Learning-based model predictive control for path tracking of autonomous vehicles considering model uncertainties

Ren Hongbin[1,2], Xie Panpan[2], Wu Zhicheng[2], Wang Yang[3]

1. State Key Laboratory of Mechanical Transmission for Advanced Equipments, Chongqing University；2.Department of Vehicle Engineering, Beijing Institute of Technology Author Address, Beijing 100081; 3. China North Vehicle Research Institute, Beijing 100072

[Abstract] In order to improve the control effect of unmanned vehicles under extreme working conditions, this paper proposes a learning-based model predictive control method to achieve optimal control of vehicles. Firstly, the Gaussian process regression is used to learn the residual model of the vehicle to compensate for the error of the dynamic model. Secondly, the covariance matrix is introduced to tighten the constraints of the vehicle to achieve the purpose of safety control. Thirdly, the uncertainty introduced by the Gaussian process is studied, and the robust asymptotic stability of the system is proved. Finally, operating conditions were designed, and biases were introduced to the vehicle's actual parameters during modeling to simulate model uncertainties. Comparative simulation verification was conducted between Gaussian Process Model Predictive Control (GP-MPC) and traditional Nonlinear Model Predictive Control (NMPC). The results show that GPMPC has a more accurate prediction of the vehicle state and can better cope with uncertainty in control. The vehicle does not exceed the road boundary when the vehicle parameters are uncertain.

Keywords: autonomous vehicle；gaussian process regression；model predictive control

Introduction

With the continuous in-depth research on autonomous driving technology, the industry’s attention to autonomous driving technology in uncertain environments is increasing. Currently, the consideration of uncertainty mainly focuses on the uncertainty of perception information and traffic participants’ intentions, while the uncertainty of vehicle parameters receives less attention. Ignoring this uncertainty can impact vehicle safety, especially under extreme conditions. Precisely controlling the vehicle’s operation under extreme conditions can effectively enhance the safety performance boundaries of autonomous vehicles. Autonomous racing cars exhibit characteristics such as high-speed operation, highly nonlinear systems, and operating close to their limits, making them an important experimental platform for studying these issues[1]-[2].

Model Predictive Control (MPC) is an advanced control strategy that ensures system stability while imposing constraints on variables. It has significant advantages in handling vehicle control under extreme conditions. In recent years, with advancements in computer hardware[3], real-time deployment of MPC on vehicles has become more straightforward.

Given that MPC predicts system behavior by solving an optimization problem based on a model, inaccuracies in the model can adversely affect control performance. Particularly in extreme conditions, relying on an inaccurate model may lead to violations of constraint boundaries. Researchers have explored robust MPC methods to handle uncertainties affecting system dynamics including TUBE MPC. Although these methods address uncertainty, their performance may be impacted due to the unchanged controller structure.

With the advancement of machine learning, scholars are increasingly considering the integration of MPC with machine learning to achieve enhanced control performance. Learning-based MPC research can be broadly categorized into four directions: learning residual models, learning control strategy parameters, learning control laws, and learning cost representations[4].

Among these, Gaussian process regression (GPR) has been widely used for learning residual models[5]. For instance, Klenske[6] utilized GPR to correct model errors, resulting in improved control performance.

However, introducing GPR significantly increases computational complexity. To address this, Carrau[7] simplified the model to enhance computational speed. Lukas Hewing[8] explored sparse approximations of GPR, inducing the process with a finite set of points and validated it on an AMZ Driverless car. However, further investigation is needed regarding the selection of original data.

Additionally, incorporating GPR during system modeling introduces uncertainty. To mitigate this, computed covariances can be used to tighten system boundaries for cautious control[9]. However, in multi-step open-loop predictions, this uncertainty rapidly accumulates, leading to large covariance values. Some researchers have adopted integral visibility methods to reasonably tighten constraints. Beyond constraint handling, auxiliary control strategies can be introduced to mitigate the impact of uncertainty.

This paper aims to improve existing GP-MPC methods, with the following main contributions:

1. Leveraging ideas from TUBE-MPC, we introduce covariance matrices to tighten vehicle constraints, ensuring control safety.

2. We analyze the stability of GP-MPC and prove the robust asymptotic stability of closed-loop systems with Gaussian residual models.

3. Comparative analysis of the proposed GP-MPC method and traditional NMPC is performed under various operating conditions.

1 Gaussian Process Regression

1.1 Gaussian Process Regression Principles

GPR is a mathematical method based on Bayesian optimization, aiming to model system residuals in a non-parametric probabilistic framework. In this approach, a Gaussian process is considered as a collection of finite random variables following Gaussian distributions[10].

The existing dataset, denoted as $D = (Z, Y) = \{(z_i, y_i)|i = 1, \cdots, n\}$, $z_i \in \mathfrak{R}^{n_z}$, consist of relevant feature state vectors (independent variables of the GP model), and let $y_i \in \mathfrak{R}^{n_d}$ represent the dependent variable (output) of the GP model. The relationship between these two can be expressed as follows:

$$y_i = d_{GP}(z) + w_k \tag{1}$$

where, d: $\mathfrak{R}^{n_z} \to \mathfrak{R}$ represents a mapping from an n_z-dimensional real-valued space to a 1-dimensional space. The term $w \sim N(0, \delta_w)$ corresponds to Gaussian noise. The primary goal of GPR is to find the mapping function d. Given a dataset D, we can compute the prior distribution:

$$d(z) \sim gp(m(z)) \tag{2}$$

Where, $m(\boldsymbol{z})$ represents the mean vector of the dataset. Typically, we assume a prior mean of 0. $k(\boldsymbol{z}, \boldsymbol{z}')$ denotes the Gram matrix of the dataset variables.

$$k(z, z') = \begin{bmatrix} k(z_1, z_1) & k(z_1, z_2) & \dots & k(z_1, z_n) \\ k(z_2, z_1) & k(z_2, z_2) & \dots & k(z_2, z_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(z_n, z_1) & k(z_n, z_2) & \dots & k(z_n, z_n) \end{bmatrix} \tag{3}$$

$k(.,.)$ refers to the kernel function. Ning[11]have compared different kernel choices and their impact on GPR results. For this work, we select the squared exponential kernel function, which can represent data features in infinite dimensions.

1.2 Properties of the Squared Exponential Kernel Function

The squared exponential kernel function can be expressed as follows:

$$k(z_i, z_j) = \sigma_f^2 \, exp\left(-\frac{1}{2}(z_i - z_j)^T M (z_i - z_j)\right) + \sigma_n^2 \tag{4}$$

Where, σ_f represents the variance of the signal and can be used to measure the degree of output variation in the model. M denotes the length scale of the Gaussian kernel, measuring the impact of input variables on the model output. σ_n represents the covariance of the noise, quantifying the prediction error of the model. The choice of hyperparameters significantly affects the training results[12]. These hyperparameters can be updated between specific steps, as detailed in[13].

The posterior distribution of the function at test points also follows the original Gaussian distribution[14].

$$\begin{bmatrix} y \\ y_* \end{bmatrix} \sim N\left(m(z_*), \begin{bmatrix} K(Z, Z) & K(Z, z_*) \\ K(z_*, Z) & k(z_*, z_*) \end{bmatrix}\right) \tag{5}$$

Specifically:

$$\bar{y}_* = k_*^T (K + \sigma_n^2 I)^{-1} \big(y - m(Z)\big) + m(z_*) \tag{6}$$

$$\Sigma(y_*) = k(z_*, z_*) - k_*^T (K + \sigma_n^2 I)^{-1} k_* \quad (7)$$

Where K is the Gram matrix, $K = K(\boldsymbol{Z}, \boldsymbol{Z})$, $\boldsymbol{k}_*$ represents the covariance function, indicating the covariance between new observed data and the original training data, $\boldsymbol{k}_* = K(\boldsymbol{z}_*, \boldsymbol{Z})$. The mean $\bar{\boldsymbol{y}}_*$ is effectively a linear function with respect to the prior dataset Y. $k(\boldsymbol{z}_*, \boldsymbol{z}_*)$ is the first part of the covariance term $\boldsymbol{\Sigma}(\boldsymbol{y}_*)$, which represents the uncertainty of the newly observed data. Subtracting the subsequent term accounts for the reduction in function distribution uncertainty based on the prior model. If the second term is close to zero, it indicates that our uncertainty remains nearly unchanged after observing the data. Conversely, if the second term is large, it implies a significant reduction in uncertainty.

1.3 Simplification Using Cholesky Decomposition

When computing $(K + \sigma_n^2 I)^{-1}$, matrix inversion is involved, which can significantly increase computational complexity and may even lead to numerical instability.

Proof of positive definiteness: The Gram matrix K is generated using the squared exponential kernel function, making it at least a positive semidefinite matrix. During matrix data input, a selection process is applied (see Section 3.3), and since the input data varies, K is positive definite. Additionally, $\sigma_n^2 I$ is also positive semidefinite, making $(K + \sigma_n^2 I)$ positive definite.

For the positive definite matrix $(K + \sigma_n^2 I)$, we can express it as the product of two lower triangular matrices:

$$(K + \sigma_n^2 I) = LL^T \quad (8)$$

Define the transition matrix:

$$\beta = L^{-T} L^{-1} (\bar{y}_* - m(Z)) \quad (9)$$

The computation proceeds by first solving $Lb = (\bar{\boldsymbol{y}}_* - m(\boldsymbol{Z}))$ to obtain b, and then solving $L^T \beta = b$ to find β. We can rewrite Equation (6) as follows:

$$\bar{y}_* = k_*^T \beta + m(z_*) \quad (10)$$

Similarly, define the matrix γ:

$$\gamma = L^{-1} k_* \quad (11)$$

Finally, we can express Equation (7) as:

$$\Sigma(y_*) = k(z_*, z_*) - \gamma^T \gamma \quad (12)$$

By introducing Cholesky decomposition, we can reduce computational complexity and enhance numerical stability.

2. GP-MPC Construction and Stability Proof

2.1 Nonlinear Model Predictive Control

2.1.1 Two-Degree-of-Freedom Bicycle Model

First, let's simplify and model the vehicle dynamics. In this study, we consider a four-wheel-drive vehicle and simplify the nominal model by adopting the classical two-degree-of-freedom bicycle model (as shown in Fig1. The state variables of the simplified model are $[X, Y, \varphi, v_x, v_Y, \omega]$, and the control inputs are $[\delta, T]$.

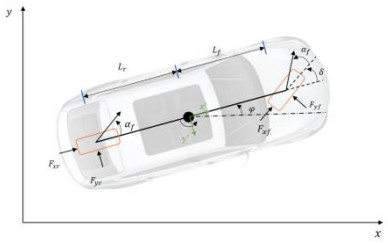


Fig 1 Two-Degree-of-Freedom Bicycle Model

$$\begin{cases} \dot{X} = v_x \cos(\varphi) - v_y \sin(\varphi) \\ \dot{Y} = v_x \sin(\varphi) - v_y \cos(\varphi) \\ \dot{\varphi} = \omega \\ \dot{v_x} = \frac{1}{m}\left(F_{r,x} + F_{f,x} cos(\delta) - F_{f,y} sin(\delta) + m v_y \omega\right) \\ \dot{v_y} = \frac{1}{m}(F_{r,y} + F_{f,x} sin(\delta) + F_{f,y} cos(\delta) - m v_x \omega) \\ \dot{\omega} = \frac{1}{I}\left(F_{f,y} cos(\delta) L_f + F_{f,x} sin(\delta) L_f - F_{r,y} L_r\right) \end{cases} \quad (13)$$

where X and Y represent the position of the vehicle's center of mass in the world coordinate system, v_x denotes the longitudinal velocity of the vehicle in the vehicle coordinate system, v_y represents the lateral velocity of the vehicle in the vehicle coordinate system, φ corresponds to the yaw angle of the vehicle, ω represents the yaw rate of the vehicle, δ is the front wheel steering angle, $F_{f,x}$ and $F_{r,x}$ denote the driving forces provided by the front and rear wheels, respectively, $F_{f,y}$ and $F_{r,y}$ represent the tire forces at the front and rear wheels, L_f and L_r are the distances from the vehicle's center of mass to the front and rear axles, respectively.

Given the use of a simple linear tire model, the tire forces can be expressed as the product of slip angles and stiffness:

$$F_{f,y} = C_f \alpha_f \quad (14)$$

$$F_{r,y} = C_r \alpha_r \tag{15}$$

The calculation of slip angles is as follows:

$$\alpha_r = atan\frac{\omega L_r - v_y}{v_x} \tag{16}$$

$$\alpha_f = -atan\frac{\omega L_f + v_y}{v_x} + \delta \tag{17}$$

2.1.2 NMPC Problem Formulation

MPC leverages prior knowledge to construct a system model, predict future states within a finite time horizon, design an optimization problem, find an optimal control sequence, and apply the first control input to the system to achieve optimal control objectives[15].

Compared to traditional PID control methods, NMPC allows for state constraints. When using NMPC, the following three steps are typically repeated:

1. Obtain the current system state.

2. Perform open-loop prediction on the system and solve the optimization problem to obtain the optimal control sequence.

3. Apply the first element of the computed optimal control sequence to the system.

Using the nominal model from Section 2.1.1, we can formulate the classical NMPC problem as follows:

$$\min_{x,u} x_{0:N}, u_{0:N-1} = \sum_{k=0}^{k=N-1} f_0(x_k, u_k) + \phi(x_N)$$

subject to given x_0, u_{i-1},

$$x_{k+1} = f_{nom}(x_k, u_k) \tag{18a}$$

$$x_k \in \chi \tag{18b}$$

$$u_k \in \mathcal{U} \tag{18c}$$

$$x_N \in \chi_f \tag{18d}$$

Where, $f_0(x_k, u_k)$ represents the stage cost function, $\phi(x_N)$ denotes the terminal cost function, χ and $\mathcal{U}$ are the constraint sets for state and control variables, respectively, χ_f represents the terminal constraint. The system state is subject to the following constraints: Vehicle speed constraint: [0m/s, 20m/s], Front wheel steering angle constraint: [-20°， 20°], Brake and acceleration constraints: [-1， 1], To avoid simultaneous braking and acceleration, we set $T_{brake} \cdot T_{acc} = 0$.

2.2 GP-MPC Problem Formulation

Traditional NMPC uses the nominal model constructed based on Equation (15) for prediction. However, due to environmental uncertainties and model inaccuracies, the deviation between predicted results and actual outcomes increases as the prediction horizon lengthens, as shown in Fig 2.

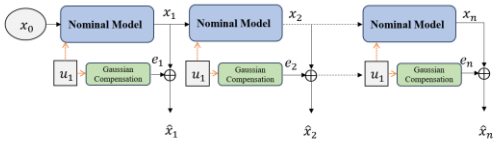


Fig 2 Nominal model and GP model prediction process

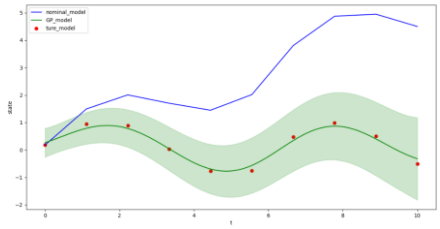


Fig 3 Schematic diagram of real situation, nominal model prediction, and GP model prediction. (Where, blue represents the prediction result of the nominal model, the red dot represents the true state of the system at the corresponding step size, the green line represents the mean predicted by the GP model, and the green area represents the uncertainty of the prediction.)

The accuracy of system predictions directly influences control performance, making an accurate model crucial. Typically, the model established based on dynamics is referred to as the nominal model f_{nom}, as depicted in Fig1. However, due to modeling limitations, there exist unmodeled parts between the nominal model and the true system. GPR can be employed to train these unmodeled components, resulting in a model that better approximates the real system. During training, to reduce data volume and enhance computational efficiency, we select relevant state and control variables that impact the output. The augmented model can be expressed as follows:

$$f_{true} = f_{nom}(x_k, u_k) + B_d(d_{GP}(z_k) + w) \tag{19}$$

Where, B_d represents the compensatory state selection matrix, which selects the state variables affected by errors. $z_k = [B_{zx}\hat{x}_k; B_{zu}u_k]$ constitutes the input data for training. It includes the partially selected measured state variables $B_{zx}\hat{x}_k$ and control variables $B_{zu}u_k$. In this study, we choose $z_k = [v_x, v_y, \omega, \delta, T]$ as the input state vector for the dictionary. And w represents

process noise, following a Gaussian distribution with $w \sim N(0, I\sigma_n^2)$.

When modeling the residual model of the system using Gaussian process regression, uncertainty is introduced. In multi-step MPC prediction, if the input states contain uncertainty, solving the optimization problem becomes challenging. Additionally, computing the posterior of the Gaussian process with uncertain inputs is complex, and the resulting distribution is no longer Gaussian[16]. To address this, we treat the mean computed by the Gaussian model as a deterministic input. In multi-step prediction, we assume that both the state and Gaussian terms follow Gaussian distributions at each step[9].

$$\begin{bmatrix} x_k^T \\ d_k^T \end{bmatrix} \sim N(\mu_k^x, \Sigma_k) = N\left(\begin{bmatrix} \mu_k^x \\ \mu_k^d \end{bmatrix}, \begin{bmatrix} \Sigma_k^x & \Sigma_k^{xd} \\ \Sigma_k^{dx} & \Sigma_k^d + \Sigma_k^w \end{bmatrix}\right) \quad (20)$$

For calculating prediction values, we can use the Unscented Transform method[17]. Drawing inspiration from Extended Kalman Filtering (EKF), we linearize the prediction mean and corresponding covariance using first-order Taylor approximation:

$$\mu_{k+1}^x = f_{norm}(\mu_k^x, u_k) + B_d \mu_k^d(\mu_k^z) \quad (21)$$

$$\Sigma_{k+1}^x = \left[\nabla_{x_k} f_{norm}, B_d\right] \Sigma_k \left[\nabla_{x_k} f_{norm}, B_d\right]^T \quad (22)$$

Where, $\nabla_{x_k} f_{norm}(\mu_k^x, u_k)$ represents the gradient of the nominal model at the operating point. Σ_k is the covariance matrix at time step k. μ_k^x denotes the predicted mean at time step (k). μ_k^z represents the mean of the selected input variables at time step (k).

For the covariance matrix Σ_k^d related to the Gaussian process, we update it using Taylor approximation:

$$\mu_k^d = \mu^d(\mu_k^x, u_k) \quad (23)$$

$$\Sigma_k^{xd} = \Sigma_k^x (\nabla_{x_k} \mu^d(\mu_k^x, u_k))^T \quad (24)$$

$$\Sigma_k^d = \Sigma_k(\mu_k^x, u_k) \nabla_{x_k} + \mu^d(\mu_k^x, u_k) \Sigma_k^x (\nabla_{x_k} \mu^d(\mu_k^x, u_k))^T \quad (25)$$

After introducing the Gaussian model, following the S-MPC approach, we reformulate the problem:

$$\min_{x,u} \; x_{0:N}, u_{0:N-1} = E\left(\sum_{k=0}^{k=N-1} f_0(x_k, u_k) + \phi(x_N)\right)$$

subject to given x_0, u_{i-1},

$$x_{k+1} = f_{nom}(x_k, u_k) \quad (26a)$$

$$u_k = K x_k + v_k \quad (26b)$$

$$P(x_k \in \chi) \geq 1 - \varepsilon_x \quad (26c)$$

$$P(u_k \in \mathcal{U}) \geq 1 - \varepsilon_u \quad (26d)$$

$$P(x_N \in \chi_f) \geq 1 - \varepsilon_x \quad (26e)$$

2.3 Robust Stability Proof for GP-MPC

The stability proof for deterministic MPC cannot be directly extended to Stochastic MPC (S-MPC). Without appropriate conditions, the control law obtained from the optimal control problem with a finite time horizon N (as in Equation (26)) is neither optimal nor optimally stable[18]. Therefore, this paper provides a detailed proof of the system's robust stability. For the system, we know the following:

1) Under mean estimation, noise with zero mean and a normal distribution does not affect the predicted mean of the model.

2) The residual model established by Gaussian process regression compensates for the nominal system and is bounded.

3) An MPC controller relying solely on the nominal model is asymptotically stable[19].

Definitions:

$\mathcal{K}$-function: A function is a K-function if it is continuous, strictly increasing, and maps from $\mathbb{R}_+ \to \mathbb{R}_+$, with a value of 0 at the input 0.

$\mathcal{KL}$-function: A function $\theta(x, y)$ which satisfies:1) maps from $\mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}_+$; 2) For $\forall x_{const} \geq 0$, $\theta(\cdot, y)$ is a $\mathcal{K}$-function; 3) For $\forall y_{const} \geq 0$, $y \to \infty$ $\theta(x, \cdot)$ decreases, and $\theta(x, y) \to 0$.

Asymptotic Stability: A system $x_{n+1} = h(x_n)$ is asymptotically stable if there exists a $\mathcal{KL}$-function θ such that for all $\forall x_i \in \chi$ with $i \geq 0$, $\|x_{n+i} - x_s\| \leq \theta(\|x_{n+i} - x_s\|, i)$.

Robust Asymptotic Stability: A system $x_{n+1} = h(x_n)$ is robust asymptotically stable if there exists a $\mathcal{KL}$-function θ such that for all disturbances d with different upper bounds, $max_i \|d_{n+i}\| \leq \xi$, and for $\forall x_i \in \chi$, $\|x_{n+i} - x_s\| \leq \theta(\|x_{n+i} - x_s\|, i)$.

Lemma. If the original closed-loop system is asymptotically stable and we introduce the residual model $\mathcal{L}(x_i, u_i)$, such that there exists $x_i \in \chi$ with $i \geq 0$, $\xi \leq max_i \mathcal{L}(x_i, u_i) \leq \tilde{\xi}$, then the new closed-loop system with perturbations in the form of $\mathcal{L}$ is robust asymptotically stable.

Problem Transformation. When we disregard the Gaussian residual model (i.e., $\mathcal{L}(x_i, u_i) \equiv 0$), the main model is asymptotically stable (i.e., deterministic MPC is stable). If we view the Gaussian model as bounded

perturbations to the original deterministic system, with a new upper bound greater than the original one, the closed-loop system becomes robust asymptotically stable. Thus, the problem can be reformulated: After introducing the Gaussian model, there exists a new upper bound $\tilde{\xi} > \xi$ such that the Gaussian residual model satisfies $\xi \leq max_i \|\mathcal{L}(x_i, u_i)\| \leq \tilde{\xi}$.

Proof. When we exclude the Gaussian process regression modeling part, the optimal solution $\mathcal{Z}_n^*$ that ensures asymptotic stability can be obtained from the nominal model. When considering Gaussian process regression, the optimal solution becomes $\hat{\mathcal{Z}}_n^*$. Let the corresponding system state difference be e_n:

$$\hat{x}_{n+1}[\hat{\mathcal{Z}}_n^*] = x_{n+1}[\mathcal{Z}_n^*] + e_n \tag{27}$$

Here, e_n includes differences due to varying control commands and model discrepancies:

$$e_n = \frac{\partial f}{\partial u}\left(u_n[\hat{\mathcal{Z}}_n^*] - u_n[\mathcal{Z}_n^*]\right) + GP_n \tag{28}$$

For system (27), if there exists $\tilde{\xi}$ such that $max_i \|e_{n+i}\| \leq \tilde{\xi}$, and for all $i \geq 0$, $\|x_{n+i} - x_s\| \leq \theta(\|x_{n+i} - x_s\|, i)$, then the system is stable. The further transformation is:

When $max_i \|\mathcal{L}(x_i, u_i)\| \geq \xi$:1) There exists $\tilde{\xi}$ such that e_n satisfies $max_i \|e_{n+i}\| \leq \tilde{\xi}$).

For $\forall i \geq 0$, $\|x_{n+i} - x_s\| \leq \theta(\|x_{n+i} - x_s\|, i)$.

Since the relationship between $\mathcal{L}(x_i, u_i)$ and the optimal solution $\hat{\mathcal{Z}}_n^*$ is continuous, when $max_i \|\mathcal{L}(x_i, u_i)\| < \xi_1$, we have $\left\|\frac{\partial f}{\partial u}\left(u_n[\hat{\mathcal{Z}}_n^*] - u_n[\mathcal{Z}_n^*]\right)\right\| < \xi_2$. Simultaneously, we satisfy $\|e_n\| \leq \left\|\frac{\partial f}{\partial u}\left(u_n[\hat{\mathcal{Z}}_n^*] - u_n[\mathcal{Z}_n^*]\right)\right\| + \|GP_n\| \leq \xi_2 + \xi$.

Therefore, there exists $\tilde{\xi} > \xi_2 + \xi$ that satisfies condition (1). When $\xi_2 > \xi$, we have $lim_{i\to\infty} \|e_{n+i}\| = 0$, satisfying condition (2). Thus, when $max \|\mathcal{L}(x_i, u_i)\| \geq \xi$, there exists $\tilde{\xi} > 2\xi$ such that the closed-loop system, considering the Gaussian residual model, is robust asymptotically stable.

3 Optimization and Algorithm Framework of GP-MPC

3.1 Use Covariance to Tighten Road Boundaries

In addressing the S-MPC problem mentioned in Section 2.3, we impose chance constraints on the state. When utilizing GPR for state prediction, this paper employs a mean simplification approach, representing the results using means.

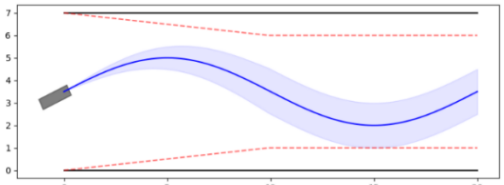


Fig 4 Road boundaries tightened. (Among them, black represents the actual road boundary, blue represents the predicted trajectory with GP compensation, and the red dotted line represents the road constraints tightened using covariance.)

To ensure that the vehicle operates within the trajectory's constraint range, a method similar to the one referenced in[12] is adopted. Specifically, we tighten the constraints using the covariance obtained from Gaussian process regression:

$$R_{GP}(\Sigma_k^{XY}) = \sqrt{\chi^2(p)\lambda_{max}(\Sigma_k^{XY})} \tag{29}$$

where, Σ_k^{XY} is a submatrix of Σ_k^x , $\lambda_{max}(\Sigma_k^{XY})$ represents the maximum eigenvalue of the covariance matrix, and $\chi^2(p)$ corresponds to the maximum violation probability for a chi-squared distribution with degrees of freedom p. The constraints before tightening are denoted as R_i and the tightened constraints can be expressed as:

$$\begin{aligned}\widetilde{R}_\iota(\Sigma_k^{XY}) &= R_i - R_{GP}(\Sigma_k^{XY}) \\ &= R_i - \sqrt{\chi^2(p)\lambda_{max}(\Sigma_k^{XY})}\end{aligned} \tag{30}$$

In practical computations, we set $\chi^2(p)$ to 1. For simplicity, this paper only tightens constraints for the first 10 prediction steps, as shown in Fig4. In the remaining prediction range, we apply constraints to the mean, similar to the approach used in [20].

3.2 Data dictionary Update

The purpose of dictionary updates is to utilize a limited set of data points from a continuous data stream to cover the feature space as comprehensively as possible. Even with a highly accurate model, it remains challenging to reflect the operational conditions during the dynamic processes of a vehicle. For instance, in the case of freight vehicles, their mass undergoes significant changes between empty and full loads, resulting in corresponding variations in the model's

feature space. Considering this, we need to update the data for the GP model, allowing the controller to account for the vehicle's dynamic variations.

The most direct approach for data updates is to input real-time data collected from the vehicle system. However, this can lead to a substantial dataset, making real-time computations difficult.

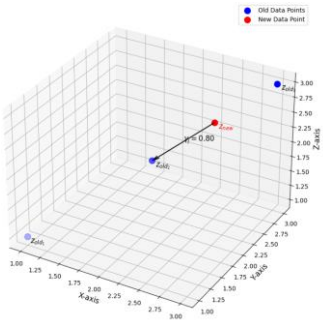


Fig 5 Schematic diagram of independence measure

Because the system data is continuously input, if not filtered during input, it will quickly reach the maximum capacity of the dataset. This may cause difficulties for the GP model to reflect system features. Therefore, data filtering is necessary[21]. We evaluate each input state point, denoted as d_{new} , using the independence metric parameter γ_{new}. After adding new input points, the covariance matrix is as follows:

$$k_{new} = \begin{bmatrix} k(Z_{old}, Z_{old}) & k(Z_{old}, z_j) \\ k^T(Z_{old}, z_j) & k(z_{new}, z_{new}) \end{bmatrix} \quad (31)$$

The independence measure for inputting new data can be expressed as:

$$\gamma_j = k(z_{new}, z_{new}) - k(z_{new}, Z_{old})(k(Z_{old}, Z_{old}) + I\lambda)k^T(z_{new}, Z_{old}) \quad (32)$$

When γ_j is large, it indicates that the data point d_{new} is relatively independent from the old data dictionary Z_{old}. In other words, the original data points do not adequately cover the system features at the current input point. Therefore, d_{new} needs to be added to the data dictionary, as shown in Fig 5.

During the data dictionary update process, we not only consider the independence metric parameter γ but also pay attention to the influence of time. We tend to assign higher trust to data points near the current time step. Thus, we redefine the independence metric as:

$$\gamma'_j = exp\left(-\frac{(n-n_i)^2}{2h}\right)\gamma_j \quad (33)$$

where, n represents the current time step, n_i corresponds to the time step of each data point, and h is a decay coefficient used to adjust the impact of time on the new independence metric.

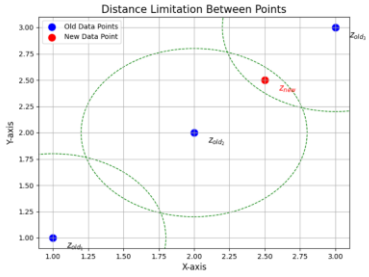


Fig 6 Schematic diagram of sparse sampling

In addition to independence measurement, we also need to limit the distance between points. As shown in Fig 6, the red points do not meet the requirements for coefficient-based sampling and are not added to the data dictionary. This sparse sampling approach aims to achieve efficient sampling[22].

$$z_i - z_j \leq \eta_d \quad (34)$$

Where η_d is the minimum distance limit between data points.

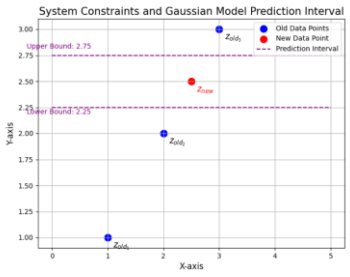


Fig 7 Schematic diagram of outlier constrain

Furthermore, we must consider the impact of outliers on the Gaussian model. For example, in certain states, the system may exceed constraint boundaries, which can interfere with the accuracy of Gaussian model predictions. Therefore, we need to restrict input points accordingly. Only points that satisfy system constraints can be added to the dictionary. Additionally, the current input point should not be too far from the predicted point of the original Gaussian process model and additive model in the previous step. Specifically:

$$m(z_{new}) - \alpha var(z_{new}) \leq y_{now} \leq m(z_{new}) + \alpha var(z_{new}) \quad (35)$$

where, $m(z_{new})$ represents the current model's mean prediction for the current state, $\boldsymbol{var}(z_{new})$ is the corresponding covariance, y_{now} is the actual system

state, and α is a tunable parameter that determines how far input data points can be from the current system's mixed model prediction. Fig7 illustrates this concept, where the purple dashed line represents the upper bound for the next sampling. If it exceeds this boundary, it is not sampled.

After the data exceeds the maximum capacity it can accommodate, it is necessary to remove the data point with the smallest independence metric value and input new data points. Once the new data points are added, the covariance matrix of the model is recalculated.

3.3 Algorithm Overview

To achieve real-time performance, the selection of original data points and optimization of corresponding hyperparameters are performed offline. In Fig8, the gray arrow represents the data processing flow.

The green box represents the offline operations. The specific process involves using an NMPC controller constructed based on a nominal model to control vehicle operation. When adding data to the dictionary, we follow the three principles outlined in Section 3.3.

Additionally, we limit the number of data points to 250. If the data points exceed this limit, we filter them based on their independence relative to the original data, introducing a time decay factor (Equation 35). This approach ensures a tendency to trust newly incoming data. After collecting data, we optimize the hyperparameters $\theta^* = [l_1, \cdots l_n, \sigma_f, \sigma_n]$ by maximizing the marginal log-likelihood $\max\left(p(\boldsymbol{y}|\boldsymbol{Z}, \boldsymbol{\Theta})\right)$.

Once the offline part is completed, we no longer update the hyperparameters of the Gaussian kernel function and only update the data dictionary itself. During prediction, we combine the nominal model with the residual model established using Gaussian process regression for prediction. Notably, the covariance used for constraint tightening is computed at the previous prediction step and corresponds to the current time step. Since the inter-frame changes are relatively small and we consider only the previous 10-time steps, the data from the previous frame still provides practical guidance for the current frame. After completing prediction and constraint updates, we incorporate this information into the model predictive control framework, solve for the optimal control sequence, and apply the first control input to the current frame, enabling rolling optimization.

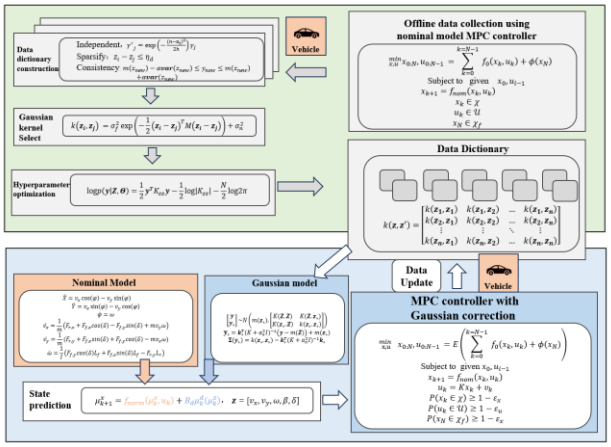


Fig 8 Algorithm Overview

4 Simulation Experiments and Results Analysis

4.1 Simulation Scenarios and Cost Function Design

We implemented the NMPC and GP-MPC algorithms in MATLAB. To better validate the reliability of the proposed algorithm, we designed a continuous curved track as the simulation scenario. The vehicle's objective is to drive as quickly as possible without deviating from the reference path. The true model is based on a two-degree-of-freedom bicycle model with nonlinear tire modeling, while the nominal model uses linear tire modeling. Additionally, we introduce biases to the mass parameter m, moment of inertia I, and the linear model coefficients C_f and C_r relative to their true values. This accounts for uncertainties during actual vehicle operation and modeling inaccuracies.

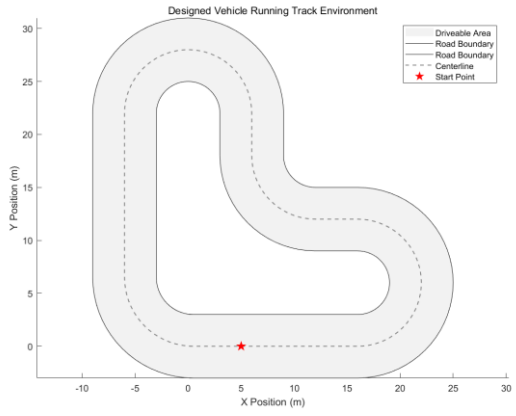


Fig 9 Vehicle tracking trajectory

As the vehicle progresses along the centerline, its state does not strictly follow the reference trajectory. The optimization goal is to drive forward as much as possible within the constraints while minimizing errors. These errors are defined as follows:

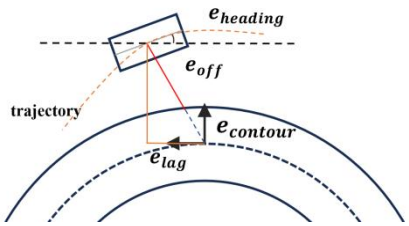


Fig 10 Error diagram

$e_{heading}$: Heading angle error, representing the angle between the vehicle's forward direction and the tangent direction of the reference trajectory.

$e_{offroad}$: Off-road boundary error, indicating the distance between the vehicle's center position and the road boundary.

$e_{contour}$: Contour error, representing the distance between the vehicle's center position and the centerline.

e_{lag}: Lag error, indicating the longitudinal distance between the vehicle's center position and the reference point.

The overall cost can be expressed as:

$$cost = e_{heading} + e_{offroad} + e_{contour} + e_{lag} \quad (36)$$

4.2 Comparative Analysis of Simulation Results

To comprehensively evaluate the effectiveness of the GP-MPC algorithm, we set the total number of simulation laps to 7, allowing us to collect more data.

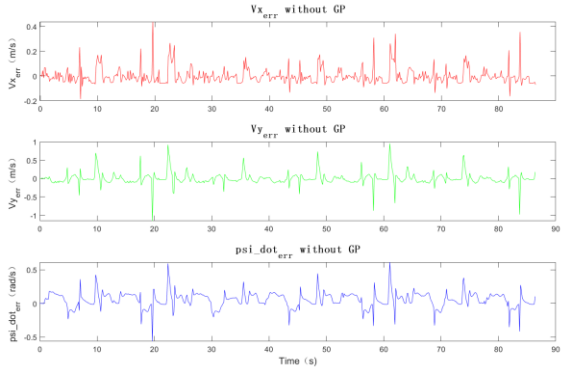


Fig11 Changes in NMPC prediction error over time

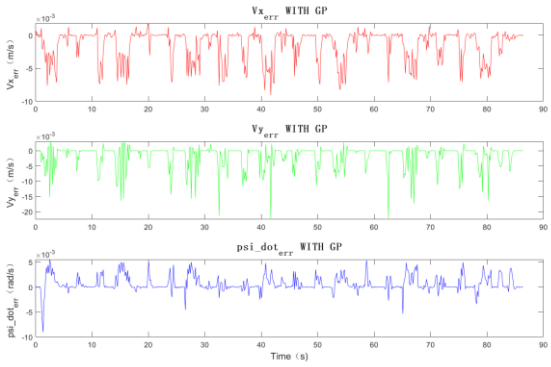


Fig 12 GP-MPC prediction error changes over time

Based on the initial state and control inputs, we predict the system's next-time-step state using both the nominal model and the model enhanced with Gaussian process regression. We assume that the next measurement corresponds to the true system state and calculate the prediction errors.

Comparing Fig11 and Fig12, it is evident that the linear nominal model constructed using dynamics exhibits significant errors when predicting v_x, v_y, and ω. Notably, the prediction error for v_y is substantial, even approaching 1 m/s. In contrast, after introducing GP as a residual model, the compensation applied to the predicted system state significantly reduces the prediction errors.

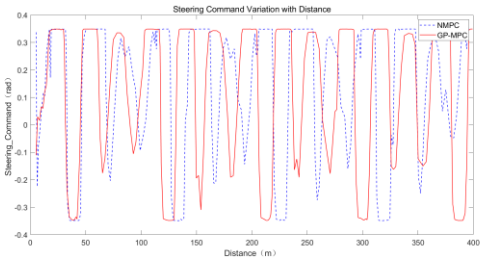


Fig 13 Steering Command Variation with Distance

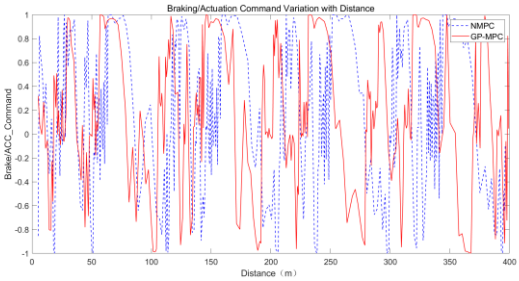
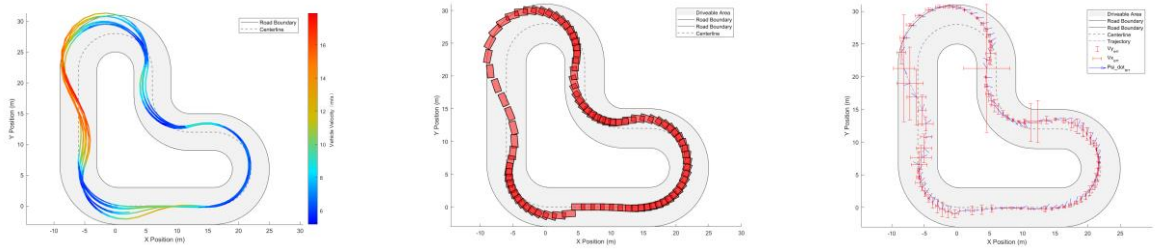


Fig 14 Braking/Actuation Command Variation with Distance

By comparing control outputs, we observe that due to its more accurate predictive model, GP-MPC produces more reasonable control outputs. Regarding changes in

steering angles, the NMPC based on the nominal model exhibits frequent discontinuities within short distances during turns. However, GP-MPC significantly improves the occurrence probability of such behavior. In terms of acceleration and braking changes, both NMPC and GP-MPC exhibit repeated discontinuities, as both lack constraints on torque variations.

The experimental results demonstrate that GP-MPC outperforms traditional NMPC in overall tracking performance. As shown in Figures 16a and 15a, GP-MPC achieves significantly improved trajectory tracking due to its reasonable predictions, maintaining stability overall. In contrast, NMPC exhibits slight oscillations during turns, which manifest as diverging trends in the trajectory.

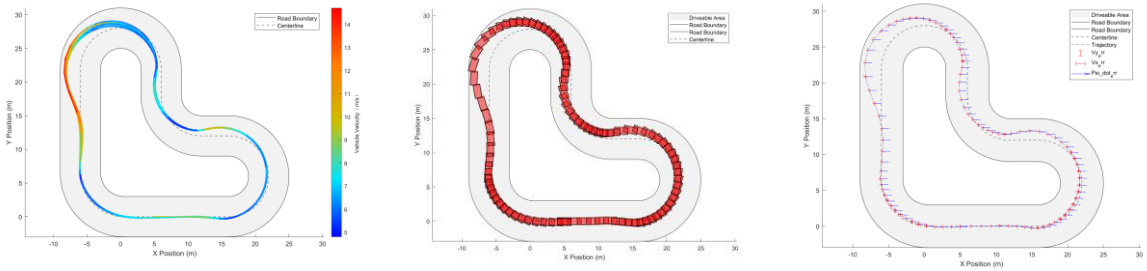


a) Trajectory diagram under NMPC conditions

b) The error at the 5th lap changes with the trajectory

c) Vehicle trajectory on the 5th lap

Fig15 NMPC control effect



a) Trajectory diagram under GP-MPC conditions

b) The error at the 5th lap changes with the trajectory

c) Vehicle trajectory on the 5th lap

Fig 16 GP-MPC control effect

Analyzing the fifth lap trajectory separately, comparing Figures 15b and 16b reveals that under NMPC control, the vehicle deviates noticeably from the road boundary near the coordinates (-5, 27). In contrast, under GP-MPC control, the vehicle strictly stays within the road boundary range. This highlights the effective application of Gaussian processes in enhancing path stability and reasonability. Such improvements are crucial for practical applications, as they can reduce the risk of accidents and enhance vehicle safety and reliability.

Figures 15c and 16c introduce error bars: the horizontal direction represents the one-step prediction error for v_x, the vertical direction represents the one-step prediction error for v_y, and the arrow indicates the prediction error for ω. When the error is 0, the arrow points in the x-direction; as the error increases, it rotates clockwise. Comparative analysis shows that introducing GP significantly reduces all prediction errors. Additionally, an interesting observation is that during the transition zones of entering and exiting curves, the model's prediction error increases significantly, coinciding with abrupt changes in lateral forces.

5 Conclusion

This paper proposed a learning-based model predictive control (MPC) approach for autonomous vehicles. Specifically, the focus is on enhancing existing

Gaussian process (GP) model predictive control to better handle system uncertainties, thereby optimizing path stability and safety during vehicle driving. Additionally, the stability of GP-MPC is analyzed, and robust asymptotic stability of the closed-loop system with Gaussian residual models is proven. In simulation experiments, significant improvements are observed: enhanced prediction accuracy, substantial reduction in prediction errors (within the 10e-2 range), and more stable actual vehicle trajectories that remain within road boundaries. These results highlight the reliability and important application prospects of the proposed approach for achieving precise vehicle control. Future research will further optimize the algorithm and validate it in real-world driving scenarios to ensure reliability and effectiveness in practical applications

Funding

This work was funded by the State Key Laboratory of Mechanical Transmission for Advanced Equipment, Chongqing University (SKLMT-MSKFKT-202221) and the National Nature Science Foundation of China under Grant 52002025, 52302508.

References

[1] Heilmeier A, Wischnewski A, Hermansdorfer L, et al. Minimum curvature trajectory planning and control for an autonomous race car[J]. Vehicle System Dynamics, 2019, 58(8):1-31.

[2] Liniger A, Domahidi A, Morari M. Optimization-based autonomous racing of 1: 43 scale RC cars[J]. Optimal Control Applications and Methods, 2015, 36(5): 628-647.

[3] Mesbah A. Stochastic model predictive control: An overview and perspectives for future research[J]. IEEE Control Systems Magazine, 2016, 36(6): 30-44.

[4] Rawlings J B, Mayne D Q, Diehl M. Model predictive control: theory, computation, and design[M]. Madison, WI: Nob Hill Publishing, 2017.

[5] Norouzi A, Heidarifar H, Borhan H, et al. Integrating machine learning and model predictive control for automotive applications: A review and future directions[J]. Engineering Applications of Artificial Intelligence, 2023, 120: 105878.

[6] Klenske E D, Zeilinger M N, Schölkopf B, et al. Gaussian process-based predictive control for periodic error correction[J]. IEEE Transactions on Control Systems Technology, 2015, 24(1): 110-121.

[7] Carrau J V, Liniger A, Zhang X, et al. Efficient implementation of randomized MPC for miniature race cars[C]//2016 European Control Conference (ECC). IEEE, 2016: 957-962.

[8] Kabzan J, Hewing L, Liniger A, et al. Learning-based model predictive control for autonomous racing[J]. IEEE Robotics and Automation Letters, 2019, 4(4): 3363-3370.

[9] Hewing L, Kabzan J, Zeilinger M N. Cautious model predictive control using gaussian process regression[J]. IEEE Transactions on Control Systems Technology, 2019, 28(6): 2736-2743.

[10] Rasmussen C E, Williams C K I. Gaussian processes for machine learning[M]. Cambridge, MA: MIT press, 2006.

[11] Ning J, Behl M. Vehicle Dynamics Modeling for Autonomous Racing Using Gaussian Processes[J]. arXiv preprint arXiv:2306.03405, 2023.

[12] Langåker H A. Cautious mpc-based control with machine learning[D]. NTNU, 2018.

[13] Hewing L, Arcari E, Fröhlich L P, et al. On simulation and trajectory prediction with gaussian process dynamics[C]//Learning for Dynamics and Control. PMLR, 2020: 424-434.

[14] Williams C K I, Rasmussen C E. Gaussian processes for machine learning[M]. Cambridge, MA: MIT press, 2006.

[15] Bemporad A, Borrelli F, Morari M. Model predictive control based on linear programming~ the explicit solution[J]. IEEE transactions on automatic control, 2002, 47(12): 1974-1985.

[16] Quinonero-Candela J, Girard A, Rasmussen C E. Prediction at an uncertain input for Gaussian processes and relevance vector machines-application to multiple-step ahead time-series forecasting[J]. 2003.

[17] Ostafew C J, Schoellig A P, Barfoot T D. Robust Constrained Learning-based NMPC enabling reliable mobile robot path tracking[J]. The International Journal of Robotics Research, 2016, 35(13):1547-1563.

[18] Zhiming Zhang. Vehicle Trajectory Planning and Control Based on Model Predictive Control [D]. Zhejiang University, 2022.

[19] Falcone P F H J D. Linear time-varying model predictive control and its application to active steering systems: Stability analysis and experimental validation[J]. International Journal of Robust and Nonlinear Control, 2008, 18(8): 862 - 875.

[20] Jesús Velasco Carrau, Liniger A, Zhang X, et al. Efficient Implementation of Randomized MPC for Miniature Race Cars[C]//2016 European Control Conference (ECC). IEEE, 2017.

[21] Nguyen-Tuong D, Peters J. Incremental online sparsification for model learning in real-time robot control[J]. Neurocomputing, 2011, 74(11):1859-1867.

[22] C. Jiang, Y. Ding, Z. Li and C. Sun. A Fast Kernel-Based Optimal Control Framework for Autonomous Driving. in IEEE Transactions on Control Systems Technology, 2023, 31(3):1296-1307