

A Tutorial on Gaussian Process Learning-based Model Predictive Control

JIE WANG*, Concordia University, Canada

YOUNMIN ZHANG, Concordia University, Canada

This tutorial provides a systematic introduction to Gaussian process learning-based model predictive control (GP-MPC), an advanced approach integrating Gaussian process (GP) with model predictive control (MPC) for enhanced control in complex systems. It begins with GP regression fundamentals, illustrating how it enriches MPC with enhanced predictive accuracy and robust handling of uncertainties. A central contribution of this tutorial is the first detailed, systematic mathematical formulation of GP-MPC in literature, focusing on deriving the approximation of means and variances propagation for GP multi-step predictions. Practical applications in robotics control, such as path-following for mobile robots in challenging terrains and mixed-vehicle platooning, are discussed to demonstrate the real-world effectiveness and adaptability of GP-MPC. This tutorial aims to make GP-MPC accessible to researchers and practitioners, enriching the learning-based control field with in-depth theoretical and practical insights and fostering further innovations in complex system control.

CCS Concepts: • Computer systems organization → Robotics; • Computing methodologies → Gaussian processes.

Additional Key Words and Phrases: Gaussian process, model predictive control, learning-based control, mobile robotics, dynamic modeling

ACM Reference Format:

Jie Wang and Youmin Zhang. 2024. A Tutorial on Gaussian Process Learning-based Model Predictive Control. *ACM Comput. Surv.* 1, 1 (April 2024), 34 pages. <https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

1 INTRODUCTION

The pursuit of sophisticated control strategies in robotics has been increasingly directed towards approaches that can effectively navigate the complexities and uncertainties inherent in real-world environments [1]. Traditional control methods often struggle with these dynamic and unpredictable aspects. This has led to a growing interest in *learning-based* control solutions, particularly Gaussian process (GP) learning-based model predictive control (GP-MPC) [2]. This method synergizes the probabilistic modeling capabilities of Gaussian processes (GPs) with the forward-looking computing and optimization features of model predictive control (MPC). The integration of GP into MPC frameworks introduces a probabilistic layer that adeptly manages uncertainties and improves the precision of future state prediction, a critical advantage in complex, real-world applications such as autonomous navigation and advanced robotics [3].

*Corresponding author.

Authors' addresses: Jie Wang, jwangjie@outlook.com, Concordia University, 1455 Blvd. De Maisonneuve Ouest, Montreal, QC, Canada, H3G 1M8; Youmin Zhang, ymzhang@encs.concordia.ca, Concordia University, 1455 Blvd. De Maisonneuve Ouest, Montreal, QC, Canada, H3G 1M8.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM 0360-0300/2024/4-ART

<https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

GP is the most widely used surrogate model in Bayesian machine learning, offering a probabilistic framework for approximating the true objective function to predict the behavior of systems [4]. Unlike deterministic models, GPs estimate a probability distribution over possible functions that fit the observed data, enabling them to predict future observations along with a measure of uncertainty (variance) [5]. This characteristic is particularly valuable in control systems, where the ability to quantify uncertainty in predictions can dramatically enhance the robustness and reliability of the control strategy [6].

MPC is an advanced forward-looking control strategy that uses a dynamic model of the system to predict and optimize the control action over a future time horizon [7]. By solving an optimization problem at each time step, MPC determines control inputs that optimize future system performance, subject to constraints on states, inputs, and outputs [8]. The predictive nature of MPC, combined with its constraint-handling capabilities, makes it highly effective for managing complex dynamic systems and constraints [9].

Integrating GP with MPC creates a new paradigm in control strategies. It enhances the accuracy of MPC's predictive model and incorporates model uncertainties directly into the control optimization process [10]. This leads to more informed, nuanced decisions considering the system's inherent dynamic uncertainties. The key challenge lies in the mathematical derivation and integration of the GP model within the MPC framework, which requires systematic techniques such as linearization for approximating non-Gaussian means and uncertainties in multi-step GP predictions.

Recent literature has provided high-quality reviews and tutorials on learning-based control, focusing on learning-based MPC [2], GP-based control [11], and the intersection of control and reinforcement learning from a safety perspective [1]. These works offer in-depth insights into the state-of-the-art in learning-based control but often assume a solid understanding of GP learning and control theories. However, there exists a gap in the comprehensive understanding of the mathematical principles underpinning the integration of GP into MPC frameworks, particularly among robotics and control communities.

This tutorial aims to fill this gap by providing an accessible introduction to GP-based MPC. Starting with an intuitive explanation of GP regression, the tutorial then delves into the mathematical foundations required for integrating GP into MPC, and finally explores advanced applications of GP-MPC in robotic and autonomous vehicle control. By providing a solid theoretical foundation coupled with practical application examples, this tutorial not only provides a comprehensive guide to understanding and implementing GP-MPC but also inspires further research and development in the field of GP-MPC.

The subsequent sections of this paper are structured as follows. Sec. 2 introduces Gaussian process regression fundamentals. Sec. 3 delves into the mathematical derivation for integrating GP predictions within MPC, particularly focusing on approximating GP means and uncertainties for multi-step forecasts. Sec. 4 discusses employing GP means within MPC without considering uncertainties; Sec. 5 and 6 present practical applications of GP-MPC in robotic controls, with Sec. 5 on advanced path-following for wheeled robots and Sec. 6 on mixed-vehicle platooning controls, integrating both GP means and uncertainties. Finally, Sec. 7 offers concluding remarks.

2 GAUSSIAN PROCESSES

Gaussian process regression (GPR) is a powerful machine learning method used for making predictions, especially useful when dealing with uncertainties and complex behaviors in systems. This section is designed to provide a solid understanding of the basics of GPR. We aim to explain GPR concepts in a way that is accessible to everyone, regardless of their prior knowledge in this area. It sets the foundation for combining GPR with MPC later on.

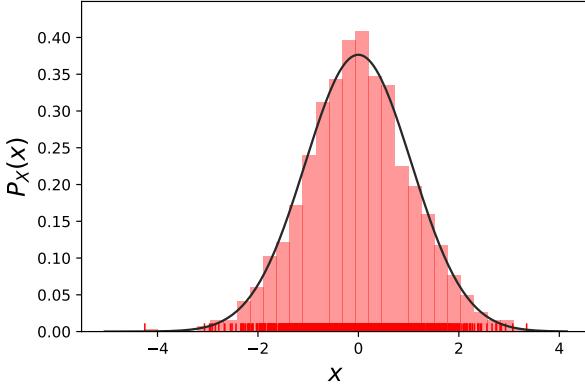


Fig. 1. Visualization of a Gaussian distribution: 1000 data points sampled from a standard Gaussian distribution are represented as red vertical bars along the X-axis. The accompanying curve, a two-dimensional bell curve, illustrates the probability density function (PDF) of the distribution [5].

2.1 Mathematical Basics

This part introduces the essential mathematical concepts that GPR is built on, including the multivariate normal (MVN) distribution, kernels, and the non-parametric nature of GPR [5].

2.1.1 Gaussian and Multivariate Normal Distributions. Central to GPR are the Gaussian/normal distribution and its generalization to multiple dimensions, the MVN distribution. These distributions form the probabilistic cornerstone for modeling and making predictions in systems with inherent uncertainties. A Gaussian distribution, or univariate normal distribution, for a variable X is defined by its probability density function (PDF):

$$P_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \quad (1)$$

where x represents the real argument of X . The behavior of X is entirely determined by its mean μ and variance σ^2 as $P_X(x) \sim N(\mu, \sigma^2)$. In Fig. 1, the PDF of a univariate normal distribution (one-dimensional) is visualized by plotting 1000 data points sampled from a normal distribution.

For systems characterized by multiple related variables $\mathbf{x} = [x_1, x_2, \dots, x_M]^\top$, the Gaussian framework extends to MVN. This extension is essential for modeling their joint behavior. The PDF of an MVN for a vector \mathbf{x} in an M -dimensional space, with a mean vector μ and a covariance matrix Σ , is expressed as [5]:

$$N(\mathbf{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{\frac{M}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left[-\frac{(\mathbf{x}-\mu)^\top \Sigma^{-1} (\mathbf{x}-\mu)}{2}\right]. \quad (2)$$

This formula captures the inter-variable correlations within vector \mathbf{x} . Here, Σ is a symmetric matrix that defines the covariance, or the joint variability, between every pair of elements in \mathbf{x} . The matrix is pivotal in determining how changes in one variable affect others, encapsulating the essence of their interdependencies. The mean vector μ and the covariance matrix Σ are crucial parameters that define this distribution. Specifically, $\mu = \mathbb{E}[\mathbf{x}] \in \mathbb{R}^M$ represents the expected value (or the average) of the vector \mathbf{x} , indicating the central tendency or the typical value each variable in \mathbf{x} is expected to take. On the other hand, $\Sigma = \text{cov}[\mathbf{x}] \in \mathbb{R}^{M \times M}$, with each element Σ_{ij} , quantifies the covariance between the i -th and j -th elements of \mathbf{x} . Covariance measures how two variables

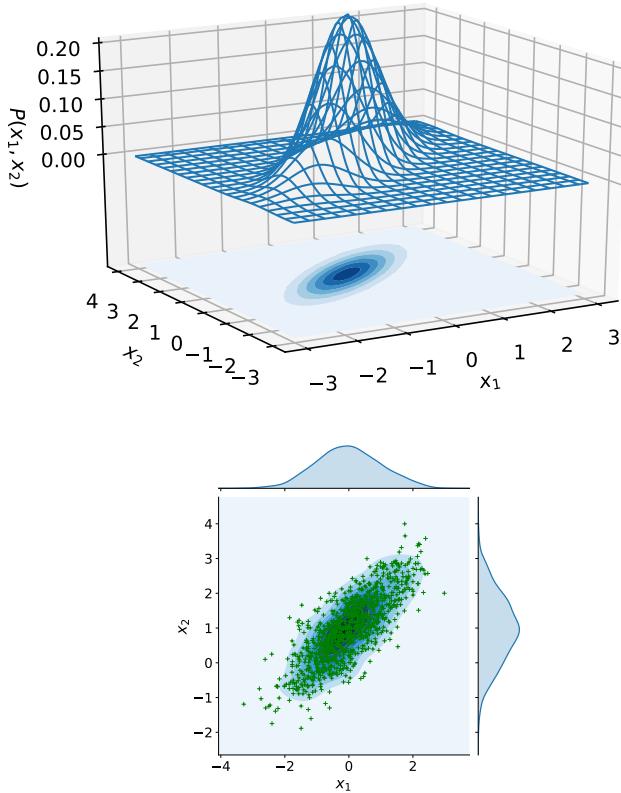


Fig. 2. Visualization of a bivariate normal distribution: The upper image shows a 3-D bell curve for probability density, and the lower images display 2-D ellipse contours indicating the correlation between joint variables x_1 and x_2 [5].

fluctuate together; a positive covariance indicates that two variables tend to move in the same direction, whereas a negative covariance signifies that they move inversely. The diagonal elements of Σ , which represent the variance of each variable, show the extent to which each variable varies from its mean, providing insights into the spread or dispersion of the data.

For intuitive understanding, a bivariate normal (BVN) distribution offers a simpler illustration of an MVN. In Fig. 2, a BVN distribution is visualized as a three-dimensional bell curve with the vertical axis (height) representing the probability density. The correlation between two variables x_1 and x_2 is illustrated by the shape of two-dimensional ellipse contour projections. The function $P(x_1, x_2)$ denotes the joint probability density of x_1 and x_2 . The BVN distribution is formulated as:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N}(\mu, \Sigma) = \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix}\right), \quad (3)$$

where μ is a two-dimensional vector with μ_1 and μ_2 represent means of x_1 and x_2 , respectively. In the matrix Σ , the diagonal elements σ_{11} and σ_{22} are variances of x_1 and x_2 , and the off-diagonal terms σ_{12} and σ_{21} describes their covariances.

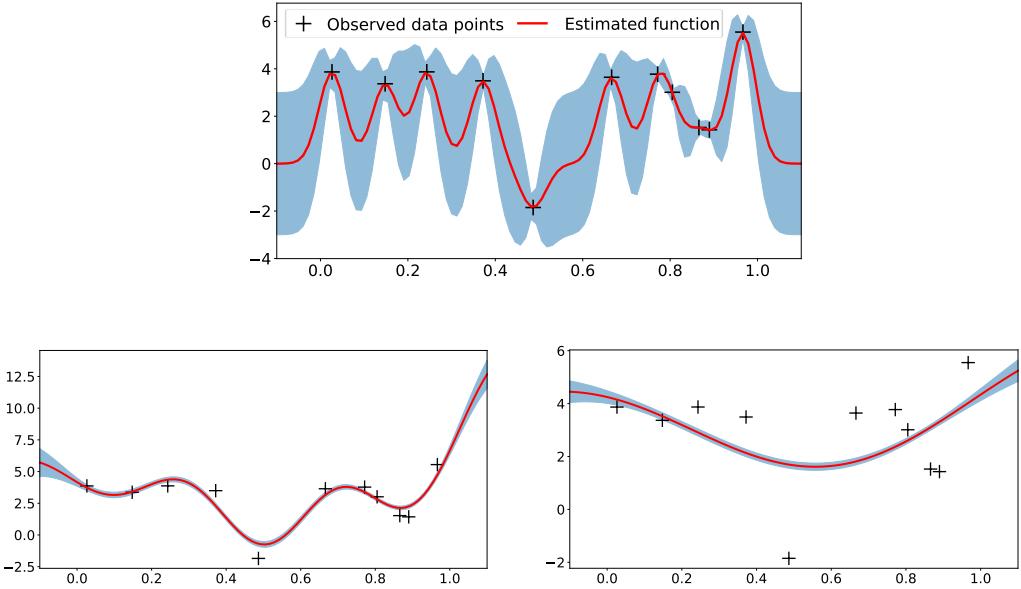


Fig. 3. Impact of the length scale hyperparameter l on model smoothness: smaller l values lead to higher sensitivity to data variations (top), medium l balances sensitivity and smoothness (lower left), and larger l values favor smoothness over sensitivity to local changes (lower right) [5].

2.1.2 Kernels. Kernel functions are pivotal in GPR to enable effective modeling of relationships between data points across high-dimensional spaces. These functions are based on the principle that data points closer together in the input space should have similar outputs, which is essential for achieving accurate regression outcomes. By controlling the model's smoothness and flexibility, kernel functions enable GPR to adeptly handle complex data patterns and facilitate accurate interpolation of observed data for precise prediction [5].

The choice of a kernel function can significantly affect the model's performance, with options ranging from the widely used radial basis function (RBF) to more specialized or custom kernels tailored to specific data properties like continuity, smoothness, periodicity, and expected trend in the data [12]. The RBF kernel, expressed as:

$$k(x_i, x_j) = \sigma_f^2 \exp\left[-\frac{1}{2}(x_i - x_j)^\top l^{-1}(x_i - x_j)\right], \quad (4)$$

is favored for its simplicity and efficacy, particularly in domains such as robotic control [1, 3]. The hyperparameters σ_f and l influence the vertical scale and the smoothness of the regression model, respectively, which are crucial in shaping the model's behavior. For instance, adjusting the length scale parameter l directly impacts the model's sensitivity to input data changes, as depicted in Fig. 3. A smaller l value increases the model's responsiveness to minor variations, potentially leading to overfitting, while a larger l enhances smoothness and generalization capabilities.

Hyperparameter optimization is crucial for customizing the GPR model to accurately reflect the underlying structure of the observed data. This process is conducted through the maximization of the log marginal likelihood [13]:

$$\theta^* = \arg \max_{\theta} \log p(\mathbf{y} | \mathbf{x}, \theta). \quad (5)$$

The process of optimization in GPR seeks a delicate balance between fitting the observed data accurately and maintaining the model's ability to generalize to new data. This equilibrium is crucial for avoiding overfitting, a scenario where the model is overly complex, mirroring the training data too closely at the expense of its predictive capability on unseen data (Fig. 3). Through meticulous adjustment of hyperparameters, the optimization process not only aligns the model with the observed data but also regulates its complexity. This regulation optimizes the predictive uncertainty $\Sigma(\mathbf{x}_*)$, thereby boosting the model's confidence and reliability across various inputs. As depicted in Fig. 4, optimizing hyperparameters significantly enhances the model's accuracy and reliability compared to scenarios where parameters are not optimized (Fig. 3).

2.1.3 Non-parametric Modeling. Understanding the key difference between parametric and non-parametric models [14] is essential for understanding the capabilities of GPR. Parametric models, characterized by a fixed set number of parameters, offer limited complexity and adaptability. For example, a linear regression model, defined as $y = \theta_1 + \theta_2 x$, assumes a direct relationship between x and y through fixed parameters θ_1 and θ_2 . While increases complexity by adding more terms, as in a polynomial model $y = \theta_1 + \theta_2 x + \theta_3 x^2$, the model is still confined to a predetermined structure, which potentially limits its ability to capture more intricate data patterns.

Typically, in regression tasks, we use a training dataset D comprising n observed points, denoted as $D = [(x_i, y_i) | i = 1, \dots, n]$, to establish a mapping from input values x to output values y through a set of basis functions $f(x)$. Parametric models simplify this process by summarizing the data with a finite set of parameters θ , allowing for predictions on new inputs \mathbf{x}_* to be made without direct reference to the original dataset D after the model has been trained. Mathematically, this concept is expressed as $P(f_* | \mathbf{x}_*, \theta, D) = P(f_* | \mathbf{x}_*, \theta)$, where f_* represents the model's predictions for new, unseen data points \mathbf{x}_* .

Non-parametric models like GPR, in contrast, do not confine their complexity to a predetermined number of parameters. They allow the model's complexity to grow with the observed dataset to enhance their flexibility and capacity to model complex relationships. By leveraging an infinite-dimensional feature space through kernels, GPR excels in tasks requiring precise predictions and uncertainty modeling. This adaptability is particularly valuable for tackling the diverse and intricate patterns found in real-world systems, where traditional parametric methods may fall short.

2.2 Gaussian Process Regression

GPR offers a unique methodology for modeling functions f from observed data \mathcal{D} , without the need for specifying a predefined function form. In general, the dataset \mathcal{D} comprises input vectors $\mathbf{x} = [x_1, \dots, x_n]^T \in \mathbb{R}^{n_a \times n}$ and corresponding output values $\mathbf{f} = [f(x_1), \dots, f(x_n)]^T \in \mathbb{R}^{1 \times n}$, where n denotes the total number of observations within \mathcal{D} . Unlike conventional regression methods that seek to identify a singular best-fit function, GPR models a distribution over possible functions that fit the observed data. This probabilistic approach is illustrated in Fig. 4, emphasizing GPR's capacity to accommodate the inherent uncertainties in modeling complex data relationships.

GPR constructs this model by assuming that the observed data can be described by an MVN, with the probability of any set of function values \mathbf{f} given inputs \mathbf{x} as:

$$P(\mathbf{f} | \mathbf{x}) = \mathcal{N}(\mathbf{f} | \boldsymbol{\mu}, \mathbf{K}), \quad (6)$$

Here, the variable $\boldsymbol{\mu} = [m(x_1), \dots, m(x_n)]$ represents the mean function, and \mathbf{K} is the covariance matrix computed using the chosen kernel function, composed of positive definite elements $K_{ij} = k(x_i, x_j)$. With no observation, we default the mean function to $m(\mathbf{x}) = 0$, assuming the data is normalized to zero mean.

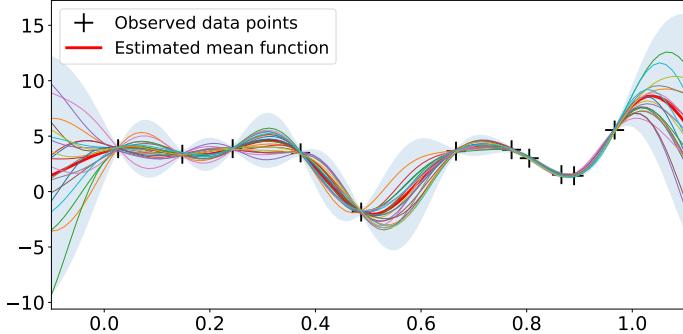


Fig. 4. GPR visualization: Black crosses denote observed data points. From these, GPR considers all possible functions (illustrated by 20 samples in assorted colors) that align with the data. The solid red line represents the GP mean function, calculated from the distribution of these possible infinite numbers of functions. The surrounding blue-shaded region illustrates the prediction uncertainty, shown as 3 times the standard deviation from the mean. This analysis employs an RBF kernel with optimized hyperparameters ($\sigma_f = 0.0067$, $l = 0.0967$), demonstrating the model's improved predictive accuracy and reliability [5].

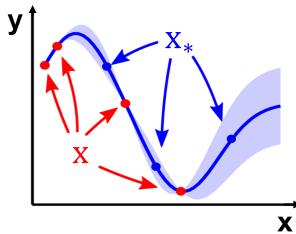


Fig. 5. An illustrative process of conducting regressions by Gaussian processes. The red points are observed data, the blue line represents the mean function estimated by the observed data points, and predictions will be made at new blue points [5].

The regression process with GP models is depicted in Fig. 5, where observed data points (red points \mathbf{x}) and the mean function f (blue line) estimated from these observed data points, predict at new points \mathbf{x}^* , resulting in f^* . Predicting new values f^* at points \mathbf{x}^* involves the joint Gaussian distribution of both observed and predicted values:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} \mu(\mathbf{x}) \\ \mu(\mathbf{x}^*) \end{bmatrix}, \begin{bmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^\top & \mathbf{K}_{**} \end{bmatrix} \right), \quad (7)$$

where $\mathbf{K} = K(\mathbf{x}, \mathbf{x})$, $\mathbf{K}_* = K(\mathbf{x}, \mathbf{x}^*)$, and $\mathbf{K}_{**} = K(\mathbf{x}^*, \mathbf{x}^*)$ represent the covariance matrices between observed data, observed and new data, and new data points, respectively. While this equation outlines the joint probability distribution $P(\mathbf{f}, \mathbf{f}^* | \mathbf{x}, \mathbf{x}^*)$ for both the observed and predictive function values, regression analysis requires the conditional distribution $P(\mathbf{f}^* | \mathbf{f}, \mathbf{x}, \mathbf{x}^*)$ to predict exclusively on \mathbf{f}^* given observed data. This conditional distribution is extracted from the joint distribution by applying the marginal and conditional distributions of the MVN theorem [15, Sec. 2.3.1] as:

$$\mathbf{f}^* | \mathbf{f}, \mathbf{x}, \mathbf{x}^* \sim \mathcal{N} (\mathbf{K}_*^\top \mathbf{K}^{-1} \mathbf{f}, \mathbf{K}_{**} - \mathbf{K}_*^\top \mathbf{K}^{-1} \mathbf{K}_*). \quad (8)$$

The conditional distribution provides the predictive mean and variance for \mathbf{f}_* , offering not only accurate predictions at new points but also quantifies the uncertainty associated with these predictions.

Real-world observations often include noises, modeled as $\mathbf{y} = \mathbf{f}(\mathbf{x}) + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is an additive Gaussian noise with variance σ_n^2 . The prior covariance of \mathbf{y} becomes $\text{cov}(\mathbf{y}) = \mathbf{K} + \sigma_n^2 \mathbf{I}$. The inclusion of observational noise refines the GPR model's predictive equations to [13]:

$$\tilde{\mathbf{f}}_* | \mathbf{x}, \mathbf{y}, \mathbf{x}_* \sim \mathcal{N}(\boldsymbol{\mu}(\mathbf{x}_*), \Sigma(\mathbf{x}_*)), \quad (9)$$

where the predictive mean $\boldsymbol{\mu}(\mathbf{x}_*)$ and variance $\Sigma(\mathbf{x}_*)$ are determined as:

$$\boldsymbol{\mu}(\mathbf{x}_*) = \mathbf{K}_{* *}^\top [\mathbf{K} + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{y}, \quad (10a)$$

$$\Sigma(\mathbf{x}_*) = \mathbf{K}_{* *} - \mathbf{K}_{* *}^\top [\mathbf{K} + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{K}_{* *}. \quad (10b)$$

In this expression, the variance function $\Sigma(\mathbf{x}_*)$ reveals that the uncertainty in predictions depends solely on the input values \mathbf{x} and \mathbf{x}_* , not on the observed outputs \mathbf{y} . This characteristic is a distinctive property of GPR [13].

A GP model describes a probability distribution over possible functions that fit a set of points. In essence, GPR is characterized by its non-parametric nature, leveraging a probability distribution over all possible functions as a mean function used for regression predictions. This approach allows for flexible modeling of complex relationships and provides a robust framework for making predictions with quantified uncertainty, making it a powerful tool in statistical learning and data analysis.

2.3 Sparse Gaussian Process

As shown in (10a) and (10b), the computational demands of the mean and variance in the standard GP model scale as $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$, respectively, which increases with the number of training data points n [16]. This scaling is due to the need to invert a large $n \times n$ covariance matrix, rendering standard GPR computationally intensive for large datasets. Such computational requirements could limit the applicability of traditional GPR in scenarios requiring real-time analysis or involving large-scale data, such as model-based control in robotics [17].

The sparse GPR technique offers a solution to these challenges by utilizing a subset of the full dataset, known as inducing points, to approximate the complete GP model [18]. This strategy focuses on condensing the full dataset's information into a smaller set of inducing variables. Consequently, this approach significantly reduces the size of the covariance matrix that needs to be inverted, thus lowering the computational complexity to depend on the number of inducing points \tilde{n} , rather than the full dataset size n . The computational complexity of sparse GP models is $\mathcal{O}(\tilde{n}^2 n)$ for the mean and $\mathcal{O}(\tilde{n}^3)$ for the variance, thereby providing a more scalable alternative to traditional GPR [19].

Choosing the right inducing points $\mathbf{x}^{\text{ind}} = [x_1^{\text{ind}}, \dots, x_{\tilde{n}}^{\text{ind}}]^\top$ is crucial for preserving the accuracy of the model. These points aim to encapsulate the essential information of the dataset, selected to reflect its underlying structure with minimal information loss. Techniques like the fully independent conditional (FIC) approximation method facilitate this by optimizing the placement of inducing points automatically to maximize the model's likelihood, which is convenient and has been widely used to achieve computational efficiency with reasonable accuracy loss [20].

3 APPROXIMATING MEAN AND UNCERTAINTY PROPAGATION

This section investigates the integration of GP with MPC, specifically targeting the creation of critical mathematical equations to approximate mean and uncertainty within the MPC prediction horizon. Such approximations are crucial due to the non-Gaussian outputs produced by GP models across multiple prediction steps. This characteristic of GP models requires approximation techniques

to enable their practical integration into MPC frameworks. By accurately approximating mean and uncertainty, GP models can be effectively combined within MPC strategies, thereby improving control strategy performance and reliability in the face of system unpredictability and variability.

Highlighting this section is a key contribution of our tutorial. It offers a systematic and detailed explanation of GP-MPC integration, marking a novel contribution that addresses a notable void in the current literature. This section lays a solid foundation for understanding GP-MPC integration theoretically, leading to comprehensive discussions about its application in real-world scenarios. Through this exposition, our tutorial distinguishes itself as a critical resource for advancing the field of learning-based control systems, particularly in applications requiring sophisticated handling of uncertainty.

Consider a nonlinear dynamical system represented by a discrete-time state-space model [19]:

$$\underbrace{x_{k+1}}_{\text{prior nominal model}} = \underbrace{f(x_k, u_k)}_{(11a)} + \underbrace{g(x_k, u_k)}_{(11b)},$$

$$= f(x_k, u_k) + B_d(d(x_k, u_k) + \omega_k).$$

In this system model, x_k and u_k represent the state and control input vectors at time step k , respectively. The system dynamics are captured by two components: a prior nominal model $f(x_k, u_k)$, which represents the known part of the system, and a learning-based model $g(x_k, u_k)$, designed to compensate for discrepancies between the nominal model and actual system behaviors. To manage the dimensionality of the learning-based model's output and direct its influence to specific subsystems or states, a matrix B_d precedes $d(x_k, u_k)$. This matrix B_d enables the model to target only a subset of the system states x , thus enhancing computational efficiency without sacrificing model fidelity. Additionally, the model includes spatially uncorrelated Gaussian noise ω_k with $\omega_k \sim \mathcal{N}(0, \Sigma^\omega)$, where $\Sigma^\omega = \text{diag}([\sigma_1^2, \dots, \sigma_{n_d}^2])$ represents the noise's diagonal variance matrix. It is worth noting that the use of B_d is mainly aimed at enhancing computational efficiency by reducing the learning-based model's output on a subset of dimensions. However, it is entirely feasible to set $B_d = I$, which allows the GP model $d(x_k, u_k)$ to influence across all system states x .

Using (9), the learning-based model $g(x_k, u_k)$ in (11a) is approximated as a GP model $d(x_k, u_k)$ in (11b) as:

$$d(x_k, u_k) \sim \mathcal{N}\left(\mu^d(x_k, u_k), \Sigma^d(x_k, u_k)\right), \quad (12)$$

which captures the learning-based adjustments to the system dynamics.

In the framework of GP modeling, the inputs at the current time step are assumed to follow a Gaussian distribution. However, the outputs of the GP model (posterior predictions) result in stochastic variables with unknown distribution. With such uncertain inputs (the predictive posterior of the current time step), the distribution of the GP output at the next time step is not Gaussian anymore [21]. This is a problem to use (11b) as the prediction model in MPC, which needs to make predictions for multiple time steps ahead.

To facilitate the integration of GP models with MPC frameworks, a common approach involves approximating the state distribution at each future time step as Gaussian, denoted by $x_k \sim \mathcal{N}(\mu_k^d, \Sigma_k^d)$ [2]. This approximation enables the use of GP models within MPC by simplifying the predictive state distributions into a form that is tractable for MPC algorithms. However, it is essential to acknowledge that this simplification may introduce inaccuracies due to the non-Gaussian nature of the true predictive distributions arising from GP models. Under this assumption, the system state x_k , control input u_k , and the GP model $d(x_k, u_k)$ are considered to have a multivariate Gaussian distribution at each time step [16], which facilitates the mathematical treatment of system dynamics

and control actions within a probabilistic framework as [19]:

$$\begin{bmatrix} x_k \\ u_k \\ d(x_k, u_k) \\ +\omega_k \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} \mu_k^x \\ \mu_k^u \\ \mu_k^d \\ \mu_k^w \end{bmatrix}, \begin{bmatrix} \Sigma_k^x & \Sigma_k^{xu} & \Sigma_k^{xd} \\ \Sigma_k^{xu^\top} & \Sigma_k^u & \Sigma_k^{ud} \\ \Sigma_k^{xd^\top} & \Sigma_k^{ud^\top} & \Sigma_k^d + \Sigma^w \end{bmatrix} \right). \quad (13)$$

Here, μ_k^x , μ_k^u , μ_k^d denote the mean values of the state x_k , control input u_k , and GP model output $d(x_k, u_k)$, respectively. The variance matrices Σ_k^x , Σ_k^u , and Σ_k^d , along with the covariance matrices Σ_k^{xu} , Σ_k^{xd} , and Σ_k^{ud} , capture the uncertainties and correlations between these variables. The term Σ^w represents the variance matrix of the noise ω_k in (11b), reflecting the uncertainty introduced by the noise in the system dynamics.

By defining the state and control input into a combined vector $z_k = [x_k^\top, u_k^\top]^\top$ and denoting the GP model output $d_k = d(x_k, u_k)$, (13) can be reformulated into a more compact representation as:

$$\begin{bmatrix} z_k \\ d_k + \omega_k \end{bmatrix} \sim \mathcal{N}(\mu_k, \Sigma_k) = \mathcal{N} \left(\begin{bmatrix} \mu_k^z \\ \mu_k^d \\ \mu_k^w \end{bmatrix}, \begin{bmatrix} \Sigma_k^z & \Sigma_k^{zd} \\ \star & \Sigma_k^d + \Sigma^w \end{bmatrix} \right), \quad (14)$$

where \star signifies symmetric elements within the covariance matrix. The notations for mean and covariance matrices are specified as follows:

$$\mu_k^z = \begin{bmatrix} \mu_k^x \\ \mu_k^u \end{bmatrix}, \quad \Sigma_k^z = \begin{bmatrix} \Sigma_k^x & \Sigma_k^{xu} \\ \Sigma_k^{xu^\top} & \Sigma_k^u \end{bmatrix}, \quad \Sigma_k^{zd} = \begin{bmatrix} \Sigma_k^{xd} \\ \Sigma_k^{ud} \end{bmatrix}. \quad (15)$$

3.1 Mean Propagation Approximation

In this section and the subsequent Sec. 3.2, we detail the process of propagating the mean and variance of the system state from x_k to x_{k+1} . We denote the mean $m(x_{k+1})$ and variance $var(x_{k+1})$ of the state x_{k+1} as μ_{k+1}^x and Σ_{k+1}^x , respectively. The propagation is based on the theoretical foundation provided in Appendix 9.1, following the steps:

$$\mu_{k+1}^x = m(x_{k+1}), \quad (16a)$$

$$= \mathbb{E}_{z_k} [\mathbb{E}_d(x_{k+1})], \quad (16b)$$

$$= \mathbb{E}_{z_k} [\mathbb{E}_d(f(x_k, u_k) + \mathbb{E}_d(B_d(d(x_k, u_k) + \omega_k)))] , \quad (16c)$$

$$= \mathbb{E}_{z_k} [\mathbb{E}_d(f(x_k, u_k)) + B_d \mu^d(x_k, u_k)]. \quad (16d)$$

Applying the mean propagation formula (A.3) in Appendix 9.1 to (16a), the mean of state x_{k+1} equals the expected value of the GP model \mathbb{E}_d at state x_{k+1} , results (16b). Incorporating the discrete-time state-space model into (16c), and then using (12) to obtain the expected output of the GP model $d(x_k, u_k)$ as $\mu^d(x_k, u_k)$, we arrive at (16d). Also, in (16d), the term $\mathbb{E}_d(f(x_k, u_k))$ represents the expected value of the function f under the GP model d at the point (x_k, u_k) , which equals to $f(x_k, u_k)$. Thus, we can express:

$$\mu_{k+1}^x = \mathbb{E}_{z_k} (f(x_k, u_k) + B_d \mu^d(x_k, u_k)), \quad (16e)$$

$$= \mathbb{E}_{z_k} (f(x_k, u_k)) + \mathbb{E}_{z_k} (B_d \mu^d(x_k, u_k)). \quad (16f)$$

To ensure a balance between computational efficiency and accuracy, similar to the approach in extended Kalman filtering, we linearize the nominal model $f(x_k, u_k)$ around the current mean

values (μ_k^x and μ_k^u) using a first-order Taylor expansion:

$$f(x_k, u_k) \approx f(\mu_k^x, \mu_k^u) + \frac{\partial f(x, u)}{\partial x} \Big|_{\mu_k^x, \mu_k^u} (x_k - \mu_k^x) + \frac{\partial f(x, u)}{\partial u} \Big|_{\mu_k^x, \mu_k^u} (u_k - \mu_k^u), \quad (17a)$$

$$= f(\mu_k^x, \mu_k^u) + \nabla f(\mu_k^z) (z_k - \mu_k^z). \quad (17b)$$

Here, $z_k = [x_k^\top, u_k^\top]^\top$ represents the combined state-input vector, and $\mu_k^z = [\mu_k^{x^\top}, \mu_k^{u^\top}]^\top$ denotes its mean. Additionally, we use $\nabla f(\mu_k^z)$ to represent the gradient of f evaluated at μ_k^z to obtain a more compact equation form as (17b). Applying (17b), the first term in (16f) is derived as:

$$\mathbb{E}_{z_k} (f(x_k, u_k)) \approx \mathbb{E}_{z_k} (f(\mu_k^x, \mu_k^u) + \nabla f(\mu_k^z) (z_k - \mu_k^z)), \quad (18a)$$

$$\approx f(\mu_k^x, \mu_k^u). \quad (18b)$$

Further by following (A.6) in Appendix 9.1, the GP mean $\mu^d(x_k, u_k)$ is approximated by its first order Taylor expansion around $x_k = \mu_k^x$ and $u_k = \mu_k^u$, thus the second term in (16f) is derived as:

$$\mathbb{E}_{z_k} (B_d \mu^d(x_k, u_k)) \approx \mathbb{E}_{z_k} (\mu^d(\mu_k^x, \mu_k^u) + \nabla \mu^d(\mu_k^z) (z_k - \mu_k^z)), \quad (19a)$$

$$\approx B_d \mu^d(\mu_k^x, \mu_k^u). \quad (19b)$$

Subscribing (18b) and (19b) to (16f), the final mean propagation approximation is:

$$\mu_{k+1}^x \approx f(\mu_k^x, \mu_k^u) + B_d \mu^d(\mu_k^x, \mu_k^u). \quad (20)$$

3.2 Uncertainty Propagation Approximation

3.2.1 Taylor Approximation. Uncertainty propagation within MPC loops is pivotal for ensuring robust decision-making under uncertainties [22]. The first widely used approach for approximation of uncertainty propagation is the Taylor approximation method, akin to the GP mean approximation discussed previously in Sec. 3.1. By linearly approximating the impact of uncertainty on system dynamics, we can efficiently predict the variance of future states, which is detailed from equations

(21a) to (21h).

$$\Sigma_{k+1}^x = \text{var}(x_{k+1}), \quad (21a)$$

$$\text{use (A.2)} = \mathbb{E}_{z_k} [\Sigma^d(x_{k+1})] + \text{var}_{z_k} (\mu^d(x_{k+1})), \quad (21b)$$

$$\text{use (12) and (20)} = \mathbb{E}_{z_k} \left(B_d \left(\Sigma^d(z_k) + \Sigma^\omega \right) B_d^\top \right) + \text{var}_{z_k} \left(f(z_k) + B_d \mu^d(z_k) \right), \quad (21c)$$

$$\text{use (A.8) and (A.9)} = B_d \left(\Sigma^d(\mu_k^z) + \Sigma^\omega \right) B_d^\top + \left(\nabla f(\mu_k^z) + B_d \nabla \mu^d(\mu_k^z) \right) \Sigma_k^z \left(\nabla f(\mu_k^z) + B_d \nabla \mu^d(\mu_k^z) \right)^\top, \quad (21d)$$

$$\begin{aligned} &= B_d \left(\Sigma^d(\mu_k^z) + \Sigma^\omega \right) B_d^\top + \nabla f(\mu_k^z) \Sigma_k^z \left(\nabla \mu^d(\mu_k^z) \right)^\top B_d^\top + \nabla f(\mu_k^z) \Sigma_k^z \nabla f^\top(\mu_k^z) \\ &\quad + B_d \left(\nabla \mu^d(\mu_k^z) \right) \Sigma_k^z \left(\nabla \mu^d(\mu_k^z) \right)^\top B_d^\top + B_d \left(\nabla \mu^d(\mu_k^z) \right) \Sigma_k^z \nabla f^\top(\mu_k^z), \end{aligned} \quad (21e)$$

$$\begin{aligned} &= \begin{bmatrix} \nabla f(\mu_k^z) & B_d \end{bmatrix} \begin{bmatrix} \Sigma_k^z & \Sigma_k^z \left(\nabla \mu^d(\mu_k^z) \right)^\top \\ \left(\nabla \mu^d(\mu_k^z) \right) \Sigma_k^z & \Sigma^d(\mu_k^z) + \nabla \mu^d(\mu_k^z) \Sigma_k^z \left(\nabla \mu^d(\mu_k^z) \right)^\top + \Sigma^\omega \end{bmatrix} \\ &\quad \begin{bmatrix} \nabla f(\mu_k^z) & B_d \end{bmatrix}^\top, \end{aligned} \quad (21f)$$

$$= \begin{bmatrix} \nabla f(\mu_k^z) & B_d \end{bmatrix} \begin{bmatrix} \Sigma_k^z & \Sigma_k^{zd} \\ \Sigma_k^{zd\top} & \Sigma_k^d + \Sigma^\omega \end{bmatrix} \begin{bmatrix} \nabla f(\mu_k^z) & B_d \end{bmatrix}^\top, \quad (21g)$$

$$= \begin{bmatrix} \nabla f(\mu_k^z) & B_d \end{bmatrix} \Sigma_k \begin{bmatrix} \nabla f(\mu_k^z) & B_d \end{bmatrix}^\top. \quad (21h)$$

It is worth remembering that the posterior predictions of the GP model in the MPC loop are not Gaussian due to multiple prediction steps needed. In order to integrate GP models into the MPC prediction model, it needs to assume the distribution of the states at every time step in the MPC loop is Gaussian distributed as $x_k \sim \mathcal{N}(\mu_k^d, \Sigma_k^d)$, thus we can obtain posterior predictions using the GP model. Intuitively, we need to consider the input uncertainty for the GP model at each time step to propagate the uncertainty in the MPC loop cumulatively.

Upon examining the definitive equation for uncertainty propagation (21g), it can be seen that for accurate uncertainty propagation when coupling a GP model $d(\cdot)$ with a nominal system model $f(\cdot)$, both the inherent uncertainty of the GP model Σ_k^d , and the covariance between the GP model's output and the combined state-control inputs Σ_k^{zd} , play pivotal roles. Comparing (21f) and (21g), we obtain the general equations for Σ_k^d and Σ_k^{zd} , articulated as:

$$\begin{bmatrix} \Sigma_k^{zd} \\ \Sigma_k^d \end{bmatrix} = \begin{bmatrix} \Sigma_k^z \left(\nabla \mu^d(\mu_k^z) \right)^\top \\ \Sigma^d(\mu_k^z) + \nabla \mu^d(\mu_k^z) \Sigma_k^z \left(\nabla \mu^d(\mu_k^z) \right)^\top \end{bmatrix}. \quad (22)$$

This equation encompasses the covariance, Σ_k^{zd} , which includes the covariance between the state variables and the GP model output Σ_k^{xd} as well as the covariance between the control inputs and the GP model outputs Σ_k^{ud} , as defined in (15). Moreover, the GP model's uncertainty variance, Σ_k^d , incorporates adjustments to both the model's posterior covariance and variance. These adjustments are informed by the gradient of the model's posterior mean relative to the input, $\nabla \mu^d(\mu_k^z)$, and the input variance, Σ_k^z . This approach ensures the precision of GP model prediction by thoroughly accounting for the dynamic interplay between input variability and output uncertainty.

3.2.2 Mean Equivalent Approximation. The mean equivalent approximation method is another commonly used strategy for approximating uncertainty propagation [23]. This technique simplifies

the process by using only the mean values of input variables without propagating uncertainties. Specifically, it assumes no uncertainty accumulation within the GP model Σ_k^d across the MPC prediction horizons and disregards the covariance between the GP model $d(z_k)$ and the state-control inputs z_k , formalized as follows:

$$\begin{bmatrix} \Sigma_k^{zd} \\ \Sigma_k^d \end{bmatrix} = \begin{bmatrix} 0 \\ \Sigma^d(\mu_k^z) \end{bmatrix}. \quad (23)$$

Without accumulating uncertainty propagation in the MPC prediction horizons, this method can lead to poor approximations of GP models. Its omission of covariance dynamics ($\Sigma_k^{zd} = 0$) can in addition deteriorate the quality of predictions when integrated with a nominal system $f(z)$ [19]. By applying (21c), the system uncertainty propagation approximation using the mean equivalent approximation method is:

$$\text{var}(x_{k+1}) = [\nabla f(\mu_k^z) \quad B_d] \begin{bmatrix} \Sigma_k^z & 0 \\ 0 & \Sigma^d(\mu_k^z) + \Sigma_w \end{bmatrix} [\nabla f(\mu_k^z) \quad B_d]^\top, \quad (24a)$$

$$= [\nabla f(\mu_k^z) \quad B_d] \Sigma_k [\nabla f(\mu_k^z) \quad B_d]^\top. \quad (24b)$$

3.3 Mean and Uncertainty Approximation

This section summarizes conclusive equations on approximating the mean (μ) and uncertainty (Σ) propagation in dynamic systems (11a):

$$\mu_{k+1}^x \approx f(\mu_k^x, \mu_k^u) + B_d \mu^d(\mu_k^x, \mu_k^u), \quad (\text{Refer back to (20)}) \quad (25a)$$

$$\Sigma_{k+1}^x \approx [\nabla f(\mu_k^x, \mu_k^u) \quad B_d] \Sigma_k [\nabla f(\mu_k^x, \mu_k^u) \quad B_d]^\top, \quad (25b)$$

where the system variance Σ_k is defined in (21g) and (24a) by using either the Taylor approximation or the mean equivalent approximation method, respectively. The difference of Σ_k using two different approximation methods offers a trade-off between computational efficiency and the accuracy of uncertainty quantification in GP-based predictive models.

Specifically, the Taylor approximation method allows for a nuanced incorporation of uncertainty through the linearization of system dynamics, whereas the mean equivalent approximation method simplifies the computational process by assuming a deterministic pathway based solely on mean values, thereby eschewing direct uncertainty propagation. This distinction underscores the importance of method selection based on the specific requirements for model accuracy and computational resources, highlighting the inherent compromises between simplicity and fidelity in the modeling of dynamic systems.

Beyond these two approximation methods discussed in Sec. 3.2, exact moment matching method [22] introduces a technique for computing the GP prediction's mean and variance directly by integrating over the input distribution without approximations. This approach accounts for the behavior of the mean function and input variability, aiming for a precise representation of prediction uncertainty. However, its practicality is constrained by the presumption that state distributions at each step within the MPC prediction horizon adhere to a Gaussian distribution. This assumption introduces inaccuracies, given the GP's posterior is non-Gaussian over multiple prediction steps. This limitation is particularly highlighted in [19], demonstrating that both Taylor approximation and exact moment matching tend to provide similar outcomes for the typical range of variation in GP's posterior mean and variance within the MPC prediction horizon. Despite the theoretical precision offered by the exact moment matching method, its substantial computational demands significantly restrict its widespread application in the propagation of mean and uncertainty, particularly when combining with MPC. The complexity of exact moment matching exacerbates the computational

burden of MPC, a framework already known for its intensive computational demands, especially in real-world robotic applications where rapid decision-making is critical.

The introduction of these three methods illuminates a spectrum of options available to mean and uncertainty propagation, from the computationally light but potentially less accurate mean equivalent approximation, through the balanced approach of Taylor approximation, to the theoretically precise but computationally intensive exact moment matching. The choice among these methods hinges on the specific trade-offs between computational efficiency, model accuracy, and the practicality of incorporating complex uncertainty quantification within GP-based predictive models.

3.4 Mean and Uncertainty Approximation with Linear Nominal Model

This section details uncertainty approximation for systems with a linear nominal model in (11a). When the nominal model is linear, it is defined as $f(x_k, u_k) = Ax_k + Bu_k$, with A and B representing the state transition and control input matrices, respectively. Leveraging equations (13) and (25b), we derive the uncertainty propagation as follows:

$$\Sigma_{k+1}^x = [A \ B \ B^d] \Sigma_k [A \ B \ B^d]^T, \quad (26a)$$

$$= [A \ B \ B^d] \begin{bmatrix} \Sigma_k^x & \Sigma_k^{xu} & \Sigma_k^{xd} \\ \Sigma_k^{xu^T} & \Sigma_k^u & \Sigma_k^{ud} \\ \Sigma_k^{xd^T} & \Sigma_k^{ud^T} & \Sigma_k^d + \Sigma^w \end{bmatrix} [A \ B \ B^d]^T. \quad (26b)$$

When employing the mean equivalent approximation method, the variance calculation results as:

$$\Sigma_{k+1}^x = [A \ B \ B^d] \begin{bmatrix} \Sigma_k^x & \Sigma_k^{xu} & 0 \\ \Sigma_k^{xu^T} & \Sigma_k^u & 0 \\ 0 & 0 & \Sigma^d(\mu_k^z) + \Sigma^w \end{bmatrix} [A \ B \ B^d]^T, \quad (27)$$

Alternatively, using the Taylor approximation method, the variance is elaborated in (28) as:

$$\Sigma_{k+1}^x = [A \ B \ B^d] \begin{bmatrix} \Sigma_k^x & \Sigma_k^{xu} & \Sigma_k^x (\nabla \mu^d(\mu_k^x))^T \\ \Sigma_k^{xu^T} & \Sigma_k^u & \Sigma_k^u (\nabla \mu^d(\mu_k^u))^T \\ (\nabla \mu^d(\mu_k^x)) \Sigma_k^x & (\nabla \mu^d(\mu_k^u)) \Sigma_k^u & \Sigma^d(\mu_k^z) + \nabla \mu^d(\mu_k^z) \Sigma_k^z (\nabla \mu^d(\mu_k^z))^T + \Sigma^w \end{bmatrix} [A \ B \ B^d]^T. \quad (28)$$

4 COMBINING GAUSSIAN PROCESSES WITH MODEL PREDICTIVE CONTROL

In Sec. 3, we demonstrated techniques for the efficient approximation of the system's mean and uncertainty propagation. Building on this, this section introduces a straightforward yet fundamental application of GP within MPC, focusing on incorporating the GP means into the MPC prediction model and omitting uncertainty propagation. Through this focused simplification, we aim to underscore the GP model's capability to enhance MPC's predictive accuracy and overall control strategy effectiveness.

Integrating a GP component into the MPC framework transforms the MPC into a stochastic MPC (SMPC) problem [2]. This adaptation necessitates a structured approach to managing chance constraints, which involves setting maximum allowable probabilities for constraint violations. The

formulation of this generalized stochastic optimal control problem is as follows:

$$\min_U \mathbb{E} \left(\sum_{k=0}^{N-1} J(x_k, u_k) \right), \quad (29a)$$

$$\text{s.t. } x_{k+1} = f(x_k, u_k) + B_d(g(x_k, u_k) + w_k), \quad (29b)$$

$$\Pr(x_{k+1} \in X_{k+1}) \geq p_x, \quad (29c)$$

$$\Pr(u_k \in \mathcal{U}_k) \geq p_u, \quad (29d)$$

$$x_0 = x(k). \quad (29e)$$

Here, $U = [u_0(\cdot), \dots, u_{N-1}(\cdot)]$ denotes the control inputs across time intervals $i = [0, \dots, N - 1]$. The optimization aims to minimize the expected sum of the cost function $J(x_k, u_k)$ over the horizon N , subject to the system dynamics as the equality constraint (29b). The chance constraints (29c) and (29d) introduce inequality constraints, ensuring that system states and control inputs remain within predefined bounds X_{k+1} and \mathcal{U}_k with associated satisfaction probabilities p_x and p_u , respectively, alongside the initial state condition (29e).

Given the stochastic nature of the prediction model, this optimization problem is computationally intractable [16]. To address this, approximation techniques for mean and uncertainty, as detailed in Sec. 3, are employed alongside strategic handling of chance constraints. These methodologies collectively enable the formulation of a more tractable, deterministic approximation of the original stochastic optimization challenge, which will be further elaborated in Section 6.1.

Focusing on showing the GP model's predictive capabilities within an MPC framework, we demonstrate a foundational GP-MPC model that only integrates the GP mean into the prediction model and excludes the uncertainty propagation [3, 24]. Such a simplification redefines (29a) as a deterministic optimization problem. We further focus on the optimization core by temporarily disregarding constraints (29e), and utilize a quadratic cost function commonly adopted in MPC formulations [24, 25]:

$$J(\mathbf{x}_k, \mathbf{u}_k) = (\mathbf{x}_{d,k+1} - \mathbf{x}_{k+1})^\top Q (\mathbf{x}_{d,k+1} - \mathbf{x}_{k+1}) + \mathbf{u}_k^\top R \mathbf{u}_k, \quad (30)$$

where \mathbf{u}_k represents a sequence of control inputs $\mathbf{u}_k = (u_k, \dots, u_{k+N-1})$ and N represents the MPC prediction horizon, $\mathbf{x}_{d,k+1}$ denotes the desired state trajectory $\mathbf{x}_{d,k+1} = (x_{d,k+1}, \dots, x_{d,k+N})$, and \mathbf{x}_{k+1} is the predicted state trajectory $\mathbf{x}_{k+1} = (x_{k+1}, \dots, x_{k+N})$, derived from the GP mean approximation (25a) by applying \mathbf{u}_k . The observable system state $x_k \in \mathbb{R}^n$ and control input $u_k \in \mathbb{R}^m$ at time k , with $Q \in \mathbb{R}^{Pn \times Pn}$ and $R \in \mathbb{R}^{Pm \times Pm}$ being positive semidefinite matrices, ensure the cost function's efficacy in guiding the system towards desired behaviors.

Adapting the prediction model within MPC to incorporate solely the GP mean, and excluding B_d for simplicity without loss of generality, the prediction model can be formulated as:

$$x_{k+1} \approx f(x_k, u_k) + \mu^d(x_k, u_k). \quad (31)$$

In general with a nonlinear nominal model $f(\cdot)$, we can linearize (31) around a point (\bar{x}_k, \bar{u}_k) as:

$$\begin{aligned} x_{k+1} &\approx f(\bar{x}_k, \bar{u}_k) + \frac{\partial f(x, u)}{\partial x} \Big|_{\bar{x}_k, \bar{u}_k} (x_k - \bar{x}_k) + \frac{\partial f(x, u)}{\partial u} \Big|_{\bar{x}_k, \bar{u}_k} (u_k - \bar{u}_k) \\ &+ \mu^d(\bar{x}_k, \bar{u}_k) + \frac{\partial \mu^d(x, u)}{\partial x} \Big|_{\bar{x}_k, \bar{u}_k} (x_k - \bar{x}_k) + \frac{\partial \mu^d(x, u)}{\partial u} \Big|_{\bar{x}_k, \bar{u}_k} (u_k - \bar{u}_k). \end{aligned} \quad (32)$$

By defining $\delta x_k = x_k - \bar{x}_k$ and $\delta u_k = u_k - \bar{u}_k$, we obtain:

$$\begin{aligned} \delta x_{k+1} + \bar{x}_{k+1} &\approx f(\bar{x}_k, \bar{u}_k) + \frac{\partial f(x, u)}{\partial x} \Big|_{\bar{x}_k, \bar{u}_k} \delta x_k + \frac{\partial f(x, u)}{\partial u} \Big|_{\bar{x}_k, \bar{u}_k} \delta u_k \\ &+ \mu^d(\bar{x}_k, \bar{u}_k) + \frac{\partial \mu^d(x, u)}{\partial x} \Big|_{\bar{x}_k, \bar{u}_k} \delta x_k + \frac{\partial \mu^d(x, u)}{\partial u} \Big|_{\bar{x}_k, \bar{u}_k} \delta u_k. \end{aligned} \quad (33)$$

If the linearization is conducted at the mean points for each time step, i.e. $\bar{x}_{k+1} = \mu_{k+1}^x$, $\bar{x}_k = \mu_k^x$ and $\bar{u}_k = \mu_k^u$, and according to (25a), we have $\bar{x}_{k+1} \approx f(\bar{x}_k, \bar{u}_k) + \mu^d(\bar{x}_k, \bar{u}_k)$. The linearized prediction model (32) becomes:

$$\delta x_{k+1} \approx \frac{\partial f(x, u)}{\partial x} \Big|_{\bar{x}_k, \bar{u}_k} \delta x_k + \frac{\partial f(x, u)}{\partial u} \Big|_{\bar{x}_k, \bar{u}_k} \delta u_k + \frac{\partial \mu^d(x, u)}{\partial x} \Big|_{\bar{x}_k, \bar{u}_k} \delta x_k + \frac{\partial \mu^d(x, u)}{\partial u} \Big|_{\bar{x}_k, \bar{u}_k} \delta u_k. \quad (34)$$

For time index $b \in \{0, \dots, N-1\}$, the (34) over the next N time steps yields a compact matrix form for delta changes:

$$\delta x_{k+b+1} \approx H_{x,k+b} \delta x_{k+b} + H_{u,k+b} \delta u_{k+b}, \quad (35)$$

where $H_{x,k+b}$ and $H_{u,k+b}$ are matrices capturing the combined effects of the linearized nominal model and the GP mean derivative as:

$$H_{x,k+b} = \frac{\partial f(x, u)}{\partial x} \Big|_{\bar{x}_{k+b}, \bar{u}_{k+b}} + \frac{\partial \mu^d(x, u)}{\partial x} \Big|_{\bar{x}_{k+b}, \bar{u}_{k+b}}, \quad (36a)$$

$$H_{u,k+b} = \frac{\partial f(x, u)}{\partial u} \Big|_{\bar{x}_{k+b}, \bar{u}_{k+b}} + \frac{\partial \mu^d(x, u)}{\partial u} \Big|_{\bar{x}_{k+b}, \bar{u}_{k+b}}. \quad (36b)$$

Defining the deviations in state and control input vectors as $\delta \mathbf{x}_{k+1} = (\delta x_{k+1}, \dots, \delta x_{k+N})$, $\delta \mathbf{x}_k = (\delta x_k, \dots, \delta x_{k+N-1})$ and $\delta \mathbf{u}_k = (\delta u_k, \dots, \delta u_{k+N-1})$ yields a compact matrix formulation for the system dynamics as follows:

$$\begin{aligned} \delta \mathbf{x}_{k+1} &= \mathbf{H}_x \delta \mathbf{x}_k + \mathbf{H}_u \delta \mathbf{u}_k, \\ &= (\mathbf{I} - \mathbf{H}_x)^{-1} \mathbf{H}_u \delta \mathbf{u}_k, \\ &= \mathbf{H}' \delta \mathbf{u}_k, \end{aligned} \quad (37)$$

where \mathbf{I} denotes the identity matrix, and $\mathbf{H}_u = \text{diag}(H_{u,k}, \dots, H_{u,k+N-1})$ encapsulates the control dynamics across the prediction horizon. The matrices \mathbf{H}_x and \mathbf{H}_u reflect the response to state and control perturbations of the system and GP model, respectively with

$$\Lambda_{x,k+1} = \begin{bmatrix} 0 & 0 \\ H_{x,k+1} & 0 \end{bmatrix}, \quad \mathbf{H}_x = \text{diag}(\Lambda_{x,k+1}, \dots, \Lambda_{x,k+N-1}). \quad (38)$$

Substituting $\delta \mathbf{x}_{k+1} = \mathbf{x}_{k+1} - \bar{\mathbf{x}}_{k+1}$, $\delta \mathbf{u}_k = \mathbf{u}_k - \bar{\mathbf{u}}_k$ and (37) to (30), results in an approximated cost function:

$$\mathbf{J}(\mathbf{x}_k, \mathbf{u}_k, \delta \mathbf{u}_k) \approx (\mathbf{x}_{d,k+1} - \bar{\mathbf{x}}_{k+1} - \mathbf{H}' \delta \mathbf{u}_k)^\top \mathbf{Q} (\mathbf{x}_{d,k+1} - \bar{\mathbf{x}}_{k+1} - \mathbf{H}' \delta \mathbf{u}_k) + (\bar{\mathbf{u}}_k + \delta \mathbf{u}_k)^\top \mathbf{R} (\bar{\mathbf{u}}_k + \delta \mathbf{u}_k). \quad (39)$$

To find the optimal $\delta \mathbf{u}_k$ that minimizes the cost function, we set its derivative with respect to $\delta \mathbf{u}_k$ to zero. The solution, detailed in the Appendix 9.2, is given by:

$$\delta \mathbf{u}_k = \left(\mathbf{H}'^\top \mathbf{Q} \mathbf{H}' + \mathbf{R} \right)^{-1} \left(\mathbf{H}'^\top \mathbf{Q} \tilde{\mathbf{x}}_{k+1} - \mathbf{R} \bar{\mathbf{u}}_k \right). \quad (40)$$

where $\tilde{\mathbf{x}}_{k+1}$ represents the desired state trajectory deviation from the predicted mean as $\tilde{\mathbf{x}}_{k+1} := \mathbf{x}_{d,k+1} - \bar{\mathbf{x}}_{k+1}$. This iterative update process for $\delta \mathbf{u}_k$, with $\bar{\mathbf{u}}_k \leftarrow \bar{\mathbf{u}}_k + \delta \mathbf{u}_k$ until it converges, ensures dynamic optimization of control actions, with only the first control input implemented at each

step. This cycle repeats, recalculating optimal controls at each new timestep, thereby adapting to evolving system dynamics and optimizing control strategies over time.

5 APPLICATION I: IMPROVED PATH FOLLOWING FOR MOBILE ROBOTS

This section, alongside the next, investigates the application of GP-MPC strategies in the realm of mobile robotics, focusing on path-following controls for wheeled robots. It introduces two innovative approaches: GP-based nonlinear MPC (GP-NMPC) [24] and GP-based feedback linearization MPC (GP-FBLMPC) [3]. Building on the integration of GP means into MPC predictions discussed in Sec. 4, we illustrate the practical application of GP-MPC in elevating system performance in real-world scenarios.

5.1 GP-based Nonlinear MPC

The most direct utilization of GP models is identifying and learning the discrepancies between theoretical nominal models and actual observed behavior in robotic systems. By precisely modeling these discrepancies, GP models significantly enhance the prediction accuracy of the system's dynamics. This improvement in predictive accuracy directly contributes to superior control performance, enabling the system to adjust its actions based on a more accurate representation of its interactions with the environment.

In mobile robotics, especially for path-following tasks, employing GP models has proven to be highly effective. Path following, a fundamental functionality for autonomous vehicles and robots, requires the robot to precisely adhere to a predetermined path without temporal constraints. This challenge is particularly challenging for wheeled robots navigating complex off-road terrains, where accurately modeling the dynamics of robot-terrain interaction becomes problematic. The work presented in [24] is a pioneering application of GP-based MPC to this challenge, demonstrating a significant improvement in path-following accuracy by adapting the MPC model to reflect actual field observations through GP learning, thereby underscoring the efficacy of GP models in complex environmental interactions.

A general robotic system model incorporating dynamic elements into kinematic equations can be expressed as:

$$\mathbf{x}_{k+1} = \underbrace{\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)}_{\text{nominal model}} + \underbrace{\mathbf{g}(\mathbf{x}_k, \mathbf{v}_{k-1}, \mathbf{u}_k, \mathbf{u}_{k-1})}_{a_k} . \quad (41)$$

In this model, $\mathbf{x}_k = (x_k, y_k, \theta_k)$ denotes the robot's pose, $\mathbf{v}_k = (v_k^{\text{act}}, \omega_k^{\text{act}})$ the actual velocity, and $\mathbf{u}_k = (v_k^{\text{cmd}}, \omega_k^{\text{cmd}})$ the control inputs (linear and angular velocity) at time k . The disturbance query state is defined as:

$$a_k = (\mathbf{x}_k, \mathbf{v}_{k-1}, \mathbf{u}_k, \mathbf{u}_{k-1}) . \quad (42)$$

Utilizing this model, at time $k - 1$, discrepancies between the nominal model and actual system behavior are estimated by a GP disturbance model as:

$$\hat{\mathbf{g}}(a_{k-1}) = \hat{\mathbf{x}}_k - \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}) . \quad (43)$$

Here, $\hat{\mathbf{x}}_k$ and $\hat{\mathbf{x}}_{k-1}$ represent the robot poses from the on-board navigation system at time k and $k - 1$ respectively, \mathbf{u}_{k-1} is the control input at time $k - 1$. In the disturbance query state $a_{k-1} = (\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{v}}_{k-2}, \mathbf{u}_{k-1}, \mathbf{u}_{k-2})$, the $\hat{\mathbf{v}}_{k-2}$ is calculated by the estimated robot poses $\hat{\mathbf{x}}_{k-1}$ and $\hat{\mathbf{x}}_{k-2}$, which indicates the disturbance query state a_k requires historic states data. In one experiment for

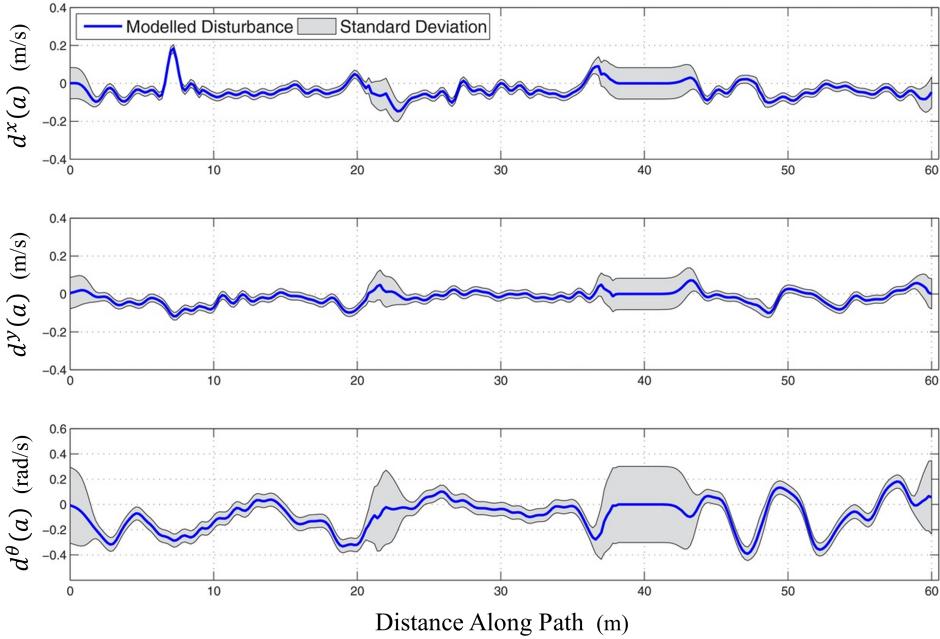


Fig. 6. Illustration of the trained GP models for the robot pose variables x , y , and θ ($d^x(a)$, $d^y(a)$, and $d^\theta(a)$ respectively), based on a 60 m field experiment dataset (44). These models, developed under the assumption of uncorrelated disturbances, demonstrate the application of (12) in creating separate GP models for each pose variable [24].

data collection, the observed disturbance datasets are collected and prepared by following (43) as:

$$\begin{aligned}\mathcal{D}_m^x &= \{\mathbf{a} = [a_1, \dots, a_{k-1}, \dots, a_m]^\top, \mathbf{g}^x(\mathbf{a}) = [g^x(a_1), \dots, g^x(a_k), \dots, g^x(a_m)]^\top\}, \\ \mathcal{D}_m^y &= \{\mathbf{a} = [a_1, \dots, a_{k-1}, \dots, a_m]^\top, \mathbf{g}^y(\mathbf{a}) = [g^y(a_1), \dots, g^y(a_k), \dots, g^y(a_m)]^\top\}, \\ \mathcal{D}_m^\theta &= \{\mathbf{a} = [a_1, \dots, a_{k-1}, \dots, a_m]^\top, \mathbf{g}^\theta(\mathbf{a}) = [g^\theta(a_1), \dots, g^\theta(a_k), \dots, g^\theta(a_m)]^\top\},\end{aligned}\quad (44)$$

Using these datasets and (12), three separate GP models, $d^x(a)$, $d^y(a)$, and $d^\theta(a)$, are trained for each variable of the robot pose, x , y , and θ respectively, by assuming the disturbances are uncorrelated. The trained GP models with 60 m field experiment data are shown in Fig. 6 [24].

Utilizing the GP-enhanced prediction model (41), a GP-NMPC algorithm [24] was developed following the methodology explained in Sec. 4. Through 3 km field tests on three different robots, the GP-NMPC demonstrated its superior path-following capabilities in challenging off-road environments. These tests highlighted its ability to learn and reduce path-following errors by effectively combining GP with MPC. However, it encounters computational efficiency challenges due to the iterative optimization inherent in NMPC, and it cannot generalize to new, untrained paths. Addressing these limitations, a novel GP-FBLMPC strategy that combines feedback linearization (FBL) with GP and MPC was proposed to improve both computational efficiency and the system's adaptability to diverse untrained new paths, thereby providing a more robust and adaptable solution for mobile robot path-following in challenging terrains.

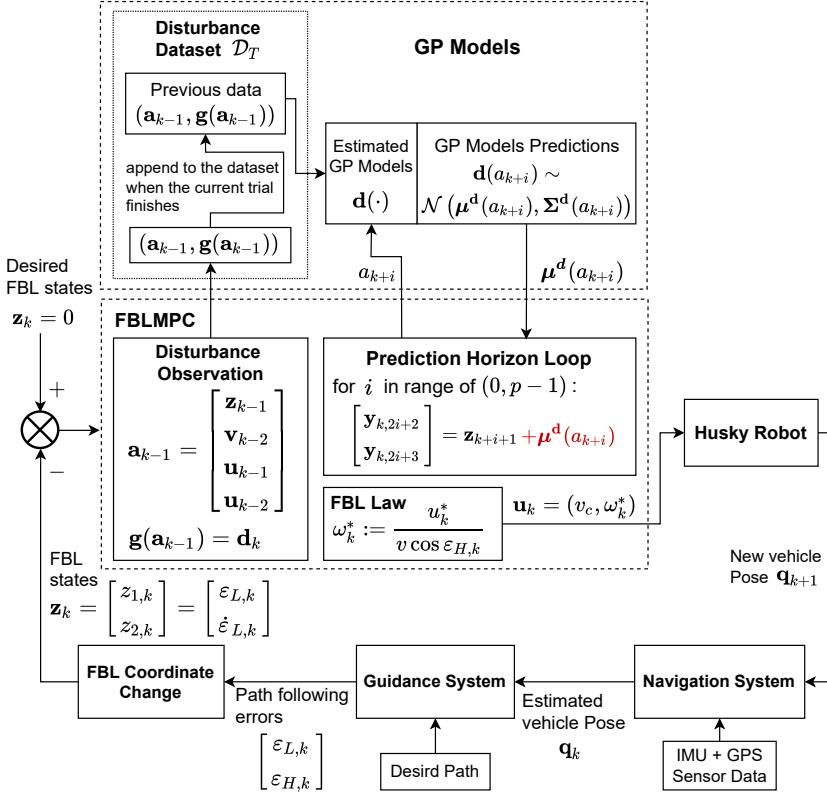


Fig. 7. Block diagram of path-following control using the GP-FBLMPC algorithm. GP models capture unmodeled system dynamics from previous experiences. Estimated GP model means ($\boldsymbol{\mu}^d$) are integrated into the system model within the MPC prediction horizon loop [3].

5.2 GP-based Feedback Linearization MPC

This section introduces the innovative GP-FBLMPC methodology [3], an advancement over the traditional GP-based nonlinear MPC (GP-NMPC). GP-FBLMPC ingeniously combines GP, MPC, and FBL techniques. This approach effectively addresses the computational efficiency and model generalization challenges encountered in GP-NMPC, positioning GP-FBLMPC as a superior strategy for path-following in mobile robotics across diverse and challenging terrains.

The essence of GP-FBLMPC lies in its integration of GP predictions within a feedback-linearized MPC strategy, offering an efficient and adaptable solution for navigating complex environments. The schematic representation provided in Fig. 7 outlines the operational framework of GP-FBLMPC, illustrating how GP models incorporate learned dynamics from past experiences into the MPC's prediction loop for enhanced path-following control.

5.2.1 Feedback Linearization in MPC. Before delving into the integration of GP with FBLMPC, it is helpful to understand the basics of feedback linearization as a standalone control strategy. Unlike traditional linearization techniques such as Taylor expansion shown in (32), which approximate nonlinear systems around a specific operating point, feedback linearization transforms the entire nonlinear system dynamics into a globally linear form [26]. This transformation is achieved through

a change of variables and the application of a control input that exactly cancels the nonlinear characteristics of the system over its entire operating range, thereby yielding a linear system representation without local approximation errors.

For wheeled robots with a differential drive mechanism, the discretized kinematic model is:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + T \begin{bmatrix} \cos \theta_k & 0 \\ \sin \theta_k & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_k \\ \omega_k \end{bmatrix}. \quad (45)$$

where x , y , and θ represent the robot's pose states, and the control inputs are the commanded forward velocity v and yaw rate ω . Defining the path-following errors as lateral $\varepsilon_L := y$ and heading $\varepsilon_H := \theta$, the instantaneous rate of change of the path-following errors are:

$$\dot{\varepsilon}_L = \dot{y} = v \sin \varepsilon_H, \quad (46a)$$

$$\dot{\varepsilon}_H = \dot{\theta} = \omega. \quad (46b)$$

The feedback linearization process begins with the introduction of new coordinates $z_1 \equiv \varepsilon_L = y$ and $z_2 \equiv \dot{\varepsilon}_L = \dot{y}$, designated as feedback linearization (FBL) states. This leads to a transformed system dynamics represented as:

$$\dot{\mathbf{z}} = \begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u = \mathbf{A}\mathbf{z} + \mathbf{B}u, \quad (47)$$

where u symbolizes the new control input. In the context of constant linear velocity, $u = \dot{z}_2 = v\omega \cos \varepsilon_H$, revealing a nonlinear relationship between u and the robot's steering rate $\omega = u/(v \cos \varepsilon_H)$. This relationship holds true under the condition that $v \neq 0$ and $\varepsilon_H \in (-\pi/2, \pi/2)$.

The FBLMPC strategy computes a sequence of control inputs \mathbf{u}_k across a finite horizon N to minimize the predicted path-following errors. The control sequence and the incremental control updates are represented by $\mathbf{u}_k := \Delta\mathbf{u}_k + \mathbf{u}_{k-1} = (u_k, u_{k+1}, \dots, u_{k+N-1})$ and $\Delta\mathbf{u}_k = (\Delta u_k, \Delta u_{k+1}, \dots, \Delta u_{k+N-1})$, respectively. By discretizing the linearized system equation (47), we get:

$$\mathbf{z}_{k+1} = \mathbf{F}\mathbf{z}_k + \mathbf{G}u_k = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} z_{1,k} \\ z_{2,k} \end{bmatrix} + \begin{bmatrix} \frac{T^2}{2} \\ T \end{bmatrix} u_k, \quad (48)$$

where $T > 0$ is a small update interval, and $\mathbf{z}_k = \mathbf{z}(kT)$, $k = 0, 1, 2, \dots$. The FBL states over the prediction horizon are described by:

$$\mathbf{y}_{k+1} := \Delta\mathbf{y}_{k+1} + \mathbf{y}_k = (\mathbf{z}_{k+1}, \mathbf{z}_{k+2}, \dots, \mathbf{z}_{k+N}), \quad (49)$$

where $\Delta\mathbf{y}_k = (\Delta\mathbf{z}_{k+1}, \Delta\mathbf{z}_{k+2}, \dots, \Delta\mathbf{z}_{k+N})$ represents the updates of FBL states over the prediction horizon. The discrete updates in control input and FBL states are defined as $\Delta u_k := u_k - u_{k-1}$ and $\Delta\mathbf{z}_k := \mathbf{z}_k - \mathbf{z}_{k-1}$, respectively. Utilizing (48), the dynamics of FBL states across the prediction horizon can be expressed as:

$$\mathbf{z}_{k+1} = \mathbf{F}\mathbf{z}_k + \mathbf{G}u_k, \quad (50a)$$

$$\Delta\mathbf{z}_{k+1} + \mathbf{z}_k = \mathbf{F}(\Delta\mathbf{z}_k + \mathbf{z}_{k-1}) + \mathbf{G}(\Delta u_k + u_{k-1}), \quad (50b)$$

$$\Delta\mathbf{z}_{k+1} = \mathbf{F}\Delta\mathbf{z}_k + \mathbf{G}\Delta u_k, \quad (50c)$$

$$\Delta\mathbf{z}_{k+2} = \mathbf{F}\Delta\mathbf{z}_{k+1} + \mathbf{G}\Delta u_{k+1}, \quad (50d)$$

⋮

$$\Delta\mathbf{z}_{k+N} = \mathbf{F}^N \Delta\mathbf{z}_k + \mathbf{F}^{N-1} \mathbf{G} \Delta u_k + \dots + \mathbf{G} \Delta u_{k+N-1}, \quad (50e)$$

which are rewritten in matrix form as:

$$\begin{bmatrix} \Delta z_{k+1} \\ \Delta z_{k+2} \\ \vdots \\ \Delta z_{k+N} \end{bmatrix} = \underbrace{\begin{bmatrix} F \\ F^2 \\ \vdots \\ F^N \end{bmatrix}}_L \Delta z_k + \underbrace{\begin{bmatrix} G & 0 & \cdots & 0 & 0 \\ FG & G & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ F^{N-1}G & F^{N-2}G & \cdots & FG & G \end{bmatrix}}_M \underbrace{\begin{bmatrix} \Delta u_k \\ \Delta u_{k+1} \\ \Delta u_{k+2} \\ \vdots \\ \Delta u_{k+N-1} \end{bmatrix}}_{\Delta u_k}, \quad (51)$$

and more succinctly as:

$$\Delta y_{k+1} = L \Delta z_k + M \Delta u_k. \quad (52)$$

In the GP-FBLMPC approach, the optimization of path-following control is achieved by minimizing the same quadratic cost function used in the GP-NMPC method, allowing for a consistent comparison of path-following performance between the two strategies:

$$J(y_k, u_k) = y_{k+1}^\top Q y_{k+1} + u_k^\top R u_k, \quad (53)$$

where $Q \in \mathbb{R}^{2N \times 2N}$ and $R \in \mathbb{R}^{N \times N}$ are the weighting matrices for path-following errors and control efforts, respectively. The term y_{k+1} represents the predicted FBL state z over the prediction horizon, and u_k is the control input sequence over the prediction horizon specified ahead of (48).

Integrating the equation of u_k , (49), and (52) into the cost function (53), the optimization problem becomes:

$$J(y_k, u_k, \Delta z_k, \Delta u_k) = (L \Delta z_k + M \Delta u_k + y_k)^\top Q (L \Delta z_k + M \Delta u_k + y_k) + (\Delta u + u_{k-1})^\top R (\Delta u + u_{k-1}). \quad (54)$$

The optimization process seeks the sequence of control adjustments, Δu_k^* , that minimizes this cost function. Given the quadratic and convex nature of J , the optimal control sequence is derived by solving the equation $\partial J(\Delta u_k)/\partial \Delta u_k = 0$ as:

$$\Delta u_k^* = -(M^\top Q M + R)^{-1} (M^\top Q (y_k + L \Delta z_k) + R u_{k-1}). \quad (55)$$

The final control sequence u_k is then obtained by adding the optimal control adjustments to the previous control sequence: $u_k^* = \Delta u_k^* + u_{k-1}$. The first element of this sequence, u_k^* , is applied in real-time to adjust the vehicle's steering rate, calculated as:

$$\omega_k^* = \frac{u_k^*}{v \cos \epsilon_{H,k}}, \quad (56)$$

where v is the vehicle's linear velocity and $\epsilon_{H,k}$ is the heading error at time k . Note that the matrices $(M^\top Q M + R)^{-1}$ and $M^\top Q$ are constants at all time steps and thus can be precomputed to significantly reduce real-time computational complexity, which is a key difference between the FBLMPC strategy and traditional nonlinear MPC.

5.2.2 GP-FBLMPC. In GP-FBLMPC, the focus shifts to using feedback linearized (FBL) states z_k as the system variables, in contrast to the robot pose states x_k used in GP-NMPC. Utilizing (41), the relationship between the actual and predicted robot pose state x_{k+1} is expressed as:

$$x_{k+1} - f(x_k, u_k) = g(x_k, v_{k-1}, u_k, u_{k-1}). \quad (57)$$

where the right side of (57) are the FBL states z_{k+1} . The GP model incorporating these FBL states is formulated as:

$$z_{k+1} = g(a_k), \quad (58)$$



Fig. 8. A Husky A200 robot is driven autonomously to follow the infinite path (left) and the track path (right) in sandy terrain. The desired paths, represented by black dotted lines, were blended into the field test scene image captured by a DJI Mini drone [3].

with the disturbance state defined as $a_k = (\mathbf{z}_k, \mathbf{v}_{k-1}, \mathbf{u}_k, \mathbf{u}_{k-1})$. Like GP-NMPC, assuming uncorrelated disturbance states z_1 and z_2 , two separate GP models $d^{z_1}(\mathbf{a})$ and $d^{z_2}(\mathbf{a})$, are developed for each FBL state.

The developed GP-FBLMPC algorithm was extensively tested across three different paths in sand and grass off-road terrains. Compared to the GP-NMPC algorithm, GP-FBLMPC performed equally well in reducing the path following errors. However, GP-FBLMPC showcases superior computational efficiency by eliminating the need for iterative optimization and requiring fewer GP models for prediction.

The GP-FBLMPC method's adaptability to different paths is another key advantage, as demonstrated in a field experiment shown in Fig. 8 [3]. In this experiment, GP models, initially trained using data from an infinite path (left image in Fig. 8), were applied to a different, track path (right image in Fig. 8) for both GP-FBLMPC and GP-NMPC setups. This was done without re-training the GP models with data specific to the track path. The results showed that GP-FBLMPC significantly reduced path errors in comparison to basic FBLMPC and notably outperformed GP-NMPC, which demonstrated worse performance compared to the standard NMPC in this instance.

Fig. 9 illustrates the alignment of GP predictions from GP-FBLMPC with actual lateral and heading errors on the track path. Notably, these predictions remain accurate despite the GP models being trained on a different path, underscoring the robust generalization capabilities of the GP-FBLMPC algorithm and its effectiveness in consistently reducing path-following errors across varied paths.

6 APPLICATION II: ENHANCED SAFETY FOR MIXED-VEHICLE PLATOONS

This section extends the application of GP-MPC to control autonomous vehicles, particularly focusing on enhancing safety in mixed-vehicle platoons. Here, GP-MPC is employed not only to integrate GP means into the MPC prediction model but also to explicitly account for GP uncertainty propagation as a constraint within the MPC optimization framework. This advanced approach significantly enhances safety in mixed-vehicle platoons by effectively modeling the uncertainties associated with human-driven vehicle (HV) behaviors, a critical aspect often overlooked in conventional control strategies.

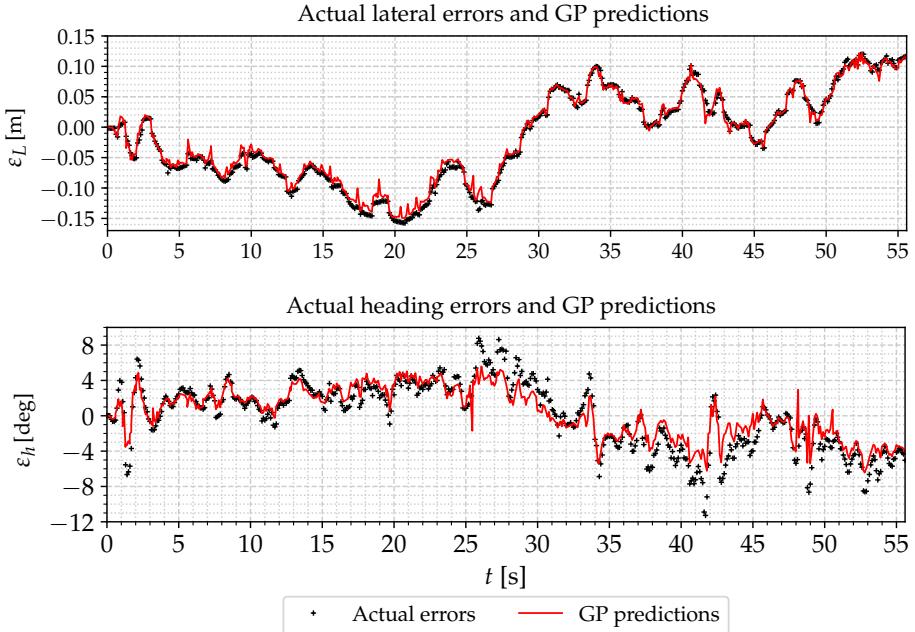


Fig. 9. Graphs depicting the lateral and heading error predictions from the GP models and the actual observed errors over time, for the GP-FBLMPC applied on the track path using GP models trained on the infinite path [3].

6.1 Standard GP-based MPC

The integration of autonomous vehicles (AVs) into existing traffic systems, especially in the domain of AV platooning, holds transformative potential for public traffic management through synchronized vehicle movements. However, the coexistence of human-driven vehicles (HVs) alongside AVs in these mixed-traffic environments introduces complex challenges. Notably, the interaction between AVs and HVs has led to an increased rate of accidents that are predominantly caused by HVs rear-ending AVs, which underscores the urgent need for innovative control strategies tailored to mixed-traffic scenarios.

Addressing this gap, [10] proposed a novel GP learning-based MPC strategy (GP-MPC) specifically designed to enhance longitudinal car-following control within mixed-vehicle platoons, as illustrated in Fig. 10. This approach is primarily aimed at improving safety in interactions between AVs and HVs. It does so by incorporating uncertainties inherent in HV behavior into the control strategy, thus ensuring safer and more efficient operations within mixed-vehicle platoons. The GP-MPC method distinguishes itself by maintaining increased minimum distances between vehicles and facilitating higher travel speeds, representing a significant advancement over traditional control methods, including standard MPC. This approach is particularly effective in complex traffic scenarios, such as emergency braking situations, where it adeptly handles the unpredictable nature of HVs by integrating quantified uncertainty estimated through GP models into the MPC framework.

6.1.1 HV Modeling. In longitudinal car-following scenarios, HVs are traditionally modeled using a nominal function with a fixed human reaction time delay to emulate general human driving behaviors. A common model used for this purpose is an autoregressive with exogenous input (ARX)

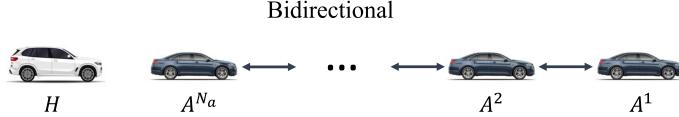


Fig. 10. A mixed vehicle platoon consists of N_a connected AVs, labeled as A^1, A^2, \dots, A^{N_a} , followed by a HV H . These AVs use a sequential bidirectional communication topology for data exchange, excluding direct communication with the HV. This configuration is motivated by recent studies highlighting that most accidents in mixed traffic involve HVs rear-ending AVs, leading to the specific platoon arrangement showcased [10].

model, as described in [10]. The ARX model for HV velocity prediction is represented as:

$$v_k^H = -c_1 v_{k-1}^H - c_2 v_{k-2}^H - c_3 v_{k-3}^H - c_4 v_{k-4}^H + b_1 v_{k-1}^{N_a} + b_2 v_{k-2}^{N_a} + b_3 v_{k-3}^{N_a} + b_4 v_{k-4}^{N_a}, \quad (59a)$$

$$= f(v_{k-1:k-4}^H, v_{k-1:k-4}^{N_a}). \quad (59b)$$

Here, v_{k-i}^H and $v_{k-i}^{N_a}$ represent the velocities of the HV and the last vehicle in the AV platoon at time step $k-i$, respectively. The coefficients $c_{1,\dots,4}$ and $b_{1,\dots,4}$ are constant coefficients associated with velocities of the HV and the last vehicle in the AV platoon at previous time steps respectively. Similar to combining GP models with the nominal model in these path-following control application examples, GP models are combined with the nominal model (59b) not only to enhance the accuracy of the HV model but also to offer a quantifiable measure of modeling uncertainties. The GP-enhanced ARX model is developed as:

$$v_k^H = \sum_{i=1}^4 -c_i v_{k-i}^H + \sum_{i=1}^4 b_i v_{k-i}^{N_a} = f(\cdot), \quad (60a)$$

$$\tilde{v}_k^H = \overbrace{v_k^H}^{\text{ARX prediction}} + \overbrace{g(v_{k-1}^H, v_{k-1}^{N_a})}^{\text{GP-based correction}}. \quad (60b)$$

In this model, \tilde{v}_k^H represents the GP-compensated velocity prediction of the HV. The GP model $g(\cdot)$ learns the divergence between the actual system behaviors and the ARX predictions. It considers both v^H (HV velocity) and v^{N_a} (velocity of the last AV). This ensures that the GP model adequately captures the dynamics of the HV as influenced by the actions of the leading AV.

Data for developing the HV model were collected using a Unity simulator, where drivers were asked to follow an AV platoon while being distracted by answering algebraic questions, mimicking real-world distracted driving scenarios. The setup is illustrated in Fig. 11. Comparative analysis using root mean square error (RMSE) against actual HV velocity data showed that the GP-enhanced HV model (60b) yielded approximately a 35.64% improvement in prediction accuracy over the standard ARX model, highlighting its effectiveness in modeling HV behavior in mixed-vehicle platoons.

6.1.2 Mixed Platooning Model. To effectively manage longitudinal tracking in mixed-vehicle fleets, a GP learning-based MPC strategy is developed, utilizing the GP-enhanced HV model (60b). This strategy integrates crucial constraints like acceleration, speed, and safe distance. A key aspect of this approach involves maintaining a safe distance within the mixed platoon. For AVs in the platoon, defined by the set $n_a = 1, 2, \dots, N_a$, the distance between consecutive vehicles must satisfy



Fig. 11. One of the drivers in a controlled experiment within a Unity driving simulator. The experiment was designed to collect data for HV modeling, simulating distracted driving conditions [10].

a minimum threshold Δ , i.e., $p_k^{n_a-1} - p_k^{n_a} \geq \Delta$. Here, N_a is the number of AVs, and p_k indicates their position. The kinematics of AVs is formulated as:

$$v_{k+1}^{n_a} = v_k^{n_a} + T \text{acc}_k^{n_a}, \quad (61a)$$

$$p_{k+1}^{n_a} = p_k^{n_a} + T v_k^{n_a}. \quad (61b)$$

where $0 < T \ll 1$ indicates the sample time, and $v_k^{n_a}$ and $\text{acc}_k^{n_a}$ denote the velocity and acceleration of A^{n_a} (shown in Fig. 10) respectively. AVs are assumed to be deterministic, with their states measured and communicated error-free, i.e., $\Sigma(v_k^{n_a}) = 0$. By applying (60b), the HV model is derived as follows:

$$\tilde{v}_k^H = v_k^H + d(v_{k-1}^H, v_{k-1}^{N_a}), \quad (62a)$$

$$\begin{aligned} p_{k+1}^H &= p_k^H + T \tilde{v}_k^H, \\ &= p_k^H + T v_k^H + T d(v_{k-1}^H, v_{k-1}^{N_a}). \end{aligned} \quad (62b)$$

In this model, \tilde{v}_k^H is the GP-enhanced HV velocity prediction, where v_k^H is the nominal velocity (computed using (59b)), and p_k^H is the position state. Using (62b) and (25a), the propagation of the HV position mean can then be derived as:

$$\mu(p_{k+1}^H) = \mu(p_k^H) + T \mu(v_k^H) + T \mu^d(v_{k-1}^H, v_{k-1}^{N_a}). \quad (63a)$$

Denoting $\mu(p_{k+1}^H)$ as $\mu_{k+1}^{p^H}$ and rewriting other terms similarly, (63a) is expressed more compactly as:

$$\mu_{k+1}^{p^H} = \mu_k^{p^H} + T v_k^H + T \mu^d(v_{k-1}^H, v_{k-1}^{N_a}), \quad (63b)$$

with the initial value $\mu_0^{p^H} = p_k^H$. Using (62b) and (27), the propagation of the HV position variance is derived as:

$$\Sigma(p_{k+1}^H) = \Sigma(p_k^H) + T^2 \Sigma^d(v_{k-1}^H, v_{k-1}^{N_a}). \quad (64a)$$

Similarly, this can be compactly written as:

$$\Sigma_{k+1}^{p^H} = \Sigma_k^{p^H} + T^2 \Sigma^d(v_{k-1}^H, v_{k-1}^{N_a}). \quad (64b)$$

with $\Sigma_0^{p^H} = 0$, this equation allows for the tracking of the HV position variance, which is vital for ensuring safety in mixed platooning scenarios. It is important to note that in this model, covariances between p_k^H and v_k^H are not considered during the HV uncertainty propagation.

6.1.3 Safe Distance Probability Constraint. To ensure a safe distance within a mixed-vehicle platoon, a probabilistic constraint, also known as a chance constraint, is established. This constraint is crucial for maintaining a safe separation between the trailing AV and the HV, particularly given the uncertain nature of HV behavior. The constraint is formulated as:

$$\Pr \left(p_k^{N_a} - (p_k^H + \Delta) > \Delta_{\text{ext}} \right) \geq p_{\text{def}} . \quad (65)$$

In this formulation, Δ is the predefined safe distance between the AVs, and $\Delta_{\text{ext}} \geq 0$ is an additional buffer to accommodate the stochastic behavior of the HV. The predefined p_{def} denotes the desired satisfaction probability.

To express this probabilistic constraint in a more deterministic form, we use the concept of a half-space constraint $X^{hs} := \{x | h^\top x \leq b\}$, $h \in \mathbb{R}^n$, where $h \in \mathbb{R}^n$, and $b \in \mathbb{R}$. The tightened constraint on the state mean, as derived in [16], is expressed as:

$$X^{hs} (\Sigma_i^x) := \left\{ x \mid h^\top x \leq b - \phi^{-1} (p_{\text{def}}) \sqrt{h^\top \Sigma_i^x h} \right\} . \quad (66)$$

In this equation, $h^\top = [-1 \ 1]$, $x := [p_k^{N_a} \ p_k^H + \Delta]^\top$, and $b = -\Delta_{\text{ext}}$. Here, ϕ^{-1} is the inverse of the cumulative distribution function (CDF). In our scenario,

$$\Sigma_k^x := \begin{bmatrix} \Sigma_k^{p^{N_a}} \\ \Sigma_k^{p^H} + \Delta \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & \Sigma_k^H \end{bmatrix} . \quad (67)$$

Considering there is no covariance between the positions of the HV and the leading AV ($\Sigma_k^{p^{N_a}} = 0$), and the position variance of the HV (Σ_k^H) is computed using (64b), we can establish a “tightened” constraint on the position state by incorporating (67) into (66). This leads to

$$p_k^{N_a} - p_k^H \geq \Delta + \Delta_{\text{ext}} + \phi^{-1} (p_{\text{def}}) \sqrt{\Sigma_k^H} . \quad (68)$$

The chance constraint for maintaining safe distances, given by (65), is reduced to a deterministic representation in (68). It can be further simplified by omitting Δ_{ext} if the satisfaction probability p_{def} is set high. By adaptively adjusting the safe distance based on the estimated uncertainties from the HV’s GP model, as per (64b), this approach ensures that the safe distance always exceeds the minimum requirement of Δ under varying conditions. This adaptive adjustment is key to enhancing safety in mixed-vehicle platoons, catering to the unpredictable nature of human driving behaviors.

6.1.4 GP-Based MPC. The proposed GP-based MPC strategy, specifically tailored for mixed-vehicle platoon scenarios, integrates an advanced HV model for a platoon consisting of N_a AVs followed

by an HV as depicted in Fig. 10. The strategy can be expressed as:

$$\min_{\mathbb{V}} \sum_{n_a=1}^{N_a} \sum_{i=k}^{k+N-1} \left\| acc_{i|k}^{n_a} \right\|_R^2 + \sum_{i=k}^{k+N} \left\| v_{i+1|k}^1 - v_{i+1|k}^{\text{ref}} \right\|_{Q_1}^2 + \sum_{n_a=2}^{N_a} \sum_{i=k}^{k+N} \left\| v_{i+1|k}^{n_a} - v_{i+1|k}^{n_a-1} \right\|_{Q_2}^2 \quad (69a)$$

$$\text{with } \mathbb{V} = \left\{ v_{i|k}^1, v_{i|k}^{n_a}, v_{i|k}^H, p_{i|k}^{n_a}, \mu_{i|k}^{p^H}, \Sigma_{i|k}^{p^H}, acc_{i|k}^{n_a} \right\}$$

subject to

$$v_{i+1|k}^{n_a} = v_{i|k}^{n_a} + T acc_{i|k}^{n_a}, \quad p_{i+1|k}^{n_a} = p_{i|k}^{n_a} + T v_{i|k}^{n_a}, \\ n_a = \{1, 2, \dots, N_a\}, \quad (69b)$$

$$v_{i|k}^H = f \left(v_{i-1:i-4|k}^H, v_{i-1:i-4|k}^{N_a} \right), \quad (69c)$$

$$\mu_{i+1|k}^{p^H} = \mu_{i|k}^{p^H} + T v_{i|k}^H + T \mu^d(v_{i-1|k}^H, v_{i-1|k}^{N_a}), \quad (69d)$$

$$\Sigma_{i+1|k}^{p^H} = \Sigma_{i|k}^{p^H} + T^2 \Sigma^d(v_{i-1|k}^H, v_{i-1|k}^{N_a}), \quad (69e)$$

$$p_{i|k}^{n_a-1} - p_{i|k}^{n_a} \geq \Delta, \quad (69f)$$

$$p_{i|k}^{N_a} - \mu_{i|k}^{p^H} \geq \Delta + \phi^{-1}(p_{\text{def}}) \sqrt{\Sigma_{i|k}^{p^H}}, \quad (69g)$$

$$v_{\min} \leq v_{i|k}^{n_a} \leq v_{\max}, \quad acc_{\min} \leq acc_{i|k}^{n_a} \leq acc_{\max}. \quad (69h)$$

The cost function (69a) emphasizes minimizing the acceleration of each AV, aligning the leader AV's velocity with the reference speed, and maintaining consistent velocities among the AVs. HV velocities are not directly controlled but are instead factored into the system through the HV model, enhancing the system's predictive accuracy and enabling the management of uncertainties associated with HV behavior. The constraints, (69b)–(69h), encompass the vehicle dynamics, including the velocity and position relationships for both AVs and the HV, originally represented by (61a), (61b), (59b), (63b), and (64b). Additionally, the constraints ensure safe distances between vehicles and adhere to predefined velocity and acceleration limits.

In testing, the GP-MPC was benchmarked against a standard MPC in scenarios involving constant-velocity tracking and emergency braking. The results demonstrated the GP-MPC's superior performance in maintaining safety margins and higher average speeds, showcasing its effectiveness in mixed-traffic environments. By incorporating HV uncertainties, the GP-MPC strategy significantly enhances safety and efficiency, offering a robust and adaptable solution for complex traffic scenarios involving a mix of autonomous and human-driven vehicles.

6.2 Sparse GP-based MPC

We have shown that GP models are exceptionally effective in modeling the complex dynamics of robotic systems. The integration of GP models with MPC, forming GP-MPC frameworks, not only considerably improves the accuracy of MPC's predictive models but also effectively integrates quantified uncertainties into the optimization loop, thereby enhancing system performance significantly. However, applying standard GP models within the MPC framework presents a notable challenge, particularly in scenarios with large-scale data as discussed in Sec. 2.3. As the training dataset size increases, the prediction time for standard GP models escalates, rendering them less feasible for real-time MPC applications.

To address this challenge, in previous path-following application examples, strategies such as employing local GP models to maintain a manageable data size (Sec. 5.1) or limiting the path lengths

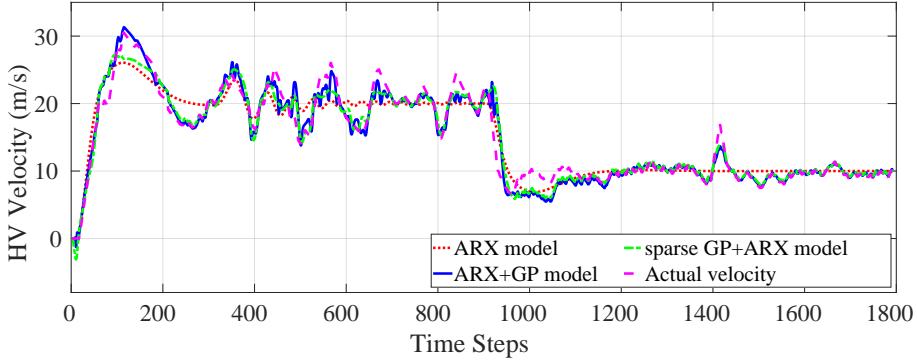


Fig. 12. Comparative analysis of velocity prediction models. Plots show velocity predictions of the ARX, ARX+GP, and sparse GP+ARX models, against actual HV velocity measurements. Compared to the ARX model, the ARX+GP exhibits approximately 35.64% improvement, and sparse GP+ARX achieves a 23.94% enhancement with significantly reduced computational demands [17].

(Sec. 5.2) were implemented. To further mitigate computational demands, they also ignore the uncertainty propagation in GP models and neglect constraints in the MPC optimization process, which potentially compromise the overall performance of GP-MPC methods. In mixed-platoon control utilizing standard GP-based MPC strategies (Sec. 6.1), the computation time reaches approximately 19 seconds per time step, underscoring their impracticality for real-time operation.

A promising solution to this computational dilemma lies in the adoption of sparse GP models. Sparse GP models are a computationally efficient alternative to standard GP models, striking a critical balance between computational efficiency and the precision of approximations [4]. This balance is essential for ensuring that the GP models remain viable for real-time applications without significantly compromising their predictive accuracy or their ability to effectively manage uncertainties, which makes it feasible to apply sophisticated GP-MPC methods in real-time, large-scale data applications, thereby expanding the potential and applicability of these advanced control systems.

6.2.1 Sparse GP+ARX Modeling. The research detailed in [17] illustrates a successful application of the fully independent conditional (FIC) approximation, a leading-edge sparse GP approach, for enhancing computational efficiency in dynamic systems modeling. The hyperparameters for the sparse GP model were directly loaded from the trained standard GP model, and 20 inducing points were automatically selected by the FIC method within the training datasets. This incorporation resulted in a remarkable boost in computational speed, where the sparse GP model required only about 5.6% of the average prediction time needed by the standard GP model.

In terms of modeling precision, the standard GP+ARX model exhibited a notable 35.64% improvement in forecasting HV velocities relative to the ARX nominal model. The sparse GP+ARX model achieved an approximate 23.94% improvement compared to the ARX model, which provides a good balance of prediction accuracy and computational efficiency. To illustrate this improvement, Fig. 12 displays a comparison of velocity predictions using the ARX model, the ARX+GP model, and the sparse GP+ARX model against measured HV velocities. The comparison clearly demonstrates that both the standard GP+ARX and the sparse GP+ARX models significantly outperform the ARX model in accurately predicting the HV behaviors. These results highlight the sparse GP+ARX



Fig. 13. The autonomous vehicle platform, UW Moose, and a following human-driven vehicle on the test track [28].

model's effectiveness in providing a good balance between enhanced modeling precision and substantial gains in computational efficiency.

6.2.2 Dynamic Sparse GP Prediction in MPC. Despite the notable reduction in computation time achieved by employing the sparse GP+ARX model for HV modeling, real-time integration of GP models within the MPC framework continues to pose challenges due to the inherently complex nature of MPC [27]. MPC involves solving an optimal control problem at each time step, adhering to constraints as outlined in equations (69b)–(69h). Addressing this challenge, [17] introduced a dynamic sparse approximation method for GP-MPC, aimed at further speeding up the process.

Leveraging the receding horizon characteristic of MPC, where the predictive trajectory at the current time step closely resembles the trajectory calculated at the previous time step, the dynamic sparse GP approach computes calculations using the previously derived trajectory. These results are then utilized to update the mean and variance for the current time step, as per equations (69d) and (69e). This strategy significantly reduces computational demands by replacing repetitive predictions for each time step within the horizon with a single sparse GP prediction for the entire prediction horizon. As a result, this dynamic sparse GP-MPC methodology becomes practicable for real-time applications, especially those requiring rapid sampling rates, thereby enhancing the feasibility and efficiency of GP-MPC schemes in time-sensitive control applications.

6.2.3 Testing with Realistic Velocity Profiles. In [10] and [17], the HV model including both the nominal ARX and GP components, was estimated with data collected in a human-in-the-loop Unity driving simulator. In [28], the HV model incorporates field experiment data, complementing the nominal model estimated with data from the simulator. This advancement involved collecting data for the GP component during field experiments where an HV followed an AV, as shown in Fig. 13. The data was gathered on an 850-meter real-world track, distinguished by its varying curvatures and bumps, to provide a detailed and realistic dataset. This dataset was pivotal in improving the GP model's accuracy and reliability. Notably, compared to the simulator-based ARX model, the field data-integrated GP model showed a significant 36.34% improvement in RMSE accuracy.

Utilizing this improved HV model, the dynamic Sparse GP-based MPC policy was tested using the worldwide harmonized light vehicle test procedure (WLTP) as the reference speed profile for

the leader HV. The WLTP, recognized for its realistic representation of driving conditions in car-following scenarios [29], makes it an ideal benchmark for evaluating the controller's performance in real-life car-following scenarios.

Fig. 14 shows the outcomes of these WLTP scenario tests. This comprehensive display includes plots of the velocity response, position trajectories, and inter-vehicle distances, providing an in-depth perspective on the system's adaptability to the varied conditions presented by the WLTP cycle. These results underscore the model's ability to dynamically adjust safety margins in response to varying driving conditions, thereby affirming its applicability and effectiveness in a range of traffic scenarios. The GP-MPC showcases enhanced safety by increasing the minimum distance between the HV and the following AV by 0.4 meters compared to the nominal MPC. Furthermore, the position plots reveal that vehicles under GP-MPC management maintain positions several meters ahead (4–9 meters) of those managed by nominal MPC, indicating more efficient traffic flow in realistic driving situations.

7 CONCLUSION

This tutorial has systematically explored the integration of Gaussian process (GP) models with model predictive control (MPC), presenting a comprehensive and accessible guide for its implementation in complex systems, especially in robotics. Our detailed mathematical exposition on GP-MPC, a novel contribution to the field, demystifies the intricacies of integrating GP models within the MPC framework. We have highlighted the importance of approximating means and variances for multi-step predictions, a crucial aspect in dynamic environments, thus addressing a significant gap in the current literature.

This tutorial began with the fundamentals of GP regression, transitioning smoothly into the integration of these principles within MPC, emphasizing the enhancement of MPC's predictive capabilities. The theoretical discussions are supported by practical applications, including robotic path-following and vehicle platooning, demonstrating GP-MPC's applicability and effectiveness in real-world scenarios.

The GP-MPC approach presented in this tutorial not only bridges the knowledge gap for practitioners but also opens new avenues for research and application in learning-based control systems. It provides the necessary tools and understanding for researchers and practitioners to navigate and innovate in the evolving landscape of robotics control, where dealing with uncertainties and dynamic environments is paramount.

The contributions of this tutorial extend beyond academic understanding, offering practical solutions and insights that are critical in advancing applications of learning-based control systems. By providing a solid theoretical background coupled with practical implementation strategies, this tutorial stands as a pivotal resource in the field of GP-MPC, encouraging further exploration and innovation.

8 ACKNOWLEDGMENTS

The author wishes to extend heartfelt thanks to Prof. Joshua Marshall at Queen's University for his invaluable support and mentorship. His guidance, particularly during the first author's postdoctoral tenure at Queen's University, provided not only academic direction but also inspiration and encouragement. This work, with its intricate explorations into the domain of GP-MPC, has been greatly influenced by his deep understanding and innovative approach towards robotics control.

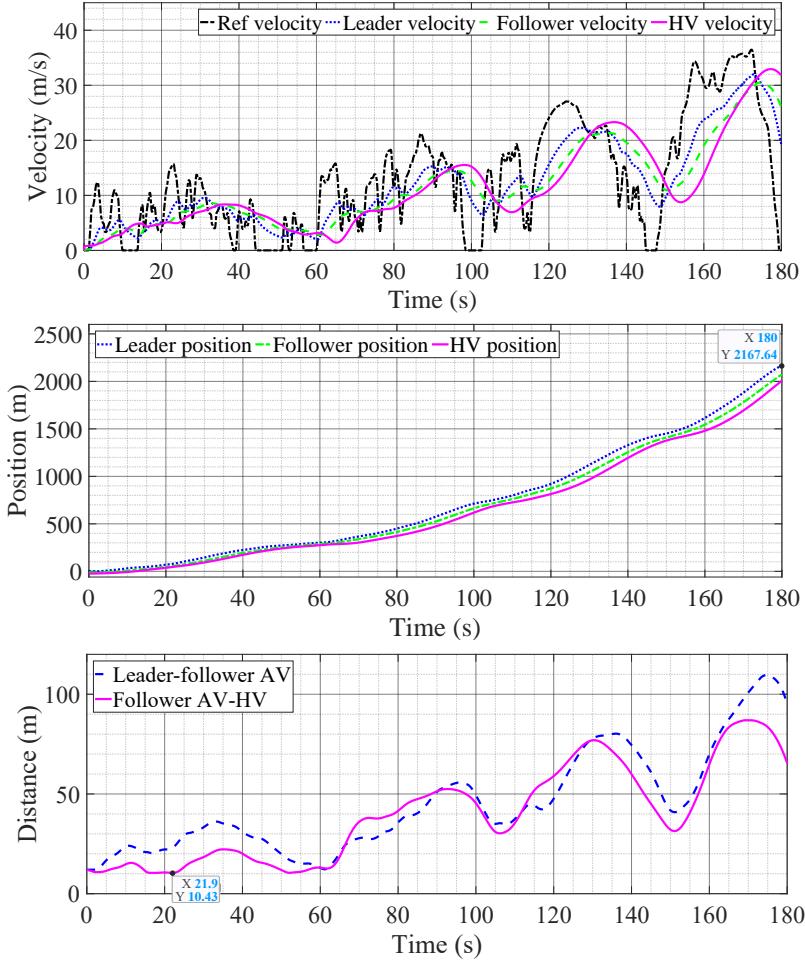


Fig. 14. WLTP scenario test results using the dynamic sparse GP-based MPC. Arranged in descending order, these plots include velocity responses, vehicle position trajectories, and inter-vehicle distances. Notably, the GP-MPC ensures increased safety by augmenting the minimum inter-vehicle distance compared to the nominal MPC. Moreover, the vehicle positions in GP-MPC scenarios are substantially ahead of the nominal MPC, indicating enhanced traffic efficiency under realistic driving conditions [28].

9 APPENDICES

9.1 Theorems of Gaussian Mean and Variance Approximation

This appendix presents the theoretical basis for Gaussian mean and variance approximation in GP regression. The foundational principles of Gaussian process (GP) regression are developed in [30] and [31], where a GP model $f(x)$ is used to approximate a function $y = f(x) + \epsilon$. The approximations of the GP mean and variance use the principles of iterated expectations and conditional variances,

respectively. These calculations are represented as follows:

$$\begin{aligned} m(x^*) &= \mathbb{E}_{x^*} [\mathbb{E}_{f(x^*)} [f(x^*)|x^*]] , \\ &= \mathbb{E}_{x^*} [\mu(x^*)] . \end{aligned} \quad (\text{A.1})$$

$$\begin{aligned} var(x^*) &= \mathbb{E}_{x^*} [\text{var}_{f(x^*)} (f(x^*)|x^*)] + \text{var}_{x^*} (\mathbb{E}_{f(x^*)} [f(x^*)|x^*]) , \\ &= \mathbb{E}_{x^*} [\Sigma(x^*)] + \text{var}_{x^*} (\mu(x^*)) . \end{aligned} \quad (\text{A.2})$$

In (A.1), $\mathbb{E}_{f(x^*)} [f(x^*)|x^*] = \mu(x^*)$ implies that the expected value of the GP model, $(\mathbb{E}_{f(x)})$, evaluated at a test point x^* , is equivalent to the mean at that test point. The notation x^* represents the test points for which predictions are desired. For clarity, (A.1) and (A.2) are expressed explicitly with a time index:

$$\begin{aligned} m(x_{k+1}^*) &= \mathbb{E}_{x_k^*} [\mathbb{E}_{f(x^*)} [f(x^*)|x_{k+1}^*]] , \\ &= \mathbb{E}_{x_k^*} [\mu(x_{k+1}^*)] . \end{aligned} \quad (\text{A.3})$$

$$\begin{aligned} var(x_{k+1}^*) &= \mathbb{E}_{x_k^*} [\text{var}_{f(x^*)} (f(x^*)|x_{k+1}^*)] + \text{var}_{x_k^*} (\mathbb{E}_{f(x^*)} [f(x^*)|x_{k+1}^*]) , \\ &= \mathbb{E}_{x_k^*} [\Sigma(x_{k+1}^*)] + \text{var}_{x_k^*} (\mu(x_{k+1}^*)) . \end{aligned} \quad (\text{A.4})$$

9.1.1 Predictive Mean Approximation. The approximation of the expectation under x_k^* of the GP mean $\mu(x_k^*)$, as described in (A.3), involves its first-order Taylor expansion around $x^* = \mu_{x_k^*}$. Following the equations in [30, Eq. 14], this can be expressed as:

$$\mu(x_k^*) \approx \mu(\mu_{x_k^*}) + \frac{\partial \mu(x^*)}{\partial x^*} \Bigg|_{x^*=\mu_{x_k^*}}^\top (x^* - \mu_{x_k^*}) + O\left(||x^* - \mu_{x_k^*}||^2\right) . \quad (\text{A.5})$$

Subsequently, the expectation $\mathbb{E}_{x_k^*} [\mu(x_k^*)]$ is approximated following the process outlined in [30, Eq. 15]:

$$\begin{aligned} \mathbb{E}_{x_k^*} [\mu(x_k^*)] &\approx \mathbb{E}_{x_k^*} \left[\mu(\mu_{x_k^*}) + \frac{\partial \mu(x^*)}{\partial x^*} \Bigg|_{x^*=\mu_{x_k^*}}^\top (x^* - \mu_{x_k^*}) \right] , \\ &= \mu(\mu_{x_k^*}) . \end{aligned} \quad (\text{A.6})$$

9.1.2 Predictive Variance Approximation. Similarly, for the expectation under x_k^* of the GP variance $\Sigma(x_k^*)$ in (A.4), we approximate $\Sigma(x_k^*)$ through its first-order Taylor expansion around $x^* = \mu_{x_k^*}$ as:

$$\Sigma(x_k^*) \approx \Sigma(\mu_{x_k^*}) + \frac{\partial \Sigma(x^*)}{\partial x^*} \Bigg|_{x^*=\mu_{x_k^*}}^\top (x^* - \mu_{x_k^*}) + O\left(||x^* - \mu_{x_k^*}||^2\right) . \quad (\text{A.7})$$

The expectation $\mathbb{E}_{x_k^*} [\Sigma(x_k^*)]$, following the derivations in [30, Eq. 18] and [19, Appendix I], is approximated as:

$$\begin{aligned} \mathbb{E}_{x_k^*} [\Sigma(x_k^*)] &\approx \mathbb{E}_{x_k^*} \left[\Sigma(\mu_{x_k^*}) + \frac{\partial \Sigma(x^*)}{\partial x^*} \Bigg|_{x^*=\mu_{x_k^*}}^\top (x^* - \mu_{x_k^*}) \right] \\ &= \Sigma(\mu_{x_k^*}) . \end{aligned} \quad (\text{A.8})$$

For the second term of (A.2), $\text{var}_{x^*}(\mu(x^*))$ is derived as follows, based on [30, Eq. 20]:

$$\begin{aligned}\text{var}_{x_k^*}(\mu(x_k^*)) &\approx \text{var}_{x_k^*} \left[\mu(\mu_{x_k^*}) + \frac{\partial \mu(x^*)}{\partial x^*} \Big|_{x^*=\mu_{x_k^*}} (x^* - \mu_{x_k^*}) \right] \\ &= \frac{\partial \mu(x^*)}{\partial x^*} \Big|_{x^*=\mu_{x_k^*}} \Sigma_{x_k^*} \frac{\partial \mu(x^*)}{\partial x^*} \Big|_{x^*=\mu_{x_k^*}}.\end{aligned}\quad (\text{A.9})$$

9.2 Calculation of the Cost Function

To compute the partial derivatives of the cost function, we need to refer to equations (69) and (81) from [32, Sec. 2.4]:

$$\frac{\partial \mathbf{x}^\top \mathbf{a}}{\partial \mathbf{x}} = \frac{\partial \mathbf{a}^\top \mathbf{x}}{\partial \mathbf{x}} = \mathbf{a}, \quad (\text{A.10})$$

$$\frac{\partial \mathbf{x}^\top \mathbf{A} \mathbf{x}}{\partial \mathbf{x}} = (\mathbf{A} + \mathbf{A}^\top) \mathbf{x}. \quad (\text{A.11})$$

Given that \mathbf{Q} and \mathbf{R} are positive semi-definite symmetric matrices in the cost function (30), we have:

$$\frac{\partial \mathbf{x}^\top \mathbf{A} \mathbf{x}}{\partial \mathbf{x}} = 2\mathbf{A}\mathbf{x}, \quad (\text{A.12})$$

The partial derivative of the cost function (39) in Sec. 4, can be calculated:

$$\begin{aligned}\frac{\partial J(\delta \mathbf{u}_k)}{\partial \delta \mathbf{u}_k} &\approx \frac{\partial ((\mathbf{x}_{d,k+1} - \bar{\mathbf{x}}_{k+1} - \mathbf{H}' \delta \mathbf{u}_k)^\top \mathbf{Q} (\mathbf{x}_{d,k+1} - \bar{\mathbf{x}}_{k+1} - \mathbf{H}' \delta \mathbf{u}_k))}{\partial \delta \mathbf{u}_k} + \frac{\partial ((\bar{\mathbf{u}}_k + \delta \mathbf{u}_k)^\top \mathbf{R} (\bar{\mathbf{u}}_k + \delta \mathbf{u}_k))}{\partial \delta \mathbf{u}_k}, \\ &= \frac{\partial ((\tilde{\mathbf{x}}_{k+1} - \mathbf{H}' \delta \mathbf{u}_k)^\top \mathbf{Q} (\tilde{\mathbf{x}}_{k+1} - \mathbf{H}' \delta \mathbf{u}_k))}{\partial \delta \mathbf{u}} + \frac{\partial ((\bar{\mathbf{u}}_k + \delta \mathbf{u}_k)^\top \mathbf{R} (\bar{\mathbf{u}}_k + \delta \mathbf{u}_k))}{\partial \delta \mathbf{u}}, \\ &= 2\mathbf{Q}(\tilde{\mathbf{x}}_{k+1} - \mathbf{H}' \delta \mathbf{u}_k)(-\mathbf{H}') + 2\mathbf{R}(\bar{\mathbf{u}}_k + \delta \mathbf{u}_k), \\ &= 2(-\mathbf{Q}\tilde{\mathbf{x}}_{k+1}\mathbf{H}' + \mathbf{Q}\mathbf{H}'\delta\mathbf{u}_k\mathbf{H}' + \mathbf{R}\bar{\mathbf{u}}_k + \mathbf{R}\delta\mathbf{u}_k), \\ &= 2(-\mathbf{H}'^\top \mathbf{Q}\tilde{\mathbf{x}}_{k+1} + \mathbf{H}'^\top \mathbf{Q}\mathbf{H}'\delta\mathbf{u}_k + \mathbf{R}\bar{\mathbf{u}}_k + \mathbf{R}\delta\mathbf{u}_k), \\ &= 2((\mathbf{H}'^\top \mathbf{Q}\mathbf{H}' + \mathbf{R})\delta\mathbf{u}_k + \mathbf{R}\bar{\mathbf{u}}_k - \mathbf{H}'^\top \mathbf{Q}\tilde{\mathbf{x}}_{k+1}), \\ &= 0.\end{aligned}\quad (\text{A.13})$$

This leads to the solution,

$$\delta \mathbf{u}_k = \left(\mathbf{H}'^\top \mathbf{Q} \mathbf{H}' + \mathbf{R} \right)^{-1} \left(\mathbf{H}'^\top \mathbf{Q} \tilde{\mathbf{x}}_{k+1} - \mathbf{R} \bar{\mathbf{u}}_k \right). \quad (\text{A.14})$$

REFERENCES

- [1] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, “Safe learning in robotics: From learning-based control to safe reinforcement learning,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 411–444, 2022.
- [2] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, “Learning-based model predictive control: Toward safe learning in control,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 269–296, 2020.
- [3] J. Wang, M. T. Fader, and J. A. Marshall, “Learning-based model predictive control for improved mobile robot path following using Gaussian processes and feedback linearization,” *Journal of Field Robotics*, vol. 40, pp. 1014–1033, 2023.
- [4] X. Wang, Y. Jin, S. Schmitt, and M. Olhofer, “Recent advances in Bayesian optimization,” *ACM Computing Surveys*, vol. 55, no. 13s, pp. 1–36, 2023.

- [5] J. Wang, "An intuitive tutorial to Gaussian process regression," *Computing in Science & Engineering*, vol. 25, no. 4, pp. 4–11, 2023.
- [6] Y. He and Y. Zhao, "Adaptive robust control of uncertain Euler–Lagrange systems using Gaussian processes," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
- [7] A. Khalid, Z. Mushtaq, S. Arif, K. Zeb, M. A. Khan, and S. Bakshi, "Control schemes for quadrotor UAV: Taxonomy and survey," *ACM Computing Surveys*, vol. 56, no. 5, pp. 1–32, 2023.
- [8] V. Deshpande and Y. Zhang, "Fault-tolerant model predictive control of a fixed-wing UAV with actuator fault estimation," *Guidance, Navigation and Control*, vol. 1, no. 04, p. 2140006, 2021.
- [9] T. Chevet, C. Vlad, C. S. Maniu, and Y. Zhang, "Decentralized MPC for UAVs formation deployment and reconfiguration with multiple outgoing agents," *Journal of Intelligent & Robotic Systems*, vol. 97, no. 1, pp. 155–170, 2020.
- [10] J. Wang, Z. Jiang, and Y. V. Pant, "Improving safety in mixed traffic: A learning-based model predictive control for autonomous and human-driven vehicle platooning," *Knowledge-Based Systems*, vol. 293, p. 111673, 2024.
- [11] M. Liu, G. Chowdhary, B. C. Da Silva, S.-Y. Liu, and J. P. How, "Gaussian processes for learning and control: A tutorial with examples," *IEEE Control Systems Magazine*, vol. 38, no. 5, pp. 53–86, 2018.
- [12] D. Duvenaud, "Automatic model construction with Gaussian processes," Ph.D. dissertation, University of Cambridge, 2014.
- [13] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.
- [14] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.
- [15] C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning*. Springer, 2006.
- [16] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using Gaussian process regression," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2736–2743, 2019.
- [17] J. Wang, Y. V. Pant, and Z. Jiang, "Learning-based modeling of human-autonomous vehicle interaction for improved safety in mixed-vehicle platooning control," *Transportation Research Part C: Emerging Technologies*, 2024.
- [18] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When Gaussian process meets big data: A review of scalable GPs," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 11, pp. 4405–4423, 2020.
- [19] L. Hewing, A. Liniger, and M. N. Zeilinger, "Cautious NMPC with Gaussian process dynamics for autonomous miniature race cars," in *2018 European Control Conference (ECC)*. IEEE, 2018, pp. 1341–1348.
- [20] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," *Advances in Neural Information Processing Systems*, vol. 18, 2005.
- [21] J. Quinonero-Candela, A. Girard, and C. E. Rasmussen, "Prediction at an uncertain input for Gaussian processes and relevance vector machines-application to multiple-step ahead time-series forecasting," Technical University of Denmark, DTU: Informatics and Mathematical Modelling, Tech. Rep., 2003.
- [22] M. P. Deisenroth, *Efficient Reinforcement Learning using Gaussian processes*. KIT Scientific Publishing, 2010.
- [23] H.-A. Langåker, "Cautious MPC-based control with machine learning," Master's thesis, Norwegian University of Science and Technology, 2018.
- [24] C. J. Ostafew, A. P. Schoellig, T. D. Barfoot, and J. Collier, "Learning-based nonlinear model predictive control to improve vision-based mobile robot path tracking," *Journal of Field Robotics*, vol. 33, no. 1, pp. 133–152, 2016.
- [25] I. Sadeghzadeh, M. Abdolhosseini, and Y. Zhang, "Payload drop application using an unmanned quadrotor helicopter based on gain-scheduled PID and model predictive control," *Unmanned Systems*, vol. 2, no. 01, pp. 39–52, 2014.
- [26] J.-J. Slotine and W. Li, *Applied Nonlinear Control*. Prentice Hall Englewood Cliffs, NJ, 1991.
- [27] M. Abdolhosseini, Y. M. Zhang, and C. A. Rabbath, "An efficient model predictive control scheme for an unmanned quadrotor helicopter," *Journal of Intelligent & Robotic Systems*, vol. 70, pp. 27–38, 2013.
- [28] J. Wang, Y. V. Pant, L. Zhao, M. Antkiewicz, and K. Czarnecki, "Enhancing safety in mixed traffic: Learning-based modeling and efficient control of autonomous and human-driven vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 2024.
- [29] A. Coppola, D. G. Lui, A. Petrillo, and S. Santini, "Eco-driving control architecture for platoons of uncertain heterogeneous nonlinear connected autonomous electric vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 24 220–24 234, 2022.
- [30] A. Girard, C. Rasmussen, and R. Murray-Smith, "Gaussian process priors with uncertainty inputs: multiple-step-ahead prediction," *Technical Report TR-2002-119, Dept. of Computer Science*, 2002.
- [31] A. Girard, C. Rasmussen, J. Q. Candela, and R. Murray-Smith, "Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting," *Advances in Neural Information Processing Systems*, vol. 15, 2002.
- [32] K. B. Petersen and M. S. Pedersen, "The matrix cookbook," 2012, version 20121115. [Online]. Available: <http://www2.compute.dtu.dk/pubdb/pubs/3274-full.html>