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

by

Zhao Feng Dai

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2025

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Zhao Feng Dai was the sole author of this thesis, which was written under the supervision of Professor Yash Vardhan Pant and Professor Stephen L. Smith. This thesis includes first-authored peer-reviewed material that has appeared in a journal published by the Institute of Electrical and Electronics Engineers (IEEE) [16].

Citation: Z. F. Dai, Y. V. Pant and S. L. Smith, "Model Predictive Control for Systems With Partially Unknown Dynamics Under Signal Temporal Logic Specifications," in IEEE Control Systems Letters, vol. 8, pp. 2931-2936, 2024, doi: 10.1109/LCSYS.2024.3519034. © 2024 IEEE.

Abstract

Autonomous systems are seeing increased deployment in real-world applications such as self-driving vehicles, package delivery drones, and warehouse robots. In these applications, such systems are often required to perform complex tasks that involve multiple, possibly inter-dependent steps that must be completed in a specific order or at specific times. One way of mathematically representing such tasks is using temporal logics. Specifically, Signal Temporal Logic (STL), which evaluates real-valued, continuous-time signals, has been used to formally specify behavioral requirements for autonomous systems.

This thesis proposes a design for a Model Predictive Controller (MPC) for systems to satisfy STL specifications when the system dynamics are partially unknown, and only a nominal model and past runtime data are available. The proposed approach uses Gaussian Process (GP) 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, a modification is discussed for improving the solution speed of the control optimization. In simulation case studies, the proposed controller increases the frequency of satisfying the STL specification compared to controllers that use only the nominal dynamics model. An initial design is also proposed that extends the controller to distributed multi-agent systems, which must make individual decisions to complete a cooperative task.

Acknowledgements

I am deeply grateful to my supervisors, Professor Yash Vardhan Pant and Professor Stephen L. Smith, for the opportunity to join their research groups and for their invaluable guidance, insight, and support throughout my Masters journey. Their feedback and our discussions have taught me so much about research and tackling problems as a researcher, and it has been a privilege to learn from them.

I also want to thank the awesome members of the Autonomous Systems Lab and the Control, Learning and Logic Group, whose work has been an inspiration, and from whom I have learned so much. In particular, I want to acknowledge Shamak Dutta, Megnath Ramesh, Rodrigue de Shaetzen, and Praneeth Kolapalli, with whom I have shared so many wonderful memories.

Dedication

To my amazing family.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Chapter 1

Introduction

Autonomous systems are seeing increased deployment in a variety of applications such as self-driving road vehicles [73], package delivery drones [6], and warehouse robots [15]. In these applications, such systems are often required to perform complex tasks, involving multiple items that must be completed in a specific sequence or at specific times. Successfully executing such tasks is a challenging problem. This motivates, firstly, formally representing these complex tasks in a mathematically sound framework, and secondly, developing control approaches that lead the system behavior to meet the requirements specified in these tasks.

An approach to formally representing complex behavioral requirements for autonomous systems is using temporal logics [7, 72]. This includes logics such as Linear Temporal Logic (LTL), Metric Temporal Logic (MTL), and STL. STL, in particular, is a specification language for real-valued, continuous-time signals, making it useful for specifying complex control tasks for dynamical systems that produce real-valued, continuous-time trajectories. STL has been used to specify safe acceleration behaviour for autonomous vehicles [5, 60], splitting and merging behavior for autonomous vehicle platoons [12], cooperative object manipulation for robotic arms [61], and efficient temperature control for buildings [70]. Formally modeling system requirements using temporal logic enables two uses for autonomous systems. The first is synthesis, where a system controller can be formulated such that the resulting system trajectory satisfies the logic formula that specifies the task. The second is verification, where the success of a controller can be formally evaluated against the task specification using the trajectories of the system. This thesis focuses on the synthesis of controllers for systems to satisfy STL tasks, a topic that has been investigated by many recent works. These works often focus on systems with perfectly known dynamics [53, 50] or dynamics with simple additive disturbances [58, 21]. Real systems, however,

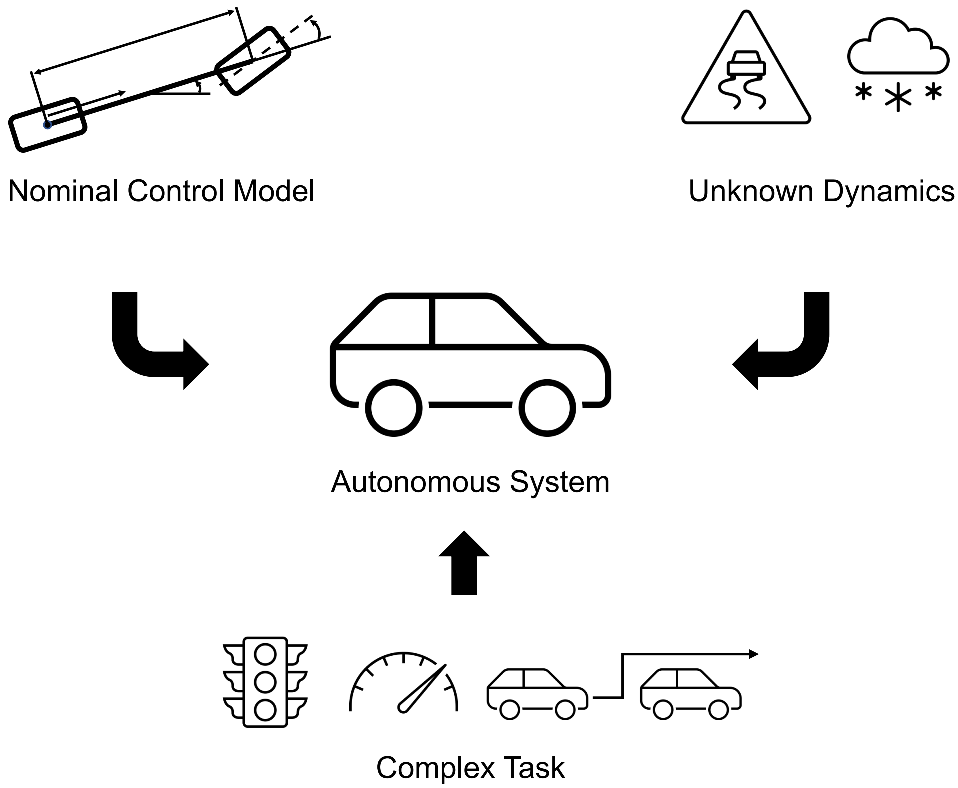

Figure 1.1: Autonomous systems, for example a road vehicle, can experience dynamics not captured in their control model while attempting to complete complex tasks.

have more complex, partially unknown dynamics not readily captured by such models. Not accounting for the effects of these dynamics can result in the closed-loop system violating the given STL specification, failing to complete the control task and potentially operating in an unsafe manner.

Example 1. *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 (Figure 1.1). The controller therefore does not have access to an accurate model of the system, which can result in the closed-loop system failing to complete the control task.*

This thesis investigates the problem of designing a controller for a system for which we first have access only to a nominal dynamics model along with past runtime data, which has to complete a control task specified in STL. The data and GP regression are utilized 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, the control task is specified using PrSTL [57]. An MPC is then formulated incorporating the learned dynamics and PrSTL specification. To the best of my knowledge, this is the first such work studying MPC for systems with learned partially unknown dynamics to satisfy STL specifications.

1.1 Thesis Contributions

The contributions of this thesis are as follows.

- 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, the proposed MPC design for satisfying STL specifications can incorporate a learned GP model of the unknown residual dynamics. The MPC uses a Mixed-Integer Nonlinear Program (MINLP) control optimization. For systems requiring high control rates, a modification to the controller is discussed that improves the solution speed of the underlying control optimization.

- Initial results for an extension of the proposed controller to distributed multi-agent systems, where each agent must make its own decisions in a cooperative manner to accomplish a task that depends on the trajectories of multiple agents.

- Two case studies showing that the proposed controller can successfully complete a specified control task at higher rates compared to existing controllers that are limited to the nominal model. A brief simulation result is also provided comparing the distributed variant of the controller with the centralized variant.

1.2 Thesis Outline

The remainder of this thesis is organized as follows: Chapter 2 reviews prior work on STL, control with tasks specified in STL, and control for systems with unknown or partially unknown dynamics; Chapter 3 details the technical background related to this work and lays out the formal problem investigated in this thesis; Chapter 4 introduces the controller design proposed to solve the problem; Chapter 5 discusses initial results extending the controller design to multi-agent systems; Chapter 6 documents results from simulation case studies testing the performance of the proposed controllers. Finally, Chapter 7 concludes the thesis and discusses limitations and potential future extensions of the work.

Chapter 2

Literature Review

This chapter reviews prior works related to the problem of interest. This includes prior works on control with STL specifications, as well as MPC for systems with unknown or partially unknown dynamics.

2.1 Control of Dynamical Systems with Signal Temporal Logic Specifications

MPC for mixed logical dynamical systems, which are systems described by interdependent physical laws, logical rules, and operating constraints, is studied in [8]. In [8], the logical rules of the system are transformed into mixed-integer inequalities, which are then used as constraints in a mixed-integer program-based MPC. In [53], a similar methodology is used to formulate an MPC for a discrete-time system subject to constraints specified using STL. In both [8] and [53], the systems considered have deterministic and perfectly known dynamics. Extensions to systems with disturbances are studied in [58, 21]. In [58], satisfying an STL specification for Linear Time-Invariant (LTI) systems with additive bounded deterministic disturbances is investigated, and a controller is proposed that leads the system trajectory to satisfy the specification for all possible realizations of the additive noise. In [21], satisfying an STL specification for LTI systems with additive stochastic disturbances with bounded support is investigated. The proposed controller guarantees that the closed-loop system satisfies the specification with the desired probability. However, these classes of disturbance models are less expressive than the unknown dynamics considered in this thesis.

STL admits a quantitative robustness semantic [19]. Informally, the robustness of a signal with respect to an STL formula is the smallest distance that the signal must be perturbed such that it no longer satisfies the formula. A more detailed discussion of STL robustness is provided in Section 3.1.2. One approach to achieving robust trajectories is formulating a controller that maximizes the robustness with which the specification is satisfied [53, 48, 50]. The robustness-maximizing controllers in [53, 48, 50] use the robustness of the system trajectory as the objective function of the control optimization. However, a robustness-maximizing control sequence determined using the known nominal model may not increase the robustness of the actual system trajectory if the full system includes unknown residual dynamics. Furthermore, by only considering trajectory robustness, robustness-maximizing controllers may sacrifice other aspects of control performance such as requiring increased control effort.

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, PrSTL, which follows the grammar of STL but uses chance constraints as predicates, was introduced in [57]. A similar logic that evaluates the risk of the robustness of STL predicates was introduced in [59]. MPC for control tasks specified using these logics have been studied in [27, 59, 71] for system dynamics with additive stochastic noise. The case of the additive stochastic noise having an unknown distribution, supported on a set, is investigated in [32], which proposes a distributionally-robust control approach using Wasserstein distance and a scenario program. Again, the additive noise models considered in these works are less expressive than the unknown dynamics considered in this paper.

Control approaches also exist outside of MPC for satisfying STL specifications. One such approach is to use time-varying Control Barrier Functions (CBFs) [37, 13, 31]. In [37], hand-designed time-varying CBFs are used to enforce the satisfaction of an STL specification by the system trajectory. In [42], a method is proposed that allows the CBFs for satisfying STL specifications to be learned, while guaranteeing that the resulting controller will lead to the satisfaction of the specification. However, the aforementioned controllers require LTI or control affine dynamics, and admit only a fragment of STL syntax.

For systems with partially unknown control affine dynamics, an approach is proposed in [39] using prescribed performance control to satisfy an STL specification. However, the controller proposed in this work also only admits a fragment of STL syntax that excludes disjunctions.

Finally, learning-based controllers for control with STL specifications are investigated in [1, 67, 35, 41]. Q-learning is used to learn a control policy for satisfying an STL specification in [1, 67], and a long short term memory network trajectory feedback controller is proposed for the same in [35]. The work in [41] learns a both a model of unknown dynamics using a feedforward neural network and a controller using a recurrent neural network to satisfy an STL specification.

2.2 Model Predictive Control for Systems with Unknown Dynamics

MPC for systems with unknown dynamics has been studied without STL. This includes MPCs that use data-driven dynamics models learned through methods such as neural networks [47, 64] and GP regression [30, 29, 11, 28, 26, 45, 18], which require little prior knowledge of the unknown dynamics. GP dynamics models, in particular, provide a measure of uncertainty for their state predictions that can be used in chance constraints. GP regression can be used to learn the entire unknown model of a system [30, 11, 45], or the unknown portion of a system with partially unknown dynamics [29, 28, 26, 18]. In [26], a GP model of the unknown residual dynamics of a system is used in the MPC. That said, these prior works on MPC with unknown dynamics only consider control tasks such as stabilization and reference tracking. In contrast, this thesis investigates MPC with both unknown dynamics and more complex temporal logic control tasks.

Chapter 3

Background

This chapter introduces background concepts related to the problem of interest, culminating in the formal problem statement.

3.1 Preliminaries

Following [26], this work considers discrete-time dynamical systems that can be modeled in the following form:

$$x_{k+1} = f(x_k, u_k) + B_d\big(g(x_k, u_k) + w_k\big), \tag{3.1}$$

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 $w_k \sim \mathcal{N}(0, \Sigma_k^w)$, $\Sigma_k^w = \mathrm{diag}\,[\sigma_{1,k}^2, ..., \sigma_{n_d,k}^2]$ 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 dimensions n and n_d for the nominal and residual dynamics are known but possibly mismatched. The matrix $B_d \in \mathbb{R}^{n \times n_d}$ is known, and matches the dimensions of $g(x_k, u_k) + w_k$ to $f(x_k, u_k)$.

3.1.1 Modeling Residual Dynamics Using Gaussian Processes

The proposed approach in this work 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 [54]. In GP regression, the random variables are the value of a function at different inputs. The GP provides a distribution over the predicted function value given a new input, conditioned on the training data.

The use of GP regression to learn the unknown residual dynamics is motivated by two reasons. Firstly, the output distribution of the GP readily allows for the formulation of chance constraints. Secondly, compared to other methods for learning unknown dynamics, such as neural networks, many prior works exist on MPC with GP models (see Section 2.2), and predicting time-series trajectories using GP models [52] that can be built on for the work in this thesis.

The training dataset for the GP is obtained from past state trajectories (via (3.1)) by calculating the mismatch y_k between the actual recorded 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 [26, 29], each output dimension of g is modeled 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. This work considers the Squared Exponential (SE) kernel:

$$K_{z_1,z_2} = \sigma^2 \exp\Big(-\frac{1}{2}(z_1 - z_2)^\top L^{-1}(z_1 - z_2)\Big),$$

where σ^2 is the output variance parameter (not to be confused with measurement noise σ^i from earlier), and L is the lengthscale parameter. 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

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

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: $\mu^d = \begin{bmatrix}\mu^{d_1} & \dots & \mu^{d_{n^d}}\end{bmatrix}^\top$, $\Sigma^d = \operatorname{diag}\begin{bmatrix}\Sigma^{d_1} & \dots & \Sigma^{d_{n^d}}\end{bmatrix}$. The GP approximates the residual $g(x_k, u_k) + w_k$, producing the system model $x_{k+1} = f(x_k, u_k) + B_d(d(x_k, u_k))$.

3.1.2 Signal Temporal Logic

STL is used to specify control tasks for deterministic systems. STL defines predicates, which are denoted in this thesis 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 are denoted in this thesis using ψ, 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, presented in Backus-Naur form, is as follows [46]:

$$\psi \coloneqq \top \,|\, p \,|\, \neg\psi \,|\, \psi_1 \wedge \psi_2 \,|\, \psi_1 \, \mathcal{U}_{[l,r]} \, \psi_2. \tag{3.2}$$

A valid STL formula can be a tautology $\top$, a predicate p, the negation of a formula $\neg\psi$, conjunction of formulas $\psi_1 \wedge \psi_2$, and the temporal operator until with two formulas as arguments $\psi_1 \mathcal{U}_{[l,r]} \psi_2$. The other operators commonly seen in STL formulas, such as disjunction, eventually, and always, can be defined using the valid STL formulas defined in (3.2).

$$\begin{aligned}
\text{Disjunction}: &\; \psi_1 \vee \psi_2 \coloneqq \neg(\neg\psi_1 \wedge \neg\psi_2) \\
\text{Eventually}: &\; \Diamond_{[l,r]}\psi \coloneqq \top \, \mathcal{U}_{[l,r]} \, \psi \\
\text{Always}: &\; \Box_{[l,r]} \, \psi \coloneqq \neg(\top \, \mathcal{U}_{[l,r]} \, \neg\psi).
\end{aligned}$$

Let (ξ, t) be a deterministic, real-valued, continuous-time signal, and $\xi(t)$ be its value at time t. Let $(\xi, t) \models \psi$ denote that the signal (ξ, t) satisfies the formula ψ. Whether (ξ, t) satisfies an STL formula ψ when evaluated at time t is defined as follows:

$$
\begin{aligned}
(\xi, t) \models p &\iff \alpha(\xi(t)) > 0 \\
(\xi, t) \models \neg\psi &\iff \neg((\xi, t) \models \psi) \\
(\xi, t) \models \psi_1 \wedge \psi_2 &\iff (\xi, t) \models \psi_1 \wedge (\xi, t) \models \psi_2 \\
(\xi, t) \models \psi_1 \vee \psi_2 &\iff (\xi, t) \models \psi_1 \vee (\xi, t) \models \psi_2 \\
(\xi, t) \models \Diamond_{[l,r]}\psi &\iff \exists t' \in [t+l, t+r] \, \text{s.t.} \, (\xi, t') \models \psi \\
(\xi, t) \models \Box_{[l,r]}\psi &\iff \forall t' \in [t+l, t+r], \, (\xi, t') \models \psi \\
(\xi, t) \models \psi_1 \mathcal{U}_{[l,r]} \psi_2 &\iff \exists t' \in [t+l, t+r] \, \text{s.t.} \, (\xi, t') \models \psi_2 \\
&\qquad \wedge \forall t'' \in [t, t'], \, (\xi, t'') \models \psi_1.
\end{aligned}
$$

Although the remaining definitions regarding STL in this subsection include negation, the rest of this work considers, without loss of generality, STL formulas written without negation [58].

STL Robustness

As STL defines predicates over real-valued signals, a notion of robustness exists representing the degree to which a signal satisfies or violates a formula. Informally, the robustness of a signal with respect to an STL formula ψ is the bound on the perturbation that a signal can receive without changing the value of its boolean satisfaction of the formula [20]. In practice, the true robustness of a signal can be expensive to compute, and a common robustness semantic which under-approximates the true robustness of a signal is used. Let (ξ, t) be a deterministic, real-valued, continuous-time signal, and let $\xi(t)$ be its value at time t, the under-approximating robustness semantic is defined as follows [17]:

$$
\begin{aligned}
\rho_p(\xi, t) &\coloneqq \alpha_p(\xi(t)) \\
\rho_{\neg\psi}(\xi, t) &\coloneqq -\rho_\psi(\xi, t) \\
\rho_{\psi_1 \wedge \psi_2}(\xi, t) &\coloneqq \min\left(\rho_{\psi_1}(\xi, t), \rho_{\psi_2}(\xi, t)\right) \\
\rho_{\psi_1 \mathcal{U}_{[l,r]} \psi_2}(\xi, t) &\coloneqq \max_{t' \in [t+l, t+r]} \left(\min \left(\rho_{\psi_2}(\xi, t'), \min_{t'' \in [t, t']} \rho_{\psi_1}(\xi, t'') \right) \right).
\end{aligned}
$$

The robustness semantics of disjunction, eventually, and always operators are defined from the above as follows:

$$\begin{aligned}
\rho_{\psi_1 \vee \psi_2}(\xi, t) &\coloneqq \max\left(\rho_{\psi_1}(\xi, t), \rho_{\psi_2}(\xi, t)\right) \\
\rho_{\Diamond_{[l,r]}}(\xi, t) &\coloneqq \max_{t' \in [t+l, t+r]} \rho_{\psi}(\xi, t') \\
\rho_{\Box_{[l,r]}}(\xi, t) &\coloneqq \min_{t' \in [t+l, t+r]} \rho_{\psi}(\xi, t').
\end{aligned}$$

A property of the robustness semantics of STL is that the robustness of a signal with respect to a formula being positive is equivalent to the signal satisfying the specification.

$$\rho_{\psi}(\xi, t) > 0 \iff (\xi, t) \models \psi.$$

Horizon Length of STL Formulas

The horizon length of an STL formula is the minimum length that a signal must have in order to evaluate whether it satisfies the formula.

Definition 1. *Let $\|\psi\|_H$ denote the horizon length of an STL formula ψ, $\|\psi\|_H$ is defined as follows [46]:*

$$\begin{aligned}
\|p\|_H &\coloneqq 0 \\
\|\neg\psi\|_H &\coloneqq \|\psi\|_H \\
\|\psi_1 \wedge \psi_2\|_H &\coloneqq \max\left(\|\psi_1\|_H,\ \|\psi_2\|_H\right) \\
\|\psi_1 \mathcal{U}_{[l,r]} \psi_2\|_H &\coloneqq \max\left(\|\psi_1\|_H,\ \|\psi_2\|_H\right) + r.
\end{aligned}$$

3.1.3 Signal Temporal Logic for Stochastic Systems

While STL can evaluate a deterministic state trajectory, the system dynamics (3.1), 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 $(\bar{\xi}, t)$ be a stochastic signal, e.g. the trajectory of a stochastic system, and $\bar{\xi}(t)$ its value at time t. The STL predicate cannot be evaluated via $\alpha_p(\bar{\xi}(t)) > 0$ as $\bar{\xi}(t)$ is a random variable. Instead, I specify a chance constraint over the satisfaction of each predicate, i.e. $\mathbb{P}(\alpha_p(\bar{\xi}(t)) > 0) > 1 - \epsilon$, $\epsilon \in (0, 1)$, 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 [57]. The formal definition of a valid PrSTL formula, presented in Backus-Naur form, is as follows [57]:

$$\psi \coloneqq p \,|\, \neg p \,|\, \psi_1 \wedge \psi_2 \,|\, \psi_1 \vee \psi_2 \,|\, \Box_{[l,r]}\psi \,|\, \Diamond_{[l,r]}\psi \,|\, \psi_1 \mathcal{U}_{[l,r]} \psi_2.$$

A valid PrSTL formula can be a predicate, a negation of a predicate, the conjunction and disjunction of formulas, the temporal operators always and eventually with a formula as an argument, and until with two formulas as its arguments. The definition of satisfying each of the PrSTL formulas is as follows

$$\begin{aligned}
(\bar{\xi}, t) \models p &\iff \mathbb{P}(\alpha(\bar{\xi}(t)) > 0) > 1 - \epsilon \\
(\bar{\xi}, t) \models \neg p &\iff \mathbb{P}(-\alpha(\bar{\xi}(t)) > 0) > 1 - \epsilon \\
(\bar{\xi}, t) \models \psi_1 \wedge \psi_2 &\iff (\bar{\xi}, t) \models \psi_1 \wedge (\bar{\xi}, t) \models \psi_2 \\
(\bar{\xi}, t) \models \psi_1 \vee \psi_2 &\iff (\bar{\xi}, t) \models \psi_1 \vee (\bar{\xi}, t) \models \psi_2 \\
(\bar{\xi}, t) \models \Box_{[l,r]}\psi &\iff \forall t' \in [t+l, t+r],\, (\bar{\xi}, t') \models \psi \\
(\bar{\xi}, t) \models \Diamond_{[l,r]}\psi &\iff \exists t' \in [t+l, t+r] \,\text{s.t.}\, (\bar{\xi}, t') \models \psi \\
(\bar{\xi}, t) \models \psi_1 \mathcal{U}_{[l,r]} \psi_2 &\iff \exists t' \in [t+l, t+r] \,\text{s.t.}\, (\bar{\xi}, t') \models \psi_2 \\
&\quad\;\; \wedge \forall t'' \in [t, t'],\, (\bar{\xi}, t'') \models \psi_1
\end{aligned}$$

Note that PrSTL only admits the negation of predicates as valid formulas. Thus, unlike STL, where the operators such as disjunction can be defined from the negation of other valid formula, these operators are included directly in the definition of PrSTL. Furthermore, although negations can be applied to PrSTL predicates, the predicates do not follow the law of excluded middle. A signal can fail both a predicate and its negation.

The horizon length of a PrSTL formula is defined as follows.

Definition 2. *Let $\|\psi\|_H$ denote the horizon length of a PrSTL formula ψ, $\|\psi\|_H$ is defined as follows [71].*

$$\begin{aligned}
\|p\|_H &\coloneqq 0 \\
\|\neg p\|_H &\coloneqq 0 \\
\|\psi_1 \wedge \psi_2\|_H &\coloneqq \max\left(\|\psi_1\|_H,\, \|\psi_2\|_H\right) \\
\|\psi_1 \vee \psi_2\|_H &\coloneqq \max\left(\|\psi_1\|_H,\, \|\psi_2\|_H\right) \\
\|\Box_{[l,r]}\psi\|_H &\coloneqq (\|\psi\|_H) + r \\
\|\Diamond_{[l,r]}\psi\|_H &\coloneqq (\|\psi\|_H) + r \\
\|\psi_1 \mathcal{U}_{[l,r]} \psi_2\|_H &\coloneqq \max\left(\|\psi_1\|_H,\, \|\psi_2\|_H\right) + r
\end{aligned}$$

A discrete-time system is considered in this thesis. When evaluating discrete-time signals such as the trajectory of a discrete-time system, the temporal operators always, eventually, and until can be expressed in terms of conjunctions and disjunctions. Let l, r represent whole numbers of time steps, the temporal operators evaluating a discrete-time signal starting from time step k can be represented in terms of conjunctions and disjunctions as follows:

$$\begin{aligned} \Box_{[l,r]}\psi &= \wedge_{i=k+l}^{k+r}\psi_i \\ \Diamond_{[l,r]}\psi &= \vee_{i=k+l}^{k+r}\psi_i \\ \psi_1\mathcal{U}_{[l,r]}\psi_2 &= \vee_{i=k+l}^{k+r}(\psi_{2,i} \wedge (\wedge_{j=k}^{i}\psi_{1,j})), \end{aligned}$$

where ψ_i denotes the value of the formula when evaluated for the signal starting from time step i.

3.2 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 $\mathbf{x} \models \psi_{\mathrm{PrSTL}}$ 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 discrete-time system trajectory $\mathbf{x}$ satisfies the PrSTL specification ψ_{PrSTL}.*

$$\min_{\Pi} \quad \mathbb{E}\big(J(\mathbf{x}, \mathbf{u})\big) \tag{3.3a}$$

$$s.t. \quad x_{k+1} = f(x_k, u_k) + B_d(d(x_k, u_k)) \tag{3.3b}$$

$$u_k = \pi_k(x_k) \tag{3.3c}$$

$$\mathbf{x} \models \psi_{PrSTL} \tag{3.3d}$$

$$x_0 = x_s.$$

As stated, the optimization posed in Problem 1 cannot readily be solved. Firstly, optimizing over all classes of control policies is an infinite-dimensional optimization problem. Furthermore, predicting $\mathbf{x}$ via (3.3b) requires a method to evaluate the GP output using stochastic predicted values as input. Finally, (3.3d) needs to be converted to mathematical constraints for the optimization. The next chapter proceeds with addressing these points in the controller design.

Chapter 4

Controller Design

In this chapter, a model predictive controller design is proposed for solving Problem 1.

4.1 Control Policies

Optimizing over all possible control policies is an infinite dimensional optimization problem. Instead, following [26], 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 shrinking horizon MPC is used, where the initial control horizon is the horizon length of the PrSTL specification and each subsequent iteration decrements the control horizon by 1 [21]. Shrinking horizon MPC allows the consideration of past states when evaluating the satisfaction of the specification. Only the first of the sequence of control inputs in the solution, u_0, is applied to the system after each iteration. 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 a Linear Quadratic Regulator (LQR) problem that tracks that solution. The feedback gains can also be set to a constant value for mild non-linearities [26].

4.2 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 (3.3b), 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 [52]. Thus, I use a computationally tractable approximation commonly used in literature to evaluate (3.3b) across multiple time-steps, which first approximates state, control, and disturbance as jointly Gaussian distributed at each time step [26, 52]:

$$\begin{bmatrix} x_k \\ u_k \\ d_k \end{bmatrix} \sim \mathcal{N}\left(\mu_k = \begin{bmatrix} \mu_k^x \\ \mu_k^u \\ \mu_k^d \end{bmatrix}, \Sigma_k = \begin{bmatrix} \Sigma_k^x & \Sigma_k^{xu} & \Sigma_k^{xd} \\ \star & \Sigma_k^u & \Sigma_k^{ud} \\ \star & \star & \Sigma_k^d \end{bmatrix}\right), \tag{4.1}$$

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 then be evaluated for a Gaussian distributed input using known approximation techniques such as the mean equivalence and Taylor approximations [26]. To aid in the exposition of these approximations, define

$$\mu_k^z = \begin{bmatrix} \mu_k^x \\ \mu_k^u \end{bmatrix}, \Sigma_k^z = \begin{bmatrix} \Sigma_k^x & \Sigma_k^{xu} \\ \star & \Sigma_k^u \end{bmatrix}, \Sigma_k^{zd} \begin{bmatrix} \Sigma_k^{xd} \\ \Sigma_k^{ud} \end{bmatrix}.$$

The mean equivalence approximation approximates the output of a GP for a stochastic input using the output for only the mean of the input distribution. For a Gaussian distributed input, the approximation is as follows:

$$\begin{gathered} \mu_k^d = \mu^d(\mu_k^z) \\ \begin{bmatrix} \Sigma_k^{zd} \\ \Sigma_k^d \end{bmatrix} = \begin{bmatrix} 0 \\ \Sigma^d(\mu_k^z) \end{bmatrix}, \end{gathered}$$

where $\mu^d(\mu_k^z)$ and $\Sigma^d(\mu_k^z)$ refer to the stacked output of the GPs for each output dimension of the residual when given the mean of the state and control as input.

The Taylor approximation of the GP output uses the Taylor approximation of the GP mean and covariance function to determine the output for a stochastic input. Given the jointly Gaussian assumption in (4.1), this results in an affine transformation of a Gaussian distributed random variable:

$$\begin{aligned}\mu_k^d &= \mu^d(\mu_k^z)\\ \begin{bmatrix}\Sigma_k^z\\ \Sigma_k^d\end{bmatrix} &= \begin{bmatrix}\Sigma_k^z(\nabla\mu^d(\mu_k^z))^\top\\ \Sigma^d(\mu_k^z)+(\nabla\mu^d(\mu_k^z))\Sigma_k^z(\nabla\mu^d(\mu_k^z))^\top\end{bmatrix}.\end{aligned}$$

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.

$$f(x_k, u_k) \approx f(\mu_k^x, \mu_k^u) + \nabla f(\mu_k^x, \mu_k^u)\left(\begin{bmatrix}x_k\\ u_k\end{bmatrix} - \begin{bmatrix}\mu_k^x\\ \mu_k^u\end{bmatrix}\right). \tag{4.2}$$

Following (4.2), the dynamics (3.3b) 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.

$$\mu_{k+1}^x = f(\mu_k^x, \mu_k^u) + B_d\mu_k^d \tag{4.3a}$$

$$\Sigma_{k+1}^x = [\nabla f(\mu_k^x, \mu_k^u)\ B_d]\Sigma_k[\nabla f(\mu_k^x, \mu_k^u)\ B_d]^\top. \tag{4.3b}$$

4.3 Cost Function

This work considers a quadratic cost as the performance objective of interest while satisfying a PrSTL specification. This takes the form

$$\mathbb{E}\big(J(\mathbf{x}, \mathbf{u})\big) = \mathbb{E}\big(\Sigma_{k=0}^N \|x_k - c_k\|_Q^2 + \Sigma_{k=0}^{N-1}\|u_k\|_R^2\big), \tag{4.4}$$

where c_k are constants. The term $\Sigma_{k=0}^N\|x_k - c_k\|_Q^2$ can be used to minimize the magnitude of certain state variables when $c_k = 0$, or the deviation of a state variable from a quantity. An example of the former case is a road vehicle that minimizes the magnitude of the steering angle state variable to reduce sharp turns. An example of the latter case is a road vehicle that minimizes the deviation from the center of a lane during a collision avoidance task in order to reduce the likelihood of creating a hazard for the vehicles in adjacent lanes. The term $\Sigma_{k=0}^{N-1}\|u_k\|_R^2$ can be used to minimize the control effort required to complete a task. For example, a road vehicle may minimize the acceleration control input to conserve fuel.

As predicted state and control are Gaussian distributed random variables following Section 4.2, $\mathbb{E}\big(J(\mathbf{x},\mathbf{u})\big)$ evaluates to

$$\sum_{k=0}^{N}\left(\|\mu_k^x - c_k\|_Q^2 + \operatorname{tr}(Q\Sigma_k^x)\right) + \sum_{k=0}^{N-1}\left(\|\mu_k^u\|_R^2 + \operatorname{tr}(R\Sigma_k^u)\right). \tag{4.5}$$

4.4 Encoding the PrSTL Task Specification as Optimization 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, i.e. $\alpha_p(x_k) = a^\top x_k + b$, $a, b \in \mathbb{R}^n$, and $x_k \sim \mathcal{N}(\mu_k^x, \Sigma_k^x)$ is Gaussian distributed following Section 4.2. This is done as follows, where $v \sim \mathcal{N}(0,1)$, and $\Phi^{-1}(\epsilon)$ is the quantile function of the standard normal distribution:

$$\begin{aligned}
\mathbb{P}(a^\top x_k + b > 0) > 1-\epsilon &\iff \mathbb{P}\left(v > \frac{-(a^\top \mu_k^x + b)}{\sqrt{a^\top \Sigma_k^x a}}\right) > 1-\epsilon \\
&\iff \mathbb{P}\left(v \le \frac{-(a^\top \mu_k^x + b)}{\sqrt{a^\top \Sigma_k^x a}}\right) < \epsilon \\
&\iff \Phi\left(\frac{-(a^\top \mu_k^x + b)}{\sqrt{a^\top \Sigma_k^x a}}\right) < \epsilon \\
&\iff a^\top \mu_k^x + b + \Phi^{-1}(\epsilon)\sqrt{a^\top \Sigma_k^x a} > 0, \qquad (4.6)
\end{aligned}$$

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:

$$\begin{array}{ll} \text{Conjunction: } \psi = \wedge_{i=1}^{n_\psi} \psi_i & \delta_k^\psi \leq \delta^{\psi_i} \; \forall i \in \{1, \ldots, n_\psi\} \\ \text{Disjunction: } \psi = \vee_{i=1}^{n_\psi} \psi_i & \delta_k^\psi \leq \sum_{i=1}^{n_\psi} \delta^{\psi_i} \end{array}$$

As discussed in Section 3.1.3, when evaluating discrete-time signals, the temporal operators until, eventually, and always can be expressed in terms of conjunctions and disjunctions, which allows them to be encoded using the same constraints as the logical operators. A final constraint $\delta_0^\psi = 1$ is added for the overall specification ψ to enforce its satisfaction at time step $k = 0$.

Example 2. *Given a discrete-time signal (x, k), where $k \in \{0, 1, 2\}$ and $x_k \sim \mathcal{N}(\mu_k^x, \Sigma_k^x)$, $\mu_k^x, \Sigma_k^x \in \mathbb{R}$, and the following PrSTL specification:*

$$\psi = \Box_{[0,2]}(p_1 \vee p_2),$$

where the predicates p_1 and p_2 are defined as $\mathbb{P}(x-1 > 0) > 1-\epsilon_1$ and $\mathbb{P}(x-2 > 0) > 1-\epsilon_2$, respectively, the following mixed-integer constraints encode the specification:

$$M(1 - \delta_k^{p_1}) + \left(\mu_k^x - 1 + \Phi^{-1}(\epsilon_1)\sqrt{\Sigma_k^x}\right) > 0 \tag{4.7a}$$

$$M(1 - \delta_k^{p_2}) + \left(\mu_k^x - 2 + \Phi^{-1}(\epsilon_2)\sqrt{\Sigma_k^x}\right) > 0 \tag{4.7b}$$

$$\delta_k^{p_1 \vee p_2} \leq \delta_k^{p_1} + \delta_k^{p_2} \tag{4.7c}$$

$$\delta^\psi \leq \delta_k^{p_1 \vee p_2} \tag{4.7d}$$

$$\delta^\psi = 1, \tag{4.7e}$$

for all $k \in \{0, 1, 2\}$, where $\delta_k^{p_1}$, $\delta_k^{p_2}$, $\delta_k^{p_1 \vee p_2} \in \{0, 1\}$. The constraints (4.7a) and (4.7b) encode whether the system state at time step k must satisfy the chance constraint predicates p_1 and p_2, respectively. The constraint (4.7c) encodes whether the disjunction $p_1 \vee p_2$ is satisfied at time step k. The constraint (4.7d) for each time step k, along with the requirement that $\delta^\psi = 1$ (4.7e), encodes that for every time step (i.e., always) from 0 to 2 the disjunction must be satisfied, as required by the specification.

4.4.1 Methods for Reducing the Number of Integer Variables

The worst-case complexity of branch and bound increases exponentially with the number of binary variables, motivating methods to reduce the number of binary variables used to encode the specification. This section discusses three of such methods.

Method 1: The first method for reducing the number of binary variables needed to encode a PrSTL specification is introducing binary variables only for the predicates. The variables δ^{ψ} introduced for formulas that are not predicates can be relaxed to continuous values in the range $[0,1]$. However, even as these variables are continuous variables, they will only take on binary values in the solution of the optimization as a result of the constraints, and the binary variables for the predicates [7]. Applying this method to the specification encoding in Example 2, the variables $\delta_k^{p_1}$ and $\delta_k^{p_2}$ for the predicates would be constrained to take values $\{0,1\}$, while $\delta_k^{p_1 \vee p_2}$ for the disjunction formula would be permitted to take any value within the range $[0,1]$. In the solution to the optimization, however, $\delta_k^{p_1 \vee p_2}$ would take on binary values $\{0,1\}$ due to its dependence on (4.7d) and (4.7e).

To aid in the exposition of two more methods for reducing the number of binary variables, the definition of a PrSTL subformula is first introduced, based on a similar definition for an STL subformula in [34].

Definition 3. *The subformula of a PrSTL formula is the formula to which the outermost (last) operator is applied. A formula that is a conjunction $\wedge$, disjunction $\vee$, or until $\mathcal{U}_{[l,r]}$ has two subformulas, while one that is a always $\Box_{[l,r]}$ or eventually $\Diamond_{[l,r]}$ has one subformula. Predicates p do not have subformulas.*

Method 2: A second method for reducing the number of binary variables is flattening encodings for formulas that have subformulas with the same operators, following a similar approach proposed in [33]. In Section 4.4, this was implicitly done when encoding a formula that consisted of either multiple conjunctions or multiple disjunctions. For example, following the encoding from Section 4.4, $\psi = \psi_1 \wedge \psi_2 \wedge \psi_3$ would be encoded with a variable δ^{ψ}, along with δ^{ψ_1}, δ^{ψ_2}, and δ^{ψ_3}. From the definition of PrSTL syntax, however, each conjunction formula consists of the conjunction operator with two subformulas. Thus, formulating ψ using the PrSTL syntax would require nesting multiple conjunction formulas, for example: $\psi = (\psi_1 \wedge \psi_2) \wedge \psi_3$. Here, the formula $\psi_1 \wedge \psi_2$ is a subformula of ψ along with ψ_3. However, encoding both ψ and its subformula $\psi_1 \wedge \psi_2$ as separate formulas introduces one more variable: $\delta^{\psi_1 \wedge \psi_2}$, in addition to δ^{ψ}, δ^{ψ_1}, δ^{ψ_2}, and δ^{ψ_3}. The extra variable is unnecessary. Flattening the formula encoding by treating ψ as $\psi_1 \wedge \psi_2 \wedge \psi_3$ removes the extra variable. The method can also be applied to a formula consisting of nested disjunctions, for example $\psi = \psi_1 \vee \psi_2 \vee \psi_3$. Since the temporal operators can be expressed in terms of conjunctions and disjunctions when encoding them as constraints for discrete-time signals, their encoding can be flattened with those of their subformulas that use the same operator, and with those of any formula of which they are the subformula and use the same operator. For example, formulas such as eventually of disjunctions and always of conjunctions can be flattened. A more detailed example of a flattened encoding is given below.

Example 3. *Given a discrete-time signal (x, k), where $k \in \{0, 1\}$ and $x_k \sim \mathcal{N}(\mu_k^x, \Sigma_k^x)$, $\mu_k^x, \Sigma_k^x \in \mathbb{R}$, and the following PrSTL specification:*

$$\psi = \Box_{[0,1]}(p_1 \wedge p_2),$$

where the predicates p_1 and p_2 are defined as $\mathbb{P}(x-1 > 0) > 1-\epsilon_1$ and $\mathbb{P}(x-2 > 0) > 1-\epsilon_2$, respectively. When evaluating the discrete-time signal (x, k), this formula is equivalent to:

$$(p_{1,0} \wedge p_{2,0}) \wedge (p_{1,1} \wedge p_{2,1})$$

This specification can be encoded without flattening, as follows:

$$\begin{aligned} M(1-\delta_k^{p_1}) + \left(\mu_k^x - 1 + \Phi^{-1}(\epsilon_1)\sqrt{\Sigma_k^x}\right) &> 0 \\ M(1-\delta_k^{p_2}) + \left(\mu_k^x - 2 + \Phi^{-1}(\epsilon_2)\sqrt{\Sigma_k^x}\right) &> 0 \\ \delta_k^{p_1 \wedge p_2} &\leq \delta_k^{p_1} \\ \delta_k^{p_1 \wedge p_2} &\leq \delta_k^{p_2} \\ \delta^{\psi} &\leq \delta_k^{p_1 \wedge p_2} \\ \delta^{\psi} &= 1. \end{aligned}$$

Flattening the always with the conjunctions produces the following encoding.

$$\begin{aligned} M(1-\delta_k^{p_1}) + \left(\mu_k^x - 1 + \Phi^{-1}(\epsilon_1)\sqrt{\Sigma_k^x}\right) &> 0 \\ M(1-\delta_k^{p_2}) + \left(\mu_k^x - 2 + \Phi^{-1}(\epsilon_2)\sqrt{\Sigma_k^x}\right) &> 0 \\ \delta^{\psi} &\leq \delta_k^{p_1} \\ \delta^{\psi} &\leq \delta_k^{p_2} \\ \delta^{\psi} &= 1. \end{aligned}$$

Formula flattening removes the variables $\delta_k^{p_1 \wedge p_2}$ for every time step k.

A flatter encoding does not always decrease the solution time of the mixed-integer optimization. The results of the case study in Section 6.4 shows one such instance when flattening more parts of the specification encoding does not improve solution speed. In [33], this behavior is attributed to the possibility that certain heuristics in the off-the-shelf mixed-integer program solvers may be more effective for specifications encoded without flattening.

Method 3: A third method for reducing the number of variables used for encoding the PrSTL specification is grouping the variables that would be introduced for a conjunction and all of its subformulas into a single variable. This eliminates both the constraints needed to encode the conjunction, as well as all of the variables needed to encode each subformula. If this method is used together with the first method of relaxing binary variables that are not for predicates, and the subformulas being grouped include predicates, then the grouped variable needs to remain a binary variable. This method is illustrated in Example 4.

Example 4. *Given a discrete-time signal (x, k), where $x_k \sim \mathcal{N}(\mu_k^x, \Sigma_k^x)$, $\mu_k^x, \Sigma_k^x \in \mathbb{R}$, and the following PrSTL specification:*

$$\psi = (p_1 \wedge p_2) \vee (p_3 \wedge p_4).$$

The predicates are p_1, p_2, p_3, p_4 are defined as $\mathbb{P}(x-1>0) > 1-\epsilon_1$, $\mathbb{P}(x-2>0) > 1-\epsilon_2$, $\mathbb{P}(x-3>0) > 1-\epsilon_1$, and $\mathbb{P}(x-4>0) > 1-\epsilon_2$, respectively. Without applying Method 3, the satisfaction of the specification by the signal at time step k is encoded as follows:

$$\begin{aligned}
M(1-\delta^{p_1}) + \left(\mu_k^x - 1 + \Phi^{-1}(\epsilon_1)\sqrt{\Sigma_k^x}\right) &> 0 \\
M(1-\delta^{p_2}) + \left(\mu_k^x - 2 + \Phi^{-1}(\epsilon_2)\sqrt{\Sigma_k^x}\right) &> 0 \\
M(1-\delta^{p_3}) + \left(\mu_k^x - 3 + \Phi^{-1}(\epsilon_3)\sqrt{\Sigma_k^x}\right) &> 0 \\
M(1-\delta^{p_4}) + \left(\mu_k^x - 4 + \Phi^{-1}(\epsilon_4)\sqrt{\Sigma_k^x}\right) &> 0 \\
\delta^{p_1 \wedge p_2} &\leq \delta^{p_1} \\
\delta^{p_1 \wedge p_2} &\leq \delta^{p_2} \\
\delta^{p_3 \wedge p_4} &\leq \delta^{p_3} \\
\delta^{p_3 \wedge p_4} &\leq \delta^{p_4} \\
\delta^{\psi} \leq \delta^{p_1 \wedge p_2} &+ \delta^{p_3 \wedge p_4} \\
\delta^{\psi} &= 1.
\end{aligned}$$

Let $\psi_1 = (p_1 \wedge p_2)$ and $\psi_2 = (p_3 \wedge p_4)$. Applying Method 3, the encoding is as follows:

$$
\begin{aligned}
M(1-\delta^{\psi_1}) + \left(\mu_k^x - 1 + \Phi^{-1}(\epsilon_1)\sqrt{\Sigma_k^x}\right) &> 0 \\
M(1-\delta^{\psi_1}) + \left(\mu_k^x - 2 + \Phi^{-1}(\epsilon_2)\sqrt{\Sigma_k^x}\right) &> 0 \\
M(1-\delta^{\psi_2}) + \left(\mu_k^x - 3 + \Phi^{-1}(\epsilon_3)\sqrt{\Sigma_k^x}\right) &> 0 \\
M(1-\delta^{\psi_2}) + \left(\mu_k^x - 4 + \Phi^{-1}(\epsilon_4)\sqrt{\Sigma_k^x}\right) &> 0 \\
\delta^{\psi} &\leq \delta^{\psi_1} + \delta^{\psi_2} \\
\delta^{\psi} &= 1.
\end{aligned}
$$

The conjunctions $p_1 \wedge p_2$ and $p_3 \wedge p_4$, as well as its subformulas, i.e., the predicates p_1, p_2, p_3, and p_4, are grouped into the same binary variables δ^{ψ_1} and δ^{ψ_2}, respectively. Thus, only two binary variables δ^{ψ_1} and δ^{ψ_2} need to be introduced for this example. Furthermore, the constraints encoding the conjunctions do not need to be introduced.

4.5 MINLP Control Optimization Formulation

This section formulates the control optimization, which is referred to as $\mathbf{P}_{\text{MINLP}}$, that is to be solved at every time step in the shrinking horizon MPC. To aid in the presentation, I define the decision variables $\boldsymbol{\mu}^{\boldsymbol{u}} = (\mu_0^u, \ldots, \mu_{N-1}^u)$ and $\boldsymbol{\delta} \in \{0,1\}^{n_\delta}$, where the latter represents the n_δ binary variables that encode the PrSTL specification. The mixed-integer constraints encoding the specification are represented by $h_\psi(\boldsymbol{\mu}^{\boldsymbol{x}}, \boldsymbol{\Sigma}^{\boldsymbol{x}}, \boldsymbol{\delta}) \geq 0$, where $\boldsymbol{\mu}^{\boldsymbol{x}} =$

$(\mu_0^x, \ldots, \mu_N^x)$ and $\mathbf{\Sigma}^{\boldsymbol{x}} = (\Sigma_0^x, \ldots, \Sigma_N^x)$. The cost function $\mathbb{E}\big(J(\mathbf{x}, \mathbf{u})\big)$ has the form of (4.5).

$$
\begin{aligned}
\min_{\boldsymbol{\mu}^{\boldsymbol{u}}, \boldsymbol{\delta}} \;& \mathbb{E}\big(J(\mathbf{x}, \mathbf{u})\big) \\
\text{s.t. } & \mu_{k+1}^x \text{via (4.3a)} \\
& \Sigma_{k+1}^x \text{via (4.3b)} \\
& \Sigma_k^u = \mathcal{K}_k \Sigma_k^x \mathcal{K}_k^\top \\
& h_\psi(\boldsymbol{\mu}^{\boldsymbol{x}}, \mathbf{\Sigma}^{\boldsymbol{x}}, \boldsymbol{\delta}) \geq 0 \\
& \mu_{k+1}^x \in \mathbb{R}^n,\, \Sigma_{k+1}^x \in \mathbb{R}^{n\times n},\, \mu_k^u \in \mathbb{R}^m,\, \Sigma_k^u \in \mathbb{R}^{m \times m} \\
& \boldsymbol{\delta} \in \{0,1\}^{n_\delta} \\
& \delta_0^\psi = 1 \\
& \mu_0^x = x_s,\, \Sigma_0^x = 0
\end{aligned}
$$

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 an MINLP with a non-convex relaxation due to the nonlinear equality constraints (4.3a) and (4.3b) for the mean and covariance dynamics.

4.6 Improving Solution Speed

For systems with fast dynamics that require high controller update rates, this thesis proposes 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 (4.3a) 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, e.g. a set-point controller (an example is provided in Section 6.4). Let $\mathcal{F}(\mu_k^x, \mu_k^u)$ represent the mean dynamics (4.3a), 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:

$$
F_k = \mathcal{F}(\mu_k^{x^r}, \mu_k^{u^r}) + \begin{bmatrix} \frac{\partial \mathcal{F}}{\partial \mu^x} & \frac{\partial \mathcal{F}}{\partial \mu^u} \end{bmatrix} \Bigg|_{\begin{bmatrix} \mu_k^{x^r} \\ \mu_k^{u^r} \end{bmatrix}} \begin{bmatrix} \mu_k^x - \mu_k^{x^r} \\ \mu_k^u - \mu_k^{u^r} \end{bmatrix}. \tag{4.9}
$$

The covariance dynamics (4.3b) can be similarly linearized. However, we found that linearizing (4.3b) can produce predictions for Σ_{k+1}^x that are not positive semi-definite, which causes numerical issues when used in the chance constraint (4.6). Instead, following [28] I propose removing the covariance dynamics (4.3b) from the optimization and approximately pre-computing the covariance matrices for each time step prior to the optimization. This is done by feeding each state and input means from the solution of the previous MPC iteration through the covariance dynamics to obtain the 1-step prediction covariance matrices at each time step, which are then given to the optimization as fixed parameters.

Define decision variables $\boldsymbol{\mu^u} = (\mu_0^u, \dots, \mu_{N-1}^u)$ and $\boldsymbol{\delta} \in \{0,1\}^{n_\delta}$, where the latter represents the n_δ binary variables that encode the PrSTL specification. Let $h_\psi(\boldsymbol{\mu^x}, \boldsymbol{\Sigma^x}, \boldsymbol{\delta}) \geq 0$, represent the mixed-integer constraints used to encode the PrSTL specification, where $\boldsymbol{\mu^x} = (\mu_0^x, \dots, \mu_N^x)$ and $\boldsymbol{\Sigma^x} = (\Sigma_0^x, \dots, \Sigma_N^x)$. The cost function $\mathbb{E}\big(J(\mathbf{x}, \mathbf{u})\big)$ has the form of (4.5). The approximate control optimization, $\mathbf{P}_{\text{LTV}}$ is as follows:

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

where $\boldsymbol{\Sigma^{x^r}} = (\Sigma_0^{x^r}, \dots, \Sigma_N^{x^r})$ refers to the pre-computed covariance matrices.

Chapter 5

Controller Design for Distributed Multi-Agent Systems

This chapter discusses initial results extending the controller design from Chapter 4 to distributed cooperative multi-agent systems. In this problem setting, the agents must cooperate to complete a task that is dependent on the state of more than one agent, in addition to their own local tasks. While the agents are cooperative, each agent is unaware of the dynamics and local constraints of the other agents. Thus, the agents must solve the problem in a distributed manner. It is assumed that all of the agents are able to communicate instantaneously with each other.

In contrast to related works on distributed control of multi-agent systems, this thesis considers agents that have partially unknown dynamics, with only a nominal model and past runtime data being available initially, as was with the single agent case investigated in the previous chapters. The dynamics of each agent are not coupled with those of any other agent. Each agent may have different dynamics.

5.1 Related Work

In [49], a distributed control scheme is proposed that extends the robustness-maximizing MPC design in [50] to multi-agent systems under STL specifications. The controller leverages a distributed block coordinate descent approach presented in [55], having every agent maximize the robustness of the coupled STL specification with respect to their own state variables. However, although the block coordinate descent approach used accepts non-convex objectives, it only accepts affine constraints, limiting the controller to LTI systems. Other works on control of multi-agent systems with tasks specified using STL, such as [43, 10, 74], also only consider LTI systems with perfectly known dynamics.

Distributed control for multi-agent systems with LTI dynamics plus additive stochastic noise under PrSTL specifications is considered in [71]. However, the distributed control procedure requires an initial centralized solution, with all of the agents' dynamics as constraints.

The works in [36, 38, 14] use CBFs for multi-agent systems to satisfy coupled and local STL specifications. However, similar to CBF-based controllers for single-agent systems under STL specifications, these controllers only admit a fragment of STL formulas, excluding disjunctions.

5.2 Problem Statement

The problem of interest is controlling a multi-agent system consisting of n_A cooperative, heterogeneous agents, where the dynamics of each agent: a) do not depend on any other agent's dynamics, and b) are partially unknown, with only a nominal model and past trajectory data being available initially, such that the trajectories of all agents together complete tasks specified in PrSTL. It is assumed that each agent in the system can maintain lossless, instant communication with every other agent.

Let $\mathcal{A}$ be the set of agents. For the PrSTL task specification, I consider the case where the overall task specification can be represented as follows for a single agent $i \in \mathcal{A}$:

$$\psi_i = \left(\bigwedge_{j \in \mathcal{A}} \psi_{i,j} \right) \wedge \psi_{i,\text{local}},$$

where $\psi_{i,j}$ is the coupled task between agents i and j, whose satisfaction depends on the trajectories of the two agents, and $\psi_{i,\text{local}}$ is the local specification for agent i, whose satisfaction depends only on its own trajectory. The overall specification ψ_{global} for the entire multi-agent system is the conjunction of each pairwise coupled task between the agents, as well as all of the agents' local tasks:

$$\psi_{\text{global}} = \left(\bigwedge_{i,j \in \mathcal{A}} \psi_{i,j} \right) \wedge \left(\bigwedge_{i \in \mathcal{A}} \psi_{i,\text{local}} \right). \tag{5.1}$$

Let $\mathbf{x}^i = (x_0^i, \ldots, x_{N-1}^i, x_N^i)$, $\mathbf{u}^i = (u_0^i, \ldots, u_{N-1}^i)$ be sequences of states and inputs, respectively, for agent i, and define a sequence of control policies $\Pi^i = (\pi_0^i, \ldots, \pi_{N-1}^i)$ for agent i, where $\pi_k^i : \mathcal{X} \to \mathcal{U}$ for each $k \in \{0, 1, \ldots, N-1\}$. Let $J(\mathbf{x}^1, \ldots, \mathbf{x}^{n_A}, \mathbf{u}^1, \ldots, \mathbf{u}^{n_A})$ denote the cost of the state and control sequence pair $(\mathbf{x}^i, \mathbf{u}^i)$ for every agent.

Problem 2. *Given n_A agents, where each agent is a system whose dynamics consist of a known nominal model plus a learned GP residual, a PrSTL specification ψ_{PrSTL} of the form in (5.1), whose satisfaction depends on the trajectories of every agent, and initial states x_s^i for each agent, provided without uncertainty. Determine Π^i for each agent such that all agent trajectories $\mathbf{x}^i$ together satisfy ψ_{PrSTL}.*

$$
\begin{aligned}
\min_{\Pi^1, \ldots, \Pi^{n_A}} \quad & \mathbb{E}\big(J(\mathbf{x}^1, \ldots, \mathbf{x}^{n_A}, \mathbf{u}^1, \ldots, \mathbf{u}^{n_A})\big) && (5.2a)\\
s.t. \quad & x_{k+1}^i = f^i(x_k, u_k) + B_d^i(d^i(x_k, u_k)) && (5.2b)\\
& u_k^i = \pi_k^i(x_k^i) && (5.2c)\\
& \mathbf{x}^1, \ldots, \mathbf{x}^{n_A} \models \psi_{PrSTL} && (5.2d)\\
& x_0^i = x_s^i,
\end{aligned}
$$

5.3 Distributed Control Optimization

As with the single-agent case in Problem 1, Problem 2 is solved with shrinking horizon MPC. Unlike Problem 1, however, the control optimization of each iteration of the MPC must be solved in a distributed manner.

One potential approach is to formulate the control optimization for each iteration of the MPC as a mixed-integer program, similar to $\mathbf{P}_{\text{MINLP}}$, and solving the resulting mixed-integer program in a distributed manner. However, although parallel distributed methods have been proposed in [43] and [65] to solve Mixed-Integer Linear Programs (MILPs), this problem class cannot handle $\mathbf{P}_{\text{MINLP}}$.

A second potential approach is to use existing techniques for solving non-convex, non-linear, continuous optimization problems. These have been used to develop distributed MPC and trajectory planners for nonlinear systems with coupled constraints [22, 56, 62]. In particular, the distributed MPC design proposed in [62] and [22] allow for both nonlinear system dynamics and non-convex coupled constraints between the agents. The proposed controller in this chapter leverages a similar control optimization design as [62]. Formulating a continuous optimization for the MPC has the additional potential benefit of improving controller scalability when compared to using mixed-integer programming. The distributed optimization method proposed in [62] is proved to converge to the optimal solution for convex problems, and to local optima for nonconvex problems based on a result in [69]. The initial design of the control optimization proposed in this thesis however, does not always converge to a solution within a set maximum number of iterations. The cause of this non-convergence, for example, whether the proposed control optimization conforms to the form for which the convergence result is shown in [69], has not yet been investigated.

For the set of agents $\mathcal{A} = \{1, \ldots, n_{\mathcal{A}}\}$, the method from [62] accepts MPC problems that have the following form when written as a centralized problem:

$$\begin{aligned}
\min_{\mathbf{x}^i, \mathbf{u}^i} \quad & \sum_{i=1}^{n_{\mathcal{A}}} J(\mathbf{x}^i, \mathbf{u}^i) && \text{(5.3a)} \\
\text{s.t.} \quad & x_{k+1}^i = f(x_k^i, u_k^i) && \text{(5.3b)} \\
& g(\mathbf{x}^i, \mathbf{u}^i) = 0 && \text{(5.3c)} \\
& h(\mathbf{x}^i, \mathbf{u}^i) \leq 0 && \text{(5.3d)} \\
& \phi(\mathbf{x}^i, \mathbf{u}^i, \mathbf{x}^j, \mathbf{u}^j) = 0, && \text{(5.3e)}
\end{aligned}$$

where the constraints (5.3b) to (5.3e) are applied for all i, $j \in \mathcal{A}$, and $k \in \{0, \ldots, N-1\}$. In the control optimization, (5.3b) are the dynamics constraints for each agent, (5.3c) are the local equality constraints for each agent, (5.3d) are the local inequality constraints for each agent, and (5.3e) are the coupled equality constraints between the agents. Only coupled equality constraints are admitted in this control optimization formulation. Inequality constraints must be converted to equality constraints via slack variables. Furthermore, the coupled equality constraints must have the following form:

$$\phi_{i,j}(\mathbf{x}^i, \mathbf{u}^i, \mathbf{x}^j, \mathbf{u}^j) = \phi_{i,j}^i(\mathbf{x}^i, \mathbf{u}^i) - \phi_{i,j}^j(\mathbf{x}^j, \mathbf{u}^j) = 0, \tag{5.4}$$

If the coupled equality constraint is not in this form, it must be decomposed into this form. For example, a bilinear constraint $x_i x_j = 0$ can be decomposed into the following coupled constraints:

$$\begin{aligned}
x_i - \check{x}_i &= 0 \\
\check{x}_j - x_j &= 0,
\end{aligned}$$

where $\check{x}_i$ is agent j's local copy of the value of x_i, and $\check{x}_j$ is agent i's local copy of x_j, and the following local constraints:

$$\begin{aligned}
x_i \check{x}_j &= 0 \\
\check{x}_i x_j &= 0,
\end{aligned}$$

where the first constraint is enforced by agent i, and the second by agent j, in their distributed subproblems, which are discussed further below.

The optimization is solved iteratively and in parallel by the agents, where each agent $i \in \mathcal{A}$ solves the following subproblem at each iteration:

$$\min_{\mathbf{x}^i,\mathbf{u}^i} \quad J(\mathbf{x}^i,\mathbf{u}^i) + q^{i\top}\left[\phi^i_{i,1}(\mathbf{x}^i,\mathbf{u}^i)^\top \quad \dots \quad \phi^i_{i,n_{\mathcal{A}}}(\mathbf{x}^i,\mathbf{u}^i)^\top\right]^\top + \rho\sum_{j=0}^{n_{\mathcal{A}}}||\phi^i_{i,j}(\mathbf{x}^i,\mathbf{u}^i) - \varphi_{i,j}||_2^2 \tag{5.5a}$$

$$\text{s.t.} \quad x^i_{k+1} = f^i(x^i_k, u^i_k) \tag{5.5b}$$

$$g^i(\mathbf{x}^i,\mathbf{u}^i) = 0 \tag{5.5c}$$

$$h^i(\mathbf{x}^i,\mathbf{u}^i) \leq 0, \tag{5.5d}$$

where $\rho > 0$, and with some abuse of notation, $\phi^i_{i,j}(\mathbf{x}^i,\mathbf{u}^i)$ is a vector containing agent i's portion all of its coupled constraints with agent j. The vector q^i is set to 0 for the first solution iteration, and updated for the next iteration based on the value of the coupled constraints determined during the current iteration,

$$q^i_{\text{next}} = q^i_{\text{curr}} + \rho\left[(\phi^i_{i,1,\text{curr}} - \phi^1_{i,1,\text{curr}})^\top \quad \dots \quad (\phi^i_{i,n_{\mathcal{A}},\text{curr}} - \phi^{n_{\mathcal{A}}}_{i,n_{\mathcal{A}},\text{curr}})^\top\right]^\top,$$

where, for example, $(\phi^i_{i,1,\text{curr}} - \phi^1_{i,1,\text{curr}})^\top$ are the stacked values of the coupled constraints between agents i and 1, written in the form of (5.4). The vector $\varphi_{i,j}$ is the stacked average values of agent i and agent j's part of their coupled constraints, and is updated as follows:

$$\varphi_{i,j,\text{next}} = \frac{\phi^i_{i,j,\text{curr}} + \phi^j_{i,j,\text{curr}}}{2}$$

The subproblems are repeatedly solved by the agents, updating the values for q^i and $\varphi_{i,j}$ at each iteration, until a convergence criteria is met. For example, when the value of the coupled constraints between the two agents are near 0.

5.4 Cost Function

Following the problem format of (5.3), cost functions of the following form are considered:

$$\mathbb{E}(J(\mathbf{x}^1,\mathbf{u}^1\dots,\mathbf{x}^{n_{\mathcal{A}}},\mathbf{u}^{n_{\mathcal{A}}})) = \mathbb{E}\left(\sum_{i=1}^{n_{\mathcal{A}}} J(\mathbf{x}^i,\mathbf{u}^i)\right) = \sum_{i=1}^{n_{\mathcal{A}}}\mathbb{E}\left(J(\mathbf{x}^i,\mathbf{u}^i)\right), \tag{5.6}$$

where the separated cost function for each agent $J(\mathbf{x}^i,\mathbf{u}^i)$ in turn is a quadratic cost with the form of (4.4).

5.5 Encoding PrSTL Task Specification as Continuous Optimization Constraints

Predicates involving only the local state variables of a single agent can be converted to constraints of the same format as (4.6). Coupled predicates $\mathbb{P}(\alpha_p(x_k^i, x_k^j)) > 1 - \epsilon$, where x_k^i, x_k^j refers to the state variables of agents i and j, which are Gaussian distributed, and $\alpha_p(x_k^i, x_k^j) = a_i x_k^i + a_j x_k^j + b$ is an affine function of x_k^i, x_k^j, can be encoded as follows:

$$a_i^\top \mu_k^{x^i} + a_j^\top \mu_k^{x^j} + b + \Phi^{-1}(\epsilon)\sqrt{a_i^\top \Sigma_k^{x^i} a_i + a_j^\top \Sigma_k^{x^j} a_j} > 0. \tag{5.7}$$

The prior encoding of the other syntax of the PrSTL specification relied on mixed-integer constraints. However, this is not possible when using the distributed optimization from Section 5.3. Instead, the specification is encoded using continuous optimization constraints.

To do so, note firstly that the local and coupled predicates are satisfied when the value of (4.6) and (5.7) are greater than 0, respectively, and define these equations as the *characteristic equations* of the predicate when the predicate p is defined on an affine predicate function of signal(s) with Gaussian distributed signal values. To simplify notation, both single-agent and dual-agent trajectories are denoted $(\bar{\xi}, t)$. The characteristic equation of $(\bar{\xi}, t)$ with respect to predicate p is denoted as $\mathcal{X}_p(\bar{\xi}, t)$. Using this definition, the characteristic equations of other PrSTL formulas are defined as follows:

$$\begin{aligned}
\mathcal{X}_{\psi_1 \wedge \psi_2}(\bar{\xi}, t) &= \min\left(\mathcal{X}_{\psi_1}(\bar{\xi}, t), \mathcal{X}_{\psi_2}(\bar{\xi}, t)\right) \\
\mathcal{X}_{\psi_1 \vee \psi_2}(\bar{\xi}, t) &= \max\left(\mathcal{X}_{\psi_1}(\bar{\xi}, t), \mathcal{X}_{\psi_2}(\bar{\xi}, t)\right) \\
\mathcal{X}_{\Box_{[l,r]}\psi}(\bar{\xi}, t) &= \min_{t' \in [t+l, t+r]} \mathcal{X}_{\psi}(\bar{\xi}, t') \\
\mathcal{X}_{\Diamond_{[l,r]}\psi}(\bar{\xi}, t) &= \max_{t' \in [t+l, t+r]} \mathcal{X}_{\psi}(\bar{\xi}, t') \\
\mathcal{X}_{\psi_1 \mathcal{U}_{[l,r]} \psi_2}(\bar{\xi}, t) &= \max_{t' \in [t+l, t+r]} \left(\min\left(\mathcal{X}_{\psi_2}(\bar{\xi}, t'), \min_{t'' \in [t, t']} \mathcal{X}_{\psi_1}(\bar{\xi}, t'')\right)\right).
\end{aligned}$$

A signal satisfies a PrSTL specification if its characteristic equation with respect to the specification is positive. Thus, given a state sequence $\mathbf{x} = (x_0, \ldots, x_N)$, the constraint $\mathcal{X}_\psi(\mathbf{x}) > 0$ would enforce the satisfaction of ψ by $\mathbf{x}$. Even though this constraint is not smooth due to the presence of the min and max operators, smooth approximations exist for these operators that in turn allow the encoding of the PrSTL specification as smooth constraints.

For the min operator, the Log-Sum-Exponential (LSE) approximation can be used [48, 23]:

$$\widetilde{\min}(a_1, \ldots, a_m) = \frac{-1}{k_{\min}} \log \big(\sum_{i=1}^{m} e^{-k_{\min} a_i}\big).$$

The parameter $k_{\min}$ can be adjusted to trade off smoothness of the approximation with accuracy. The LSE approximation under-approximates the true minimum of its arguments, with the approximation error being bounded as follows [48, 23]:

$$\min\,(a_1, \ldots, a_m) - \widetilde{\min}(a_1, \ldots, a_m) \leq \frac{\log(m)}{k_{\min}}.$$

The under-approximation of the true minimum is a desirable quality when applying this approximation to enforce the satisfaction of a PrSTL specification, enforcing that the value of the characteristic equation is greater than 0 with extra margin.

Although the LSE approximation can also be used for the max operator, this leads to an over-approximation, which is not desirable. Instead, following [23], I use the following approximation for the max operator, which under-approximates the true max:

$$\widetilde{\max}(a_1, \ldots, a_m) = \frac{\sum_{i=1}^{m} a_i e^{k_{\max} a_i}}{\sum_{i=1}^{m} e^{k_{\max} a_i}}.$$

Similarly to $k_{\min}$, $k_{\max}$ is an adjustable parameter. Assuming, without loss of generality, that $m \geq 2$ and $\{a_1, \ldots, a_m\}$ is sorted from largest to smallest, the approximation error is bounded as follows [23]:

$$\max\,(a_1, \ldots, a_m) - \widetilde{\max}(a_1, \ldots, a_m) \leq \frac{a_1 - a_m}{\frac{e^{k_{\max}(a_1 - a_2)}}{m-1} + 1}.$$

An example of the smooth encoding of a PrSTL specification is provided below.

Example 5. *Given a discrete-time signal (x, k), where $k \in \{0, 1, 2\}$ and $x_k \sim \mathcal{N}(\mu_k^x, \Sigma_k^x)$, $\mu_k^x, \Sigma_k^x \in \mathbb{R}$, and the following PrSTL specification:*

$$\psi = \Box_{[0,2]}(p_1 \vee p_2),$$

where the predicates p_1 and p_2 are defined as $\mathbb{P}(x-1 > 0) > 1-\epsilon_1$ and $\mathbb{P}(x-2 > 0) > 1-\epsilon_2$, respectively, the following constraint encodes the specification:

$$\frac{-1}{k_{min}} \log \big(e^{-k_{min}\mathcal{X}_0^{p_1 \vee p_2}} + e^{-k_{min}\mathcal{X}_1^{p_1 \vee p_2}} + e^{-k_{min}\mathcal{X}_2^{p_1 \vee p_2}}\big) > 0,$$

where

$$\mathcal{X}_k^{p_1 \vee p_2} = \frac{\mathcal{X}_k^{p_1} \, e^{k_{max} \mathcal{X}_k^{p_1}} + \mathcal{X}_k^{p_2} \, e^{k_{max} \mathcal{X}_k^{p_2}}}{e^{k_{max} \mathcal{X}_k^{p_1}} + e^{k_{max} \mathcal{X}_k^{p_2}}},$$

and $\mathcal{X}_k^{p_1}$ *and* $\mathcal{X}_k^{p_2}$ *are as follows:*

$$\mathcal{X}_k^{p_1} = \mu_k^x - 1 + \Phi^{-1}(\epsilon)\sqrt{\Sigma_k^x}$$
$$\mathcal{X}_k^{p_2} = \mu_k^x - 2 + \Phi^{-1}(\epsilon)\sqrt{\Sigma_k^x}$$

Remark 1. *The under-approximation of the characteristic equation of the specification introduces extra conservativeness in addition to the chance constrain predicates. In practice, the constraint on the value of the characteristic equation being positive may need to be relaxed to maintain the feasibility of the problem.*

5.6 Distributed Control Optimization Formulation

The value of state variables and the characteristic equation value of coupled predicates are treated as coupled constraints.

I first present the centralized form of the control optimization, $\mathbf{P}_{\text{cNLP}}$.

$$\min_{\boldsymbol{\mu}^{\boldsymbol{u}^{\boldsymbol{1}}}, \ldots, \boldsymbol{\mu}^{\boldsymbol{u}^{\boldsymbol{n}_{\mathcal{A}}}}} \quad \sum_{i=1}^{n_{\mathcal{A}}} \mathbb{E}(J(\mathbf{x}^i, \mathbf{u}^i)) \tag{5.8a}$$

$$\text{s.t.} \quad \mu_{k+1}^{x^i} \text{via}(4.3a) \tag{5.8b}$$

$$\Sigma_{k+1}^{x^i} \text{via}(4.3b) \tag{5.8c}$$

$$\Sigma_k^{u^i} = \mathcal{K}_k^i \Sigma_k^{x^i} \mathcal{K}_k^{i\top} \tag{5.8d}$$

$$\tilde{\mathcal{X}}_{\psi_i}(\boldsymbol{\mu}^{\boldsymbol{x}^{\boldsymbol{i}}}, \boldsymbol{\Sigma}^{\boldsymbol{x}^{\boldsymbol{i}}}) > 0 \tag{5.8e}$$

$$\tilde{\mathcal{X}}_{\psi_{i,j}}(\boldsymbol{\mu}^{\boldsymbol{x}^{\boldsymbol{i}}}, \boldsymbol{\Sigma}^{\boldsymbol{x}^{\boldsymbol{i}}}, \boldsymbol{\mu}^{\boldsymbol{x}^{\boldsymbol{j}}}, \boldsymbol{\Sigma}^{\boldsymbol{x}^{\boldsymbol{j}}}) > 0 \tag{5.8f}$$

$$\mu_{k+1}^{x^i} \in \mathbb{R}^{n^i}, \, \Sigma_{k+1}^{x^i} \in \mathbb{R}^{n^i \times n^i}, \mu_k^{u^i} \in \mathbb{R}^{m^i}, \Sigma_k^{u^i} \in \mathbb{R}^{m^i \times m^i} \tag{5.8g}$$

$$\mu_0^{x^i} = x_s^i, \, \Sigma_0^{x^i} = 0, \tag{5.8h}$$

where $i, j \in \mathcal{A}$, and $k \in \{0, \ldots, N-1\}$. The constraints (5.8e) and (5.8f) require that the (approximate) characteristic equations evaluate to above 0. The centralized control optimization is solved iteratively in a distributed manner, where the distributed subproblem solved by an agent at each iteration is denoted $\mathbf{P}_{\text{dNLP}}$.

The constraint for each coupled PrSTL specification (5.8f) is decomposed for the distributed subproblems using the following constraints:

$$\begin{aligned}
\mu^{x^i}_{i,j} - \check{\mu}^{x^i}_{i,1} &= 0 \\
\check{\mu}^{x^j}_{i,j} - \mu^{x^j}_{i,1} &= 0 \\
\Sigma^{x^i}_{i,j} - \check{\Sigma}^{x^i}_{i,1} &= 0 \\
\check{\Sigma}^{x^j}_{i,j} - \Sigma^{x^j}_{i,1} &= 0 \\
\mathcal{X}^{p,i}_{i,j} - \mathcal{X}^{p,j}_{i,j} &= 0,
\end{aligned}$$

where $\mu^{x^i}_{i,j}$, $\Sigma^{x^i}_{i,j}$ are the state means and covariances of agent i that are needed to evaluate the predicates in specification $\psi_{i,j}$. The variables $\check{\mu}^{x^j}_{i,j}$ and $\check{\Sigma}^{x^j}_{i,j}$ are agent i's local copy of agent j's state means and covariances needed to evaluate the predicates in $\psi_{i,j}$. Similarly, $\mu^{x^j}_{i,j}$, $\Sigma^{x^j}_{i,j}$ are the state means and covariances of agent j needed to evaluate the predicates in $\psi_{i,j}$, and $\check{\mu}^{x^i}_{i,1}$ and $\check{\Sigma}^{x^i}_{i,1}$ are agent j's local copy of agent i's variables needed to evaluate predicates in $\psi_{i,j}$. Finally, $\mathcal{X}^{p,i}_{i,j}$ is the characteristic equation for a predicate evaluated by agent i using $\mu^{x^i}_{i,j}$, $\Sigma^{x^i}_{i,j}$, $\check{\mu}^{x^j}_{i,j}$, and $\check{\Sigma}^{x^j}_{i,j}$. The same predicate is also evaluated by agent j using its variables to determine $\mathcal{X}^{p,j}_{i,j}$. The coupled constraints enforce that each agent's local copy of another agent's state must be equal to the other agent's actual state, and the characteristic equation value of the same predicate evaluated by both agents must be equal. These constraints apply to all of the predicates in the coupled specification between the two agents, including those whose evaluation only depends on the state variables of one agent. The actual characteristic equation constraint for the coupled specification (5.8f) is treated as a local constraint by each agent, and evaluated using their own state variables, the local copies of the other agent's state variables, and the characteristic equation values of the predicates.

To aid in the exposition of the distributed subproblem solved by each agent in each iteration of the optimization, $\mathbf{P}_{\text{dNLP}}$, let $\text{vec}(\Sigma^x_k)$ be the vectorization of the covariance matrix Σ_k^x. The subproblem $\mathbf{P}_{\text{dNLP}}$ is as follows:

$$\min_{\boldsymbol{\mu}^{\boldsymbol{u},\boldsymbol{i}}} \quad \mathbb{E}(J(\mathbf{x}^i, \mathbf{u}^i)) + q^{i\top}\boldsymbol{\phi} + \rho \sum_{j=0}^{n_{\mathcal{A}}} \left\| \begin{bmatrix} \boldsymbol{\mu}_{\boldsymbol{i,j}}^{\boldsymbol{x^i}\top} & \text{vec}(\boldsymbol{\Sigma}_{\boldsymbol{i,j}}^{\boldsymbol{x^i}})^\top & \mathcal{X}_{i,j}^{p\top} \end{bmatrix}^\top - \varphi_{i,j} \right\|_2^2 \tag{5.9}$$

$$\text{s.t.} \quad \mu_{k+1}^{x,i} \text{via}(4.3a) \tag{5.10}$$

$$\Sigma_{k+1}^{x,i} \text{via}(4.3b) \tag{5.11}$$

$$\tilde{\mathcal{X}}_{\psi_i}(\boldsymbol{\mu}^{\boldsymbol{x^i}}, \boldsymbol{\Sigma}^{\boldsymbol{x^i}}) > 0 \tag{5.12}$$

$$\tilde{\mathcal{X}}_{\psi_{i,j}}(\boldsymbol{\mu}^{\boldsymbol{x^i}}, \boldsymbol{\Sigma}^{\boldsymbol{x^i}}, \check{\boldsymbol{\mu}}^{\boldsymbol{x^j}}, \check{\boldsymbol{\Sigma}}^{\boldsymbol{x^j}}) > 0, \tag{5.13}$$

where $\boldsymbol{\phi}$ is as follows:

$$\boldsymbol{\phi} = \begin{bmatrix} \begin{bmatrix} \boldsymbol{\mu}_{\boldsymbol{i,1}}^{\boldsymbol{x^i}\top} & \dots & \boldsymbol{\mu}_{\boldsymbol{i,n_{\mathcal{A}}}}^{\boldsymbol{x^i}\top} \end{bmatrix}^\top \\ \begin{bmatrix} \text{vec}(\boldsymbol{\Sigma}_{\boldsymbol{i,1}}^{\boldsymbol{x^i}})^\top & \dots & \text{vec}(\boldsymbol{\Sigma}_{\boldsymbol{i,n_{\mathcal{A}}}}^{\boldsymbol{x^i}})^\top \end{bmatrix}^\top \\ \begin{bmatrix} \check{\boldsymbol{\mu}}_{\boldsymbol{i,1}}^{\boldsymbol{x^1}\top} & \dots & \check{\boldsymbol{\mu}}_{\boldsymbol{i,n_{\mathcal{A}}}}^{\boldsymbol{x^{n_{\mathcal{A}}}}\top} \end{bmatrix}^\top \\ \begin{bmatrix} \text{vec}(\check{\boldsymbol{\Sigma}}_{\boldsymbol{i,1}}^{\boldsymbol{x^1}\top}) & \dots & \text{vec}(\check{\boldsymbol{\Sigma}}_{\boldsymbol{i,n_{\mathcal{A}}}}^{\boldsymbol{x^{n_{\mathcal{A}}}}\top})^\top \end{bmatrix}^\top \\ \begin{bmatrix} \mathcal{X}_{i,1}^{p\top} & \dots & \mathcal{X}_{i,n_{\mathcal{A}}}^{p\top} \end{bmatrix}^\top \end{bmatrix}^\top$$

where $\boldsymbol{\mu}_{\boldsymbol{i,j}}^{\boldsymbol{x^i}}$, $\text{vec}(\boldsymbol{\Sigma}_{\boldsymbol{i,j}}^{\boldsymbol{x^i}})$ are the stacked vectors of all of the state means and vectorized covariance matrices of agent i that are required to evaluate the coupled specification $\psi_{i,j}$, $\check{\boldsymbol{\mu}}_{\boldsymbol{i,j}}^{\boldsymbol{x^j}}$, $\text{vec}(\check{\boldsymbol{\Sigma}}_{\boldsymbol{i,j}}^{\boldsymbol{x^j}})$ are the stacked vectors of agent i's local copy of the state means and vectorized covariance matrices of agent j that are required to evaluate the coupled specification $\psi_{i,j}$, and $\mathcal{X}_{i,j}^{p\top}$ contains all of the characteristic equation values of the predicates in the coupled specification $\psi_{i,j}$. The remaining logic of the coupled PrSTL specification $\psi_{i,j}$ are encoded as local inequality constraints (5.13). The local specification ψ_i is encoded in the local inequality constraints (5.12). The distributed subproblem is a Nonlinear Program (NLP), for which off-the-shelf solvers are available.

Chapter 6

Experiments

This chapter discusses simulation results evaluating the controllers presented in this thesis. The chapter primarily focuses on the Learned Residual-informed (LRi) controller from Chapter 4. However, initial results are briefly discussed in Section 6.5 for the Multi-Agent Learned Residual-informed (MA-LRi) controller from Chapter 5 as well.

The LRi controller is evaluated through two simulation case studies. The first case study is on the temperature control of a CSTR, which is a chemical reactor tank equipped with a cooling jacket to manage heat released from the reaction. The second case study is on a double lane change maneuver for an autonomous road vehicle.

All simulations are performed on a laptop with an Apple M1 Pro CPU and 16GB of RAM[1].

6.1 Baselines

The LRi controller is evaluated firstly against existing MPC baselines, 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 baseline enforces an STL specification via mixed-integer constraints while using only the nominal model [53]. The second baseline maximizes the STL robustness of the predicted trajectory in its objective, using a smooth approximation of STL

[1]Simulation code for the LRi controller case studies can be found at `https://github.com/jackzfdai/GPMPC-for-Temporal-Logic`.

robustness that removes the need for integer variables in the control optimization [50]. These two baseline controllers are referred to as Nominal and Smooth Robust, respectively.

Additionally, the LRi controller is compared against a controller with an existing design, but having access to the full dynamics model of the system, which acts as an oracle. The systems in the case studies have nonlinear dynamics with additive stochastic noise. To the best of my knowledge, no existing work is available that provides an MPC design considering STL specifications for such systems. The closest works on MPC under STL or PrSTL specifications for stochastic systems are [71] and [21], which are limited to LTI models with additive stochastic noise. Instead, the robustness-maximizing Smooth Robust design from [50] is used with the full nonlinear simulation dynamics but without the small additive stochastic noise term.

6.2 Evaluating Controller Performance

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 [57, 71]. In the problem setting considered in this thesis, however, these distributions are unknown and may differ from the predictions of the learned model. Instead, as an indication of controller performance, this work evaluates the frequency with which the sampled closed-loop trajectories returned by each controller satisfy the corresponding STL specification ψ_{STL}, which 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].

[2] For discrete-time systems, the PrSTL chance constraint predicates can be specified such that $\mathbf{x} \models \psi_{\text{PrSTL}} \implies \mathbb{P}(\mathbf{x} \models \psi_{\text{STL}}) > 1 - \lambda$ via Boole's inequality [71]. 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.

6.3 Continuously-Stirred Tank Reactor Temperature Control

The first case study considers the temperature control of a CSTR equipped with a cooling jacket, which has the continuous-time dynamics [66]

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

where the system states x_1, x_2 are reactor composition and reactor temperature, respectively, the control input u is cooling jacket temperature, and $\mathcal{D}a$, $\mathcal{B}$, γ and β are parameters with values from [63]. All variables and parameters are dimensionless, and units of time are in minutes. The continuous-time model is discretized using 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 $\Sigma_w = \operatorname{diag}\left(\begin{bmatrix}0.001 & 0.001\end{bmatrix}\right)$ 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 the nominal model controls the full system to a set-point. The dataset records the mismatch between the predicted and simulated states and has 18 points, which I found sufficient to improve task completion success rate while maintaining average solver time within the control interval.

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

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

The specification length is $H = 6$ minutes, corresponding to a discrete-time control horizon of $N = 12$. For simplicity, all predicates in the PrSTL control task specification used by the LRi controllers have the same ϵ value. We include results for $\epsilon = \{0.01, 0.10, 0.20, 0.30\}$. The PrSTL specification is encoded as mixed-integer constraints for the control optimization. The variable used to encode the always operator, which is represented as a conjunction across time steps for the discretized system, as well as its subformulas, are grouped. Thus, the variable δ^{ψ} for the overall specification is also the variable encoding the disjunction $(x_1 > C_{th}) \vee (x_2 < T_{th})$ for each time step. As δ^{ψ} is constrained by $\delta^{\psi} = 1$, the variable is replaced with 1 in all of the constraints that it appears in. This leaves $1 \leq \delta_k^{x_1 > C_{th}} + \delta_k^{x_2 < T_{th}}$ for each time step, as well as the predicate constraints for each time step, as the only constraints, and $\delta_k^{x_1>C_{th}}$, $\delta_k^{x_2<T_{th}}$ of the predicates for each time step as the only binary variables. The STL specification used by the Nominal controller is encoded in a similar way.

The Nominal and LRi controllers minimize a control cost $\sum_{k=0}^{N-1} \|u_k\|_R^2$, or its expectation, respectively, as their objective. The Smooth Robust controller and the oracle maximize the robustness of the trajectory. The MINLP solver Bonmin [51] is used for the Nominal and LRi controllers, and the NLP solver Ipopt [4] is used for the Smooth Robust controller and the oracle. I 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. Figures 6.1 and 6.2 show 30 closed-loop trajectories returned from one of the sampled initial states by the baseline controllers and the oracle, and by the different ϵ configurations of the LRi controllers, respectively. The Nominal controller trajectories in Figure 6.1 terminate early because the applied inputs are unable to keep the system from reaching a state that violates ψ. Table 6.1 compares the rate of each controller successfully completing the task, and the average and 5-th and 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, but only for completed closed-loop trajectories, i.e. control cost for trajectories that terminated early are not included in the results. The LRi controllers return higher success rates than both baseline controllers, with the LRi ($\epsilon_{0.20}$) configuration matching the oracle's performance. Furthermore, the LRi controllers use at least 30 times smaller control effort on average than the Smooth Robust controller, which has the closer success rate to the LRi controllers out of the two baselines. This difference in control effort can be seen between the plot for u of the Smooth Robust controller in the middle row of Figure 6.1, compared to the plots for u of the LRi controllers in Figure 6.2. 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 solver times of the LRi controllers are on average within the 30s control interval, but some iterations exceed this limit.

Decreasing the value of the parameter ϵ in the PrSTL specification given to the LRi controller leads to an increase in success rate to a point, after which the increasingly conservative chance constraints lead the the control optimization to become infeasible more often.

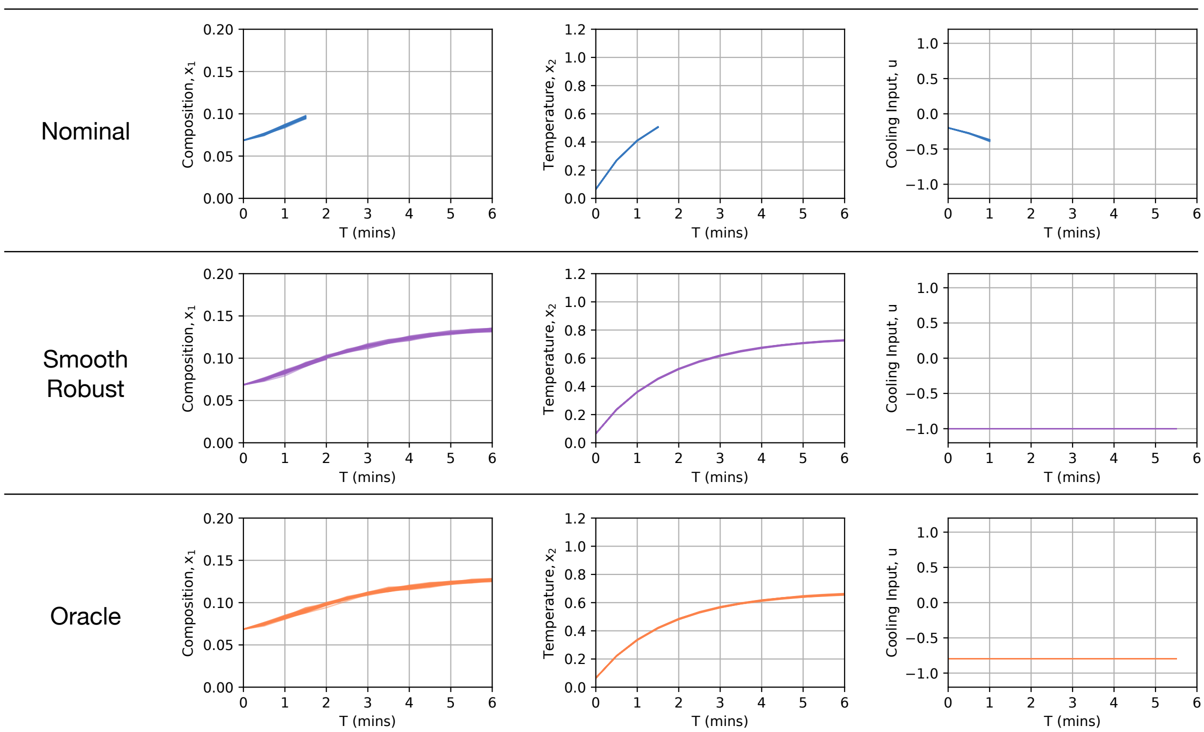

Figure 6.1: Comparison of 30 closed-loop CSTR temperature control trajectories from one initial state between the Nominal and Smooth Robust baselines, and the oracle.

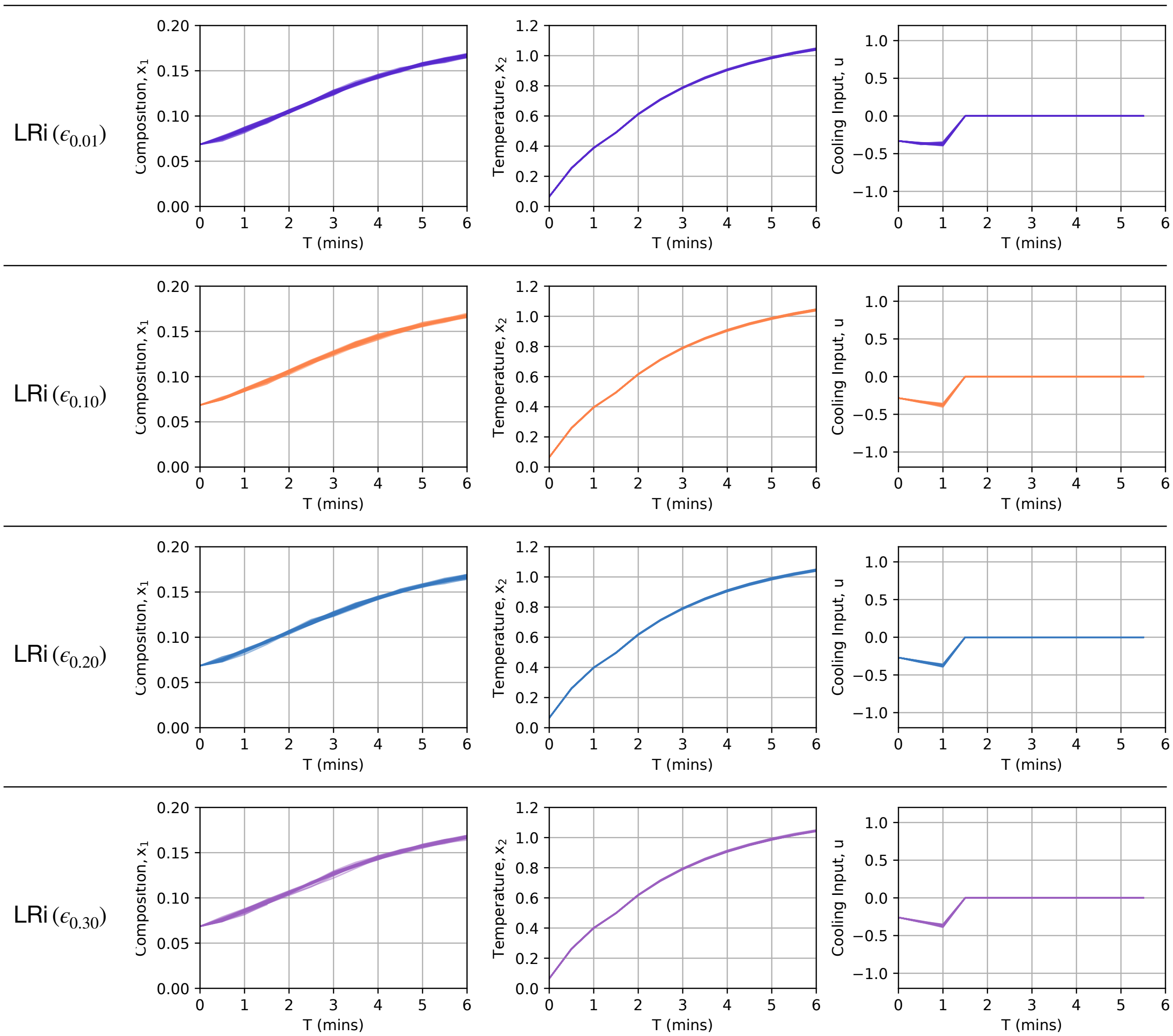

Figure 6.2: Comparison of 30 closed-loop CSTR temperature control trajectories from one initial state for the different LRi controller configurations.

Table 6.1: CSTR temperature control controller performance comparison.

Controller	Success Rate (%)	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)
Smooth 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)

6.4 Autonomous Car 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 [2]:

$$\begin{aligned}
\dot{s}_x &= v\cos(\theta) \\
\dot{s}_y &= v\sin(\theta) \\
\dot{\lambda} &= v_\lambda \\
\dot{v} &= a_{\text{long}} \\
\dot{\theta} &= \frac{v}{l_{wb}}\tan(\lambda),
\end{aligned}$$

where l_{wb} is the vehicle wheelbase. The state variables are position s_x, s_y, steering angle λ, longitudinal 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 [2], and is additionally modified to vary friction across the width of the road. Zero-mean Gaussian noise with covariance

$$\Sigma_w = \operatorname{diag}\left(\begin{bmatrix}0.05 & 0.005 & 0.0001 & 0.001 & 0.0005 & 0.0001 & 0.0001\end{bmatrix}\right)$$

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.

Given the higher control rates of this system, the LRi controller variant from Section 4.6 is used, which we refer to as Learned Residual-informed (Approximate) (LRi-A). The Nominal controller also uses an LTV approximation, but of the nonlinear nominal dynamics, to achieve comparable solve times, as the nonlinear nominal dynamics create a computationally demanding MINLP even without the GP model. The Smooth Robust controller and the oracle use the nonlinear nominal model and simulation model (excluding process noise), respectively. In both controllers, the initial reference trajectories for state and input (or their means for the LRi-A controller) are obtained from a set-point controller and drives the vehicle across the map in a straight line without considering a temporal logic specification. The LRi-A and Nominal controllers use the solver Gurobi [24]. The Smooth Robust controller and the oracle, which do not require linearization, use Ipopt [4].

The double lane change task is specified as follows:

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

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 requirement to avoid the obstacle (i.e. stay in the clear zone) $s \in C$ is encoded as a disjunction of four predicates, the satisfaction of each predicate corresponds to being outside one side of the rectangular obstacle. The lane area to which the vehicle must return is represented as the conjunction of four predicates, satisfying all of which means the vehicle position is inside all of the edges that define the lane. The heading range Θ is defined by the conjunction of two predicates representing the lower and upper bounds for the vehicle heading. The specification length H is 4s, corresponding to a discrete-time control horizon of $N = 32$. As with the CSTR case study, all predicates in the PrSTL specification of the control task have the same ϵ value, and results are included for $\epsilon = \{0.01, 0.10, 0.20, 0.30\}$. The LRi-A and Nominal controllers are tested with two mixed-integer constraint encodings for the PrSTL specification that differ in how the formula is flattened. In Encoding 1, the encoding of the formula always $\Box_{[0,H]} ((s \in L) \wedge (v > v_{\min}) \wedge (\theta \in \Theta))$ is not flattened with its subformulas, even though they are also conjunctions. In encoding 2, the encoding for always is flattened with its conjunction subformulas.

I uniformly sample 30 initial states from a region and 30 closed-loop trajectories from each initial state for each controller. Figure 6.3 shows 30 closed-loop position 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. Figure 6.4 shows the values of the closed-loop control inputs applied by each controller over the same 30 trajectories from one of the sampled initial states. Table 6.2 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 of the LRi-A controller 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.

The Nominal and LRi-A controllers formulated with either Encoding 1 or Encoding 2 returned identical results for success rate and nearly identical results for control effort. However, the controllers formulated with Encoding 1 return reduced solver times compared to controllers formulated with Encoding 2. The LRi-A controllers' solver times with either encoding are on average within the 0.125s control interval.

Table 6.2: Double lane change controller performance comparison.

Controller	Success Rate (%)	Control Cost [Avg. (5%, 95%)]	Solver Time (s) [Avg. (5%, 95%)]
Nominal Encoding 1	0	2.0 (0.1, 15.8)	**0.02 (<0.01, 0.06)**
Nominal Encoding 2	0	2.0 (0.1, 15.9)	0.07 (<0.01, 0.23)
Smooth Robust	84	683.6 (533.2, 891.4)	0.17 (<0.01, 0.54)
LRi-A ($\epsilon_{0.01}$) Encoding 1	**90**	**0.3 (<0.1, 0.4)**	**0.01 (<0.01, 0.04)**
LRi-A ($\epsilon_{0.01}$) Encoding 2	**90**	**0.3 (<0.1, 0.4)**	0.06 (<0.01, 0.27)
LRi-A ($\epsilon_{0.10}$) Encoding 1	81	**0.6 (<0.1, 0.4)**	**0.01 (<0.01, 0.04)**
LRi-A ($\epsilon_{0.10}$) Encoding 2	81	**0.6 (<0.1, 0.4)**	0.06 (<0.01, 0.26)
LRi-A ($\epsilon_{0.20}$) Encoding 1	76	**0.9 (<0.1, 0.4)**	**0.01 (<0.01, 0.04)**
LRi-A ($\epsilon_{0.20}$) Encoding 2	76	**0.9 (<0.1, 0.4)**	0.06 (<0.01, 0.27)
LRi-A ($\epsilon_{0.30}$) Encoding 1	71	**1.0 (<0.1, 0.4)**	**0.01 (<0.01, 0.05)**
LRi-A ($\epsilon_{0.30}$) Encoding 2	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)

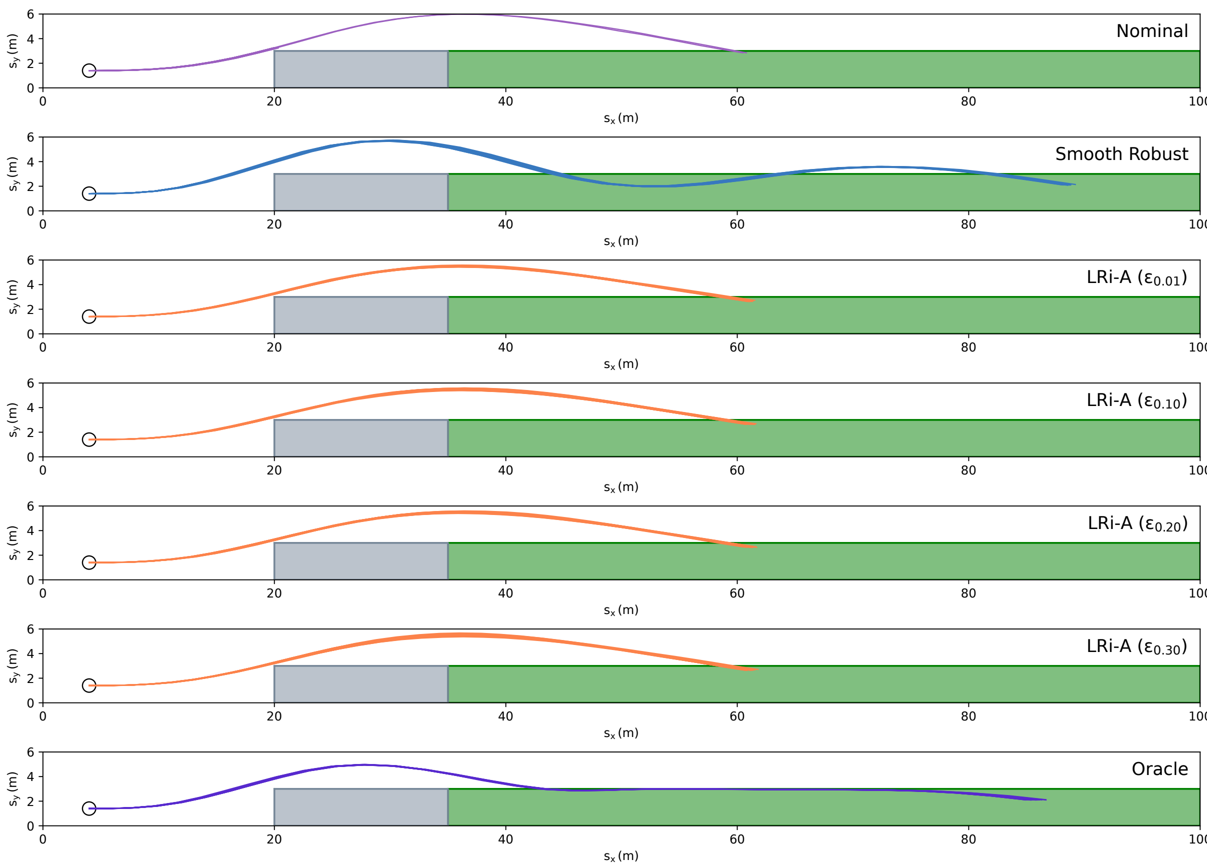

Figure 6.3: Comparison of 30 closed-loop double lane change trajectories from one initial state between the baseline, oracle, and LRi-A controllers. The grey box is the obstacle to be avoided, the green box is the lane that the vehicle must return to.

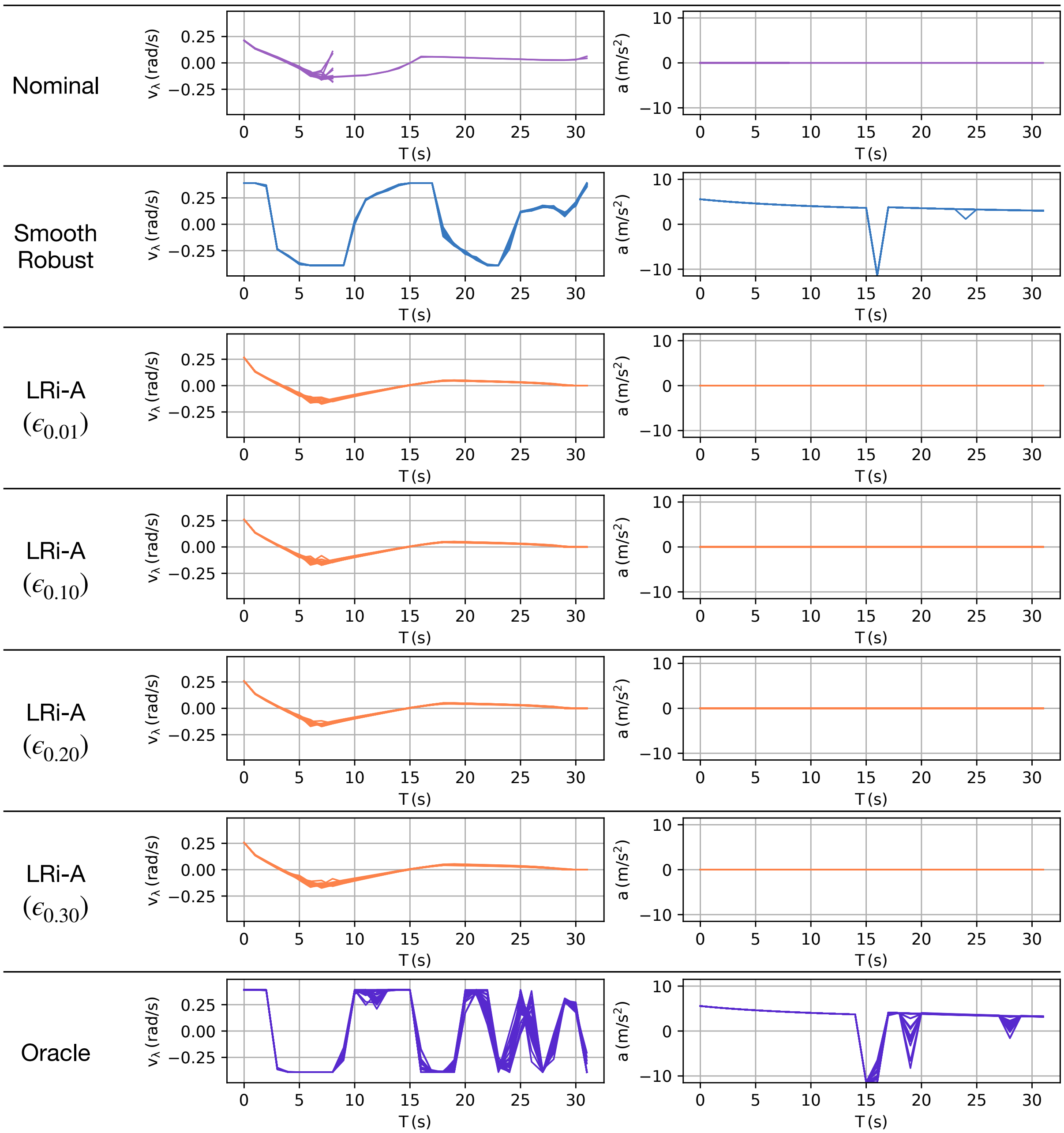

Figure 6.4: Comparison of closed-loop control inputs for the double lane change task applied over 30 trajectories from one initial state between the baseline, oracle, and LRi-A controllers.

6.5 Multi-Agent Continuously-Stirred Tank Reactor Temperature Control

This case study evaluates the distributed controller proposed in Chapter 5, which is referred to as the MA-LRi controller. The MA-LRi controller is evaluated using a simulation case study on the temperature control of two CSTRs. The two CSTR agents have the same simulation dynamics as the CSTR in the single agent case study from Section 6.3, including the disturbance that is an exponential function of the control input. The nominal model and learned GP residual model are also the same as in Section 6.3. The control task is to always maintain both the composition and temperature of reactor agent 1 above those of reactor agent 2, and is specified as follows:

$$\psi = \square_{[0,H]}(x_1^1 > x_2^1) \vee (x_1^2 > x_2^2).$$

The specification length is $H = 6$ minutes, corresponding to a discrete-time control horizon of $N = 12$. In the MA-LRi controller, the constraint that the under-approximated characteristic equation value of the system trajectory with respect to the PrSTL task specification be greater than 0 is relaxed to instead be greater than -0.18 in order to maintain the feasibility of the problem.

The MA-LRi controller is compared with three different centralized controllers: the Nominal and Smooth Robust controller baselines, which include the nominal dynamics of both reactors in the optimization, and an LRi controller configured with $\epsilon = 0.20$, which includes the nominal plus learned dynamics of both reactors in the optimization. The Nominal and LRi controllers minimize the control effort cost $\sum_{k=0}^{N-1}(\|u_k^1\|_2^2 + \|u_k^2\|_2^2)$ and its expectation, respectively, as their objective. Each agent's subproblem in the MA-LRi controller includes the expectation of the control cost for that agent in the objective.

Simulations are run to sample 10 trajectories from one initial state for each controller. Figure 6.5 shows plots comparing the 10 trajectories returned by each controller. Table 6.3 compares the performance of the controllers. The column labeled Agent in Table 6.3 with values 1 or 2 is used to denote reactor agent 1 and reactor agent 2. For the centralized Nominal, Smooth Robust, and LRi controllers, one instance of the centralized optimization is solved for each iteration of the MPC. For the distributed MA-LRi controller, multiple iterations of the distributed optimization take place for each iteration of the MPC. Thus, the solver time for each MPC iteration is determined by summing the process time taken for all of the subproblem iterations during the single MPC iteration. Two solver times are reported, corresponding to reactor agents 1 and 2.

Table 6.3: Multi-agent CSTR temperature control controller performance comparison.

Controller	Agent	Success Count (/10)	Control Cost [Avg. (5%, 95%)]	Solver Time (s) [Avg. (5%, 95%)]
Nominal	1	2	**<0.1 (<0.1, <0.1)**	1.4 (<0.1, 2.4)
	2		**<0.1 (<0.1, <0.1)**	
Smooth Robust	1	**10**	**<0.1 (<0.1, <0.1)**	**<0.1 (<0.1, <0.1)**
	2		12.0 (12.0, 12.0)	
LRi ($\epsilon_{0.20}$)	1	4	**<0.1 (<0.1, <0.1)**	134.0 (<0.5, 284.3)
	2		**<0.1 (<0.1, <0.1)**	
MA-LRi ($\epsilon_{0.20}$)	1	5	**<0.1 (<0.1, <0.1)**	88.8 (0.6, 765.6)
	2		0.1 (0.1, 0.1)	106.6 (0.7, 932.3)

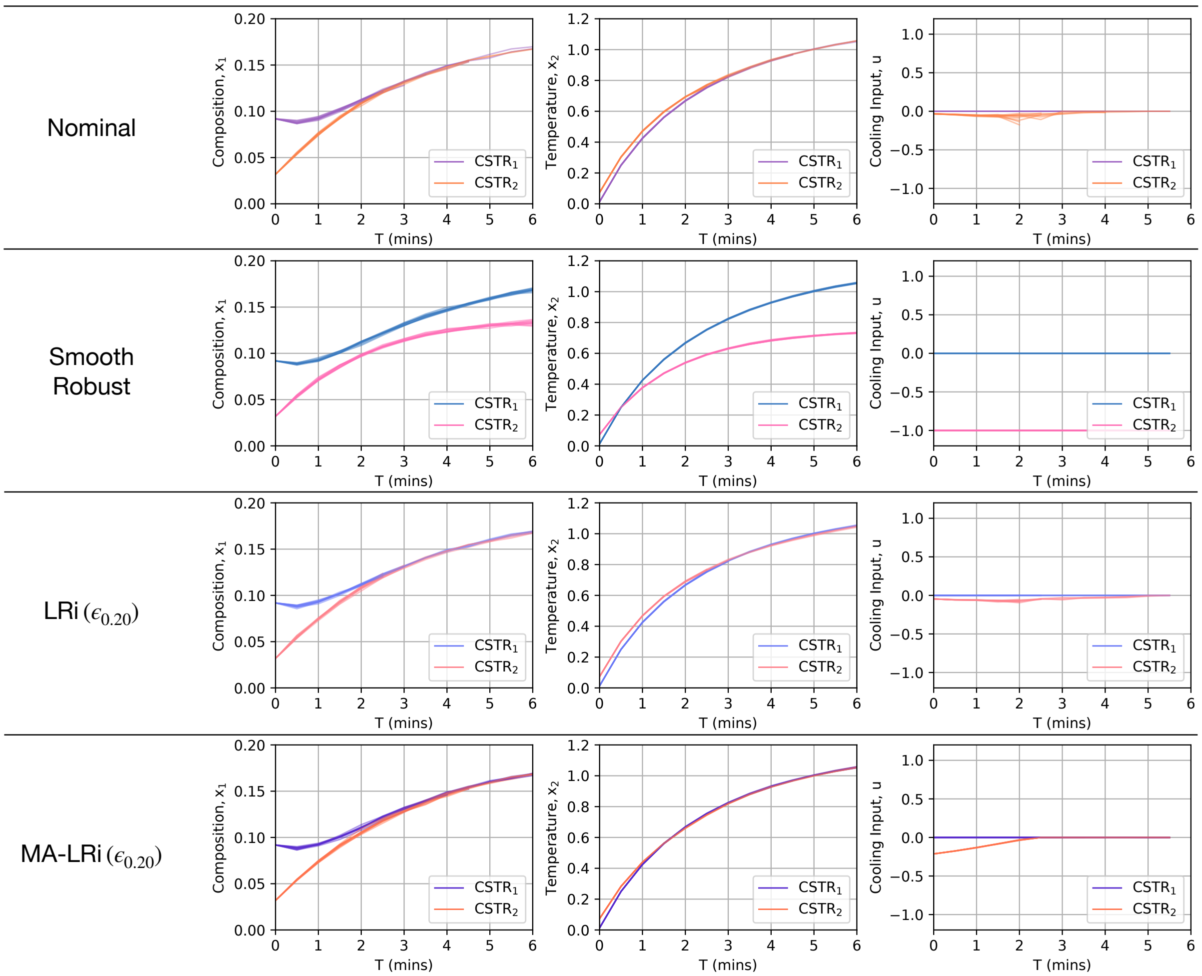

Figure 6.5: Comparison of 10 closed-loop multi-agent CSTR temperature control trajectories from one initial state for centralized Nominal, Smooth Robust, and LRi controllers, and the distributed MA-LRi controller.

Chapter 7

Conclusion

This thesis investigated the problem of controlling a system to satisfy an STL specification when the controller does not have access to the true system dynamics, but only a nominal model and past runtime data. An MPC design is proposed that first uses Gaussian Process regression and past trajectory data to learn a stochastic model of the unknown residual dynamics. To address uncertainty in the predicted state trajectory arising from the learned stochastic model, PrSTL is used to specify the control task. The task specification is encoded as mixed-integer constraints in the control optimization. The resulting control optimization is an MINLP, and a convex local approximation is presented for systems with fast control rates. The proposed controller is assessed in two simulation case studies. The first case study is on a temperature control task for a CSTR, which uses the MINLP control optimization. The second case study is on a double lane change maneuver by an autonomous road vehicle, and uses the convex approximation to the MINLP control optimization due to the higher controller update rates of the system. In these case studies, the proposed controller is compared to existing MPC baselines, which would be limited to the nominal model when only this model and past runtime data are available. Compared to these baselines, the proposed controller returned higher success rates in completing the specified control task. The proposed controller was also compared to an MPC that has access to the full simulation dynamics model, which acts as an oracle. The proposed controller approached the success rate of the oracle in completing the specified control task. An extension of the proposed controller to multi-agent systems is introduced, where the agents must complete a task that depends on the trajectories of multiple agents. This controller uses a smooth approximation of the PrSTL specification and a distributed MPC design. In a simulation case study on the distributed control of two CSTRs with a coupled control task, the distributed controller performed similarly in success rate to the original, centralized version of the proposed controller.

7.1 Limitations and Future Work

The following are some limitations and potential extensions to the work in this thesis.

Online GP Updates: The first limitation that could be addressed in future work would be methods for addressing changes in the unknown residual dynamics over time. To this end, the controller could incorporate updating the GP model of the residual dynamics online.

Control With No Prior Data: Another potential extension would be developing a controller for a scenario where there are unknown residual dynamics but no prior data available. The controller would have to satisfy a task specification with limited initial knowledge of the unknown dynamics, and potentially determine its control inputs considering actions that lead to faster learning of the unknown dynamics. Many related works exist for simultaneously learning the unknown dynamics of and controlling a system without considering control tasks specified in STL or similar logics. In [9], an algorithm is proposed that allows both a nominal plus GP residual model and a control policy to be learned and updated while maintaining stability guarantees. In [25], a distributionally robust MPC with a learned GP dynamics model is proposed, which is robust to differences between the learned GP distribution and the true distribution. In [41], the control policy and unknown dynamics are simultaneously learned for a system that must satisfy an STL specification. The work proposes using CBF to improve safety, while maximizing the expectation of robustness of the STL formula. As with other works using CBF, the controller design is limited to control affine systems. A controller that can consider more general nonlinear dynamics and simulataneously improve its model of the system and satisfy the STL specification would thus improve upon both the work in this thesis and the state of the art.

References

[1] Derya Aksaray, Austin Jones, Zhaodan Kong, Mac Schwager, and Calin Belta. Q-learning for robust satisfaction of signal temporal logic specifications. In *2016 IEEE Conference on Decision and Control*, pages 6565–6570, Las Vegas, NV, USA, December 2016. IEEE.

[2] Matthias Althoff, Markus Koschi, and Stefanie Manzinger. CommonRoad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium*, pages 719–726, 2017.

[3] Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 European Control Conference*, pages 3420–3431. IEEE, June 2019.

[4] Andreas Waechter, Carl Laird. Ipopt, 2024.

[5] Nikos Arechiga. Specifying safety of autonomous vehicles in signal temporal logic. In *2019 IEEE Intelligent Vehicles Symposium*, pages 58–63, 2019.

[6] Giulio Attenni, Viviana Arrigoni, Novella Bartolini, and Gaia Maselli. Drone-based delivery systems: A survey on route planning. *IEEE Access*, 11:123476–123504, 2023.

[7] Calin Belta and Sadra Sadraddini. Formal methods for control synthesis: An optimization perspective. *Annual Review of Control, Robotics, and Autonomous Systems*, 2(1):115–140, May 2019.

[8] Alberto Bemporad and Manfred Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, March 1999.

[9] Felix Berkenkamp, Angela P Schoellig, Matteo Turchetta, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *Conference on Neural Information Processing Systems*, 2017.

[10] Ali Tevfik Büyükkoçak, Derya Aksaray, and Yasin Yazicioglu. Distributed planning of multi-agent Systems with coupled temporal logic specifications. In *AIAA Scitech 2021 Forum*. American Institute of Aeronautics and Astronautics, January 2021.

[11] Gang Cao, Edmund M.-K. Lai, and Fakhrul Alam. Gaussian process model predictive control of an unmanned quadrotor. *Journal of Intelligent & Robotic Systems*, 88(1):147–162, 2017.

[12] Maria Charitidou and Dimos V. Dimarogonas. Splitting and merging control of multiple platoons with signal temporal logic. In *2022 IEEE Conference on Control Technology and Applications*, pages 1031–1036, Trieste, Italy, August 2022. IEEE.

[13] Maria Charitidou and Dimos V. Dimarogonas. Receding horizon control with online barrier function design under signal temporal logic specifications. *IEEE Transactions on Automatic Control*, 68(6):3545–3556, June 2023.

[14] Maria Charitidou and Dimos V. Dimarogonas. Distributed MPC with continuous-time STL constraint satisfaction guarantees. *IEEE Control Systems Letters*, 8:211–216, 2024.

[15] Ítalo Renan Da Costa Barros and Tiago Pereira Nascimento. Robotic mobile fulfillment systems: A survey on recent developments and research opportunities. *Robotics and Autonomous Systems*, 137:103729, March 2021.

[16] Zhao Feng Dai, Yash Vardhan Pant, and Stephen L. Smith. Model predictive control for systems with partially unknown dynamics under signal temporal logic specifications. *IEEE Control Systems Letters*, 8:2931–2936, 2024.

[17] Alexandre Donzé and Oded Maler. Robust satisfaction of temporal logic over real-valued signals. In *Formal Modeling and Analysis of Timed Systems*, volume 6246, pages 92–106. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[18] Leroy D'Souza and Yash Vardhan Pant. Stochastic hybrid model predictive control using Gaussian processes for systems with piecewise residual dynamics. In *2023 American Control Conference*, pages 2670–2677, San Diego, CA, USA, May 2023. IEEE.

[19] Georgios E. Fainekos and George J. Pappas. Robustness of temporal logic specifications. In *Formal Approaches to Software Testing and Runtime Verification*, pages 178–192. Springer, 2006.

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

[21] Samira S. Farahani, Rupak Majumdar, Vinayak S. Prabhu, and Sadegh Soudjani. Shrinking horizon model predictive control with signal temporal logic constraints under stochastic disturbances. *IEEE Transactions on Automatic Control*, 64(8):3324–3331, 2019.

[22] L. Ferranti, R. R. Negenborn, T. Keviczky, and J. Alonso-Mora. Coordination of multiple vessels via distributed nonlinear model predictive control. In *2018 European Control Conference*, pages 2523–2528, Limassol, June 2018. IEEE.

[23] Yann Gilpin, Vince Kurtz, and Hai Lin. A smooth robustness measure of signal temporal logic for symbolic control. *IEEE Control Systems Letters*, 5(1):241–246, 2021.

[24] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024.

[25] Astghik Hakobyan and Insoon Yang. Learning-based distributionally robust motion control with Gaussian processes. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 7667–7674, Las Vegas, NV, USA, October 2020. IEEE.

[26] Lukas Hewing, Juraj Kabzan, and Melanie N. Zeilinger. Cautious model predictive control using Gaussian process regression. *IEEE Transactions on Control Systems Technology*, 28(6):2736–2743, 2020.

[27] Susmit Jha, Vasumathi Raman, Dorsa Sadigh, and Sanjit A. Seshia. Safe autonomy under perception uncertainty using chance-constrained temporal logic. *Journal of Automated Reasoning*, 60(1):43–62, January 2018.

[28] Juraj Kabzan, Lukas Hewing, Alexander Liniger, and Melanie N. Zeilinger. Learning-based model predictive control for autonomous racing. *IEEE Robotics and Automation Letters*, 4(4):3363–3370, 2019.

[29] Edgar D. Klenske, Melanie N. Zeilinger, Bernhard Scholkopf, and Philipp Hennig. Gaussian process-based predictive control for periodic error correction. *IEEE Transactions on Control Systems Technology*, 24(1):110–121, 2016.

[30] J. Kocijan, R. Murray-Smith, C.E. Rasmussen, and A. Girard. Gaussian process model based predictive control. In *2004 American Control Conference*, pages 2214–2219 vol.3, 2004.

[31] Arash Bahari Kordabad, Maria Charitidou, Dimos V. Dimarogonas, and Sadegh Soudjani. Control barrier functions for stochastic systems under signal temporal logic tasks. In *2024 European Control Conference*, pages 3213–3219, Stockholm, Sweden, June 2024. IEEE.

[32] Arash Bahari Kordabad, Eleftherios E. Vlahakis, Lars Lindemann, Dimos V. Dimarogonas, and Sadegh Soudjani. Distributionally Robust Control for Chance-Constrained Signal Temporal Logic Specifications, September 2024. arXiv:2409.03855 [eess].

[33] Vincent Kurtz and Hai Lin. Mixed-integer programming for signal temporal logic with fewer binary variables. *IEEE Control Systems Letters*, 6:2635–2640, 2022.

[34] Karen Leung, Nikos Aréchiga, and Marco Pavone. Backpropagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods. *The International Journal of Robotic Research*, 42(6):356–370, May 2023.

[35] Karen Leung and Marco Pavone. Semi-supervised trajectory-feedback controller synthesis for signal temporal logic specifications. In *2022 American Control Conference*, pages 178–185, Atlanta, GA, USA, June 2022. IEEE.

[36] Lars Lindemann and Dimos V. Dimarogonas. Decentralized robust control of coupled multi-agent systems under local signal temporal logic tasks. In *2018 American Control Conference*, pages 1567–1573, Milwaukee, WI, June 2018. IEEE.

[37] Lars Lindemann and Dimos V. Dimarogonas. Control barrier functions for signal temporal logic tasks. *IEEE Control Systems Letters*, 3(1):96–101, January 2019.

[38] Lars Lindemann and Dimos V. Dimarogonas. Decentralized control barrier functions for coupled multi-agent systems under signal temporal logic tasks. In *2019 European Control Conference*, pages 89–94, Naples, Italy, June 2019. IEEE.

[39] Lars Lindemann and Dimos V. Dimarogonas. Funnel control for fully actuated systems under a fragment of signal temporal logic specifications. *Nonlinear Analysis: Hybrid Systems*, 39:100973, 2021.

[40] Lars Lindemann, Lejun Jiang, Nikolai Matni, and George J. Pappas. Risk of stochastic systems for temporal logic specifications. *ACM Transactions on Embedded Computing Systems*, 22(3):1–31, 2023.

[41] Wenliang Liu, Mirai Nishioka, and Calin Belta. Safe model-based control from signal temporal logic specifications using recurrent neural networks. In *2023 International Conference on Robotics and Automation*, pages 12416–12422, 2023.

[42] Wenliang Liu, Wei Xiao, and Calin Belta. Learning robust and correct controllers from signal temporal logic specifications using BarrierNet. In *2023 IEEE Conference on Decision and Control*, pages 7049–7054, Singapore, Singapore, December 2023. IEEE.

[43] Zhiyu Liu, Bo Wu, Jin Dai, and Hai Lin. Distributed communication-aware motion planning for multi-agent systems from STL and SpaTeL specifications. In *2017 IEEE Conference on Decision and Control*, pages 4452–4457, Melbourne, Australia, December 2017. IEEE.

[44] Zonglin Liu and Olaf Stursberg. Efficient solution of distributed MILP in control of networked systems. *IFAC-PapersOnLine*, 53(2):6723–6729, 2020.

[45] Michael Maiworm, Daniel Limon, and Rolf Findeisen. Online learning-based model predictive control with Gaussian process models and stability guarantees. *International Journal of Robust and Nonlinear Control*, 31(18):8785–8812, December 2021.

[46] Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166. Springer, 2004.

[47] Yunpeng Pan and Jun Wang. Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks. *IEEE Transactions on Industrial Electronics*, 59(8):3089–3101, 2012.

[48] Yash Vardhan Pant, Houssam Abbas, and Rahul Mangharam. Smooth Operator: Control using the smooth robustness of temporal logic. In *2017 IEEE Conference on Control Technology and Applications*, pages 1235–1240, Mauna Lani Resort, HI, USA, August 2017. IEEE.

[49] Yash Vardhan Pant, Houssam Abbas, and Rahul Mangharam. Distributed trajectory planning for multi-rotor uavs with signal temporal logic objectives. In *2022 IEEE Conference on Control Technology and Applications*, pages 476–483, Trieste, Italy, August 2022. IEEE.

[50] Yash Vardhan Pant, Houssam Abbas, Rhudii A. Quaye, and Rahul Mangharam. Fly-by-Logic: Control of multi-drone fleets with temporal logic objectives. In *2018*

ACM/IEEE International Conference on Cyber-Physical Systems, pages 186–197, 2018.

[51] Pierre Bonami, Pietro Belotti, John J. Forrest, Lazlo Ladanyi, Carl Laird, Jon Lee, Francois Margot, Andreas Waechter. Bonmin, 2024.

[52] Joaquin Quinonero-Candela, Agathe Girard, and Carl Edward Rasmussen. Prediction at an uncertain input for Gaussian processes and relevance vector machines application to multiple-step ahead time-series forecasting. Technical report, Technical University of Denmark, DTU, 2003.

[53] Vasumathi Raman, Alexandre Donze, Mehdi Maasoumy, Richard M. Murray, Alberto Sangiovanni-Vincentelli, and Sanjit A. Seshia. Model predictive control with signal temporal logic specifications. In *2014 IEEE Conference on Decision and Control*, pages 81–87, 2014.

[54] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, Mass., 3. print edition, 2008.

[55] Meisam Razaviyayn, Mingyi Hong, Zhi-Quan Luo, and Jong-Shi Pang. Parallel successive convex approximation for nonsmooth nonconvex optimization. In *Conference on Neural Information Processing Systems*, 2014.

[56] Felix Rey, Zhoudan Pan, Adrian Hauswirth, and John Lygeros. Fully decentralized ADMM for coordination and collision avoidance. In *2018 European Control Conference*, pages 825–830, Limassol, June 2018. IEEE.

[57] Dorsa Sadigh and Ashish Kapoor. Safe control under uncertainty with probabilistic signal temporal logic. In *Robotics: Science and Systems*, 2016.

[58] Sadra Sadraddini and Calin Belta. Robust temporal logic model predictive control. In *2015 Annual Allerton Conference on Communicatio, Control and Computing*, pages 772–779, 2015.

[59] Sleiman Safaoui, Lars Lindemann, Dimos V. Dimarogonas, Iman Shames, and Tyler H. Summers. Control design for risk-based signal temporal logic specifications. *IEEE Control Systems Letters*, 4(4):1000–1005, 2020.

[60] Yunus Emre Sahin, Rien Quirynen, and Stefano Di Cairano. Autonomous vehicle decision-making and monitoring based on signal temporal logic and mixed-integer

programming. In *2020 American Control Conference*, pages 454–459, Denver, CO, USA, July 2020. IEEE.

[61] Mayank Sewlia, Christos K. Verginis, and Dimos V. Dimarogonas. Cooperative object manipulation under signal temporal logic tasks and uncertain dynamics. *IEEE Robotics and Automation Letters*, 7(4):11561–11568, October 2022.

[62] Ola Shorinwa and Mac Schwager. Distributed model predictive control via separable optimization in multiagent networks. *IEEE Transactions on Automatic Control*, 69(1):230–245, January 2024.

[63] Abhinav Sinha and Rajiv Kumar Mishra. Temperature regulation in a continuous stirred tank reactor using event triggered sliding mode control. *IFAC-PapersOnLine*, 51(1):401–406, 2018.

[64] Enrico Terzi, Fabio Bonassi, Marcello Farina, and Riccardo Scattolini. Learning model predictive control with long short-term memory networks. *International Journal of Robust and Nonlinear Control*, 31(18):8877–8896, 2021.

[65] Andrea Testa, Alessandro Rucco, and Giuseppe Notarstefano. Distributed mixed-integer linear programming via cut generation and constraint exchange. *IEEE Transactions on Automatic Control*, 65(4):1456–1467, April 2020.

[66] A. Uppal, W.H. Ray, and A.B. Poore. On the dynamic behavior of continuous stirred tank reactors. *Chemical Engineering Science*, 29(4):967–985, 1974.

[67] Harish Venkataraman, Derya Aksaray, and Peter Seiler. Tractable reinforcement learning of signal temporal logic objectives. *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, 120:308–317, 2020.

[68] Eleftherios E. Vlahakis, Lars Lindemann, and Dimos V. Dimarogonas. Distributed sequential receding horizon control of multi-agent systems under recurring signal temporal logic. In *2024 European Control Conference*, pages 305–310, Stockholm, Sweden, June 2024. IEEE.

[69] Yu Wang, Wotao Yin, and Jinshan Zeng. Global Convergence of ADMM in Nonconvex Nonsmooth Optimization. *Journal of Scientific Computing*, 78(1):29–63, January 2019.

[70] Tumin Wu, Mohammed M. Olama, Seddik M. Djouadi, Jin Dong, Yaosuo Xue, and Teja Kuruganti. Signal temporal logic control for residential HVAC systems to accommodate high solar PV penetration. In *2020 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference*, pages 1–5, 2020.

[71] Tiange Yang, Yuanyuan Zou, Shaoyuan Li, and Yaru Yang. Distributed model predictive control for probabilistic signal temporal logic specifications. *IEEE Transactions on Automation Science & Engineering*, pages 1–11, 2024.

[72] Xiang Yin, Bingzhao Gao, and Xiao Yu. Formal synthesis of controllers for safety-critical autonomous systems: Developments and challenges. *Annual Reviews in Control*, 57:100940, 2024.

[73] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020.

[74] Xiaoyi Zhou, Tiange Yang, Yuanyuan Zou, Shaoyuan Li, and Hao Fang. Multiple subformulae cooperative control for multiagent systems under conflicting signal temporal logic tasks. *IEEE Transactions on Industrial Electronics*, 70(9):9357–9367, September 2023.