Prescribed Performance Control Guided Policy Improvement for Satisfying Signal Temporal Logic Tasks

Peter Varnai and Dimos V. Dimarogonas[1]

***Abstract*— Signal temporal logic (STL) provides a user-friendly interface for defining complex tasks for robotic systems. Recent efforts aim at designing control laws or using reinforcement learning methods to find policies which guarantee satisfaction of these tasks. While the former suffer from the trade-off between task specification and computational complexity, the latter encounter difficulties in exploration as the tasks become more complex and challenging to satisfy. This paper proposes to combine the benefits of the two approaches and use an efficient prescribed performance control (PPC) base law to guide exploration within the reinforcement learning algorithm. The potential of the method is demonstrated in a simulated environment through two sample navigational tasks.**

I. INTRODUCTION

Temporal logics (TLs) have gained considerable attention for their convenience and expressive power in specifying complex tasks for a variety of systems. While the field has its roots in formal verification theory [1], recent successful applications include areas in control such as task and motion planning for robotic systems [2]. In this paper, we focus on the controller synthesis problem for nonlinear systems subject to tasks specified by signal temporal logic (STL), a temporal logic originally introduced in the context of monitoring [3]. In STL, the logical predicates stem from real-valued functions of the system states and the temporal specifications include explicit timing requirements.

STL task specifications have lately been studied from a control perspective in the sense of how to ensure their satisfaction. Proposed approaches for controller synthesis include model predictive control (MPC) [4], [5], barrier function- [6], and prescribed performance control (PPC)-based methods [7]. These methods rely heavily on knowledge of system dynamics and exhibit a trade-off between their computational complexity and the range of system dynamics and STL task fragments they can handle.

The recent use of reinforcement learning (RL) methods in the field of robotics [8] and linear temporal logics [9] have motivated research into their applicability for satisfying STL tasks as well. RL is able to deal with unknown dynamics and allows real-time computational expenses to be transferred offline by training from gathered experiences. For the purpose of task satisfaction, an STL description of the task becomes suitable because STL is equipped with various robustness measures that quantify the degree of its satisfaction for a given system trajectory in its entirety [10]. Therefore, these measures inherently constitute a descriptive reward to be maximized for task satisfaction and have been shown to be effective for trajectory-based RL methods such as temporal logic policy search (TLPS) [11]. TLPS is based on the policy improvement with path integrals (PI^2) algorithm [12], a form of sampling-based methods also studied for solving linear temporal logic tasks [13]. The STL robustness measures have also been adapted to serve as step-based intermediate rewards for Q-learning [14]. Practical implementations of RL are hindered by the high cost of trial and error (e.g., time-consuming sampling) on which these algorithms rely.

The main contribution of this paper is to combine the benefits of model-based STL control laws and the reinforcement learning approaches. More specifically, we propose to use a simple and efficient PPC law as a basis for the PI^2 algorithm in order to approximately solve optimal control problems for nonlinear systems subject to STL task specifications using partial knowledge of the system dynamics. The learning part allows (locally) optimal solutions to be found under environmental uncertainties, while the base law aids in satisfying the STL task and thus leads to effective and robust exploration towards the optimum. The advantages of the approach are illustrated by two simulated scenarios. Although our study regards the trajectory-based PI^2 algorithm, the idea of guided exploration should also be applicable to other RL methods.

Due to space limitations, some details regarding the presented results have been omitted from this paper. For an extended version, the interested reader is referred to [15].

II. PRELIMINARIES

A. System description

We consider nonlinear systems of the following form:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + g(\boldsymbol{x})\boldsymbol{u} + \boldsymbol{w}, \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0, \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{u} \in \mathbb{R}^m$, and $\boldsymbol{w} \in \mathbb{R}^n$ are the state, input, and process noise, respectively. The noise $\boldsymbol{w}$ is assumed to be zero-mean Gaussian white noise with covariance $\boldsymbol{\Sigma}_w \in \mathbb{R}^{n\times n}$, while the functions $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ are locally Lipschitz continuous with $g(\boldsymbol{x})g^{\mathrm{T}}(\boldsymbol{x})$ positive definite for all $\boldsymbol{x} \in \mathbb{R}^n$. The system starts in an initial state $\boldsymbol{x}_0 \in \mathbb{R}^n$.

A *trajectory* $\tau_{[0,T]}$ of the system (1) is defined by the signals $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$ throughout its evolution from $\boldsymbol{x}_0$ under the input $\boldsymbol{u}(t)$ during $t \in [0, T]$. For brevity, we omit the time bounds and simply denote the trajectory $\tau_{[0,T]}$ by τ. Signal values at time t are also abbreviated as, e.g., $\boldsymbol{x}_t := \boldsymbol{x}(t)$.

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, the Swedish Research Council (VR), the SSF COIN project, and the EU H2020 Co4Robots project.

[1]Both authors are with the Division of Decision and Control Systems, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, 114 28 Stockholm, Sweden. varnai@kth.se (P. Varnai), dimos@kth.se (D. V. Dimarogonas)

B. Signal temporal logic (STL)

STL is a predicate logic defined over continuous-time signals [3]. The *predicates* μ are evaluated according to a corresponding function $h^\mu : \mathbb{R}^n \to \mathbb{R}$ as true($\top$) if $h^\mu(\boldsymbol{x}) \geq 0$ or false($\bot$) if $h^\mu(\boldsymbol{x}) < 0$. Predicates can be recursively combined using Boolean logic and temporal operators to form increasingly complex *formulas* (or *task specifications*) $\phi := \top \mid \mu \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \mathcal{U}_{[a,b]} \phi_2$. The time bounds of the *until* operator $\mathcal{U}_{[a,b]}$ are given as $a, b \in [0, \infty)$ with $a \leq b$. The commonly used temporal operators *eventually* and *always* follow from $F_{[a,b]}\phi = \top \mathcal{U}_{[a,b]} \phi$ and $G_{[a,b]}\phi = \neg F_{[a,b]} \neg \phi$. A signal $\boldsymbol{x}(t)$ is said to satisfy an STL expression ϕ at time t, i.e., $(\boldsymbol{x}, t) \vDash \phi$, by the following semantics [7]: $(\boldsymbol{x}, t) \vDash \mu \Leftrightarrow h^\mu(\boldsymbol{x}(t)) \geq 0$; $(\boldsymbol{x}, t) \vDash \neg\phi \Leftrightarrow \neg((\boldsymbol{x}, t) \vDash \phi)$; $(\boldsymbol{x}, t) \vDash \phi_1 \wedge \phi_2 \Leftrightarrow (\boldsymbol{x}, t) \vDash \phi_1 \wedge (\boldsymbol{x}, t) \vDash \phi_2$; and $(\boldsymbol{x}, t) \vDash \phi_1 \mathcal{U}_{[a,b]} \phi_2 \Leftrightarrow \exists t_1 \in [t+a, t+b]$ such that $(\boldsymbol{x}, t_1) \vDash \phi_2$ and $(\boldsymbol{x}, t_2) \vDash \phi_1$ for $\forall t_2 \in [t, t_1]$.

STL is equipped with various robustness measures ρ that quantify the extent to which a task specification is satisfied [10]. In this work, we employ the so-called *spatial robustness* metric, evaluated as follows for formulas used herein: $\rho^\mu(\boldsymbol{x}, t) = h^\mu(\boldsymbol{x}(t))$; $\rho^{\neg\phi}(\boldsymbol{x}, t) = -\rho^\phi(\boldsymbol{x}, t)$; $\rho^{\phi_1 \wedge \phi_2}(\boldsymbol{x}, t) = \min\left(\rho^{\phi_1}(\boldsymbol{x}, t) \rho^{\phi_2}(\boldsymbol{x}, t)\right)$; $\rho^{F_{[a,b]}\phi}(\boldsymbol{x}, t) = \max_{t' \in [t+a, t+b]} \rho^\phi(\boldsymbol{x}, t')$; and finally $\rho^{G_{[a,b]}\phi}(\boldsymbol{x}, t) = \min_{t' \in [t+a, t+b]} \rho^\phi(\boldsymbol{x}, t')$. An important property of this robustness metric is that the positiveness of its value indicates whether the corresponding task specification is satisfied.

C. Prescribed performance control (PPC) for STL tasks

In the following, we review a gradient-based control law advocated by [7] for satisfying STL tasks for dynamics of the form (1). The method uses ideas from prescribed performance control [16] to guide the robustness metric of logical predicates in time, thereby ensuring their desired temporal behavior. The resulting control law will serve as a basis for guiding learning, as detailed in Section IV-B.

Consider the following subset of STL formulas:

$$\psi := \top \mid \mu \mid \neg\mu \mid \psi_1 \wedge \psi_2, \tag{2a}$$

$$\phi := G_{[a,b]}\psi \mid F_{[a,b]}\psi \mid F_{[a_1,b_1]} G_{[a_2,b_2]} \psi, \tag{2b}$$

where the robustness metric ρ^ψ associated with each *non-temporal formula* ψ is assumed concave or convex. The main idea of PPC is to achieve satisfaction of the *temporal formula* ϕ by controlling the evolution of ρ^ψ in time such that it stays bounded between two prescribed curves (a *funnel*) related to the *always* or *eventually* operators. For example, in case of an *always* task, the lower curve remains at or above 0 during the $[a, b]$ time interval to ensure $\rho^\psi(\boldsymbol{x}(t)) \geq 0$ and therefore satisfaction of the task ϕ as $\rho^\phi = \min_{t \in [a,b]} \rho^\psi(\boldsymbol{x}(t)) \geq 0$. We note that the class of satisfiable tasks (2b) could be extended by studying how such funnels should be constructed.

The two prescribed boundaries for ρ^ψ are defined by a curve $\gamma(t) \in \mathbb{R}$ and a parameter $\rho_{\max} \in \mathbb{R}$. These are chosen such that the task ϕ will be satisfied if $\gamma(t) < \rho^\psi(\boldsymbol{x}(t)) < \rho_{\max}$ holds for all $t \in [a, b]$. Under some assumptions, this satisfaction is achieved by the control law:

$$\boldsymbol{u}^\phi(\boldsymbol{x}, t) = \epsilon(\boldsymbol{x}, t) g^{\mathrm{T}}(\boldsymbol{x}) \frac{\partial \rho^\psi(\boldsymbol{x})}{\partial \boldsymbol{x}}, \tag{3}$$

where ϵ is the so-called transformed error:

$$\epsilon(\boldsymbol{x}, t) = S(\xi(\boldsymbol{x}, t)) \text{ where } \xi(\boldsymbol{x}, t) = \frac{\rho_{\max} - \rho^\psi(\boldsymbol{x})}{\rho_{\max} - \gamma(t)}. \tag{4}$$

Here, the transformation function $S(\xi)$ maps the interval $(0, 1)$ to $(-\infty, \infty)$ in a monotonically increasing manner. This ensures that ρ^ψ stays within its prescribed funnel since ξ approaches 0 or 1 as the upper or lower boundaries are neared. The assumptions require the noise $\boldsymbol{w}$ to remain in some bounded set $\mathcal{W} \subset \mathbb{R}^n$ and the functions $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ to be locally Lipschitz continuous. Furthermore, it must be possible to set the derivative $\dot\rho(\boldsymbol{x})$ arbitrarily through the input term $g(\boldsymbol{x})\boldsymbol{u}$. Note, however, that knowledge of $f(\boldsymbol{x})$ in the system dynamics (1) is *not* required to evaluate $\boldsymbol{u}^\phi(\boldsymbol{x}, t)$!

To compute the derivative $\partial\rho^\psi(\boldsymbol{x})/\partial\boldsymbol{x}$ in (3), [7] uses a differentiable under-approximation for conjunctions of propositions: $\rho^{\psi_1 \wedge \psi_2}(\boldsymbol{x}, t) \approx -\ln(e^{-\rho^{\psi_1}(\boldsymbol{x}, t)} + e^{-\rho^{\psi_2}(\boldsymbol{x}, t)})$.

III. PROBLEM FORMULATION

The main problem examined herein is given as follows.

Problem 1. Consider the system (1) starting from an initial state $\boldsymbol{x}_0 \in \mathbb{R}^n$ within the time frame $t \in [0, T]$. Design control inputs subject to the constraints $\boldsymbol{u} \in \mathcal{U} \subseteq \mathbb{R}^m$ which guarantee that the system satisfies a given STL task φ composed of the conjunction of M temporal formulas of the form (2b):

$$\varphi = \phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_M, \tag{5}$$

with a robustness degree of at least $\rho_{\min} \geq 0$ and with respect to minimizing a given cost function $C(\tau) : \tau \to \mathbb{R}$ of the generated system trajectory. Only the input term $g(\boldsymbol{x})$ is considered known from the system dynamics (1).

The cost function $C(\tau)$ indicates preference for one task satisfying trajectory over another, and we assume a solution exists to the outlined optimization problem. The control inputs are sought over $t \in [0, T]$ in the form of a time-varying policy $\pi_\theta(\boldsymbol{u}_t | \boldsymbol{x}_t, t)$ parameterized by θ, which returns the input $\boldsymbol{u}_t$ to the system given the current state $\boldsymbol{x}_t$ and time t.

A similar problem has been formulated and examined for (a broader range of) completely unknown system dynamics in the context of truncated linear temporal logic (TLTL), a language comparable to STL, by [11]. Therein, the goal was to find a policy that maximizes the expected robustness measure corresponding to a general TLTL task specification. The authors proposed temporal logic policy search (TLPS), a method based on PI^2, to find such a controller, which was shown to surpass the performance of alternative state-of-the-art algorithms capable of dealing with such a problem.

This work shows that TLPS can be further improved by incorporating available knowledge of the system dynamics into the algorithm. Namely, this will be done by using the PPC control law introduced in section II-C to guide PI^2 for an increased rate of convergence and robustness to process noise. We also extend the PI^2 framework to allow optimizing system trajectories subject to STL tasks for general $C(\tau)$ costs; task satisfaction is thus treated as a constraint instead of as the target of optimization, in contrast to [11]. So far, our approach applies to the range of system dynamics (1) and STL formulas (5) to which the discussed PPC control law is applicable. We intend to extend this range and examine the method's fundamental limitations in future work.

IV. SOLUTION

The proposed solution to Problem 1 is based on policy improvement with path integrals (PI^2), a trajectory-based RL algorithm [12]. PI^2 is advantageous in case the system dynamics are (partially) unknown or if the control problem is difficult to solve, e.g., using traditional feedback controllers. This is the case as we aim at both meeting a robustness requirement for satisfying an STL formula and minimizing the trajectory cost $C(\tau)$ under input constraints with knowledge of the system dynamics limited to the input term $g(\boldsymbol{x})$.

A. The PI^2 framework

Policy improvement finds a control policy π under which the generated system trajectory τ minimizes a given objective function $J(\tau)$[1]. Here, π is modeled as a time-varying control policy over a time horizon of length T as:

$$\pi_\theta(\boldsymbol{u}_t|\boldsymbol{x}_t, t) = \hat{\boldsymbol{u}}(\boldsymbol{x}_t, t) + \boldsymbol{k}_t(\theta), \quad t \in [0, T], \tag{6}$$

where $\hat{\boldsymbol{u}}(\boldsymbol{x}_t, t) \in \mathbb{R}^m$ is a so-called *base control law* and $\boldsymbol{k}_t(\theta) \in \mathbb{R}^m$ is a feedforward term parameterized by the unknown θ. We allow degrees of freedom for every time-step in the form $\theta = \{\theta_0, \ldots, \theta_T\}$, with each $\theta_t \in \mathbb{R}^n$. A simple feedforward is then $\boldsymbol{k}_t := \theta_t$ as in [11] or [17]. Here we search for the time differentials of these terms using $\boldsymbol{k}_t = \int_0^t \theta_\tau \mathrm{d}\tau$, arguing that the optimal control actions should generally differ marginally from one time instance to another.

The PI^2 algorithm computes a (locally) optimal parameter θ that minimizes $J(\tau)$ in an iterative fashion, starting from an initial guess $\theta^{(0)}$. The main steps for the (k)-th iteration of its variant employed herein are summarized as follows from a combination of the works [11], [12], and [18]:

1) Generate $i = 1, \ldots, N$ samples of parameters $\tilde{\theta}_{t,i} = \mathcal{N}(\theta_t^{(k)}, \mathbf{C}_t^{(k)})$ for each time step t and obtain the system trajectory τ_i under each corresponding control policy $\pi_{\tilde{\theta}_i}$. The covariances $\mathbf{C}_t^{(0)} \in \mathbb{R}^{n \times n}$ are initialized by tuning and will be adapted by the algorithm. Sampling from such Gaussian distributions allows exploration of the parameter space for (locally) optimal policies.
2) Compute the cost $J_i = J(\tau_i)$ of each trajectory τ_i and a corresponding weight w_i using:
$$w_i = \frac{e^{-\frac{1}{\eta} J_i}}{\sum_{j=1}^N e^{-\frac{1}{\eta} J_j}}. \tag{7}$$
The *temperature parameter* $\eta > 0$ controls the aggressiveness of selecting greedily from the sampled trajectories towards minimizing the objective $J(\tau)$.
3) Update the policy parameters and apply covariance matrix adaptation using weighted averaging [18]:
$$\theta_t^{(k+1)} = \sum\nolimits_{i=1}^N w_i \tilde{\theta}_{t,i}, \tag{8a}$$
$$\mathbf{C}_t^{(k+1)} = \mathbf{C}_{t,\min} + \sum_{i=1}^N w_i (\tilde{\theta}_{t,i} - \theta_t)(\tilde{\theta}_{t,i} - \theta_t)^{\mathrm{T}}. \tag{8b}$$
The term $\mathbf{C}_{t,\min}$ enforces a minimal amount of exploration in subsequent iterations.

The PI^2 algorithm repeats these steps until a given number of K iterations or convergence of $\theta^{(k)}$.

Remark 1. The work [12] lays out the theoretical foundation of PI^2 and proves convergence for specific objectives $J(\tau)$. However, the algorithm and its variants are said to perform well even in case the required assumptions do not hold.

B. Base control law

The base control law in (6) is often taken as the linear state feedback $\hat{\boldsymbol{u}}(\boldsymbol{x}_t) = -\mathbf{K}_t \boldsymbol{x}_t$ such as in [11] and [17]. This choice is general enough to handle cases where the system dynamics are unknown. However, considering the case where there is partial information available in the knowledge of the input matrix $g(\boldsymbol{x})$, we propose to take advantage of the existing PPC controller introduced in Section II-C in order to guide the search procedure towards satisfying the given STL task. Using the PPC law as a basis for PI^2 offers two main advantages over the linear state feedback. First, it leads to faster convergence of the algorithm, which is important from the practical perspective of sample-efficiency. Second, it can be expected to diminish the algorithm's sensitivity to noise and algorithm hyperparameters. These characteristics will be evaluated in Section V and are due to the feedback nature of the PPC law, as it guides the system towards task satisfaction.

For each so-called *elementary* temporal task ϕ_i in the task specification (5), an elementary controller is defined from (3) as:

$$\boldsymbol{u}_{\phi_i}(\boldsymbol{x}, t) = -\epsilon_i(\boldsymbol{x}, t) g^{\mathrm{T}}(\boldsymbol{x}) \frac{\partial \rho^{\psi_i}(\boldsymbol{x})}{\partial \boldsymbol{x}}. \tag{9}$$

Since the STL task φ is composed of a conjunction of such elementary tasks, the linear combination of these elementary controls can serve well as a base law towards satisfying φ:

$$\hat{\boldsymbol{u}}(\boldsymbol{x}, t) := \sum\nolimits_{i=1}^M \beta_i \boldsymbol{u}_{\phi_i}(\boldsymbol{x}, t). \tag{10}$$

The coefficients $\beta_i \in \mathbb{R}$ are such that $\sum_{i=1}^M \beta_i = 1$. The simulations presented in Section V simply use $\beta_i = 1/M$.

While the elementary control laws $\boldsymbol{u}_{\phi_i}$ would individually guarantee the satisfaction of each corresponding task ϕ_i, this cannot be said about their linear combination $\hat{\boldsymbol{u}}(\boldsymbol{x}, t)$ with respect to the task φ. There exist other controllers that can handle such a broader subset of STL tasks [6]; however, these are much more expensive to evaluate than the simple PPC law employed herein. We argue that since PI^2 relies on a multitude of sampled trajectories and will find a task satisfying policy in either case, this computational efficiency makes the PPC law a more attractive choice as the base controller.

[1]Here, $J(\tau)$ is a general objective that differs from the target cost $C(\tau)$ introduced earlier and will be defined in Section IV-C.

The choice of $S(\xi)$ in (4) greatly impacts the performance of PI^2, because it determines how aggressively the base law steers the system away from the prescribed robustness boundaries. This transformation function should grow unbounded as the edges of the funnel are neared to avoid crossing them due to the noise $\boldsymbol{w}$ or the term $f(\boldsymbol{x})$ in the dynamics, e.g., by mapping $\xi \in (0,1)$ to $(-\infty, \infty)$ as in [7]. Here, we are not interested in the theoretical task satisfaction guarantee offered by such a choice, as this is lost when combining the elementary controls in (10) anyway. Instead, we aim to avoid numerical issues caused by possibly extremely high values of $S(\xi)$ and its derivative around the interval $\xi \in [0,1]$. Exploration in PI^2 becomes difficult if the base control is changing drastically from one time instance to another in such a case. Therefore, we propose to use a joined linear and exponential transformation $S(\xi)$ parameterized by $\alpha, \beta, \kappa > 0$, and $0 < \xi_c < 1$ that maps $\xi \in [0,1]$ to $[0, B]$:

$$S(\xi) := \begin{cases} \max\left(0, \frac{\beta}{\xi_c}\xi\right), & \xi \le \xi_c, \\ m + \alpha(e^{\kappa\xi} - 1), & \xi > \xi_c. \end{cases} \quad (11)$$

This construction automatically satisfies $S(0) = 0$ at the upper robustness boundary $\xi = 0$. The linear part reaches the given value β at ξ_c, whereas α and κ for the exponential part are chosen to achieve a continuous derivative at the transition $\xi = \xi_c$ and have $S(1) = B$ at the lower robustness boundary.

C. Objective function definition

The objective function $J(\tau)$ plays a central role in PI^2 and must be defined such that the control objectives stated in Problem 1 are achieved through its minimization. Namely, we wish to find trajectories minimizing the cost $C(\tau)$ while satisfying the STL formula φ with robustness $\rho_{\text{min}} \ge 0$. To enforce this constraint, $C(\tau)$ is augmented with a penalty term to obtain:

$$J(\tau) := J^{\lambda}(\tau, \rho) = C(\tau) + P^{\lambda}(\rho), \quad (12)$$

where the penalty function $P^{\lambda}(\rho)$ is parameterized by $\lambda \in \mathbb{R}$ and satisfies $P^{\lambda}(\rho \ge \rho_{\text{min}}) \to 0$ and $P^{\lambda}(\rho < \rho_{\text{min}}) \to \infty$ as $\lambda \to \infty$. The function used in this work is $P^{\lambda}(\rho) = \lambda(\rho_{\text{min}} - \rho)^3$ if $\rho < \rho_{\text{min}}$ and 0 otherwise. The constraint $\rho \ge \rho_{\text{min}}$ is enforced by progressively increasing λ throughout PI^2.

To ensure proper discrimination between sampled trajectories, it is important to normalize the sampled $i = 1, \ldots, N$ J_i^{λ} costs [12]. In our case, some costs may have extremely high values due to constraint penalization, yet we still aim to discriminate the rest of the trajectories. Defining the value J_{ϵ}^{λ} for which the ϵ% (e.g., 25%) of all sampled J_i^{λ} costs fall below it, our proposed normalization is thus:

$$\bar{J}_i^{\lambda} = -h\eta \frac{J_i^{\lambda} - \min_j J_j^{\lambda}}{J_{\epsilon}^{\lambda} - \min_j J_j^{\lambda}}, \quad (13)$$

where h controls the range of the normalized values, e.g., $h = 10$ used in the case study simulations herein, and η is the temperature parameter from (7). This elitist strategy tunes the normalization for the top ϵ-th percentile of the sampled trajectories and corresponding costs. The normalized cost values are then used to calculate the weights in (7).

The proposed solution for Problem 1 is fully summarized as Algorithm 1 below. For improved convergence and decreased sensitivity to hyperparameters, a Nesterov-type acceleration scheme [19] is employed, as seen in steps 12-13.

Algorithm 1 PPC guided PI^2 solution to Problem 1

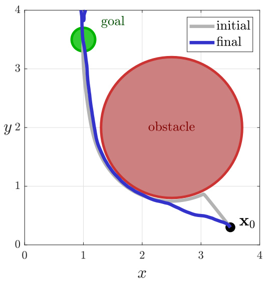
Require: Initial parameter estimates $\theta_t^{(0)}$, covariances $\mathbf{C}_t^{(0)}$, sample batch size N, iteration number K, penalty λ
1: $\alpha^{(0)} := 1$, $\hat{\theta}_t^{(0)} := \theta_t^{(0)}$ for $\forall t = 0, \ldots, T$
2: **for** $k = 1 \ldots K$ **do**
3:     **for** $i = 1 \ldots N$ **do**
4:         Sample policy parameters $\tilde{\theta}_{t,i} = \mathcal{N}(\hat{\theta}_t^{(k-1)}, \mathbf{C}_t^{(0)})$
5:         Obtain τ_i under the PPC guided policy $\pi_{\tilde{\theta}_i}$
6:     **end for**
7:     Compute the normalized costs $\bar{J}_i^{\lambda}$ for each trajectory τ_i using (12) and (13)
8:     Compute weights w_i using (7) with normalized costs
9:     **for** each time step $t = 0, \ldots, T$ **do**
10:         $\theta_t^{(k)} = \sum_{i=1}^{N} w_i \tilde{\theta}_{t,i}$
11:         $\mathbf{C}_t^{(k)} = \mathbf{C}_{t,\min} + \sum_{i=1}^{N} w_i (\tilde{\theta}_{t,i} - \theta_t)(\tilde{\theta}_{t,i} - \theta_t)^{\mathrm{T}}$
12:         $\alpha^{(k)} = (1 + \sqrt{4\left(\alpha^{(k-1)}\right)^2 + 1})/2$
13:         $\hat{\theta}_t^{(k)} = \theta_t^{(k)} + (\alpha^{(k-1)} - 1)(\theta_t^{(k)} - \theta_t^{(k-1)})/\alpha^{(k)}$
14:     **end for**
15:     Increase penalty term λ
16: **end for**
17: **return** $\theta = \theta^{(K)}$

V. RESULTS

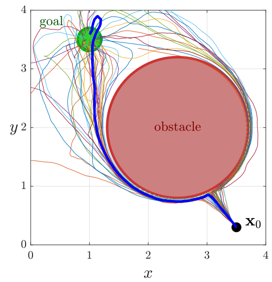
In this section, we present simulation results of the proposed PI^2 algorithm applied to two sample scenarios. The first involves a simple navigational task with the purpose of illustrating the main advantages of using the PPC base law for improved convergence and robustness. The second presents a more complicated scenario to demonstrate the applicability of the technique to a more practical problem. In both scenarios, the feedforward terms of the control policy are parameterized by $\theta = [\theta_0, \ldots, \theta_T]$ with $\boldsymbol{k}_t = \sum_{t'=0}^{t} \theta_{t'}$, the discrete form of the expression described in Section IV-A. For comparison, Algorithm 1 is also implemented using the linear state feedback $\hat{\boldsymbol{u}}(\boldsymbol{x}_t) = -\mathbf{K}_t \boldsymbol{x}_t$ as the base law in step 5; the two variants are referred to as 'LIN' and 'PPC'.

A. Simple navigational task

Consider an omnidirectional robot described by the simplified dynamics $\dot{\boldsymbol{x}} = [\dot{x}\ \dot{y}]^{\mathrm{T}} = [u_x\ u_y]^{\mathrm{T}} = \boldsymbol{u}$, where $\|\boldsymbol{u}\|_2 \le 1$. The robot is initially located at $\boldsymbol{x}_0 = [3.0\ 0.3]^{\mathrm{T}}$ and is tasked with navigating to an $r_g = 0.2$ radius goal region at $\boldsymbol{x}_g = [1.0\ 3.5]^{\mathrm{T}}$ within 10 seconds while avoiding a large circular obstacle of radius $r_o = 1.2$ centered at $\boldsymbol{x}_o = [2.5\ 2.0]^{\mathrm{T}}$. We aim for a robustness of at least $\rho_{\text{min}} = 0.05$, and to minimize the time this is first attained for the subtask ψ_1 of reaching the goal region, i.e., $C(\tau) = \arg\min_t \{t : \rho^{\psi_1}(t) \ge \min\left(\rho_{\text{min}}, \max_t \rho^{\psi_1}(t)\right)\}$. The minimum between ρ_{min} and $\max_t \rho^{\psi_1}(t)$ is taken in order to define an appropriate cost

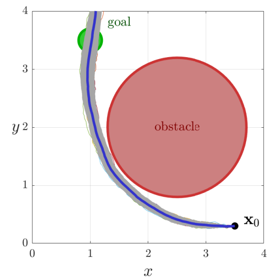


(a)

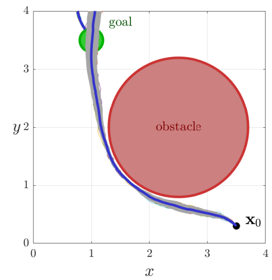


(b)

Fig. 1: (a) Initial and obtained trajectories for the simple navigational task scenario using PPC guided PI^2. The result achieves cost $C(\tau) = 4.50$ with robustness $\rho^{\varphi} = 0.046$ for task satisfaction. (b) Visualization of the first iteration of the underlying PI^2 algorithm. 50 trial trajectories are shown, along with the iteration's obtained result highlighted in blue.

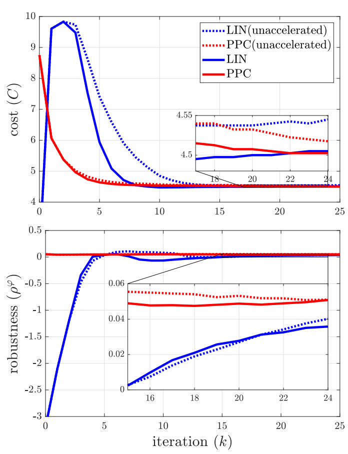


(a) LIN: $C = 4.697 \pm 0.229$, $\rho^{\varphi} = 0.104 \pm 0.070$

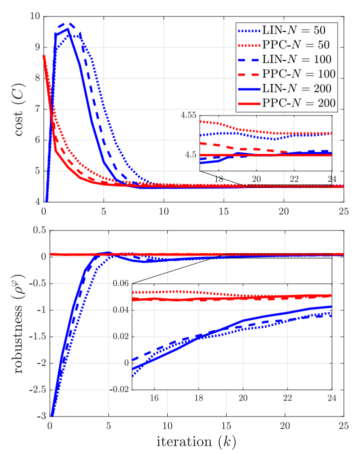


(b) PPC: $C = 4.553 \pm 0.256$, $\rho^{\varphi} = 0.054 \pm 0.018$

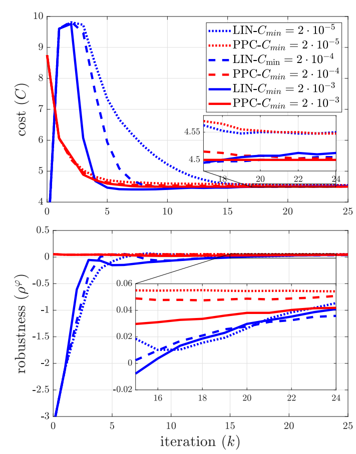
Fig. 2: Robustness of the PI^2 algorithm variants with respect to system noise. The shaded gray area corresponds to 2 standard deviations of a Gaussian distribution fitted to 30 sample runs of the obtained controllers. With the PPC-based law, the optimal robustness is achieved with lower variance.



(a) Employing Nesterov-type acceleration



(b) Sample size N



(c) Minimum covariance $C_{\min}$

Fig. 3: Effect of two chosen PI^2 hyperparameters and Nesterov's acceleration scheme on the algorithm's performance. The results show an average of 20 sample runs in each case. The PPC base law aids task satisfaction and allows for efficient exploration directly towards the cost $C(\tau)$ of interest, achieving faster convergence with less sensitivity to hyperparameters.

for the case when ψ_1 is not yet satisfied with the desired robustness. The scenario is simulated for $T = 10$s with resolution $\Delta t = 0.05$s. A formal STL description of the task, prescribed performance funnels and controllers used as a guiding law in PI^2, and other algorithm hyperparameters are omitted here due to space constraints and we refer the reader to the extended paper [15]. The navigational task is depicted in Fig. 1a, along with results from PPC guided PI^2.

We first examine the scenario without any process noise. Fig. 3 compares the convergence rate of Algorithm 1 between using linear state feedback (with $\mathbf{K}_t = \mathbf{I}_2$) and the described PPC law as a basis control law. The graphs were obtained by varying different hyperparameters of the algorithm and averaging 20 randomized runs for each case. It is clear that the PPC law outperforms its linear feedback-based counterpart both in terms of improved convergence rate and lower sensitivity to the examined parameters. Applying the Nesterov acceleration scheme leads to improvements in both cases, though the difference is less evident for the PPC variant due to the simplicity of the studied navigational task.

Next, we include a disturbance $\boldsymbol{w}$ in the system dynamics with covariance $\mathbf{\Sigma}_w = 0.2\mathbf{I}_2$, 20% of the input bound. The Nesterov acceleration scheme was turned off for a better final result as it is known to amplify the effects of noise and hinder convergence in later iterations [20]. A sample result from both the PPC and linear state feedback-based PI^2 variants is depicted in Fig. 2. The figure shows that the PPC variant provides more robustness against noise due to the feedback nature of the base controller continuously correcting for its influence. The effect is most visible near the obstacle, and leads to a lower variation of the robustness measure, which in turn allows a more optimal solution in terms of the cost $C(\tau)$ to be found while aiming to keep $\rho^{\varphi} \geq \rho_{\min}$.

B. Complex task

Consider two ground vehicles and a drone described by the following (2D) single integrator dynamics, subject to the consensus protocol [21] with additional free inputs, i.e., $\dot{\boldsymbol{x}}(t) = -0.1(\mathbf{L} \otimes \mathbf{I}_2)\boldsymbol{x}(t) + \boldsymbol{u}(t)$. This equation fits the system (1) with known input term $g(\boldsymbol{x}) = \mathbf{I}$ and unknown

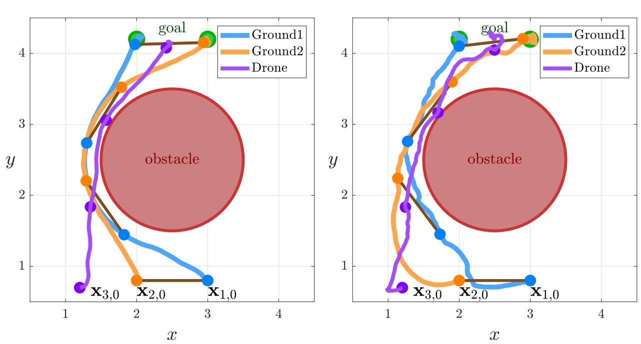
$f(\boldsymbol{x}) = -0.1(\mathbf{L} \otimes \mathbf{I}_2)$. The subscripts $\boldsymbol{x}_i$ and $\boldsymbol{u}_i$, $i = 1, 2, 3$, will refer to the location and inputs of the i-th robot. The input constraint is $\|\boldsymbol{u}_i\|_2 \leq 1$ for each robot. The matrix $\mathbf{L}$ is the so-called Laplacian of the graph describing agent connections within the consensus protocol [21]; assuming a complete graph it becomes $\mathbf{L} = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$. The robots' initial locations are $\boldsymbol{x}_{1,0} = [3.0\ 0.8]^{\mathrm{T}}$, $\boldsymbol{x}_{2,0} = [2.0\ 0.8]^{\mathrm{T}}$, and $\boldsymbol{x}_{3,0} = [1.2\ 0.7]^{\mathrm{T}}$. The ground robots are tasked with reaching and staying within $r_g = 0.1$ meters of $\boldsymbol{x}_{g1} = [2.0\ 4.2]^{\mathrm{T}}$ and $\boldsymbol{x}_{g2} = [3.0\ 4.2]^{\mathrm{T}}$ within $7s$ while maintaining a mutual distance between $d_{12}^{\min} = 1 - \Delta d_{12}$ and $d_{12}^{\max} = 1 + \Delta d_{12}$, $\Delta d_{12} = 0.1$. Furthermore, they must avoid a circular obstacle of radius 1m centered at $\boldsymbol{x}_o = [2.5\ 2.5]^{\mathrm{T}}$ by $r_o = 1.2$ during this maneuver (e.g., to leave space for a carried object). The drone is tasked with reaching and staying within $r_a = 0.1$ meters from the middle of the two ground robots within 3s. The goal is to satisfy this task with minimal robustness $\rho_{\min} = 0.02$ while minimizing the sum of each robot's extended energy, i.e., $C(\tau) = \sum_{i=1}^{3} \int_0^T \boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{u}_i$. The scenario is simulated for $T = 10$s with resolution $\Delta t = 0.01$s. For a formal task description and algorithm parameters, the reader is again referred to the extended version [15] of this paper. The scenario and sample results are shown in Fig. 4.

Examining the resulting trajectories in detail, we can see that the distance traveled by the ground robots is minimized and their speed is such that the goals are reached at the latest possible 7s in order to reduce the input efforts. The drone also maintains a nearly straight path around the middle of the two ground robots while lagging behind to minimize its input effort. Without the PPC base law as a guide, we were unable to tune the parameters for the LIN variant of PI^2 to achieve task satisfaction with a remotely optimal cost.

VI. CONCLUSIONS

In this work, we examined the possibility of using a PPC base law to guide the PI^2 reinforcement learning algorithm in order to solve optimal control problems involving STL task specifications. The method offers multiple benefits, such as increased computational efficiency and robustness to noise and hyperparameters, as well as the ability to cope with more complicated task specifications. These advantages were illustrated in a simulation study of two sample scenarios.

The results give incentive for developing STL base laws that guarantee task satisfaction for a wider range of system dynamics and under increasingly complex task specifications. Further research possibilities include automating hyperparameter choices for the proposed algorithm, as well as extending the method to the multi-agent domain by decentralizing the base control and learning aspects.



(a) PPC: $\rho^{\varphi} = 0.02$, $C = 8.01$ (b) LIN: $\rho^{\varphi} = -0.04$, $C = 11.4$

Fig. 4: Sample solutions to the complex task scenario of Section V-B. The robots are shown at 4 evenly spaced time instances until the goal areas are reached around $t = 7$s.

REFERENCES

[1] A. Pnueli, "The temporal logic of programs," in *IEEE Symposium on Foundations of Computer Science*, 1977, pp. 46–57.

[2] S. Saha and A. A. Julius, "Task and motion planning for manipulator arms with metric temporal logic specifications," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 379–386, 2018.

[3] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 2004, pp. 152–166.

[4] S. Sadraddini and C. Belta, "Robust temporal logic model predictive control," in *IEEE Allerton Conference on Communication, Control, and Computing*, 2015, pp. 772–779.

[5] L. Lindemann and D. V. Dimarogonas, "Robust control for signal temporal logic specifications using discrete average space robustness," *Automatica*, vol. 101, pp. 377–387, 2019.

[6] ——, "Control barrier functions for signal temporal logic tasks," *IEEE Control Systems Letters*, vol. 3, no. 1, pp. 96–101, 2019.

[7] L. Lindemann, C. K. Verginis, and D. V. Dimarogonas, "Prescribed performance control for signal temporal logic specifications," in *IEEE Conference on Decision and Control*, 2017, pp. 2997–3002.

[8] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[9] D. Sadigh, E. S. Kim, S. Coogan, S. S. Sastry, and S. A. Seshia, "A learning based approach to control synthesis of Markov decision processes for linear temporal logic specifications," in *IEEE Conference on Decision and Control*, 2014, pp. 1091–1096.

[10] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2010, pp. 92–106.

[11] X. Li, Y. Ma, and C. Belta, "A policy search method for temporal logic specified reinforcement learning tasks," in *IEEE American Control Conference*, 2018, pp. 240–245.

[12] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *Journal of Machine Learning Research*, vol. 11(Nov), pp. 3137–3181, 2010.

[13] J. Fu, I. Papusha, and U. Topcu, "Sampling-based approximate optimal control under temporal logic constraints," in *International Conference on Hybrid Systems: Computation and Control*. ACM, 2017, pp. 227–235.

[14] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, "Q-learning for robust satisfaction of signal temporal logic specifications," in *IEEE Conference on Decision and Control*, 2016, pp. 6565–6570.

[15] P. Varnai and D. V. Dimarogonas, "Prescribed performance control guided policy improvement for satisfying signal temporal logic tasks," *arXiv preprint arXiv:1903.04340*, 2019, to appear in the 2019 *IEEE American Control Conference*.

[16] C. P. Bechlioulis and G. A. Rovithakis, "Robust adaptive control of feedback linearizable MIMO nonlinear systems with prescribed performance," *IEEE Transactions on Automatic Control*, vol. 53, no. 9.

[17] Y. Chebotar, M. Kalakrishnan, A. Yahya, A. Li, S. Schaal, and S. Levine, "Path integral guided policy search," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 3381–3388.

[18] F. Stulp and O. Sigaud, "Path integral policy improvement with covariance matrix adaptation," in *International Conference on Machine Learning*. Omnipress, 2012, pp. 1547–1554.

[19] Y. E. Nesterov, "A method for solving the convex programming problem with convergence rate $O(1/k^2)$," in *Dokl. Akad. Nauk SSSR*, vol. 269, 1983, pp. 543–547.

[20] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International Conference on Machine Learning*, 2013, pp. 1139–1147.

[21] M. Mesbahi and M. Egerstedt, *Graph theoretic methods in multiagent networks*. Princeton University Press, 2010, vol. 33.