon(x, U)| \leq \varepsilon$.

Theorem 3.1: Consider an MTL formula φ and reals $\varepsilon > 0$ and $k \geq 1$. Define the *smooth robustness* $\tilde{\rho}_\varphi$, obtained by substituting $\widetilde{\mathbf{dist}}_\varepsilon$ for $\mathbf{dist}$, $\widetilde{\max}_k$ for $\max$, and $\widetilde{\min}_k$ for $\min$, in Def. 2.2. Then for any trajectory $\mathbf{x}$, it holds that

$$|\rho_\varphi - \tilde{\rho}_\varphi| \leq \delta_\varphi$$

where δ_φ is (a) independent of the evaluation time t, and (b) goes to 0 as $\varepsilon \to 0$ and $k \to \infty$.

Proof: We will prove a stronger result that implies the theorem. When $\mathbf{x}$ or t are clear from the context, we will drop them from the notation.

The proof is by structural induction on φ, and works by carefully characterizing the approximation error.

Case $\varphi = p \in AP$. $\rho_\varphi(\mathbf{x}, t)$ is given by either $\mathbf{dist}x_t\mathcal{O}(p)$ or $-\mathbf{dist}x_t\mathcal{O}(p)$, and $\tilde{\rho}_\varphi(\mathbf{x}, t)$ is given by either $\widetilde{\mathbf{dist}}_\varepsilon(x_t, \mathcal{O}(p))$ or $-\widetilde{\mathbf{dist}}_\varepsilon(x_t, \mathcal{O}(p))$, respectively. Either way, $|\tilde{\rho}_\varphi(\mathbf{x}, t) - \rho_\varphi(\mathbf{x}, t)| \leq \varepsilon$. Indeed, ε satisfies the conditions on δ_φ.

Case $\varphi = \neg\varphi_1$ $|\rho_{\neg\varphi_1}(\mathbf{x}, t) - \tilde{\rho}_{\neg\varphi_1}(\mathbf{x}, t)| = |-\rho_{\varphi_1}(\mathbf{x}, t) + \tilde{\rho}_{\varphi_1}(\mathbf{x}, t)| \leq \delta_{\varphi_1}$, and δ_{φ_1} satisfies (a)-(b) by the induction hypothesis.

Case $\varphi = \varphi_1 \vee \varphi_2$. If the same sub-formula φ_i achieves the max for both $\rho_{\varphi_1}(\mathbf{x}, t) \sqcup \rho_{\varphi_2}(\mathbf{x}, t)$ and $\tilde{\rho}_{\varphi_1}(\mathbf{x}, t) \sqcup \tilde{\rho}_{\varphi_2}(\mathbf{x}, t)$, then by induction hypothesis we immediately obtain $|\rho_\varphi(\mathbf{x}, t) - \tilde{\rho}(\mathbf{x}, t)| \leq \delta_{\varphi_i}$.

Otherwise if, say, $\rho_\varphi = \rho_{\varphi_1}$ and $\tilde{\rho}_\varphi = \tilde{\rho}_{\varphi_2}$ then

$$\rho_{\varphi_1} - \delta_{\varphi_1} \leq \tilde{\rho}_{\varphi_1} \leq \tilde{\rho}_{\varphi_2} \implies \rho_{\varphi_1} - \tilde{\rho}_{\varphi_2} \leq \delta_{\varphi_1}$$

Also

$$\tilde{\rho}_{\varphi_2} \leq \rho_{\varphi_2} + \delta_{\varphi_2} \leq \rho_{\varphi_1} + \delta_{\varphi_2} \implies -\delta_{\varphi_2} \leq \rho_{\varphi_1} - \tilde{\rho}_{\varphi_2}$$

Therefore

$$-(\delta_{\varphi_1} \sqcup \delta_{\varphi_2}) \leq \rho_{\varphi_1} - \tilde{\rho}_{\varphi_2} \leq \delta_{\varphi_1} \sqcup \delta_{\varphi_2} \Leftrightarrow |\rho_{\varphi_1} - \tilde{\rho}_{\varphi_2}| \leq \delta_{\varphi_1} \sqcup \delta_{\varphi_2}$$

Similarly, if $\rho_\varphi = \rho_{\varphi_2}$ and $\tilde{\rho}_\varphi = \tilde{\rho}_{\varphi_1}$, we have $|\rho_{\varphi_2} - \tilde{\rho}_{\varphi_1}| \leq \delta_{\varphi_1} \sqcup \delta_{\varphi_2}$. So in all cases,

$$|\rho_{\varphi_1} \sqcup \rho_{\varphi_2} - \tilde{\rho}_{\varphi_1} \sqcup \tilde{\rho}_{\varphi_2}| \leq \delta_{\varphi_1} \sqcup \delta_{\varphi_2}$$

Therefore by the triangle inequality and (8)

$$|\rho_{\varphi_1} \sqcup \rho_{\varphi_2} - \widetilde{\max}_k(\tilde{\rho}_{\varphi_1}, \tilde{\rho}_{\varphi_2})| \leq \delta_{\varphi_1} \sqcup \delta_{\varphi_2} + \ln(2)/k = \delta_\varphi$$

Clearly, δ_φ satisfied (a)-(b).

The case $\varphi_1 \wedge \varphi_2$ is treated similarly.

$\varphi = \varphi_1 \mathcal{U}_I \varphi_2$. Before proving this case, we will need the following lemma, which is provable by induction on n:

Lemma 3.1: If $\varphi = \varphi_1 \wedge \ldots \wedge \varphi_n$ or $\varphi = \varphi_1 \vee \ldots \vee \varphi_n$, $n \geq 2$, then $|\rho_\varphi - \tilde{\rho}_\varphi| \leq \sqcup_{1\leq i\leq n}\delta_{\varphi_i} + \ln(n)/k$.

We now proceed with the proof of the last case. Recall that $\rho_{\varphi_1\mathcal{U}_I\varphi_2}(\mathbf{x}, t) = \sqcup_{t'\in t+_{\mathbb{T}}I}(\rho_{\varphi_2}(\mathbf{x}, t') \sqcap \sqcap_{t''\in[t,t')}\rho_{\varphi_1}(\mathbf{x}, t''))$. Starting with the innermost sub expression $\rho_\psi := \sqcap_{t''\in[t,t')}\rho_{\varphi_1}(\mathbf{x}, t'')$, we have, by Lemma 3.1

$$|\rho_\psi - \tilde{\rho}_\psi| \leq \sqcup_{t''\in[t,t')}\delta_{\varphi_1}^{t''} + \ln(t' - t)/k \quad (10)$$

where $\delta_{\varphi_1}^{t''}$ is the bound for approximating $\rho_{\varphi_1}(\mathbf{x}, t'')$. But δ_φ does not depend on the time at which the formula is evaluated. Therefore the bound in (10) becomes

$$|\rho_\psi - \tilde{\rho}_\psi| \leq \delta_{\varphi_1} + \ln(t' - t)/k \quad (11)$$

To avoid introducing a dependence on time, we further upper-bound by

$$|\rho_\psi - \tilde{\rho}_\psi| \leq \delta_{\varphi_1} + \ln(hrz(\varphi))/k := \delta_\psi$$

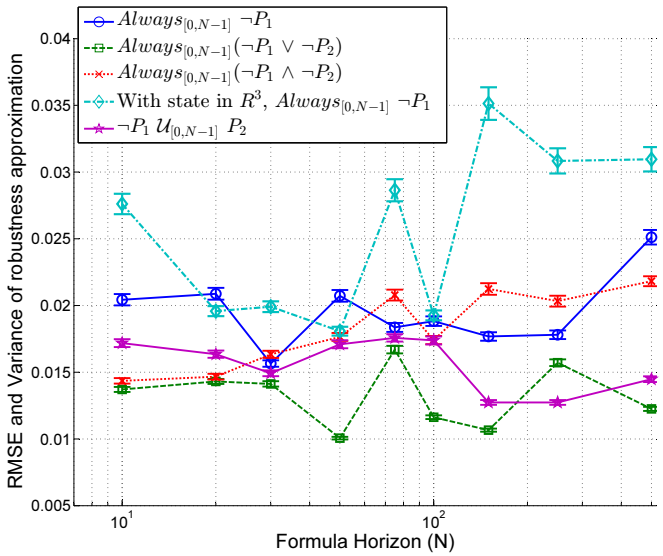

Fig. 3: RMSE and variance of robustness approximation error against formula horizon, evaluated on 1000 randomly generated trajectories for the system in (13). Unless noted, the systems are 2D. Note that the approximation errors and their variances are very small, showing the accuracy and stability of the smooth approximation. Color in online version.

where, recall, $hrz(\varphi)$ is the horizon of φ (see Section II-A).

Continuing with the sub-expression $\rho_\alpha = \rho_{\varphi_2}(\mathbf{x}, t') \bigcap \rho_\psi$, by the induction hypothesis it holds that $|\rho_\alpha - \tilde{\rho}_\alpha| \leq \delta_{\varphi_2} \sqcup \delta_\psi + \ln(2)/k := \delta_\alpha$. Finally, the top-most max operator introduces the total error

$$\begin{aligned} |\rho_\varphi - \tilde{\rho}_\varphi| &\leq \delta_\alpha + \ln(|I|)/k \\ &= \delta_{\varphi_2} \sqcup \delta_\psi + \ln(2)/k + \ln(|I|)/k \\ &= \delta_{\varphi_2} \sqcup (\delta_{\varphi_1} + \ln(hrz(\varphi))/k) + \ln(2|I|)/k \\ &= \delta_\varphi \end{aligned} \tag{12}$$

The first inequality obtains from the fact that δ_α is independent of evaluation time and Lemma 3.1. The bound δ_φ obeys (a)-(b). This concludes the proof. ■

IV. Approximation and Control

We implemented the smooth approximation to the semantics of MTL, and tested it on several examples.

A. Approximation error for robustness

We evaluated the robustness ρ_φ and its approximation $\tilde{\rho}_\varphi$ for five formulae, with $hrz(\varphi_i) = N$. P_1 and P_2 are atomic propositions for state being in two different polyhedra. Each formula's robustness is evaluated on 1000 randomly-generated trajectories of varying lengths N, so we can examine the error's variation with the horizon. The trajectories were produced by a 2-or-3 dimensional system, (13) with input and state saturation. Fig. 3 shows the Root Mean Square (RMSE) of the approximation, $\sqrt{(1/1000)\sum_{\mathbf{x}}(\rho_\varphi(\mathbf{x}) - \tilde{\rho}_\varphi(\mathbf{x}))^2}$, and variance bars around it. As seen, the approximation error is small for all cases. Note that while the RMSE increased with the system dimension (4^{th} formula in Fig. 3), it was observed that the relative error remained very small i.e. the increase in error is explained by an increase in the robustness's magnitude.

B. Robustness maximization for control

To solve Problem P_ρ given in (4), we replace the true robustness ρ_φ by its smooth approximation $\tilde{\rho}_\varphi$. We thus obtain Problem $P_{\tilde{\rho}}$. We call this approach of optimizing the smooth approximation of robustness for control, **Smooth Operator** (SOP). We illustrate the approach on a simple linear system, with a reach-while-avoid type of specification which is common in literature, e.g. in [24]. More extensive case studies are presented in Section V.

Optimization solver. We use Sequential Quadratic Programming (SQP) to solve the optimization problem $P_{\tilde{\rho}}$. SQP solves constrained non-linear optimization problems, like $P_{\tilde{\rho}}$, by creating a sequence of quadratic approximations to the problem and solving these approximate problems. SQP enjoys various convergence-to-(local)-optima properties, depending on the assumptions we place on the problem. See [22, Section 2.9]. For example, for SQP to converge to a strict local minimum (a minimum that is strictly smaller than any objective function value in an open neighborhood around it), it suffices that 1) all constraint functions be twice Lipschitz continuously differentiable. In our case, this includes function f in (4a), and the problems we solve satisfy this requirement. And, 2) at points in the search space that lie on the boundary of the inequality-feasible set (where the inequality constraints are satisfied with equality), there exists a search direction towards the interior of the feasible set that does not violate the equality constraints (the so-called Mangasarian-Fromowitz constraint qualification) [22, Assumption 2.9.1]. This is also true in our case since our equality constraints come from the dynamics and are always enforced for any $\mathbf{u}$.

Solver initialization. To initialize SQP when solving $P_{\tilde{\rho}}$ (i.e., to give it a starting value for $\mathbf{u}$), we can either:

a) solve an inexpensive feasibility linear program with constraints (4b)-(4d).

b) Or, generate a random input sequence respecting $u_k \in U$. Such a trajectory will be very fast to generate and feasible w.r.t the dynamics.

Note, the resulting initial trajectory could violate the specification (as it does in every example we study in this paper) and we only enforce that it satisfy the dynamics and state constraints.

Comparison to BluSTL. We compare the proposed Smooth Operator (SOP) to (among other methods) BluSTL, a state of the art MILP-based approach. BluSTL has two modes of operation: mode (B) or *Boolean*, which aims at satisfying the specification without maximizing its robustness, and mode (R) or *Robust*, which attempts to maximize robustness. The proposed SOP method optimizes robustness and so naturally runs in mode (R). To emulate mode (B), we terminate the optimization as soon as $\tilde{\rho} \geq \epsilon_{\text{Meyer}}$, which in turn implies that $\rho_\varphi \geq 0$.

Since the focus of this paper is on robustness optimization, we set the cost of control to zero for all methods involved

TABLE I: Example 3. Runtimes (mean and standard deviation, in seconds) for Smooth Operator (SOP) and BluSTL (BlS) in modes (B) and (R), over 100 runs with random initial states and different formula horizons N. BluSTL (R) did not finish (see text).

N	BlS(B)	SOP(B)	SOP(R)	SOP(R)
20	0.96 ± 0.82	$\mathbf{0.31 \pm 0.13}$	NA	3.30 ± 1.25
30	1.37 ± 1.72	$\mathbf{0.33 \pm 0.25}$	NA	5.85 ± 2.74
40	3.86 ± 5.10	$\mathbf{0.60 \pm 0.29}$	NA	12.36 ± 6.04
50	4.36 ± 12.97	$\mathbf{0.74 \pm 0.30}$	NA	30.05 ± 18.23
100	16.77 ± 27.84	$\mathbf{1.21 \pm 0.25}$	NA	69.70 ± 13.16
200	53.88 ± 14.18	$\mathbf{4.19 \pm 1.18}$	NA	126.11 ± 20.43

in the examples, unless stated otherwise. This corresponds to $\gamma = 0$ in $P_{\tilde{\rho}}$. The goal of $P_{\tilde{\rho}}$ with $\gamma = 0$ is to find a trajectory that maximizes $\tilde{\rho}$, hence there is no need for the additional lower bound constraint (Eq. 4e) on $\tilde{\rho}$. This is why we also set $\delta = 0$ in $P_{\tilde{\rho}}$ for all examples that follow.

Example 3: We control the following linear system

$$x_{k+1} = x_k + u_k \tag{13}$$

to satisfy the specification

$$\varphi = \Box_{[0,20]} \neg (x \in \text{Unsafe}) \wedge \Diamond_{[0,20]} (x \in \text{Terminal})$$

with the sets Unsafe $= [-1, 1]^2$ and Terminal $= [2, 2.5]^2$. The state space is $X = [-2.5, 2.5]^2$, $U = [-0.5, 0.5]^2$. Unless otherwise indicated, we use $\gamma = 0$ in Eq. 4 to focus on satisfaction in this illustrative example. Experiments were run on a quad-core Intel i5 3.2GHz processor with 24GB RAM, running MATLAB 2016b.

Results. Fig. 4 shows the trajectories of length $N = 20$ obtained by SOP and BluSTL in modes (B) and (R), starting from the same initial point $x_0 = [-2, -2]'$. Both BluSTL (B) and SOP (B) produce satisfying trajectories. The trajectory from SOP (R) ends in the middle of the terminal set, resulting in a higher robustness than mode (B), as expected. In mode (R), BluSTL could not finish a single instance of robustness maximization within 100 hours on both the above machine and on a more powerful 8 core Intel Xeon machine with 60GB RAM, leading us to believe that the corresponding MILP was not tractable.

SOP $(R, \gamma = 0.1)$ takes into account a control cost $l(x_k, u_k) = ||x_k||_2^2$ that penalizes longer trajectories. The resulting trajectory is shorter but has a lower robustness than SOP (R, $\gamma = 0$), (0.236 vs 0.247).

For further evaluation, we ran 100 instances of the problem, varying the trajectory's initial state in $[-2.5, -1.5] \times [-2.5, 2.5]$. We also varied the formula horizon N (and hence the size of the problem) from 20 to 200 time steps. Table I shows the execution times.

Analysis. As seen in table I, SOP is consistently faster than BluSTL for the *boolean* mode, and displays smaller variances in runtimes. Note also that the problem solved here is very similar to the one used in [24], which uses another MILP-based method. While the underlying dynamics differ and their numbers are reported on a more power machine, these results suggest that our runtimes compare favourably with those in [24].

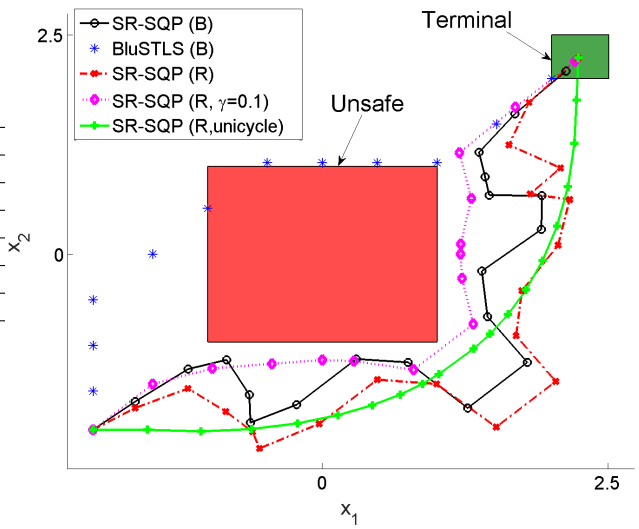

Fig. 4: The first 4 trajectories are for the linear system (13). See Ex. 3. The last trajectory, SOP (R, unicycle), is from the non-linear system in Sec. IV-B.1. Colors in online version.

In Robust mode, across all 100 experiments, SOP results in an average $\rho_\varphi = 0.247$ with a standard deviation less than 0.005. This gets very close to the upper bound on robustness, which is 0.25. This bound is achieved by a trajectory reaching (in less than N time steps) the center of the Terminal set while always staying more than 0.25 distant from Unsafe.

1) A Non-linear example: Unicycle: Since SQP can handle non-linear (twice differentiable) constraints, our method can also deal with non-linear dynamics whereas the MILP-based methods have to linearize dynamics to solve the system. The following example shows SOP applied in a one-shot manner to the unicycle dynamics ($\dot{x}_t = v_t \cos(\theta_t)$, $\dot{y}_t = v_t \sin(\theta_t)$, $\dot{\theta}_t = u_t$) discretized at 10Hz. For the specification of Ex. 3, the resulting trajectory of length 20 steps obtained by SOP (R) (in Robust mode) is shown in fig.4, starting from an initial state of $[-2, -2, 0]$. The resulting robustness is 0.248, which is close to the global optimum of 0.25. This shows that SOP can indeed handle non-linear dynamics without the need for explicit linearization as is done in MILP-based methods.

V. Case Studies

We solve a robustness maximization problem for the control of two systems using the following methods.

- SOP in (B)oolean and (R)obust modes.
- BluSTL in modes (B) and (R).
- R-SQP, which uses SQP to optimize the *exact* non-smoothed robustness ρ_φ.
- SA, which uses Simulated Annealing to optimize ρ_φ.

For both case studies, we compute the wavelet approximation of the distance function to the sets $\mathcal{O}(p)$ off-line. Next, we solve the control problem (4) as a single shot, finite horizon constrained optimization.

The code to reproduce these results can be found at `https://github.com/yashpant/SmoothOperator0`. Future release of the code will

TABLE II: HVAC. Runtimes (mean and std deviation, in seconds) for Smooth Operator (SOP), BluSTL (BlS) and Simulated annealing (SA) over 100 runs with random initial states. BluSTL (R) could not find satisfying trajectories.

BlS (B)	SOP (B)	SOP (R)	SA (R)
0.041 ± 0.002	$\mathbf{0.014 \pm 0.002}$	2.532 ± 0.26	8.56 ± 0.31

focus on re-usability of the code and making it handle any system.

A. HVAC Control of a building for comfort

The first example is the Heating, Ventilation and Air Conditioning (HVAC) control of a 4-state model of a single zone in a building. Such a model is commonly used in literature for evaluation of predictive control algorithms [13]. The control problem we solve is similar to the example used in [23], where the objective is to bring the zone temperature to a comfortable range when the zone is occupied (given predictions on the building occupancy). The specification is:

$$\varphi = \Box_I(\text{ZoneTemp} \in [22, 28]) \tag{14}$$

Here, I is the time interval where the zone is occupied and the range of temperatures (in Celsius) deemed comfortable is ($[22, 28]$). For the control horizon, we consider a 24 hour period, in which the building is occupied from time steps 10 to 19 (i.e. $I = [10, 19]$), i.e. a 10-hour workday.

Note: In this particular problem, the maximum robustness achievable is 3, which can be achieved by setting the room temperature at 25C for the interval I. With this insight, the problem of maximizing robustness can be solved with a quadratic program with linear constraints and the cost $\sum_{k\in I}(x_{4k} - 25)^2$ to be minimized. This indeed results in a trajectory with the global optimal robustness of 3, but is a method tailored to the particular problem.

SOP, which is a general purpose technique, results in a robustness which is just 0.02% less than the global optimal value. In the example that follows this one, we take a specification which cannot be trivially turned into a quadratic program.

System dynamics. The single-zone model, discretized at a sampling rate of 1 hour (which is common in building temperature control) is of the form:

$$x_{k+1} = Ax_k + Bu_k + B_d d_k \tag{15}$$

Here, A, B and B_d matrices are from the hamlab ISE model [27]. $x \in \mathbb{R}^4$ is the state of the model, the 4^{th} element of which is the zone temperature, the others are auxiliary temperatures corresponding to other physical properties of the zone (walls and facade). The input to the system, $u \in \mathbb{R}^1$, is the heating/cooling energy. $b_d \in \mathbb{R}^3$ are disturbances (due to occupancy, outside temperature, solar radiation). We assume these are known a priori. The control problem we solve is of the form in (4), with γ and δ both set to zero (correspondingly, not cost for control in BluSTL), and $X = [0, 50]^4$, $U = [-1000, 2000]$.

Results. For comparison across all methods, we run 100 instances of the problem, starting from random initial states $x_0 \in [20, 21]^4$. SA, R-SQP and SOP are initialized with the same initial input sequences $\mathbf{u}$. The final trajectories after optimization from all methods are shown in Fig.5 for a particular instance with $x_0 = [21, 21, 21, 21]'$. To reduce clutter, we do not visualize trajectories from SA and R-SQP in mode (B).

Analysis. In Boolean mode, SOP, BluSTL, and SA all find satisfying trajectories across all 100 instances, while R-SQP does not find one for any run and always exits at a local minima. Execution times for SOP and BluSTL are shown in Table II, while the runtimes for SA (B) are $3.7 \pm 2.3s$. R-SQP has run-times in the order of minutes. This is because of Matlab's failed attempts at approximating a gradient that does not exist.

In Robust mode, SOP and SA both result in trajectories that satisfy φ, with an average robustness of 2.9994 and 2.8862 respectively. On the other hand, R-SQP often returns violating trajectories (average $\rho_\varphi = -0.1492$). Somewhat surprisingly, BluSTL does not manage to find a satisfying trajectory (average $\rho = -2.71$) for any of the 100 runs. One particular instance is shown in Fig. 5. As shown in Table II, SOP takes $2.5s$ on average. SA (R) takes $8.56 \pm 0.31s$ on average. R-SQP again takes minutes on average to return non-satisfying trajectories across all 100 runs.

1) Receding horizon implementation: Our scheme can also be implemented online in a receding (shrinking) horizon method similar to [23]. Note that the control horizon in Eq. 4 is as at least long as the formula horizon N, and evaluating $\tilde{\rho}$ in general requires a trajectory of length N. For each time step $k = 0, \ldots, N$, we solve the problem of Eq.4 while constraining the variables (the previously applied inputs and states, when applicable) for times steps $< k$ to their actual values. In this scheme, the length of the optimization remains N, but the number of free variables keeps on shrinking as k increases. For initializing SOP at each time step, the sequence of inputs (or states) computed at time $k-1$ is used as an initial solution for the optimization at time k. We implemented this scheme for the HVAC control problem with additional unknown disturbances in the dynamics of the form:

$$x_{k+1} = Ax_k + Bu_k + B_d d_k + B_d w_k$$

Here, w_k is a uniform random variable centred around the known d_k with a width of 10% of element wise magnitude of d_k. This can be thought of as prediction errors in the disturbances like solar radiation and outside temperature. Over 100 runs with random initial states as before, the online application of SOP (in *robust* mode) resulted in an average robustness value of 2.91. In terms of execution time, the first iteration takes times of the order of those in table II, and subsequent iterations take a fraction of that time (average for one instance $0.0151s$). This is because we re-use the input sequence at time $k-1$ as an initial guess for the solver at time k. Since at the initial time step we have achieved near global robustness maxima, the subsequent SQP optimizations terminate much faster while

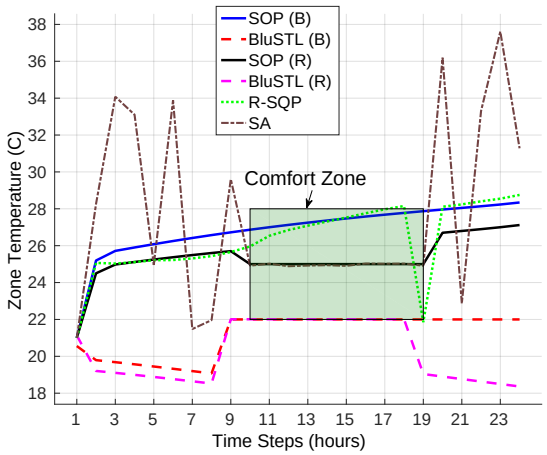

Fig. 5: Zone temperatures. The green rectangle shows the comfortable temperature limit of 22-28 C, applicable during time steps 10-19 (when the building is occupied). Color in online version.

the formulation takes into account change in the state due to disturbance values by making small changes to the input sequence being computed at time $k > 0$. The high value of average robustness and the small execution time per iteration show the applicability of SOP as an online closed loop control method.

B. Autonomous ATC for quad-rotors

Air Traffic Control (ATC) offers many opportunities for automation to allow safer and more efficient landing patterns. The constraints of ATC are complex and contain many safety rules [16]. In this example we formalize a subset of such rules, similar to those in example 1, for an autonomous ATC for quad-rotors in MTL. We demonstrate how the smoothed robustness is used to generate control strategies for safely and robustly manoeuvring two quad-rotors in an enclosed airspace with an obstacle.

The specification. The specification for the autonomous ATC with two quad-rotors is:

$$\begin{aligned}\varphi = {} & \Diamond_{[0,N-1]}(q_1 \in \text{Terminal}) \wedge \Diamond_{[0,N-1]}(q_2 \in \text{Terminal}) \wedge \\ & \Box_{[0,N-1]}(q_1 \in \text{Zone}_1 \implies z_1 \in [1,5]) \wedge \\ & \Box_{[0,N-1]}(q_2 \in \text{Zone}_1 \implies z_2 \in [1,5]) \wedge \\ & \Box_{[0,N-1]}(q_1 \in \text{Zone}_2 \implies z_1 \in [0,3]) \wedge \\ & \Box_{[0,N-1]}(q_2 \in \text{Zone}_2 \implies z_2 \in [0,3]) \wedge \\ & \Box_{[0,N-1]}(\neg(q_1 \in \text{Unsafe})) \wedge \Box_{[0,N-1]}(\neg(q_2 \in \text{Unsafe})) \wedge \\ & \Box_{[0,N-1]}(||q_1 - q_2||_2^2 \geq d_{min}^2) \end{aligned} \tag{16a}$$

Here q_1 and q_2 refer to the position of the two quad-rotors in (x, y, z)-space, and z_1 and z_2 refer to their altitude. The specification says that, within a horizon of N steps, both quad-rotors should: a) Eventually visit the terminal zone (e.g. to refuel or drop package), b) Follow altitude rules in two zones, Zone_1 and Zone_2 which have different altitude floors and ceilings, c) Avoid the Unsafe set, and d) always maintain a safe distance between each other (d_{min}).

Note that turning the specification into constraints for the control problem is no longer simple. This is due to the $\Diamond$ operator, which would require a MILP formulation to be accounted for. In addition, the minimum separation and altitude rules for the two zones cannot be turned into convex constraints for the optimization. As will be seen below, our approach allows us to keep the non-convexity in the cost function, and have convex (linear) constraints on the optimization problem.

System dynamics. The airspace and associated sets for the specification φ are hyper-rectangles in $\mathbb{R}^3$ (visualized in Fig. 6), except the altitude floor and ceiling limit, which is in $\mathbb{R}^1$. In simulation, d_{min} is set to 0.2 m.

The quad-rotor dynamics are obtained via linearization around hover, and discretization at 5-Hz. Similar models have been used for control of real quad-rotors with success ([21]). For simulation, we set the mass of either quad-rotor to be 0.5 kg. The corresponding linearized and discretized quad-rotor dynamics are given as:

$$\begin{bmatrix} \dot{x}_{k+1} \\ \dot{y}_{k+1} \\ \dot{z}_{k+1} \\ x_{k+1} \\ y_{k+1} \\ z_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0.2 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0.2 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0.2 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_k \\ \dot{y}_k \\ \dot{z}_k \\ x_k \\ y_k \\ z_k \end{bmatrix} + \begin{bmatrix} 1.96 & 0 & 0 \\ 0 & -1.96 & 0 \\ 0 & 0 & 0.4 \\ 0.196 & 0 & 0 \\ 0 & -0.196 & 0 \\ 0 & 0 & 0.04 \end{bmatrix} \begin{bmatrix} \theta_k \\ \phi_k \\ \mathrm{T}_k \end{bmatrix} \tag{17}$$

Here, the state consists of the velocities and positions in the x, y, z co-ordinates. The inputs to the system are the desired roll angle θ, pitch angle ϕ and thrust T.

The control problem. For the autonomous ATC problem for two quad-rotors, we solve (4) with $\tilde{\rho}$ in the objective instead of ρ. Note, we set $\gamma = 0$ here, following logically from existing ATC rules (see sec.I), which do not have an air-craft specific cost for fuel, or distance traveled. Because of this, we can also set $\delta = 0$ and simply maximize (smooth) robustness (subject to system dynamics and constraints) to get trajectories that satisfy φ. For the control problem, X and U represent the bounds on the states (Airspace and velocity limits) and inputs respectively, for both quad-rotors. f represents the linearized dynamics of (17) applied to two quad-rotors, and $N = 21$. The initial state for the first quad-rotor is $[2, 2, 2, 0, 0, 0]'$ and for the second, $[2, -2, 2, 0, 0, 0]'$.

Results. For each approach (except BluSTL), we ran three optimizations, starting from three different trajectories to initialize the optimization. In this case study, we only aim to maximize robustness, i.e. operate in the *robust* mode. BluSTL, in either *boolean* or *robust* mode could not find a solution for this problem (ran over 100 hours without terminating) and so is excluded from the rest of this comparison. This suggests that having a complex specification like the one in this problem, non-trivial dynamics, and the given formula horizon, results in a MILP that is simply intractable to solve. We believe that this example highlights a fundamental limitation of MILP based approaches.

These *initial trajectories* can be obtained in practice by a fast trajectory generator. Note that the three initial trajectories all have negative robustness, i.e. they violate φ.

Fig.6 shows the three trajectories obtained after applying SOP, with the three initial trajectories as initial guesses for the optimizations. All three trajectories obtained by SOP satisfy the specification φ. To avoid visual clutter, we do not show the trajectories obtained from the other methods on the figure. Instead, we summarize the results in Table III which shows the true robustness of the three initial trajectories, and

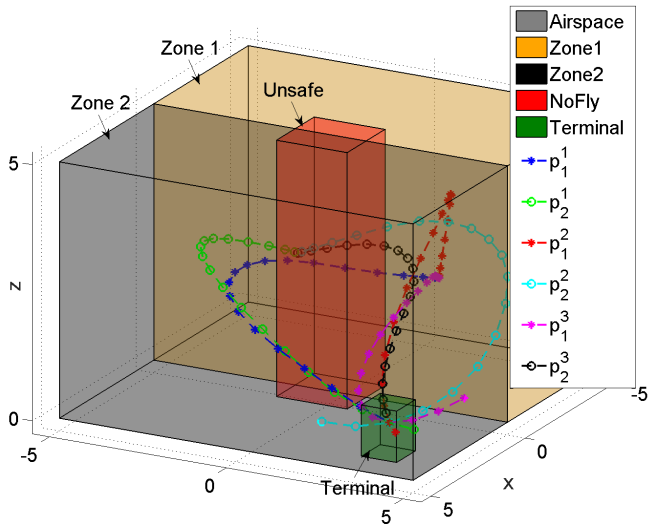

Fig. 6: Trajectories obtained via SQP on smooth robustness, with three different initial trajectories acting as initial solutions for the SQP. Note, all 3 trajectories satisfy φ. Here, q_i^j refers to the positions of the i^{th} trajectory for the j^{th} quadrotor. A real-time playback of trajectories can be seen in `https://youtu.be/gofB_HLwFGA`.

the true robustness for the trajectories obtained via the three methods, SOP, SA, and R-SQP.In order to keep the study easy to interpret, we use two quad-rotors, but it is straight forward to scale the specification and the optimization to account for more.

Analysis. It is seen that SOP and R-SQP satisfy φ for all instances, while SA satisfies it only once. Note that in all three cases, R-SQP results in trajectories with the same robustness value, which is less than the robustness value achieved in SOP. We conjecture that this is because R-SQP is getting stuck at local minima at points of non-differentiability of the objective, as illustrated in Example 2. On further investigation, we also noticed that the robustness value achieved is due to the segment of the φ corresponding to $\Diamond_{[0,N]}(q_2 \in \text{Terminal})$. R-SQP does not drive the trajectory (for quad-rotor 2) deeper inside the set Terminal, unlike the proposed approach, SOP, even though the minimum separation property is far from being violated. This lends credence to our hypothesis of SQP terminating on a local minima, which is the flag MATLAB's optimization gives.

TABLE III: Robustness of final trajectory ρ^* for three optimization runs with different initial trajectories ($\mathbf{x}_0$),none of which satisfy the specification).

Run	$\rho(\mathbf{x}_0)$	SOP $\rho^*[\tilde{\rho}^*]$	SA: ρ^*	R-SQP: ρ^*
1	-0.8803	**0.3689** [0.4107]	-0.2424	0.1798
2	-0.7832	**0.3688** [0.4106]	-0.5861	0.1798
3	-0.0399	**0.3689** [0.4107]	0.0854	0.1798

C. Discussion

With two case studies on dynamic systems, we show the applicability and consistently good performance of SOP (our method), which outperforms both SA R-SQP and BluSTL, which does not scale well enough to solve the second case study. For every instance we covered, SOP finds trajectories that satisfy the specification, while the other methods do not always do so.

While we solve the control problem in a single-shot, finite horizon manner, in general, for a real-time implementation, the problem can be solved in a receding horizon manner (similar to [21], [13]). Or, it can be solved in a manner where the state and actions of the past are stored an added as constraints at each time step while the look-ahead horizon of the optimization shrinks (similar to [23]). This will be explored further in future work. We have shown previously [21] that control of an actual quad-rotor with the dynamics in (17) is possible on a low computation power platform. The control algorithm there involved solving multiple quadratic programs at even higher sampling rates ($20Hz$), in a receding horizon manner. Future work will include a C implementation of SOP, which will allow us to experiment on real platforms, like the aforementioned quad-rotors.

VI. Conclusions

We present a method to obtain smooth (infinity differentiable) approximations to the robustness of MTL formulae, with bounded and asymptotically decaying approximation error. Empirically, we show that the approximation error is indeed small for a variety of commonly used MTL formulae. Through several examples, we show how we leverage the smoothness property of the approximation for solving a control problem by maximizing the smooth robustness, using SQP, an off-the-shelf gradient descent optimization technique. A similar approach can also be used for falsification by minimizing the smooth robustness over a set of possible initial states for a closed loop system. We compare our technique (SOP) to two other approaches for robustness maximization for control of two dynamical systems, with state and input constraints, and show how our approach consistently outperforms the other two and can be used for control of systems to satisfy MTL specifications.

References

[1] H. Abbas and G. Fainekos. Linear hybrid system falsification through local search. In *Automated Technology for Verification and Analysis*, volume 6996 of *LNCS*, pages 503–510. Springer, 2011.

[2] H. Abbas and G. Fainekos. Convergence proofs for simulated annealing falsification of safety properties. In *Proc. of 50th Annual Allerton Conference on Communication, Control, and Computing*. IEEE Press, 2012.

[3] H. Abbas and G. Fainekos. Computing descent direction of MTL robustness for non-linear systems. In *American Control Conference*, 2013.

[4] H. Abbas, A. Winn, G. Fainekos, and A. A. Julius. Functional gradient descent method for metric temporal logic specifications. In *2014 American Control Conference*, pages 2312–2317, June 2014.

[5] J. Cortes. Discontinuous dynamical systems. *IEEE Control Systems*, 28(3):36–73, June 2008.

[6] J. Deshmukh, G. Fainekos, J. Kapinski, S. Sankaranarayanan, A. Zutshi, and Xiaoqing Jin. Beyond single shooting: Iterative approaches to falsification. In *2015 American Control Conference (ACC)*, pages 4098–4098, July 2015.

[7] A. Dokhanchi, B. Hoxha, and G. Fainekos. Online monitoring for temporal logic robustness. In *Proceedings of Runtime Verification*, 2014.

[8] A. Donzé and O. Maler. *Robust Satisfaction of Temporal Logic over Real-Valued Signals*, pages 92–106. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[9] T. Dreossi, T. Dang, A. Donze, J. Kapinski, X. Jin, and J. V. Deshmukh. A trajectory splicing approach to concretizing counterexamples for hybrid systems. In *NASA Symposium on Formal Methods*, 2015.

[10] G. Fainekos and G. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, September 2009.

[11] G. E. Fainekos, S. Sankaranarayanan, K. Ueda, and H. Yazarel. Verification of automotive control applications using s-taliro. In *2012 American Control Conference (ACC)*, pages 3567–3572, June 2012.

[12] G.E. Fainekos, A. Girard, and G. Pappas. *Temporal Logic Verification Using Simulation*, pages 171–186. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[13] A. Jain, M. Behl, and R. Mangharam. Data predictive control for building energy management (submitted to the acc). 2017.

[14] S. Kirkpatrick, D. Gelatt Jr., and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[15] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[16] M. Z. Li and M. S. Ryerson. Modeling and estimating airspace movements using air traffic control transcription data, a data-driven approach. In *International Conference on Research in Air Transportation*, 2016.

[17] M. M. Makela and P. Neittaanmaki. *Nonsmooth optimization*. World Scientific, 1992.

[18] S. G. Mallat. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. Academic Press, 2008.

[19] T. Nghiem, S. Sankaranarayanan, G. Fainekos, F. Ivancic, A. Gupta, and G. Pappas. Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems. In *Hybrid Systems: Computation and Control*, 2010.

[20] Joël Ouaknine and James Worrell. Some recent results in metric temporal logic. In *Proceedings of the 6th International Conference on Formal Modeling and Analysis of Timed Systems*, FORMATS '08, pages 1–13, Berlin, Heidelberg, 2008. Springer-Verlag.

[21] Y. V. Pant, K. Mohta, H. Abbas, T. X. Nghiem, J. Devietti, and R. Mangharam. Co-design of anytime computation and robust control. In *RTSS*, pages 43–52, Dec 2015.

[22] E. Polak. *Optimization: Algorithms and Consistent Approximations*. Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[23] V. Raman, A. Donze, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia. Model predictive control with signal temporal logic specifications. In *53rd IEEE Conference on Decision and Control*, pages 81–87, Dec 2014.

[24] S. Saha and A. Agung Julius. An milp approach for real-time optimal controller synthesis with metric temporal logic specifications. In *Proceedings of the 2016 American Control Conference (ACC)*, 2016.

[25] S. Sankaranarayanan and G. Fainekos. Falsification of temporal properties of hybrid systems using the cross-entropy method. In *ACM International Conference on Hybrid Systems: Computation and Control*, 2012.

[26] S. Sankaranarayanan and G. Fainekos. Simulating insulin infusion pump risks by in-silico modeling of the insulin-glucose regulatory system. In *International Conference on Computational Methods in Systems Biology*, 2012.

[27] A Van Schijndel. Integrated heat, air and moisture modeling and simulation in hamlab. In *IEA Annex 41 working meeting, Montreal, May*, 2005.

[28] Victor Vermehren Valenzuela, , and H. M. de Oliveira. Close expressions for meyer wavelet and scale function. 2015. Arxiv 1502.00161.