8-2017

Smooth Operator: Control using the Smooth Robustness of Temporal Logic

Yash Vardhan Pant
University of Pennsylvania, yashpant@seas.upenn.edu

Houssam Abbas
University of Pennsylvania, habbas@seas.upenn.edu

Rahul Mangharam
University of Pennsylvania, Philadelphia, rahulm@seas.upenn.edu

Smooth Operator: Control using the Smooth Robustness of Temporal Logic

Abstract

Modern control systems, like controllers for swarms of quadrotors, must satisfy complex control objectives while withstanding a wide range of disturbances, from bugs in their software to attacks on their sensors and changes in their environments. These requirements go beyond stability and tracking, and involve temporal and sequencing constraints on system response to various events. This work formalizes the requirements as formulas in Metric Temporal Logic (MTL), and designs a controller that maximizes the robustness of the MTL formula. Formally, if the system satisfies the formula with robustness r, then any disturbance of size less than r cannot cause it to violate the formula. Because robustness is not differentiable, this work provides arbitrarily precise, infinitely differentiable, approximations of it, thus enabling the use of powerful gradient descent optimizers. Experiments on a temperature control example and a two-quadrotor system demonstrate that this approach to controller design outper- forms existing approaches to robustness maximization based on Mixed Integer Linear Programming and stochastic heuristics. Moreover, it is not constrained to linear systems.

Keywords

Temporal Logic, Robustness, Optimization, Control with MTL specifications

Disciplines

Computer Engineering | Electrical and Computer Engineering

Smooth Operator: Control using the Smooth Robustness of Temporal Logic

Yash Vardhan Pant*, Houssam Abbas*, Rahul Mangharam

***Abstract*— Modern control systems, like controllers for swarms of quadrotors, must satisfy complex control objectives while withstanding a wide range of disturbances, from bugs in their software to attacks on their sensors and changes in their environments. These requirements go beyond stability and tracking, and involve temporal and sequencing constraints on system response to various events. This work formalizes the requirements as formulas in Metric Temporal Logic (MTL), and designs a controller that maximizes the *robustness* of the MTL formula. Formally, if the system satisfies the formula with robustness r, then any disturbance of size less than r cannot cause it to violate the formula. Because robustness is not differentiable, this work provides arbitrarily precise, infinitely differentiable, approximations of it, thus enabling the use of powerful gradient descent optimizers. Experiments on a temperature control example and a two-quadrotor system demonstrate that this approach to controller design outperforms existing approaches to robustness maximization based on Mixed Integer Linear Programming and stochastic heuristics. Moreover, it is not constrained to linear systems.**

I. Controlling for robustness

The errors in a cyber-physical control system like an automated air traffic controller 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 system can be designed to be robust to that source of error. In general, however, unforeseen and unmodeled issues will occur and the controller has to deal with them at runtime. To help deal with unforeseen problems at runtime, the system's controller must make decisions that not only satisfy the system's requirements (like a maximum response time to an event), but satisfy them *robustly*. Intuitively, the requirements are robustly satisfied if a disturbance to the system does not cause it to violate them. 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. Sample rules for the Chicago O'hare airport include (A) When an aircraft enters any of 3 designated zones, it must stay between that zone's altitude floor and ceiling, and (B) If the airspace is too busy, an aircraft must remain in either holding zones 6 or 7, until some maximum amount of time expires.

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

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) by encoding every possibility at each (discrete) time point with integer variables, such a direct encoding can easily involve an exorbitant number of variables. For complex requirements, with many variables involved, this encoding process can also be 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 Logic (MTL) [1]. MTL is a formal language for expressing reactive requirements with constraints on their time of occurrence and sequencing, such as those of the ATC. The advantage of first expressing the requirements in MTL is that MTL formulas are more succinct and legible, and less error-prone, than the corresponding directly-encoded MIP. In this sense, MTL bridges the gap between the ease of use of natural language and the rigour of mathematical formulation. For example, ATC rule (A) can be formalized with the following MTL formula ($\Box$ means 'Always', q is an aircraft and q_z is its altitude).

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

Rule (B) can be formalized as follows.

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

This says that Always ($\Box$), if airport is Busy, then sometime t_1 to t_2 seconds later ($\Diamond_{[t_1,t_2]}$), the plane goes into one of two Holding areas. It stays there $\mathcal{U}$ntil the airport is not ($\neg$) busy, which must happen before duration MaxHolding elapses.

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 [2], [3], [4]. See Related Work and the

*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.

Experiments. The second method, upon which this work builds, uses the *robustness of MTL specifications* [5], [6]. Robustness is a rigorous notion that has been used successfully for the testing and verification of automotive systems [7], [8], medical devices [9], and general CPS. Given an MTL specification φ 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 violates it ($\rho_\varphi(\mathbf{x}) < 0$). Its magnitude $|\rho_\varphi(\mathbf{x})|$ measures how *robustly* the spec is satisfied or violated. 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 and non-differentiable, which makes its online 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, usually have a fewer number of parameters to be set, and important issues like step-size selection are rigorously addressed.

Contributions. This paper presents smooth (infinitely differentiable) approximations to the robustness function of arbitrary MTL formulae. The smooth approximation is proven to always be within a user-defined error of the true robustness, and this is illustrated experimentally. This allows running powerful and rigorous off-the-shelf gradient descent optimizers. We leverage this to maximize the smooth robustness for control of a system to robustly satisfy its MTL specification. Through multiple examples, the proposed control method is shown to be faster and to yield more robust trajectories than various current heuristics and MIP-based approaches. The results are demonstrated on a case study for an autonomous ATC for two quad-rotors, where the MIP-based approach fails to yield a satisfying controller. While this work does not tackle the non-convexity of MTL robustness 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 [10], cross-entropy [11] and RRTs [8]; non-smooth optimization [12]; Mixed Integer Linear Programming (MILP) [2], [3]; and iterative approximations [13], [14], [15]. Black-box heuristics are the most commonly used approach. 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 their lack of guarantees as explained earlier. Because the robustness is non-smooth, the work in [12] 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 [2], 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 [2] and this paper. In general, MILP solvers can only guarantee achieving local optima. Note also that [2] requires all constraints to be linear, so all atomic propositions must involve half-spaces ($p : a'x \leq b$), which is not a restriction in the current work. Another MILP based approach is presented in [3] where constraints are added when necessary, in order to reduce MILP complexity. The work closest to this paper is [13], [14]. 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. The use of fast smooth approximations of robustness circumvents most of the above issues and gets closer to real-time control by robustness maximization.

II. Robustness of MTL formulae

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

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

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 finite control input sequence $\mathbf{u} = (u_0, \ldots, u_{T-1}), u_t \in U$, a *trajectory* of the system is the unique sequence of states $\mathbf{x} = (x_0, \ldots, x_T)$ s.t. for all t, x_t is in X and obeys (1). All temporal intervals that appear in this paper are implicitly discrete-time, e.g. $[a, b]$ means $[a, b] \cap \mathbb{N}$. The set $\{0, 1, \ldots, T\} \subset \mathbb{N}$ will be abbreviated as $\mathbb{T}$. For an interval $I \subset \mathbb{N}$, let $t + I = \{t + a \mid a \in I\}$. 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 (MTL)

The controller of $\mathcal{H}$ is designed to make the closed loop system (1) satisfy a specification expressed in MTL [1]. 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:

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

where $I \subset \mathbb{R}$ is a time interval. Here, $\top$ is the Boolean True, p is an atomic proposition, $\neg$ and $\wedge$ are the Boolean negation 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 disjunction

($\vee$), implication ($\implies$), Always ($\Box$) and Eventually ($\Diamond$) operators can be defined 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):

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

As $\mathcal{O}$ is fixed in this paper, it is dropped 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, $hrz(\varphi) \in \mathbb{N}$, calculable as shown in [16]. For example, the horizon of $\Box_{[0,2]}(\Diamond_{[2,4]} p)$ is 2+4=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 minimum Euclidian distance from x to the closure $\overline{A}$ of A.

Definition 2.2 (Robustness[17]): The *robustness* of φ relative to $\mathbf{x}$ at time t is recursively defined as

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

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: [17] For any $\mathbf{x} \in X^{\mathbb{T}}$ and MTL formula φ, if $\rho_\varphi(\mathbf{x},t) < 0$ then $\mathbf{x}$ violates 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.

[1]Strictly, a controller s.t. the closed-loop behavior satisfies the formula.

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 (1). The larger $\rho_\varphi(\mathbf{x},t)$, the 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

Let φ be an MTL formula with horizon N. The goal of the present work is to solve the following problem P_ρ.

$$\begin{aligned}
P_\rho :\ \max_{\mathbf{u}} \rho_\varphi(\mathbf{x}) - \gamma \sum_{k=0}^{N-1} l(x_{k+1}, u_k) && \text{(2a)} \\
\text{s.t. } x_{k+1} = f(x_k, u_k),\ \forall k = 0, \ldots, N-1 && \text{(2b)} \\
x_k \in X,\ \forall k = 0, \ldots, N && \text{(2c)} \\
u_k \in U,\ \forall k = 0, \ldots, N-1 && \text{(2d)} \\
\delta \rho_\varphi(\mathbf{x}) \geq \delta \epsilon_{\min} && \text{(2e)}
\end{aligned}$$

Here, $\mathbf{u} = (u_0, \ldots, u_{N-1})$, $l(x_{k+1}, u_k)$ 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. Because ρ_φ uses the non-differentiable functions $\mathbf{dist}$, max and min, it is itself non-differentiable. The next three sub-sections introduce smooth approximations to each of these functions.

A. Approximating the distance function

The distance function $\mathbf{dist}(\cdot, U)$ is in $L_2(\mathbb{R}^n)$, so it can be approximated arbitrarily well using a Meyer wavelet expansion [18]. 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 [19] and is infinitely differentiable. An n-D wavelet can be obtained using the tensor product construction [18]. Let E be the set of vertices of the unit hypercube $[0,1]^n$. For every $e = (e_1, e_2, \ldots, e_n) \in E$ and $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, define $\Psi^e : \mathbb{R}^n \to \mathbb{R}$ by $\Psi^e(x) = \psi^{e_1}(x_1) \ldots \psi^{e_n}(x_n)$. Given $k \in \mathbb{Z}$ and $\mathbf{j} \in \mathbb{Z}^n$, a *dyadic cube* in $\mathbb{R}^n$ is a set of the form $I = 2^{-k}(\mathbf{j} + [0,1]^n)$. Let D be the set of all dyadic cubes in $\mathbb{R}^n$ obtained by varying k over $\mathbb{Z}$ and $\mathbf{j}$ 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 $\langle h, g \rangle := \int_{\mathbb{R}^n} h(x) g(x) dx$. The desired approximation is obtained by truncating this expansion after a finite number of terms, i.e., by using a *finite* set $D' \subsetneq D$

$$\mathbf{dist}(x, U) \approx \widetilde{\mathbf{dist}}_\varepsilon(x, U) := \sum_{I \in D'} \sum_{e \in E} c_I^e \Psi_I^e(x) \qquad (3)$$

where ε is the approximation error magnitude. Using more coefficients yields a better approximation. 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.

B. Smooth max and min

The following standard smooth approximations of m-ary max and min are used. Let $k \geq 1$.

$$\widetilde{\max}_k(a_1, \ldots, a_m) \quad := \quad \frac{1}{k}\ln(e^{ka_1} + \ldots + e^{ka_m}) \quad (4)$$

$$\widetilde{\min}_k(a_1, \ldots, a_m) \quad := \quad -\widetilde{\max}(-a_1, \ldots, -a_m) \quad (5)$$

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 $\widetilde{\max}_1(\mathbf{a}) \approxeq \ln e^{a_1} = a_1 = \max(\mathbf{a})$. 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,

$$0 \leq \widetilde{\max}_k(a_1, \ldots, a_m) - \max(a_1, \ldots, a_m) \leq \ln(m)/k \quad (6)$$

$$0 \leq \min(a_1, \ldots, a_m) - \widetilde{\min}_k(a_1, \ldots, a_m) \leq \ln(m)/k \quad (7)$$

with the maximum error is achieved when all the a_i's are equal. Indeed, assume a_1 is the largest number, then $\widetilde{\max}_k(\mathbf{a}) - a_1 \leq k^{-1}\ln\left(\frac{\sum_i e^{ka_i}}{e^{ka_1}}\right) \leq \ln m/k$.

C. Overall approximation

Putting the pieces together yields the approximation error for the robustness of any MTL formula.

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

$$|\rho_\varphi(\mathbf{x}, t) - \tilde{\rho}_\varphi(\mathbf{x}, t)| \leq \delta_\varphi$$

where δ_φ is (a) independent of the evaluation time t, and (b) goes to 0 as $\varepsilon \to 0$ and $k \to \infty$.
The proof is in on the online technical report [20]. The proof also gives an *explicit computation* for δ_φ in terms of ε, k and φ, which can be computed beforehand and made as small as desired.

IV. Approximation and Control

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

A. Approximation error

We evaluated the robustness ρ_φ and its approximation $\tilde{\rho}_\varphi$ for five formulae. The horizon N of each formula is varied, and at each value of N we generate 1000 trajectories of system $x_{k+1} = x_k + u_k$ with input and state saturation, by feeding it random input sequences. Fig. 1 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 errors and their variances are small, showing the accuracy and stability of the smooth approximation. Note that while the RMSE increased with the system dimension (4th formula in Fig. 1), 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.

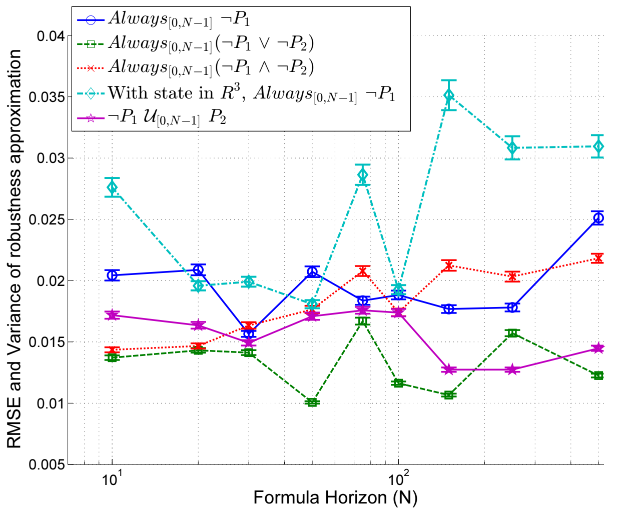

Fig. 1: Robustness approximation error against formula horizon, evaluated on 1000 randomly generated trajectories for Example 2. Unless noted, the systems are 2D. Color in online version.

B. Robustness maximization for control

Problem P_ρ given in (2) is solved by replacing the true robustness ρ_φ by its smooth approximation $\tilde{\rho}_\varphi$, and setting $\epsilon_{\min}$ to the value of the smooth approximation error. We thus obtain Problem $P_{\tilde{\rho}}$. This approach is labeled *Smooth Operator* (SOP).

Optimization solver. Problem $P_{\tilde{\rho}}$ is solved using Sequential Quadratic Programming (SQP). SQP solves constrained non-linear optimization problems by creating a sequence of quadratic approximations to the problem and solving these approximate problems. SQP enjoys various convergence-to-(local)-optima properties [21, 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 (2a), 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 there exists a search direction towards the interior of the feasible set that does not violate the equality constraints [21, Assumption 2.9.1]. This is also true for $P_{\tilde{\rho}}$ since the 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 solve an inexpensive feasibility linear program with constraints (2b)-(2d), or generate a random input sequence respecting $u_t \in U$. The resulting initial trajectory could violate the specification (as it does in every example we study in this paper) and it is only required to satisfy the dynamics and state constraints.

Comparisons to BluSTL. The tool BluSTL implements the MILP approach of [2] and is used in the experiments.

TABLE I: Example 2. 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

It 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). SOP emulates mode (B) by terminating the optimization as soon as $\tilde{\rho}_\varphi \geq \epsilon_{\text{Meyer}}$, which implies $\rho_\varphi \geq 0$. ϵ_{Meyer} can be computed explicitly using the approach in the online report [20].

Example 2: The linear system $x_{k+1} = x_k + u_k$ is controlled 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, $\gamma = \delta = 0$ in Eq. (2) to focus on robustness maximization in this illustrative example. Experiments were run on a quad-core Intel i5 3.2GHz processor with 24GB RAM, running MATLAB 2016b.

Results. Fig. 2 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 in Boolean mode, and displays smaller variances in runtimes. Note also that the problem solved here is very similar to the one used in [3], which uses another MILP-based method. While the underlying dynamics differ and their numbers are reported on a more power machine, SOP numbers compare favourably with those in [3].

In (R) mode, across 100 experiments, SOP has 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 $<$ 20 time steps) the center of the Terminal set while remaining more than 0.25 distant from Unsafe.

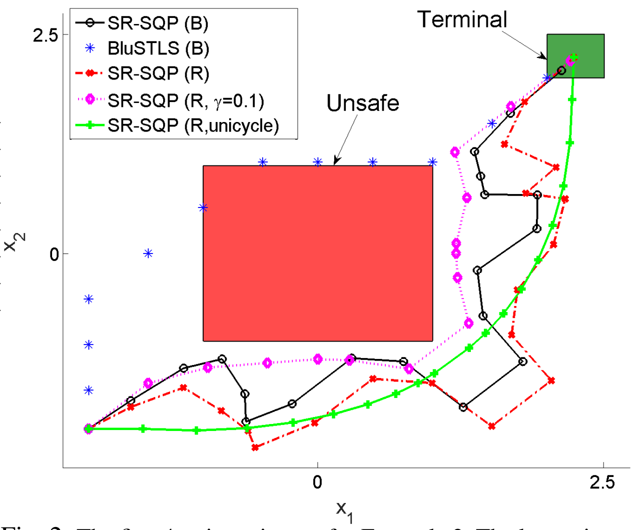

Fig. 2: The first 4 trajectories are for Example 2. The last trajectory, SOP(R, unicycle), is from Example 3. Colors in online version.

Example 3 (Nonlinear system): Since SQP can handle non-linear (twice differentiable) constraints, Smooth Operator can also deal with non-linear dynamics whereas the MILP-based methods have to linearize the 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. 2, the resulting trajectory of length 20 steps obtained by SOP(R) is shown in Fig. 2, 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.

V. Case studies

This section focuses on evaluating the efficiency of Smooth Operator (SOP) by testing it on two systems and comparing to existing approaches.

- 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, the wavelet approximation to the distance function is computed off-line. The control problem (2) is solved as an open-loop, single-shot, finite-horizon constrained optimization. This is then used in a shrinking horizon scheme in Sec V-A.1.

The code to reproduce these results can be found at `https://github.com/yashpant/SmoothOperator0`. Future versions of the code will focus on re-usability of the code.

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

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

zone in a building. Such a model is commonly used in literature for evaluation of predictive control algorithms [22]. The control problem is similar to the example used in [2]. The control horizon is a 24 hour period. The objective is to bring the zone temperature to a comfortable range, $[22, 28]$ Celsius, when the zone is occupied during the hours 10-to-19. The specification is:

$$\varphi = \Box_{[10,19]}(\text{ZoneTemp} \in [22, 28]) \quad (8)$$

Note: For this particular specification, the maximum robustness is 3, achievable by setting the room temperature at 25C during the interval $[10, 19]$. Thus the problem can be solved by minimizing the cost $\sum_{10 \le k \le 19}(x_{4k} - 25)^2$ with linear constraints, which is a problem-specific approach. SOP, which is a general purpose technique, results in a robustness which is just 0.02% smaller than the global optimum. Section V-B shows a specification that 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:

$$x_{k+1} = Ax_k + Bu_k + B_d d_k \quad (9)$$

Here, A, B and B_d matrices are from the hamlab ISE model [23]. $x \in \mathbb{R}^4$ 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. 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) assumed known a priori. The control problem we solve is of the form in (2), with γ and δ both set to zero (correspondingly, no 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 are shown in Fig.3, for $x_0 = [21, 21, 21, 21]'$. To reduce clutter, trajectories from SA and R-SQP in mode (B) are not shown.

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 minimum. 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.

In Robust mode, SOP and SA both result in trajectories that satisfy φ, with an average robustness of 2.99 and 2.88 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

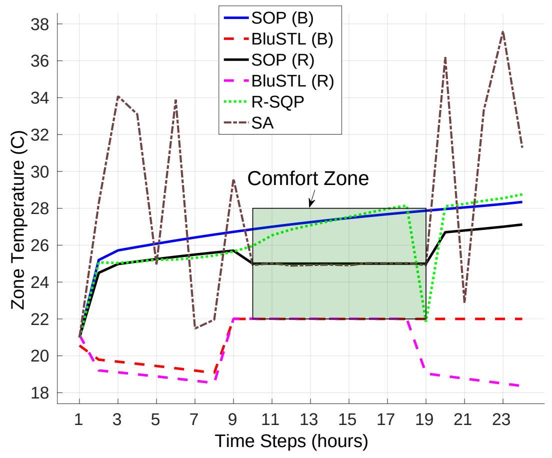

Fig. 3: Zone temperatures. The green rectangle shows the comfortable temperature limit of 22-28 C, applicable during time steps 10-19 (when the building is occupied). Color in online version.

particular instance is shown in Fig. 3. 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) Shrinking horizon implementation: SOP can also be applied online in a shrinking horizon fashion similar to [2]. The control horizon in Eq. (2) equals the formula horizon N. For each time step $k = 0, \ldots, N$, problem (2) is solved while constraining the previously applied inputs and states (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 computed at time $k-1$ is used as a feasible solution for the optimization at time k. We implemented this scheme for the HVAC control problem with additional unknown disturbances in d_k term of Eq. (9). These disturbances (in $\mathbb{R}^3$) are uniform random variables centred around the known d_k with an interval 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 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 [24]. 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:

$$\begin{aligned}\varphi = &\Diamond_{[0,N-1]}(q_1 \in \text{Terminal}) \wedge \Diamond_{[0,N-1]}(q_2 \in \text{Terminal}) \wedge \\ &\Box_{[0,N-1]}(q_1 \in \text{Zone}_1 \implies z_1 \in [1,5]) \wedge \\ &\Box_{[0,N-1]}(q_2 \in \text{Zone}_1 \implies z_2 \in [1,5]) \wedge \\ &\Box_{[0,N-1]}(q_1 \in \text{Zone}_2 \implies z_1 \in [0,3]) \wedge \\ &\Box_{[0,N-1]}(q_2 \in \text{Zone}_2 \implies z_2 \in [0,3]) \wedge \\ &\Box_{[0,N-1]}(\neg(q_1 \in \text{Unsafe})) \wedge \Box_{[0,N-1]}(\neg(q_2 \in \text{Unsafe})) \wedge \\ &\Box_{[0,N-1]}(||q_1 - q_2||_2^2 \geq d_{min}^2) \end{aligned} \tag{10a}$$

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. 4), 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 ([25]). 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 in [20]. The state for a quad-rotor $x \in \mathbb{R}^6$ consists of the velocities and positions in the x, y, z co-ordinates respectively. 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 (2) 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 ($P_{\tilde{\rho}}$), 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 applied to two quad-rotors, and $N = 21$. The initial state for the first quad-rotor is $[0, 0, 0, 2, 2, 2]'$ and for the second, $[0, 0, 0, 2, -2, 2]'$.

Results. For each approach (except BluSTL), we ran three optimizations, starting from three different trajectories to initialize the optimization. This can be thought of as a multi-start optimization, and these *initial trajectories* can be obtained in practice by a fast trajectory generator. All three initial trajectories have negative robustness, i.e. they violate φ. 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/horizon length results in a MILP that is intractable to solve. We believe that this example highlights a fundamental limitation of MILP based approaches.

Fig.4 shows the three trajectories obtained after applying SOP, all of which 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 the true robustness for the trajectories obtained via the three methods, SOP, SA, and R-SQP. Unlike previous examples, we did not explicitly compute the gradient of the robustness for φ. Because of this run-times are much slower as MATLAB has to numerically compute the gradient using finite-differences, resulting in overheads that were not incurred in the other examples. Despite this, SOP takes the order of 30 minutes for the optimization, while SA and R-SQP take over 4 hours to do so. Including explicit gradients should result in a significant speed up as was observed for the other examples.

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 (see Ex.2 in [20]). 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 3 runs with different initial trajectories ($\mathbf{x}_0$), none of which satisfy φ.

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

VI. Discussion and 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.

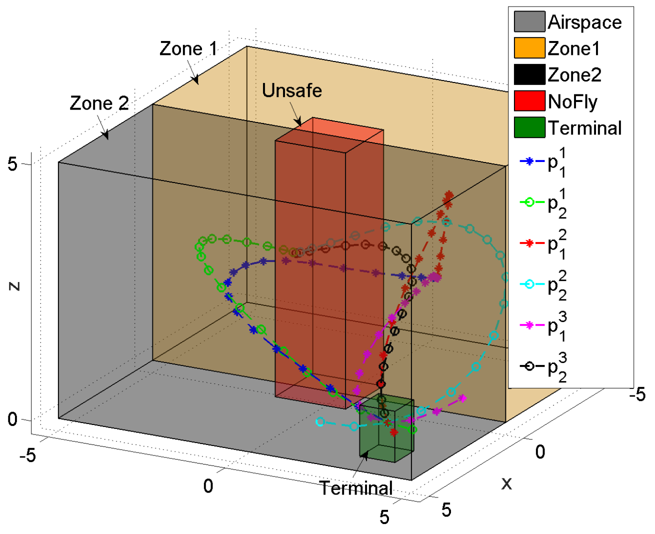

Fig. 4: Trajectories obtained via SQP on smooth robustness, with three different initial trajectories acting as initial solutions for the SQP. Note, all 3 trajectories satisfy φ. Here, p_i^j refers to the positions of the i^{th} trajectory for the j^{th} quadrotor. A real-time playback of trajectories can be seen in `https://youtu.be/FU3Rg1Jb7Fw`.

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 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 methods. While for most examples, 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 an online manner as in Sec. V-A.1. Future work will include a C implementation of SOP, which will allow us to experiment on real platforms, like the aforementioned quad-rotors, and also expand.

Acknowledgements

We gratefully acknowledge the fruitful discussions with Vasumathi Raman about the MILP approach.

References

[1] J. Ouaknine and J. Worrell, "Some recent results in metric temporal logic," in *Proceedings of the 6th International Conference on Formal Modeling and Analysis of Timed Systems*, ser. FORMATS '08, 2008.

[2] 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*, Dec 2014, pp. 81–87.

[3] S. Saha and A. A. 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.

[4] S. Karaman and E. Frazzoli, "Linear temporal logic vehicle routing with applications to multi-uav mission planning," *International Journal of Robust and Nonlinear Control*, vol. 21, no. 12, pp. 1372–1395, 2011. [Online]. Available: http://dx.doi.org/10.1002/rnc.1715

[5] G. Fainekos, A. Girard, and G. Pappas, *Temporal Logic Verification Using Simulation*. Springer Berlin Heidelberg, 2006.

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

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] ——, "Linear hybrid system falsification through local search," in *Automated Technology for Verification and Analysis*, ser. LNCS, vol. 6996. Springer, 2011, pp. 503–510.

[14] H. Abbas, A. Winn, G. Fainekos, and A. A. Julius, "Functional gradient descent method for metric temporal logic specifications," in *American Control Conference*, June 2014.

[15] J. Deshmukh, G. Fainekos, J. Kapinski, S. Sankaranarayanan, A. Zutshi, and X. Jin, "Beyond single shooting: Iterative approaches to falsification," in *American Control Conference*, July 2015.

[16] A. Dokhanchi, B. Hoxha, and G. Fainekos, "Online monitoring for temporal logic robustness," in *Proc. of Runtime Verification*, 2014.

[17] G. Fainekos and G. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, 2009.

[18] R. A. DeVore, "Nonlinear approximation," *Acta Numerica*, vol. 7, p. 51150, Jan 1998.

[19] V. Vermehren Valenzuela and H. M. de Oliveira, "Close expressions for meyer wavelet and scale function," 2015, arxiv 1502.00161.

[20] Y. V. Pant, H. Abbas, and R. Mangharam, "Technical report: Control using the smooth robustness of temporal logic," 2017, university of Pennsylvania Scholarly Commons. [Online]. Available: http://repository.upenn.edu/mlab_papers/98/

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

[22] A. Jain, M. Behl, and R. Mangharam, "Data predictive control for building energy management," in *American Control Conference*, 2017.

[23] A. Van Schijndel, "Integrated heat, air and moisture modeling and simulation in hamlab," in *IEA Annex 41 working meeting*, 2005.

[24] 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.

[25] 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*, 2015.