Functional Gradient Descent Method for Metric Temporal Logic Specifications

Houssam Abbas, Andrew Winn, Georgios Fainekos and A. Agung Julius

*Abstract***— Metric Temporal Logic (MTL) specifications can capture complex state and timing requirements. Given a nonlinear dynamical system and an MTL specification for that system, our goal is to find a trajectory that violates or satisfies the specification. This trajectory can be used as a concrete feedback to the system designer in the case of violation or as a trajectory to be tracked in the case of satisfaction. The search for such a trajectory is conducted over the space of initial conditions, system parameters and input signals. We convert the trajectory search problem into an optimization problem through MTL robust semantics. Robustness quantifies how close the trajectory is to violating or satisfying a specification. Starting from some arbitrary initial condition and parameter and given an input signal, we compute a descent direction in the search space, which leads to a trajectory that optimizes the MTL robustness. This process can be iterated to reach local optima (min or max). We demonstrate the method on examples from the literature.**

I. Introduction

The development of control laws for nonlinear systems still remains a formidable challenge despite the wealth of results on nonlinear control. Many practitioners and researchers still prefer to use classic optimal control [1] and Proportional-Integral-Derivative (PID) control [2]. There are a number of reasons for such a choice. The most important one is that such methods are automatic or almost automatic and, especially in industry, the control engineers might desire to avoid complex mathematical derivations. In other cases, PID control is sufficient to achieve the desired results [3] or accurate mathematical models might not be available.

Usually, Model-Based Development (MBD) software tools like Simulink Design Optimization$^{\text{TM}}$ can search over system parameters so that the system output satisfies certain time and frequency domain specifications. S-TALIRO [4] and BREACH [5] can be used for both falsification and open loop control design for Metric Temporal Logic (MTL) [6] and STL specifications, respectively. MTL can capture requirements on the correct sequencing of events, conditional reachability, safety requirements and real-time constraints between various events. In the MTL falsification problem you are given a formal requirement in MTL and the goal is to find system operating conditions and parameters that generate behaviors which do not satisfy the requirements. In [7], the *robustness* value of an MTL formula [8] with respect to a system simulation is considered as the cost function for an optimization problem. By minimizing the robustness value over the system simulations one can discover incorrect or non-robust system behaviors, and by maximizing the robustness value one can synthesize robust optimal control trajectories with respect to the MTL specification, all within the same framework.

All the aforementioned tools treat the system as a black-box in order to handle systems of arbitrary complexity. They need an accurate system simulator and, usually, as the fidelity of the model increases so does the simulation time. Thus, the total number of simulations needed before simulation-based tools can provide an answer becomes critical.

In [9], the first steps were taken towards addressing the problem of reducing the number of simulations for the MTL falsification problem. That work dealt with deterministic autonomous nonlinear systems where the search space was the set of initial conditions (and any parameters) x_0 of the system. If $f(x_0)$ denotes the MTL robustness value of the trajectory starting from x_0, then the goal of [9] was to compute a vector d such that $f(x_0+d) < f(x_0)$. In this paper, the search space consists of the set of initial conditions x_0 in $\mathbb{R}^N$ and the set of square-integrable input signals of duration $T > 0$. If a trajectory is computed starting from an initial condition x_0 and under a given input signal $u(\cdot)$ yields property robustness $f(x_0, u)$, then a descent element (d_{x_0}, d_u) should be computed so that starting from $x_0 + d_{x_0}$ and under input $u + d_u$, the system has robustness $f(x_0 + d_{x_0}, u + d_u) < f(x_0, u)$.

Our solution combines the previous works in [9], [10]. In brief, the work in [10] deals with the problem of optimizing a differentiable integral cost function over the output trajectories of the system starting from a given an input signal. The method is based on the calculus of variations, but it uses a gradient descent based approach to solve the optimization problem without formulating the optimality conditions given by the Minimum Principle.

Contributions: In this paper, we present a method for the computation of descent directions for reducing specification robustness for nonlinear dynamical systems. In particular, given an arbitrary MTL specification, we determine a critical point on the system trajectory which if changed, then the MTL robustness will be changed as well. We derive the equations which, given (x, u), will give a descent direction (d_x, d_u) that provably reduces robustness. Finally, we demonstrate the applicability of our approach on some nonlinear models from the literature.

This work was partially supported by the NSF awards CNS-1116136, CNS-1218109, CNS-1319560 and by the Department of Defense SMART Scholarship.

H. Abbas and G. Fainekos are with the Schools of Engineering at Arizona State University, Tempe, AZ, E-mail: {hyabbas, fainekos}@asu.edu

A. Winn and A. A. Julius are with the Department of Electrical, Computer, and Systems Engineering at Rensselaer Polytechnic Institute, Troy, NY, E-mail: winna@rpi.edu, agung@ecse.rpi.edu

II. Problem Formulation

We consider a dynamical system with state $x \in X \subset \mathbb{R}^n$

$$\dot{x} = F(t, x, u) \tag{1}$$

for a C^1 flow $F : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ with initial conditions $x_0 \in X_0$, and control input signal $u \in L^2[0, T]$ which takes values in a bounded subset U of $\mathbb{R}^m$: $u(t) \in U$ $\forall t$. T is the trajectory duration, and is fixed throughout this paper, so we may write L^2 without ambiguity. As explained earlier, x_0 can include any system parameters (which are assumed constant throughout a simulation). The letter w will denote an element $w = (x_0, u)$ of $X_0 \times L^2$. Standard assumptions apply - see [9], [10].

Assumption 2.1: For every $w = (x, u) \in X_0 \times L^2$, there exists a unique solution $s_w(\cdot) :\mapsto \mathbb{R}^n$ to the ODE (1). The solution s_w is absolutely continuous. The flow F is locally bounded, that is, for all compact subsets $S \subset [0, T] \times X_0 \times U$, there exists $m > 0$ such that $F(S) \subset m\mathbf{B}$, where $\mathbf{B}$ is the unit ball centered at 0.

We formally capture specifications regarding the correct system behavior using Metric Temporal Logic (MTL) formulae [6]. An MTL formula is a formal logical statement expressing some property that the system must satisfy. It is built by combining *atomic propositions* using logical and temporal operators. Logical operators are the *conjunction* ($\wedge$), *disjunction* ($\vee$), *negation* ($\neg$), and *implication* ($\rightarrow$). The temporal operators include *eventually* ($\Diamond_{\mathcal{I}}$), *always* ($\Box_{\mathcal{I}}$) and *until* ($U_{\mathcal{I}}$). For example, MTL can capture the requirement that "all the trajectories s_w attain a value in the set $[10, +\infty)$" ($\Diamond_{[0,\infty)} s_w(t) \geq 10$), or that "whenever the value of s_w exceeds 10, then it should go below 7 within 5 sec and remain there for at least 10 sec" ($\Box(s_w(t) \geq 10 \rightarrow \Diamond_{[0,5]} \Box_{[0,10]} s_w(t) \leq 7)$).

If we associate a set $\mathcal{O}(p)$ with each atomic proposition $p \in AP$ such that p is true of the states in $\mathcal{O}(p)$, then the above properties can be written as $\Diamond_{[0,\infty)} p_1$ with $\mathcal{O}(p_1) = [10, +\infty)$, and $\Box(\neg p_1 \rightarrow \Diamond_{[0,5]} \Box_{[0,10]} p_2)$ with $\mathcal{O}(p_2) = (-\infty, 7]$. We can quantify how robustly a system trajectory s_x satisfies a specification ϕ in MTL [8]. Namely, we define a function of the trajectory, $\rho_\phi(s_w)$, which takes positive values if s_w satisfies ϕ and negative values otherwise. Its magnitude $|\rho_\phi|$ quantifies how well the specification is satisfied or falsified. The process of falsifying a specification ϕ, i.e. detecting a system behavior that does not satisfy ϕ, can thus be re-cast as the problem of finding trajectories with negative ρ_ϕ-values. On the other hand, the optimal control can be posed as the problem of maximizing the positive robustness. Since the solutions to (1) are assumed unique, the search can be performed over the initial states $x \in X_0$ (including any system parameters) and control signals $u \in L^2$, and can be improved by computing local descent directions for ρ_ϕ.

Problem 1: Given $x \in X_0$, $u \in L^2$, and a formula ϕ, find a vector $dx \in \mathbb{R}^n$ and signal $du \in L^2$ such that there exists an $\overline{h} > 0$ for which

$$\rho_\phi(x + h \cdot dx, u + h \cdot du) < \rho_\phi(x, u) \; \forall \; h \in (0, \overline{h})$$

A general MTL formula will involve multiple propositions p_i and their sets $\mathcal{O}(p_i)$. The following proposition simplifies our task:

Proposition 2.1: Consider an MTL formula ϕ and a trajectory s_w of (1) such that $[\![\phi, \mathcal{O}]\!](s_w, 0) > 0$. If assumption 2.1 holds, and for each $p \in AP$, $\mathcal{O}(p)$ is a closed half-space, then there exist a *critical time* $t_r \in [0, T]$ and a *critical proposition* $p \in AP$ which appears in ϕ such that $\rho_\phi(w) = \inf_{z \in \mathcal{O}(p)} \|s_w(t_r) - z\|$.

In this paper, we derive the descent vector relative to only one $\mathcal{O}(p)$ at a time; this is the content of Problem 2. The choice of which $\mathcal{O}(p)$ to descend towards at any given time is decided by the following heuristic: the current target set is always the set $\mathcal{O}(p)$ where p is the critical proposition defined in Proposition 2.1. Other heuristics are possible. By focusing on one $\mathcal{O}(p)$, the problem is reduced to falsification of a safety formula, of the form: $\phi = \Box(\neg p)$ where $\mathcal{O}(p) = \mathcal{U}$ is the set of 'unsafe' system states.

The robustness then reduces to:

$$\rho_\phi(x, u) = f(x, u) \triangleq \min_{0 \leq t \leq T} d_{\mathcal{U}}(s_{(x,u)}(t)) \tag{2}$$

where $d_{\mathcal{U}}(y) \triangleq \inf_{z \in \mathcal{U}} \|y - z\|$ is the distance of a point y from $\mathcal{U}$.

The function f is non-differentiable, and generally non-convex. The special problem is then:

Problem 2: Given $x \in X_0$, $u \in L^2$, $\mathcal{U} \subset \mathbb{R}^n$, and f defined in (2), find $dx \in \mathbb{R}^n$ and $du \in L^2$ such that there exists an $\overline{h} > 0$ for which

$$f(x + h \cdot dx, u + h \cdot du) < f(x, u) \; \forall \; h \in (0, \overline{h})$$

If we treat $H \triangleq X \times L^2$ as a Hilbert space, then it can be seen that $dw = (dx, du) \in H$ is a descent direction in H. In the remainder of this paper, we will write L^2 for L^2 with no risk of ambiguity since T is fixed throughout.

III. Computing a Descent Direction

Recall that $H = X_0 \times L^2$ is the Hilbert search space, and let $w \in H$ be an element of that space. For convenience we define $s_w(t) = s_{x_0}(t; u)$.

Before continuing we will prove a result that will allow us to calculate our descent direction using a convex differentiable manifold.

Theorem 3.1: Let $w_1 \in H$ with critical time $t_{r,1}$ as defined in Proposition 2.1. Define

$$z(t; w) = \text{argmin}_{z \in \mathcal{U}} \|\bar{s}(t; w) - z\|, \tag{3}$$

and

$$J(w) = \|\bar{s}(t_{r,1}; w) - z(t_{r,1}; w_1)\|. \tag{4}$$

Suppose that there exists $w_2 \in H$ with critical time $t_{r,2}$ such that $J(w_2) < J(w_1)$. Then the robustness of the trajectory $s_{w_2}(\cdot)$ is smaller than that of $s_{w_1}(\cdot)$: $f(w_2) < f(w_1)$.

Proof: By (2) we see that

$$\begin{aligned} f(w_2) &= \min_{0 \leq t \leq T} \inf_{z \in \mathcal{U}} \|s_{w_2}(t) - z\|, \\ &\leq J(w_2) < J(w_1) = f(w_1) \end{aligned}$$

■

Our goal is to minimize the robustness. To do so we generate a sequence $(w_i) \in H$ such that $f(w_{i+1}) < f(w_i)$ as follows: first, w_0 is given. Then, we iteratively generate w_{i+1} from w_i by identifying a critical time $t_{r,i}$ and the corresponding closest unsafe point $z(t_{r,i}, w_i)$. We define the function

$$J_i(w) \triangleq G(s_w(t_{r,i})) \triangleq \|z(t_{r,i}, w_i) - s_w(t_{r,i})\| \qquad (5)$$

We then calculate a descent direction $\hat{w} \in H$ and set $w_{i+1} = w_i + h\hat{w}$, where h is the step-size. The step-size is adapted on-line: increased if a descent is obtained, and reduced if no descent is achieved. Note that with respect to $s_w(t_{r,i})$ (5) is differentiable everywhere except for the origin, at which point the trajectory has reached the unsafe set and falsification has been shown.

We now adapt the results presented in [10] to find an update direction $\hat{w}$ for w that locally decreases $J_i(w)$, which in turn will decrease the robustness, as per Thm.3.1.

Let $dJ_i(w; \hat{w})$ be the Fréchet derivative of $J_i(w)$ in the direction $\hat{w}$, and let $\hat{x}_0$ and $\hat{u}$ be the projections of $\hat{w}$ onto X_0 and L^2. This derivative can be written as a scalar valued linear functional of $\hat{w}$ as follows:

$$dJ_i(w; \hat{w}) \triangleq \langle q, \hat{w} \rangle \triangleq \int_0^T q_u(\tau)\hat{u}(\tau)\, \mathrm{d}\tau + q_{x_0}^T \hat{x}_0, \qquad (6)$$

where q_{x_0} and q_u are the projections of $q \in H$ onto X_0 and U. For brevity we shall use the following notations

$$\begin{aligned}
\frac{\partial G}{\partial x} &\triangleq \left.\frac{\partial G}{\partial x}\right|_{s(t_{r,i};x_0,u)} \in \mathbb{R}^{1\times n}, \\
\frac{\partial F(t)}{\partial x} &\triangleq \left.\frac{\partial F}{\partial x}\right|_{(t,s(t;x_0,u),u)} \in \mathbb{R}^{n\times n}, \\
\frac{\partial F(t)}{\partial u} &\triangleq \left.\frac{\partial F}{\partial u}\right|_{(t,s(t;x_0,u),u)} \in \mathbb{R}^{n\times m}.
\end{aligned}$$

Let $ds_w(t; \hat{w})$ represent the functional derivative of $s_w(t)$ in the direction $\hat{w}$. Using the Taylor series based approach in [10] we see that

$$\begin{aligned}
dJ_i(w, \hat{w}) &= \frac{\partial G}{\partial x} s_w(t_{r,i}; \hat{w}), \qquad (7)\\
ds_w(t; \hat{w}) &= \\
&\int_0^t \left(\frac{\partial F(\tau)}{\partial x} ds_w(\tau; \hat{w}) + \frac{\partial F(\tau)}{\partial u}\hat{u}(\tau) \right) d\tau \qquad (8)
\end{aligned}$$

Now suppose that

$$\begin{aligned}
ds_w(t; \hat{w}) &= \langle p(t), \hat{w} \rangle \\
&= \int_0^t p_u(t,\tau)\hat{u}(\tau)d\tau + p_{x_0}(t)^T \hat{x}_0. \qquad (9)
\end{aligned}$$

Plugging this equation into the right hand side of (8), rearranging terms and swapping the order of integration and equating terms with (9) yields

$$\begin{aligned}
p_u(t,\tau) &= \int_\tau^t \frac{\partial F(\xi)}{\partial x} p_u(\xi,\tau) d\xi + \frac{\partial F(\tau)}{\partial u} \\
p_{x_0}(t) &= \int_0^t \frac{\partial F(\tau)}{\partial x} p_{x_0}(\tau) d\tau
\end{aligned}$$

We can solve for $p_u(t,\tau)$ and $p_{x_0}(t)$ by solving the initial value problem

$$\begin{aligned}
\frac{d}{dt} p_u(\cdot,\tau) &= \frac{\partial F(t)}{\partial x} p_u(t,\tau), \qquad (10)\\
\frac{d}{dt} p_{x_0} &= \frac{\partial F(t)}{\partial x} p_{x_0}(t), \\
p_u(\tau,\tau) &= \frac{\partial F(\tau)}{\partial u}, \\
p_{x_0}(0) &= \mathbf{I}_{n\times n}
\end{aligned}$$

By combining (7) with (9) and comparing to (6), we find that

$$\begin{aligned}
q_u(\tau) &= \frac{\partial G}{\partial x} p_u(t_{r,i}, \tau) \\
q_{x_0} &= \frac{\partial G}{\partial x} p_{x_0}(t_{r,i})
\end{aligned}$$

Thus, to find a $dJ_i(w; \hat{w})$ that is negative, we can set $\hat{w}$ in the inner product (6) to be $-q$, that is

$$\begin{aligned}
\hat{u}(\tau) &= -\frac{\partial G}{\partial x} p_u(t_{r,i}, \tau), \qquad (11)\\
\hat{x}_0 &= -\frac{\partial G}{\partial x} p_{x_0}(t_{r,i}). \qquad (12)
\end{aligned}$$

In order to calculate the update direction for a continuous input on a digital computer, we represent the input function by a linear combination of finitely many basis functions. In each example presented in Section IV, we consider a basis of either rectangular or triangular pulses that are evenly spaced through time. Then we can calculate an exact update to the input parameters using, for example, the sensitivity analysis tools provided by SundialsTB toolbox [11].

Our approach is a local descent optimization; it can easily be used within a multi-start scheme (where a local optimization is performed from several initial points in the search space), as illustrated in Example 2.

IV. Experiments

Example 1: Our first example is a *linear* 81-dimensional RLC circuit. The sensitivity ODE is then itself linear [9]. The safety specification requires the output voltage to always be less than 1.5V:

$$\phi_{RLC} = \Box(x_{41} \leq 1.5)$$

Starting from $x_0 = [0, 0]$ and a constant input of 0, the descent converges to a falsifying trajectory shown in Fig.1, with a near-step input. △

Example 2: The following example is 3-dimensional system modeling the variation of glucose and insulin levels in the blood, following a meal intake [12]. The model was developed to help design insulin infusion schedules for diabetes patients, e.g. as done in [13]. It is given by

$$\dot{x}(t) = \begin{bmatrix} -p_1 x_1(t) - x_2(t)(x_1(t) + G_b) + Be^{-kt} \\ -p_2 x_2(t) + p_3 x_3(t) \\ -n(x_3(t) + I_b) + \frac{1000}{60V_I} u(t) \end{bmatrix}$$

State x_1 represents the level of glucose in the blood plasma above a given basal value G_b, x_2 is proportional

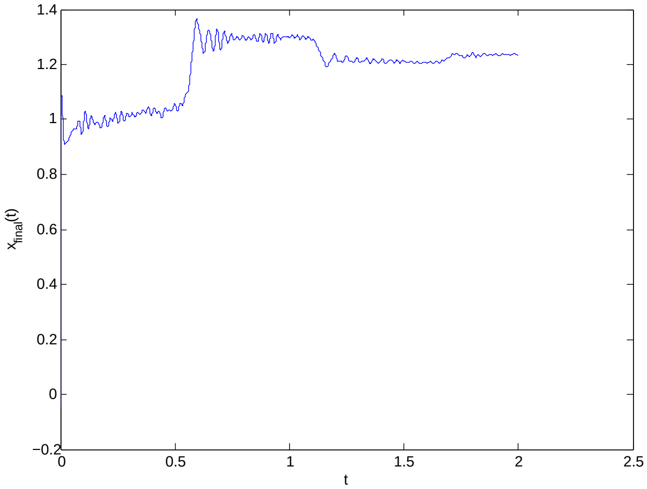


Fig. 1: Example 1 (RLC). Falsifying trajectory returned by descent.

to the level of insulin that is effective in controlling blood glucose level, and x_3 represents the level of insulin above a given basal value I_b. The search space for $[x_1, x_2, x_3]$ is $[6, 10] \times [0.05, 0.1] \times [-0.1, 0.1]$. The input $u(t)$ represents a direct infusion of insulin meant to control the glucose level. $u(t)$ is therefore also referred to as an 'infusion schedule'. The p_i, n, B and k are model parameters. We fix duration $T = 200$. Consider first the following specification:

$$\phi_1 = \Box_{[0,20]} x_1 \in [-2, 10] \wedge \Box_{[20,200]} x_1 \in [-1, 1]$$

ϕ_1 specifies that glucose level should remain in the range $[-2, 10]$ for the first 20 seconds, and should remain in the range $[-1, 1]$ for the last 180 seconds. Our goal is to design an infusion schedule such that the glucose level satisfies ϕ_1. This can be posed as the problem of falsifying $\neg\phi_1$. We decided to search over the initial values of the ODE (1), the input u, and the parameter p_3: p_3 varies between diabetes patients, and its estimated value for normal subjects is 1.3e-5 [12]. Larger p_3 values imply that the insulin injection $u(t)$ will have a greater effect on the plasma glucose level $x_1(t)$. The search range for p_3 is therefore fixed to [1e-5,1e-3]. Thus the outcome of the optimization is a set of initial conditions (patient's state at meal time), a continuous infusion schedule, and a class of patients (as described by the parameter p_3) for which the schedule is appropriate.

Starting from a constant input signal at 0.1, and $[x_0, p_3]$ =[8, 0.08, 0, 1.3e-5], the initial robustness is 1.5399. The optimization returned a decision w with robustness 0.678 in 12 iterations. The final p_3 value is 2.03e-5. It is interesting to note that the final input shows an injection at the beginning of time, followed by a constant infusion (Fig. 2). This is the type of infusion schedule advocated as being optimal in [12, Section III], under the nominal p_3 value and the cost function $C(u) = \int_0^T x_1^2(t)dt$. Our descent method produced this schedule with relatively little computational effort, and provides more information on the classes of patients for which it is appropriate.

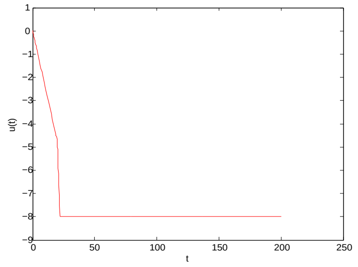
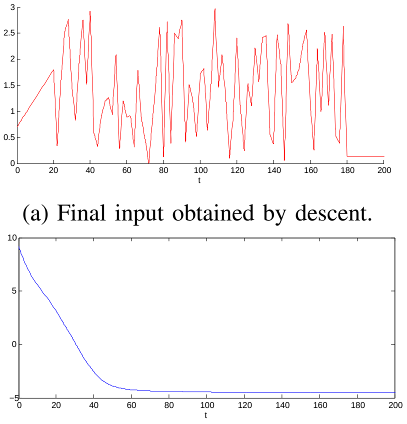


Fig. 2: Example 2 (Insulin). Final input $u(t)$ found by descent algorithm, scaled to highlight the initial impulse.



(a) Final input obtained by descent.

(b) Final trajectory returned by descent.

Fig. 3: Example 2 (Insulin). A profile obtained by multi-start.

Consider next the following specification

$$\phi_2 = \Box(p_{hg} \wedge \mathbf{X}\ \neg p_{hg} \rightarrow \Diamond_{[0,10]}(\Box_{[0,20]} \neg p_{hg}))$$

with $\mathcal{O}(p_{hg}) = \{x \mid x_1 \geq 9.44\}$. ϕ_2 expresses that if the glucose level rises above 9.44 mmol/L (meaning hyperglycemia), it should dip below 9.44 within 10secs and stay there for at least 20secs. Starting from $[6, 0.05, 0.1, 0.0001]$, the descent keeps the initial value of x_1, since one way to satisfy ϕ_2 is to never go above the dangerous level of 9.44. To see how the schedule might need to be adapted for different values of p_3, we ran a random multi-start simulation, were we uniformly sample the search space (we used 50 samples), and from each sample we run a local descent. Fig.3a shows a falsifying input profile significantly different from the one in Fig.2, with p_3 =8e-4. Whether the shown input schedule is practicable with today's technology is not assessed, but the point is that different classes of patients (and different initial states) might require different schedules. The resulting glucose trajectory is shown in Fig.3b, demonstrating a quick decrease towards safer levels of glucose. $\triangle$

Example 3: This example is a 6-dimensional system that models a quadrotor moving through a vertical plane [10].

The system dynamics are given by:

$$\begin{aligned}\ddot{X} &= \mu(w_X - \dot{X}) - \frac{u_1}{m}\sin\theta\\ \ddot{Y} &= \mu(w_Y - \dot{Y}) - g + \frac{u_1}{m}\cos\theta\\ \ddot{\theta} &= u_2\end{aligned}$$

Here m denotes the object's mass, g denotes gravitational acceleration, μ is the coefficient of friction with the air, $w_.$ is the wind velocity along each axis, and u_i is the control input. We define the XY coordinate of the object's center of mass as the system's output.

For this example, we consider the task of verifying the safety of controllers that drive the quadrotor from one side of a hill over to the other side without hitting the hill or the ground. On the other side is a desired goal region, which the quadrotor must reach within 12 seconds and stay there afterwards. This requirement can be represented by the MTL specification,

$$\phi = \Box_{[12,\infty]}p_1 \wedge \Box_{[0,12]}\neg p_2 \wedge \Box_{[0,12]}\neg p_3,$$

Here, p_1 represents the goal set, p_2 represents the ground, and p_3 represents the hill. The sets $\mathcal{O}(p_1), \mathcal{O}(p_2), \mathcal{O}(p_3)$ used in our experiments are shown graphically in Figure 4.

First, we designed a reference tracking feedback controller by linearizing the system around a hovering operating point. The system is assumed to be initially hovering at location $[x, y] = [-8, 2]$ and that there is no wind during the simulation. Although this controller works well in the nominal case, it is prudent to consider what happens if the system does not begin at the expected initial state, or if there is any wind disturbance. To this end, we treat the wind velocity as an input to the system bounded by ±2 m/s. We use the algorithm presented in this paper to search over bounded sets of initial conditions and horizonal wind profiles.

When the optimization was first run, the system was falsified mainly by shifting the intial x position to the left and by having the wind blow the quadrotor to the left. The updated initial condition and wind disturbance thus caused the quadrotor to fly into the ground in a way that is not expected for the nominal performance. This algorithm was able to quickly find this major design flaw, as shown in Figure 4.

After fixing the reference signal to maintain a height of 2 for all points to the left of the starting location, the optimization was rerun. After running for 7 iterations, the algorithm found that the initial conditions $[x, y, \theta, \dot{x}, \dot{y}, \dot{\theta}] = [0, 0, 0.005, 0, 0, 0.098]$ and the wind profile shown in Figure 5 was able to falsify ϕ, specifically by slowing the horizontal progression of the quadrotor so that it was not in the goal set at time $t = 12$.

V. Related Work

The work that appears in [14], [15] is the closest to the results that we present here in terms of methods utilized. In [15], sensitivity analysis is used to compute neighborhoods of trajectories that always remain close enough and, thus, perform coverage of the initial conditions. These results were later extended in [14] to estimating parameter ranges and initial conditions for the satisfaction of STL properties. Even though our solution leads to sensitivity calculations, our objective is very different from the work in [14]. Our goal is to develop the local search tools needed in order to improve the performance of stochastic MTL falsification/optimal control methods [7], [16]. Moreover, we can search simultaneously over the initial conditions, parameters and the input signals. Finally, stochastic falsification methods avoid the state-explosion problems that occur when attempting to cover a high-dimensional set of parameters.

Different versions of the optimal control problem under Linear Temporal Logic (LTL) specifications are presented in [17], [18]. The authors in [17] take a mathematical programming approach, while [18] develops an automata based approach. Unlike MTL, LTL does not allow the specification of timing intervals for the Until operator $\mathcal{U}_\mathcal{I}$ (and by extension, the Always and Eventually operators). This timing interval is necessary for expressing real-time constraints on the succession of events, which is important in many control applications. The problem of optimal control for vehicle routing for MTL specifications is addressed in [19]. However, the results in [19] apply to specifications without nested temporal operators and finite transition systems.

Our work in this paper can also be viewed as an optimal control problem over hybrid systems. Since in our *implementation* we parameterize the input signals with a finite number of parameters at specific points in time, we can view the system as a parametric hybrid automaton where the mode switches occur at specific time instants. Then the goal is to compute the system parameters and initial conditions such that the MTL robustness is minimized. However, we remark that our theoretical results do not require the finite parameterization of the input function space.

In terms of optimal control over hybrid systems, [20] calculates numerically a descent direction for a class of switched systems. First, we remark that our original cost

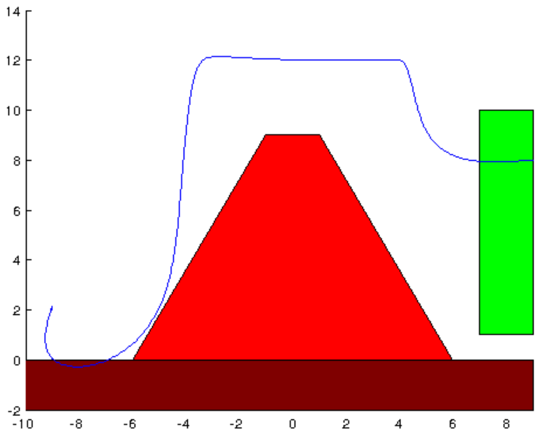


Fig. 4: Falsification of Quadrotor with poor reference signal

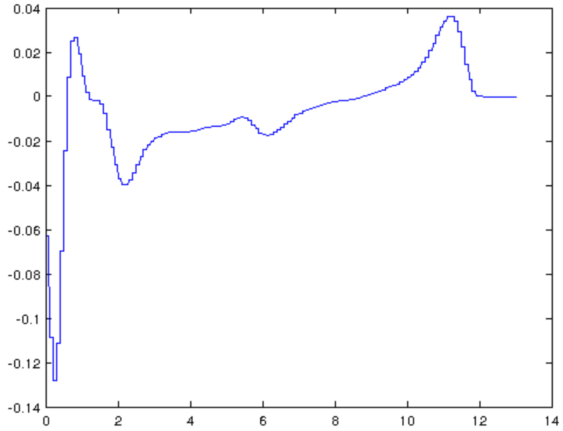


Fig. 5: Falsifying Wind Profile for Quadrotor System

function is non-differentiable so it does not satisfy the assumptions in [20]. In our current *numerical implementation* each subproblem that we solve, i.e., descent to a specific set, satisfies the assumptions in [20]. Thus, our solution could be utilizing the results in [20] to solve more general problems in the future. Similar remarks hold for the optimal control problem formulated in [21]. Finally, in [22], we demonstrated that in the case of linear hybrid systems improvements in the convergence rate of stochastic search algorithms can be achieved by adding a local search step.

VI. Conclusions

We have presented the derivation of the equations that can be used for the computation of Metric Temporal Logic (MTL) robustness descent vectors in the set of initial conditions, parameter space and input function space for nonlinear dynamical systems. These results are necessary for enabling "gray box" MTL falsification and open loop control methods for dynamical systems. One important advantage of the proposed approach is that our framework can be readily used for MTL falsification and/or optimal control methods within any Model Based Development (MBD) tool that supports sensitivity analysis. For instance, Simulink can provide such functionality [23]. In the future, we will focus on extending our new approach to hybrid systems using, for instance, the decomposition method proposed in [24]. Also of interest is the interplay between stochastic search methods [25] and local gradient descent [26].

References

[1] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Two Volume Set*, 2nd ed. Athena Scientific, 2000.

[2] K. Ogata, *Modern Control Engineering*, 4th ed. Prentice Hall, 2001.

[3] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The GRASP multiple micro uav testbed," *IEEE Robotics and Automation Magazine*, vol. 17, no. 3, pp. 56–65, 2010.

[4] Y. S. R. Annapureddy, C. Liu, G. E. Fainekos, and S. Sankaranarayanan, "S-taliro: A tool for temporal logic falsification for hybrid systems," in *Tools and algorithms for the construction and analysis of systems*, ser. LNCS, vol. 6605. Springer, 2011, pp. 254–257.

[5] A. Donze, "Breach, a toolbox for verification and parameter synthesis of hybrid systems," in *Computer Aided Verification*, ser. LNCS. Springer, 2010, vol. 6174, pp. 167–170.

[6] R. Koymans, "Specifying real-time properties with metric temporal logic." *Real-Time Systems*, vol. 2, no. 4, pp. 255–299, 1990.

[7] H. Abbas, G. E. Fainekos, S. Sankaranarayanan, F. Ivancic, and A. Gupta, "Probabilistic temporal logic falsification of cyber-physical systems," *ACM Transactions on Embedded Computing Systems*, vol. 12, no. s2, May 2013.

[8] G. Fainekos and G. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, September 2009.

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

[10] A. K. Winn and A. Julius, "Optimization of human generated trajectories for safety controller synthesis," in *American Control Conference (ACC), 2013*, 2013, pp. 4374–4379.

[11] R. Serban and A. Hindmarsh, "Cvodes: the sensitivity-enabled ode solver in sundials," in *Proceedings of IDETC/CIE*, 2005.

[12] M. Fisher, "A semiclosed-loop algorithm for the control of blood glucose levels in diabetics," *Biomedical Engineering, IEEE Transactions on*, vol. 38, no. 1, pp. 57–61, 1991.

[13] 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.

[14] A. Donze, E. Fanchon, L. M. Gattepaille, O. Maler, and P. Tracqui, "Robustness analysis and behavior discrimination in enzymatic reaction networks," *PLoS ONE*, vol. 6, no. 9, p. e24246, 09 2011.

[15] A. Donze and O. Maler, "Systematic simulation using sensitivity analysis," in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 4416. Springer, 2007, pp. 174–189.

[16] 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.

[17] S. Karaman, R. Sanfelice, and E. Frazzoli, "Optimal control of mixed logical dynamical systems with linear temporal logic specifications," in *IEEE Conf. on Decision and Control*, 2008.

[18] E. A. Gol and C. Belta, "Time-constrained temporal logic control of multi-affine systems," *Nonlinear Analysis: Hybrid Systems*, vol. 10, pp. 21–33, 2013.

[19] S. Karaman and E. Frazzoli, "Vehicle routing problem with metric temporal logic specifications," in *IEEE Conference on Decision and Control*, Dec. 2008, pp. 3953 –3958.

[20] H. Gonzalez, R. Vasudevan, M. Kamgarpour, S. S. Sastry, R. Bajcsy, and C. J. Tomlin, "A descent algorithm for the optimal control of constrained nonlinear switched dynamical systems," in *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*, ser. HSCC '10. ACM, 2010, pp. 51–60.

[21] H. Axelsson, Y. Wardi, M. Egerstedt, and E. Verriest, "Gradient descent approach to optimal mode scheduling in hybrid dynamical systems," *Journal of Optimization Theory and Applications*, vol. 136, no. 2, pp. 167–186, 2008.

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

[23] Z. Han and P. J. Mosterman, "Towards sensitivity analysis of hybrid systems using simulink," in *Proceedings of the 16th international conference on Hybrid systems: computation and control*. ACM, 2013, pp. 95–100.

[24] A. Zutshi, S. Sankaranarayanan, J. V. Deshmukh, and J. Kapinski, "A trajectory splicing approach to concretizing counterexamples for hybrid systems," in *IEEE Conference on Decision and Control*, 2013.

[25] H. Abbas and G. Fainekos, "Linear hybrid system falsification through local search," in *Automated Technology for Verification and Analysis, 9th International Symposium*, ser. LNCS, vol. 6996, Taipei, Taiwan, October 2011.

[26] D. Hristu and K. Morgansen, "Limited communication control," *Systems & Control Letters*, vol. 37, pp. 193–205, 1999.