Model Predictive Control for Systems with Partially Unknown Dynamics Under Signal Temporal Logic Specifications

Zhao Feng Dai, Yash Vardhan Pant, *Member, IEEE*, and Stephen L. Smith, *Senior Member, IEEE*

***Abstract*— In this work, we design a model predictive controller (MPC) for systems to satisfy Signal Temporal Logic (STL) specifications when the system dynamics are partially unknown, and only a nominal model and past runtime data are available. Our approach uses Gaussian process regression to learn a stochastic, data-driven model of the unknown dynamics, and manages uncertainty in the STL specification resulting from the stochastic model using Probabilistic Signal Temporal Logic (PrSTL). The learned model and PrSTL specification are then used to formulate a chance-constrained MPC. For systems with high control rates, we discuss a modification for improving the solution speed of the control optimization. In simulation case studies, our controller increases the frequency of satisfying the STL specification compared to controllers that use only the nominal dynamics model.**

***Index Terms*— Autonomous systems, predictive control for nonlinear systems, stochastic optimal control.**

I. Introduction

SIGNAL Temporal Logic (STL) is a specification language for real-valued, continuous-time signals, making it useful for specifying complex control tasks for dynamical systems. Examples include safe acceleration behaviour for autonomous vehicles [1] and efficient temperature control for buildings [2]. As a result, many recent works have studied the Model Predictive Control (MPC) of systems such that the resulting trajectory satisfies requirements specified in STL. These works often focus on systems with perfectly known dynamics [3], [4] or dynamics with simple additive disturbances [5], [6]. Real systems, however, can have more complex, partially unknown dynamics not readily captured by such models. For example, a road vehicle experiences dynamics such as tire forces and varying road friction that are not readily captured by common vehicle control models like the kinematic bicycle or by simple additive disturbances. These dynamics, therefore, remain unknown to the controller, and can result in the closed-loop system violating the given STL specification.

In this paper, we design a model predictive controller for a system for which we first have access only to a nominal dynamics model along with past runtime data. We utilize the data and Gaussian Process (GP) regression to learn an additive stochastic residual model, capturing the mismatch between the nominal model and the true system dynamics. The full resulting model, consisting of the nominal model plus learned residual, is stochastic. Thus, to manage uncertainty in the task specification, we specify the control task using Probabilistic Signal Temporal Logic (PrSTL) [7]. We then formulate an MPC incorporating the learned dynamics and PrSTL specification. To the best of our knowledge, this is the first such work studying MPC for systems with learned partially unknown dynamics to satisfy STL specifications.

Related Work: MPC under STL specifications has been studied for systems with fully known dynamics [3]. Extensions to systems with disturbances include satisfying an STL specification robustly for systems with additive bounded deterministic disturbances [5] and probabilistically for additive stochastic disturbances with bounded support [6]. These classes of disturbance models are less expressive than the unknown dynamics we consider in this paper. An alternative to explicitly modeling disturbances is to use the quantitative (robustness) semantic admitted by STL [8] to formulate a controller that maximizes the robustness with which the specification is satisfied [3], [4]. However, a control sequence determined using the known nominal model may not increase trajectory robustness if the full system includes unknown residual dynamics.

Being defined over deterministic signals, STL cannot directly evaluate a stochastic system trajectory, which limits its use in MPC for systems with uncertainty. To handle uncertainty, Probabilistic Signal Temporal Logic (PrSTL), which follows the grammar of STL but uses chance constraints as predicates, was introduced in [7]. A similar logic that evaluates the risk of the robustness of STL predicates was introduced in [9]. MPC for control tasks specified using these logics have been studied in [9] and [10] for system dynamics with additive stochastic noise. Again, this noise model is less expressive than the unknown dynamics considered in this paper.

Apart from MPC approaches, [11] and [12] use prescribed performance control to find a model-free control law for a system with partially unknown, control affine dynamics to satisfy an STL specification. However, these controllers only admit a fragment of STL syntax that excludes disjunctions.

MPC for systems with unknown dynamics has been studied without STL. This includes MPCs that use data-driven dynamics models learned through neural networks [13], [14] and GP

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

The authors are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: {zf3dai, yash.pant, stephen.smith}@uwaterloo.ca)

regression [15]–[17], which require little prior knowledge of the unknown dynamics. GP dynamics models, in particular, provide a measure of uncertainty for their predictions that can be used in chance constraints. In [15], a GP model of the unknown residual dynamics of a system is used for MPC. That said, these prior works on MPC with unknown dynamics only consider control tasks such as stabilization and reference tracking. In contrast, we investigate MPC with both unknown dynamics and more complex temporal logic control tasks.

Contributions: We provide an MPC design for satisfying STL specifications when the system dynamics are partially unknown, with only a nominal model and past trajectory data available. In contrast to existing MPC designs, which are limited to using the known nominal model or simple additive disturbances in this scenario, our MPC design for satisfying STL specifications can incorporate a learned GP model of the unknown residual dynamics. We provide two case studies showing that our controller can successfully complete a specified control task at higher rates compared to existing controllers that are limited to the nominal model.

II. PROBLEM FORMULATION

Following [15], we consider discrete-time dynamical systems that can be modeled in the following form:

$$x_{k+1} = f(x_k, u_k) + B_d\big(g(x_k, u_k) + w_k\big), \tag{1}$$

where $x_k \in \mathcal{X} \subseteq \mathbb{R}^n$ and $u_k \in \mathcal{U} \subseteq \mathbb{R}^m$ are the system state and control input at time k, respectively, and $w_k \sim \mathcal{N}(0, \Sigma_k^w)$, $\Sigma_k^w = \text{diag}\,[\sigma_{1,k}^2, ..., \sigma_{n_d,k}^2]$ is i.i.d. process noise. Here, $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ represents a known nominal model of the system. The term $B_d\big(g(x_k, u_k)+w_k\big)$ captures the "mismatch" between the nominal model f and the true system dynamics, where we refer to $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{n_d}$ as the *unknown residual dynamics*. We assume both f and g are differentiable (at least once) with respect to the state and input. The matrix $B_d \in \mathbb{R}^{n \times n_d}$ is known.

A. Modeling Residual Dynamics Using GP Regression

Our approach first infers function values for the residual g from past state trajectory data using Gaussian process regression. A Gaussian process is a collection of random variables, any finite number of which have a joint distribution [18]. In GP regression, the random variables are the value of a function at different inputs. The training dataset for the GP is obtained from past state trajectories (via (1)) by calculating the mismatch y_k between the actual next state x_{k+1} for a given starting state x_k and control input u_k and the next state predicted by the nominal dynamics, $f(x_k, u_k)$:

$$y_k = g(x_k, u_k) + w_k = B_d^+\big(x_{k+1} - f(x_k, u_k)\big),$$

where B_d^+ is the Moore-Penrose pseudoinverse of B_d. For simplicity, we introduce shorthand $z_k = \begin{bmatrix} x_k^\top & u_k^\top \end{bmatrix}^\top$. Following [15], [17], we model each output dimension of g using a separate GP $d_i(z_k)$, where $i \in \{1, \dots, n_d\}$. The full GP model is represented by $d(z_k) = \begin{bmatrix} d_1(z_k) & \dots & d_{n_d}(z_k) \end{bmatrix}^\top$. Let y_k^i be one dimension of y_k. Let $\mathbf{z} = [z_0, z_1, \dots]$ and $\mathbf{y}^i = [y_0^i, y_1^i, \dots]$, then $\mathbf{z}, \mathbf{y}^i$ forms the training dataset for the GP d_i. The training data is jointly distributed following $(\mathbf{y}^i) \sim \mathcal{N}\big(\mathbf{m}^i(\mathbf{z}), K_{\mathbf{zz}}^i + I(\sigma^i)^2\big)$, where $\mathbf{m}^i(\mathbf{z}) = \begin{bmatrix} m^i(z_1) & m^i(z_2) & \dots \end{bmatrix}$ is the mean function, and σ^i is measurement noise. The covariance matrix $K_{\mathbf{zz}}^i$ is the Gram matrix of the input data, with a single matrix element $[K_{\mathbf{zz}}^i]_{r,c} = k(z_r, z_c)$ defined using a kernel function $k(z_r, z_c)$ that gives the covariance between two data points. In this paper we consider the Squared-Exponential kernel [15]. The distribution of GP output d_i conditioned on the training data is $(d_i|\mathbf{y}^i) \sim \mathcal{N}\big(\mu^{d_i}(z), \Sigma^{d_i}(z)\big)$, where

$$\mu^{d_i}(z) = K_{z\mathbf{z}}^i (K_{\mathbf{zz}}^i + I(\sigma^i)^2)^{-1} \mathbf{y}^i$$
$$\Sigma^{d_i}(z) = K_{zz}^i - K_{z\mathbf{z}}^i (K_{\mathbf{zz}}^i + I(\sigma^i)^2)^{-1} K_{\mathbf{z}z}^i.$$

The full GP model output $d(x_k, u_k) \sim \mathcal{N}(\mu^d, \Sigma^d)$ is constructed by stacking μ^{d_i}, Σ^{d_i} from each output dimension: $\mu^d = \begin{bmatrix} \mu^{d_1} & \dots & \mu^{d_{n^d}} \end{bmatrix}^\top$, $\Sigma^d = \text{diag}\begin{bmatrix} \Sigma^{d_1} & \dots & \Sigma^{d_{n^d}} \end{bmatrix}$. The GP approximates the residual $g(x_k, u_k) + w_k$, producing the system model $x_{k+1} = f(x_k, u_k) + B_d(d(x_k, u_k))$.

B. Signal Temporal Logic and Stochastic Systems

STL is used to specify control tasks for deterministic systems. STL defines predicates, which we denote using p, that have an associated predicate function $\alpha_p : \mathbb{R}^n \to \mathbb{R}$. A predicate is true if and only if $\alpha_p(x) > 0$, $x \in \mathbb{R}^n$. STL formulas, which we denote using ψ and ϕ, are constructed using predicates p, logical operators negation $\neg$, conjunction $\wedge$ and disjunction $\vee$, and temporal operators until $\mathcal{U}$, eventually $\Diamond$, and always $\Box$. The formal definition of STL can be found in [19]. We consider, without loss of generality, STL formulas written without negation [5].

While STL can evaluate a deterministic state trajectory, the system dynamics (1), as well as its learned representation using the GP residual model, are stochastic, and the notion of controlling a stochastic system to satisfy an STL specification is not well-defined. Let (ξ, t) be a stochastic signal, e.g. the trajectory of a stochastic system, and $\xi(t)$ its value at time t. The STL predicate cannot be evaluated via $\alpha_p(\xi(t)) > 0$ as $\xi(t)$ is a random variable. Instead, we specify a chance constraint over the satisfaction of each predicate, i.e. $\mathbb{P}(\alpha_p(\xi(t)) > 0) > 1-\epsilon$, $\epsilon \in (0, 1)$, which manages prediction uncertainty for the system trajectory resulting from the learned dynamics model. The resulting control task specification is a PrSTL formula. PrSTL follows STL grammar, but defines predicates that are true if and only if a chance constraint is satisfied [7].

C. Problem Statement

Let $\mathbf{x} = (x_0, \dots, x_{N-1}, x_N)$, $\mathbf{u} = (u_0, \dots, u_{N-1})$ be sequences of states and inputs, respectively, and define a sequence of control policies $\Pi = (\pi_0, \dots, \pi_{N-1})$, where $\pi_k : \mathcal{X} \to \mathcal{U}$ for each $k \in \{0, 1, \dots, N-1\}$. Let $J(\mathbf{x}, \mathbf{u})$ denote the cost of the state and control sequence pair $(\mathbf{x}, \mathbf{u})$, and $\mathbf{x} \models \psi_{\text{PrSTL}}$ denote that $\mathbf{x}$ satisfies specification ψ_{PrSTL}.

Problem 1: Given a PrSTL specification ψ_{PrSTL}, a system whose dynamics consist of a nominal model plus learned GP residual, and an initial state x_s provided without uncertainty, determine Π such that the system trajectory $\mathbf{x}$ satisfies ψ_{PrSTL}.

$$\min_{\Pi} \quad \mathbb{E}\big(J(\mathbf{x},\mathbf{u})\big) \tag{2a}$$
$$s.t. \quad x_{k+1} = f(x_k,u_k) + B_d(d(x_k,u_k)) \tag{2b}$$
$$u_k = \pi_k(x_k) \tag{2c}$$
$$\mathbf{x} \models \psi_{\text{PrSTL}} \tag{2d}$$
$$x_0 = x_s.$$

As stated, the optimization posed in Problem 1 cannot readily be solved. Firstly, optimizing over all classes of control policies is an infinite-dimensional problem. Furthermore, predicting $\mathbf{x}$ via (2b) requires a method to evaluate the GP output using stochastic predicted values as input. Finally, (2d) needs to be converted to mathematical optimization constraints. We proceed with addressing these points in the controller design.

III. CONTROLLER DESIGN

As in [15], we consider the following class of affine feedback policies: $u_k = \pi_k(x_k) = \mathcal{K}_k(x_k - \mu_k^x) + \mu_k^u$, where $\mathcal{K}_k$, $k \in \{0,1,\dots,N-1\}$ are pre-selected feedback gains. The solution to Problem 1 is thus a sequence of control inputs $(u_0,\dots,u_{N-1})$. Whereas each input mean μ_k^u is deterministic, each u_k is a random variable due to the dependence on x_k, with the exception of u_0 as x_0 is known without uncertainty. To solve Problem 1 we use shrinking horizon MPC, where the initial control horizon is the horizon length of the PrSTL specification [10] and each subsequent iteration decrements the control horizon by 1 [6]. Only u_0 from the solution of each iteration is applied to the system. The pre-selected feedback gains $\mathcal{K}_k$ can be determined in an MPC setting by, for example, linearizing the dynamics around the solution from the previous iteration and solving an LQR problem that tracks that solution.

A. Multiple-Step State Prediction through GP Dynamics

Solving the optimization in Problem 1 requires generating multiple-step predictions for the system state over the control horizon via (2b), which in turn requires evaluating the GP residual model $d(x_k,u_k)$ using predicted values for x_k,u_k as inputs. However, the predicted inputs themselves are stochastic for $k > 0$, and the posterior of a GP when the input is a stochastic distribution cannot be computed analytically [20]. Thus, we use a computationally tractable approximation commonly used in literature to evaluate (2b) across multiple time-steps by approximating state, control, and disturbance as jointly Gaussian-distributed at each time step [15], [20]:

$$\begin{bmatrix} x_k \\ u_k \\ d_k \end{bmatrix} \sim \mathcal{N}\left(\mu_k = \begin{bmatrix} \mu_k^x \\ \mu_k^u \\ \mu_k^d \end{bmatrix}, \Sigma_k = \begin{bmatrix} \Sigma_k^x & \Sigma_k^{xu} & \Sigma_k^{xd} \\ \star & \Sigma_k^u & \Sigma_k^{ud} \\ \star & \star & \Sigma_k^d \end{bmatrix}\right),$$

where $\Sigma_k^u = \mathcal{K}_k\Sigma_k^x\mathcal{K}_k^\top$ and $\Sigma_k^{xu} = \Sigma_k^x\mathcal{K}_k^\top$. The posterior of the GP, $d_k \sim \mathcal{N}(\mu_k^d,\Sigma_k^d)$, as well as Σ_k^{xd} and Σ_k^{ud}, can be evaluated for a Gaussian-distributed input using known approximation techniques such as the mean equivalence and Taylor approximations [15]. Descriptions of these methods are omitted here for brevity but can be found in [15].

The jointly Gaussian state and input can be propagated through nonlinear nominal dynamics f via its first order Taylor approximation about μ_k^x,μ_k^u. This makes the propagation of x_k,u_k through f an affine transformation of a Gaussian distribution, but may be less accurate for highly nonlinear f.

$$f(x_k,u_k) \approx f(\mu_k^x,\mu_k^u) + \nabla f(\mu_k^x,\mu_k^u)\left(\begin{bmatrix} x_k \\ u_k \end{bmatrix} - \begin{bmatrix} \mu_k^x \\ \mu_k^u \end{bmatrix}\right). \tag{3}$$

Following (3), the dynamics (2b) becomes an affine function of Gaussian random variables x_k, u_k, d_k, and can be written as two constraints updating the state mean and covariance.

$$\mu_{k+1}^x = f(\mu_k^x,\mu_k^u) + B_d\mu_k^d \tag{4a}$$
$$\Sigma_{k+1}^x = [\nabla f(\mu_k^x,\mu_k^u)\ B_d]\Sigma_k[\nabla f(\mu_k^x,\mu_k^u)\ B_d]^\top. \tag{4b}$$

B. Cost Function

We use a quadratic cost as the performance objective of interest while satisfying a PrSTL specification. This takes the form $\mathbb{E}\big(J(\mathbf{x},\mathbf{u})\big) = \mathbb{E}\big(\Sigma_{k=0}^N\|x_k\|_Q^2 + \Sigma_{k=0}^{N-1}\|u_k\|_R^2\big)$, and can be used to minimize the magnitude of certain state variables and the control effort used to complete the task. For example, a road vehicle may minimize the magnitude of the steering angle state variable to reduce sharp turns, and the acceleration control input to conserve fuel. As predicted state and control are Gaussian-distributed random variables following Section III-A, $\mathbb{E}\big(J(\mathbf{x},\mathbf{u})\big)$ evaluates to

$$\sum_{k=0}^{N}\left(\|\mu_k^x\|_Q^2 + \text{tr}(Q\Sigma_k^x)\right) + \sum_{k=0}^{N-1}\left(\|\mu_k^u\|_R^2 + \text{tr}(R\Sigma_k^u)\right). \tag{5}$$

C. Encoding the PrSTL Specification as Constraints

The chance constraint $\mathbb{P}(\alpha_p(x_k) > 0) > 1-\epsilon$ for each PrSTL predicate can be rewritten as deterministic, convex constraints over μ^x that are tightened by Σ^x when the predicate function $\alpha_p(x_k)$ is affine in x_k, and $x_k \sim \mathcal{N}(\mu_k^x,\Sigma_k^x)$ is Gaussian-distributed following Section III-A:

$$\mathbb{P}(a^\top x_k + b > 0) > 1-\epsilon \iff a^\top\mu_k^x + b + \Phi^{-1}(\epsilon)\sqrt{a^\top\Sigma_k^x a} > 0, \tag{6}$$

where $\Phi^{-1}(\epsilon)$ is the quantile function of the standard normal distribution. Mixed-integer constraints encode the logic of the specification. A binary variable $\delta_k^p \in \{0,1\}$ is introduced along with the following constraint for each predicate p in ψ_{PrSTL}, such that p is satisfied at time step k if $\delta_k^p = 1$:

$$M(1-\delta_k^p) + \left(a^\top\mu_k^x + b + \Phi^{-1}(\epsilon)\sqrt{a^\top\Sigma_k^x a}\right) > 0,$$

where M is a sufficiently large positive scalar.

Similarly, a binary variable $\delta_k^\psi \in \{0,1\}$ and mixed-integer constraints are introduced for every PrSTL formula ψ, which consist of an operator and its arguments ϕ, at the necessary time steps k such that ψ holds at k if $\delta_k^\psi = 1$. For logical operators conjunction and disjunction, these constraints are:

Conjunction: $\psi = \wedge_{i=0}^{n_\phi}\phi_i$	$\delta_k^\psi \le \delta^{\phi_i}\ \forall i \in \{0,\dots,n_\phi\}$
Disjunction: $\psi = \vee_{i=0}^{n_\phi}\phi_i$	$\delta_k^\psi \le \sum_{i=0}^{n_\phi}\delta^{\phi_i}$

When evaluating discrete-time signals, the temporal operators can be expressed in terms of conjunctions and disjunctions:

Always: $\psi = \square_{[l,r]}\phi$	$\psi_k = \wedge_{i=k+l}^{k+r}\phi_i$
Eventually: $\psi = \Diamond_{[l,r]}\phi$	$\psi_k = \vee_{i=k+l}^{k+r}\phi_i$
Until: $\psi = \phi_1 \mathcal{U}_{[l,r]}\phi_2$	$\psi_k = \vee_{i=k+l}^{k+r}(\phi_{2,i} \wedge (\wedge_{j=k}^{i}\phi_{1,j}))$

which allows them to be encoded using the same constraints as the logical operators. Given a specification ψ, a final constraint $\delta_0^\psi = 1$ is added to enforce ψ at time step $k = 0$.

D. MINLP Control Optimization Formulation

We now formulate the control optimization, $\mathbf{P}_{\text{MINLP}}$, that is to be solved at every time step in the shrinking horizon MPC. To aid in the presentation, we define the decision variables $\boldsymbol{\mu^u} = (\mu_0^u, \ldots, \mu_{N-1}^u)$ and $\boldsymbol{\delta} \in \{0,1\}^{n_\delta}$, where $\boldsymbol{\delta}$ represents the n_δ binary variables that encode the PrSTL specification ψ. The mixed-integer constraints encoding ψ are represented by $h_\psi(\boldsymbol{\mu^x}, \boldsymbol{\Sigma^x}, \boldsymbol{\delta}) \geq 0$, where $\boldsymbol{\mu^x} = (\mu_0^x, \ldots, \mu_N^x)$ and $\boldsymbol{\Sigma^x} = (\Sigma_0^x, \ldots, \Sigma_N^x)$. The cost function has the form of (5).

$$
\begin{aligned}
&\min_{\boldsymbol{\mu^u}, \boldsymbol{\delta}} \ \mathbb{E}\big(J(\mathbf{x}, \mathbf{u})\big) \\
&s.t.\ \mu_{k+1}^x \text{via (4a)},\ \Sigma_{k+1}^x \text{via (4b)},\ \Sigma_k^u = \mathcal{K}_k \Sigma_k^x \mathcal{K}_k^\top \\
&\quad h_\psi(\boldsymbol{\mu^x}, \boldsymbol{\Sigma^x}, \boldsymbol{\delta}) \geq 0 \\
&\quad \mu_{k+1}^x \in \mathbb{R}^n,\ \Sigma_{k+1}^x \in \mathbb{R}^{n\times n},\ \mu_k^u \in \mathbb{R}^m,\ \Sigma_k^u \in \mathbb{R}^{m\times m} \\
&\quad \boldsymbol{\delta} \in \{0,1\}^{n_\delta} \\
&\quad \delta_0^\psi = 1,\ \mu_0^x = x_s,\ \Sigma_0^x = 0,
\end{aligned}
$$

where N is the control horizon and $k \in \{0, \ldots, N-1\}$. The initial state x_s is given with no uncertainty. The optimization $\mathbf{P}_{\text{MINLP}}$ is a Mixed-Integer Nonlinear Program (MINLP) with a nonconvex relaxation due to the nonlinear equality constraints (4a) and (4b) for the mean and covariance dynamics.

IV. Improving Solution Speed

For systems with fast dynamics that require high controller update rates, we propose a convex local approximation to $\mathbf{P}_{\text{MINLP}}$ that can be faster to solve. The approximation uses the idea that solution changes are small between MPC iterations for high controller update rates. Firstly, we linearize the mean dynamics (4a) around a reference trajectory to produce a linear time-varying (LTV) approximation. The solution from the previous MPC iteration can be used as the reference trajectory for time steps $k > 0$. The initial reference trajectory can be obtained from a controller that does not necessarily consider the specification (see Section V-B). Let $\mathcal{F}(\mu_k^x, \mu_k^u)$ represent the mean dynamics (4a), and let $F_k(\mu_k^x, \mu_k^u)$ be the linearization of $\mathcal{F}$ at a time step k. For a reference state and input (x_k^r, u_k^r), where $x_k^r \sim \mathcal{N}(\mu_k^{x^r}, \Sigma_k^{x^r})$ and $u_k^r \sim \mathcal{N}(\mu_k^{u^r}, \Sigma_k^{u^r})$, F_k is determined as follows:

$$
F_k = \mathcal{F}(\mu_k^{x^r}, \mu_k^{u^r}) + \begin{bmatrix} \frac{\partial \mathcal{F}}{\partial \mu^x} & \frac{\partial \mathcal{F}}{\partial \mu^u} \end{bmatrix} \Bigg|_{\begin{bmatrix} \mu_k^{x^r} \\ \mu_k^{u^r} \end{bmatrix}} \begin{bmatrix} \mu_k^x - \mu_k^{x^r} \\ \mu_k^u - \mu_k^{u^r} \end{bmatrix}. \quad (8)
$$

The covariance dynamics (4b) can be similarly linearized. However, we found that linearizing (4b) can produce predictions for Σ_{k+1}^x that are not positive semi-definite, causing numerical issues in the chance constraint (6). Instead, following [16], we propose approximately pre-computing the covariance matrices, thus removing (4b) from the optimization. This is done by feeding each state and input means from the previous MPC iteration's solution through the covariance dynamics to obtain the 1-step prediction covariance matrices at each time step, which are given to the optimization as fixed parameters.

V. Case Studies

We compare our Learned Residual-informed (LRi) controller against existing MPCs, which are limited to using the nominal model when this model and past trajectory data are the only system information available, and the full dynamics are unknown. The first MPC enforces an STL specification via mixed-integer constraints [3], the second maximizes the STL robustness of the predicted trajectory as its objective, using a smooth approximation of STL robustness that removes the need for integer variables [4]. We refer to these two baseline MPCs as Nominal and Smooth Robust, respectively. We also include a second Smooth Robust MPC that maximizes STL robustness with the full simulation model (without the small additive process noise) as an oracle with which to compare our controller's ability to complete a specified task. For simplicity, all predicates in the PrSTL task specification used by the LRi controllers have the same ϵ value. We include results for $\epsilon = \{0.01, 0.10, 0.20, 0.30\}$. All simulations are performed on a laptop with an Apple M1 Pro CPU and 16GB of RAM[1].

Remark 1: In prior works involving control under PrSTL specifications, the closed-loop system is guaranteed by construction to satisfy the specification as the uncertainty in the chance constraint predicates have known distributions [7], [10]. In our setting, however, these distributions are unknown and may differ from the predictions of the learned model. Instead, as an indication of controller performance, we evaluate the frequency with which the sampled closed-loop trajectories returned by each controller satisfy an STL specification ψ_{STL} that has the same structure as the PrSTL specification ψ_{PrSTL} but replaces each PrSTL predicate defined on a chance constraint $\mathbb{P}(\alpha(x_k) > 0) > 1 - \epsilon$ with a corresponding STL predicate defined on $\alpha(x_k) > 0$ [2].

A. Continuously-Stirred Tank Reactor Control

The first case study considers the temperature control of a Continuously-Stirred Tank Reactor (CSTR) equipped with a cooling jacket, which has the continuous-time dynamics [21]

$$
\begin{aligned}
\dot{x}_1 &= -x_1 + \mathcal{D}a(1 - x_1)e^{\frac{x_2}{1+x_2/\gamma}} \\
\dot{x}_2 &= -x_2 + \mathcal{B}\mathcal{D}a(1 - x_1)e^{\frac{x_2}{1+x_2/\gamma}} + \beta(u - x_2),
\end{aligned}
$$

where the system states x_1, x_2 are reactor composition and temperature, respectively, the control input u is cooling jacket temperature, and $\mathcal{D}a$, $\mathcal{B}$, γ and β are parameters with values from [22]. All variables are dimensionless, and units of time are in minutes. The continuous-time model is discretized using

[1] Simulation code is available at `https://github.com/jackzfdai/GPMPC-for-Temporal-Logic`.

[2] For discrete-time systems, the PrSTL chance constraint predicates can be specified such that $\mathbf{x} \models \psi_{\text{PrSTL}} \implies \mathbb{P}(\mathbf{x} \models \psi_{\text{STL}}) > 1 - \epsilon$ via Boole's inequality [10]. This allows more direct controller evaluation by estimating $\mathbb{P}(\mathbf{x} \models \psi_{\text{STL}})$ from data, but can lead to conservative chance constraints.

TABLE I

CSTR TEMP. CONTROL CONTROLLER PERFORMANCE COMPARISON.

Controller	Success (%)	Control Cost [Avg. (5%, 95%)]	Solver Time (s) [Avg. (5%, 95%)]
Nominal	16	**<0.1 (<0.1, <0.1)**	0.6 (<0.1, 1.3)
Sm. Robust	43	12.0 (12.0, 12.0)	**<0.1 (<0.1, <0.1)**
LRi ($\epsilon_{0.01}$)	63	0.2 (<0.1, 0.6)	11.8 (0.1, 94.1)
LRi ($\epsilon_{0.10}$)	78	0.3 (<0.1, 0.5)	9.5 (<0.1, 39.1)
LRi ($\epsilon_{0.20}$)	**87**	0.4 (<0.1, 2.3)	7.9 (0.1, 36.2)
LRi ($\epsilon_{0.30}$)	77	0.4 (<0.1, 2.0)	7.5 (<0.1, 32.0)
Oracle	87	7.6 (7.6, 7.6)	<0.1 (<0.1, <0.1)

the Runge-Kutta method (RK4) with an interval of 0.5 minutes to obtain the nominal control model $f(x_k, u_k)$.

The simulation model adds a disturbance to the temperature dynamics $\dot{x}_2$ that is an exponential function of u. The disturbance increases as u decreases (i.e. greater cooling is applied). Zero-mean Gaussian noise with covariance $\Sigma_w = \text{diag}([0.001\ 0.001])$ is added to x_1 and x_2 at the end of each simulation step. GP training data is gathered from closed-loop simulations where an MPC using only f controls the full system to a set-point. The dataset records mismatch between the predicted and simulated states and has 18 points, which we found sufficient to improve task completion success rate while maintaining average solver time within the control interval.

The control task is to allow composition below a threshold C_{th} only when temperature is below T_{th}, and is specified as:

$$\psi = \square_{[0,H]}\big((x_1 > C_{th}) \vee (x_2 < T_{th})\big).$$

The specification length H is 6 minutes, giving a discrete-time control horizon of $N = 12$. The Nominal and LRi controllers minimize a control cost $\sum_{k=0}^{N-1} \|u_k\|_R^2$, or its expectation, respectively. The Smooth Robust controller and the oracle maximize the robustness of the trajectory. The MINLP solver Bonmin is used for the Nominal and LRi controllers, and the NLP solver Ipopt is used for the Smooth Robust controller and the oracle. We uniformly sample 30 initial states that do not initially violate the task specification, and sample 30 closed-loop trajectories from each initial state for each controller. Figure 1 shows 30 closed-loop trajectories returned from one of the sampled initial states by each controller. The Nominal controller trajectories in Figure 1 terminate early because the applied inputs are unable to keep the system from reaching a state that violates ψ. Table I compares the success rate of each controller in completing the task, and the average and 5-th, 95-th percentiles of their control cost and solver time. The control cost of a trajectory is evaluated using $\sum_{k=0}^{N-1} \|u_k\|_R^2$ over the closed-loop inputs. The LRi controllers return higher success rates than both baselines, with the $\epsilon_{0.20}$ configuration matching the oracle's performance. Furthermore, the LRi controllers use on average at least 30 times smaller control effort than the Smooth Robust controller, which has the closer success rate of the two baselines. This difference in control effort can be seen in the two center plots for u in Figure 1. Whereas the Smooth Robust controller saturates u at -1, the LRi controller reduces the magnitude of u once x_2 no longer needs to be kept below T_{th}. The LRi controller solver times are on average within the 30s control interval, but some iterations exceed this limit.

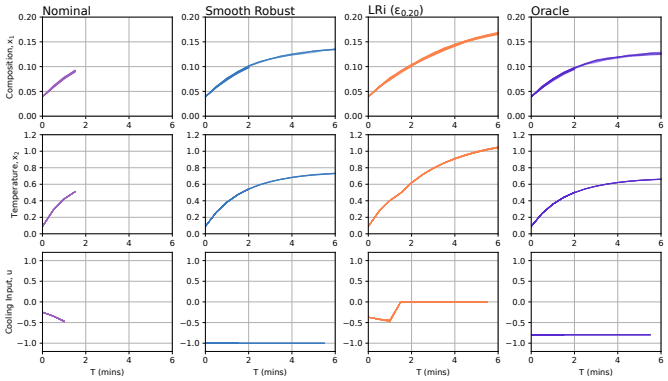

Fig. 1. Comparison of 30 closed-loop CSTR temperature control trajectories from one initial state for the baseline, oracle, and LRi controllers. LRi results are shown for the highest success rate configuration ($\epsilon_{0.20}$).

B. Autonomous Road Vehicle Double Lane Change

The second case study considers a double lane change by an autonomous road vehicle. The nominal vehicle control model is based on the kinematic single track model [23]:

$$\dot{s}_x = v\cos(\theta), \quad \dot{s}_y = v\sin(\theta),$$
$$\dot{\lambda} = v_\lambda, \quad \dot{v} = a_{\text{long}}, \quad \dot{\theta} = (v/l_{wb})\tan(\lambda),$$

where l_{wb} is the vehicle wheelbase. The state variables are position s_x, s_y, steering angle λ, velocity v, and heading θ. The control inputs are steering velocity v_λ and longitudinal acceleration a_{long}. The nominal control model is obtained using RK4 with a discretization interval of 0.125s.

The vehicle simulation model is a higher dimensional single track model with slip and tire forces [23], and is additionally modified to vary friction across the width of the road. Zero-mean Gaussian noise with covariance $\Sigma_w = \text{diag}([0.05\ 0.005\ 0.0001\ 0.001\ 0.0005\ 0.0001\ 0.0001])$ is added to the vehicle states after each simulation step. GP training data is gathered from closed-loop simulation of the Nominal and Smooth Robust controllers performing a similar task on a different map configuration and with different initial conditions. The dataset includes 127 points.

The double lane change task is specified as follows:

$$\psi = \square_{[0,H]}\,(s \in C)$$
$$\wedge\ \Diamond_{[0,H]}\left(\square_{[0,H]}\left((s \in L) \wedge (v > v_{\min}) \wedge (\theta \in \Theta)\right)\right).$$

This specification requires that the vehicle position $s = [s_x\ s_y]^\top$ avoids an obstacle (i.e. stays in clear zone C) at all times, and eventually return to and remain in the lane L, maintaining a minimum speed and driving straight (i.e. heading θ within range Θ). The specification length H is 4s, corresponding to a discrete-time control horizon of $N = 32$.

Given the higher control rates of this system, we use the LRi controller variant from Section IV, which we refer to as LRi-A (Approximate). The Nominal controller also uses an LTV approximation, but of the nonlinear nominal dynamics, to achieve comparable solve times. In both controllers, the initial reference trajectories for state and input (or their means for the LRi-A controller) drive the vehicle across the map in a line without considering a temporal logic task and are obtained from a set-point controller. The LRi-A and Nominal controllers

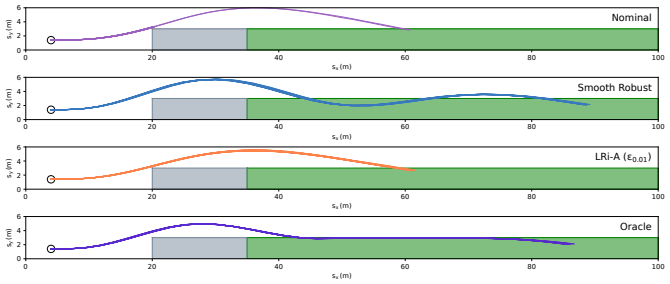

Fig. 2. Comparison of 30 closed-loop double lane change trajectories from one initial state for the baseline, oracle, and LRi-A controllers. LRi-A results shown are for the highest success rate configuration ($\epsilon_{0.01}$).

TABLE II
DOUBLE LANE CHANGE CONTROLLER PERFORMANCE COMPARISON.

Controller	Success (%)	Control Cost [Avg. (5%, 95%)]	Solver Time (s) [Avg. (5%, 95%)]
Nominal	0	2.0 (0.1, 15.9)	**0.07 (<0.01, 0.23)**
Sm. Robust	84	683.6 (533.2, 891.4)	0.17 (<0.01, 0.54)
LRi-A ($\epsilon_{0.01}$)	**90**	**0.3 (<0.1, 0.4)**	**0.06 (<0.01, 0.27)**
LRi-A ($\epsilon_{0.10}$)	81	**0.6 (<0.1, 0.4)**	**0.06 (<0.01, 0.26)**
LRi-A ($\epsilon_{0.20}$)	76	**0.9 (<0.1, 0.4)**	**0.06 (<0.01, 0.27)**
LRi-A ($\epsilon_{0.30}$)	71	**1.0 (<0.1, 0.4)**	**0.07 (<0.01, 0.27)**
Oracle	100	713.0 (620.2, 755.8)	0.24 (0.01, 0.78)

use the solver Gurobi. The Smooth Robust controller and the oracle, which do not require linearization, use Ipopt.

We uniformly sample 30 initial states from a region and 30 closed-loop trajectories from each initial state for each controller. Figure 2 shows 30 closed-loop trajectories returned by each controller from one of the sampled initial states. The Smooth Robust and oracle trajectories are visually longer as they drive the system at higher speeds, but span the same length of time as complete trajectories from other controllers shown in the figure. Table II compares the controllers in success rate, control cost, and solver time. The LRi-A controllers return higher success rates than the Nominal baseline. The $\epsilon_{0.01}$ configuration also improves upon the Smooth Robust baseline's success rate and approaches that of the oracle. As well, when compared to the Smooth Robust controller, which has the closest success rate of the two baselines, the LRi-A controllers use on average 2-3 orders of magnitude smaller control effort. Thus, the LRi-A controllers avoid the oscillatory behaviour seen in the Smooth Robust controller's trajectory (Figure 2, second plot), which is caused by large control inputs. The LRi-A controllers' solver times are within the 0.125s control interval on average, although some iterations at the start of the control horizon exceed this limit.

VI. CONCLUSION

In this paper, we consider the problem of controlling a system with unknown residual dynamics to satisfy an STL specification. We propose an MPC design using a learned GP model for the unknown dynamics and PrSTL to model the STL specification. The resulting control optimization is an MINLP, and a convex local approximation is presented for systems with fast control rates. In case studies, our controller returns higher success rates in completing the specified control task compared to baseline MPCs that are limited to the nominal dynamics model. Potential future extensions include incorporating runtime data to update the GP model online, and implementing the controller on a ground robot.

REFERENCES

[1] N. Arechiga, "Specifying safety of autonomous vehicles in signal temporal logic," in *Proc. Intell. Veh. Symp.*, pp. 58–63, 2019.

[2] T. Wu, M. M. Olama, S. M. Djouadi, J. Dong, Y. Xue, and T. Kuruganti, "Signal temporal logic control for residential HVAC systems to accommodate high solar PV penetration," in *Proc. Innovative Smart Grid Technol. Conf.*, pp. 1–5, 2020.

[3] 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 *Proc. Conf. on Decis. and Contr.*, pp. 81–87, 2014.

[4] Y. V. Pant, H. Abbas, R. A. Quaye, and R. Mangharam, "Fly-by-logic: Control of multi-drone fleets with temporal logic objectives," in *Proc. Int. Conf. Cyber Phys. Syst.*, pp. 186–197, 2018.

[5] S. Sadraddini and C. Belta, "Robust temporal logic model predictive control," in *Allerton Conf. on Comm., Contr. and Comp.*, pp. 772–779, 2015.

[6] S. S. Farahani, R. Majumdar, V. S. Prabhu, and S. Soudjani, "Shrinking horizon model predictive control with signal temporal logic constraints under stochastic disturbances," *IEEE Trans. Automat. Contr.*, vol. 64, no. 8, pp. 3324–3331, 2019.

[7] D. Sadigh and A. Kapoor, "Safe control under uncertainty with probabilistic signal temporal logic," in *Proc. Robot.: Sci. Syst.*, 2016.

[8] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications," in *FATES RV*, pp. 178–192, Springer, 2006.

[9] S. Safaoui, L. Lindemann, D. V. Dimarogonas, I. Shames, and T. H. Summers, "Control design for risk-based signal temporal logic Specifications," *IEEE Contr. Syst. Lett.*, vol. 4, no. 4, pp. 1000–1005, 2020.

[10] T. Yang, Y. Zou, S. Li, and Y. Yang, "Distributed model predictive control for probabilistic signal temporal logic specifications," *IEEE Trans. Automat. Sci. & Eng.*, pp. 1–11, 2024.

[11] P. Varnai and D. V. Dimarogonas, "Prescribed performance control guided policy improvement for satisfying signal temporal logic tasks," in *Proc. American Contr. Conf.*, pp. 286–291, 2019.

[12] L. Lindemann and D. V. Dimarogonas, "Funnel control for fully actuated systems under a fragment of signal temporal logic specifications," *Nonlinear Analysis: Hybrid Systems*, vol. 39, p. 100973, 2021.

[13] Y. Pan and J. Wang, "Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks," *IEEE Trans. Ind. Electron.*, vol. 59, no. 8, pp. 3089–3101, 2012.

[14] E. Terzi, F. Bonassi, M. Farina, and R. Scattolini, "Learning model predictive control with long short-term memory networks," *Int. J. Robust Nonlinear Contr.*, vol. 31, no. 18, pp. 8877–8896, 2021.

[15] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using Gaussian process regression," *IEEE Trans. Contr. Syst. Technol.*, vol. 28, no. 6, pp. 2736–2743, 2020.

[16] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-based model predictive control for autonomous racing," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 3363–3370, 2019.

[17] E. D. Klenske, M. N. Zeilinger, B. Scholkopf, and P. Hennig, "Gaussian process-based predictive control for periodic error correction," *IEEE Trans. Contr. Syst. Technol.*, vol. 24, no. 1, pp. 110–121, 2016.

[18] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. Adaptive computation and machine learning, Cambridge, Mass.: MIT Press, 3. print ed., 2008.

[19] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *FORMATS/FTRTFT*, pp. 152–166, Springer, 2004.

[20] J. Quinonero-Candela, A. Girard, and C. E. Rasmussen, "Prediction at an uncertain input for Gaussian processes and relevance vector machines application to multiple-step ahead time-series forecasting," tech. rep., Technical University of Denmark, DTU, 2003.

[21] A. Uppal, W. Ray, and A. Poore, "On the dynamic behavior of continuous stirred tank reactors," *Chem. Eng. Sci.*, vol. 29, no. 4, pp. 967–985, 1974.

[22] A. Sinha and R. K. Mishra, "Temperature regulation in a continuous stirred tank reactor using event triggered sliding mode control," *IFAC-PapersOnLine*, vol. 51, no. 1, pp. 401–406, 2018.

[23] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. Intell. Veh. Symp.*, pp. 719–726, 2017.