Control Engineering Practice 101 (2020) 104482

Contents lists available at ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac

Ultra-local model predictive control: A model-free approach and its application on automated vehicle trajectory tracking

Zejiang Wang, Junmin Wang *

Walker Department of Mechanical Engineering, The University of Texas at Austin, Austin, TX 78712, United States

ARTICLE INFO

Keywords:
Automated vehicle
Model-free control
Predictive control
Trajectory following
Ultra-local model

ABSTRACT

Model predictive control (MPC) has been extensively utilized in the automotive applications, such as autonomous vehicle path planning and control, hybrid-vehicle energy management, and advanced driver-assistance system design. As a typical model-based control law, MPC relies on a system model to predict the state evolution of the manipulated plant within the prediction horizon. However, a representative yet concise mathematical description of the controlled plant may not always be available in practice. Therefore, model-free strategies, e.g., identification for control and direct data-driven control, have been incorporated into the predictive control framework. Nonetheless, existing model-free predictive controllers usually require reliable datasets and employ complex nonconvex optimizations to identify the underlying system model. Furthermore, their control performances are fundamentally limited by the quality of the training data. Inspired by the model-free control, this paper proposes the ultra-local model predictive control (ULMPC), which is a novel and straightforward model-free predictive control technique with no need for the computationally-extensive model learning process. The proposed ULMPC is implemented for automated vehicle trajectory following. Carsim-Simulink joint simulations and indoor experimental field tests with a scaled car demonstrate its effectiveness.

1. Introduction

Model predictive control (MPC), as a modern control technique capable of systematically handling plant constraints, has flourished in autonomous vehicle path-planning and control (Brown, Funke, Erlien, & Gerdes, 2017), energy management and optimization (Weißmann, Görges, & Lin, 2018), and advanced driver-assistance system development (Ercan, Carvalho, Tseng, Gökaşan, & Borrelli, 2018). The standard procedure for formulating an MPC problem begins with a performance-oriented cost function, which penalizes both the state tracking errors and the amounts of the control inputs within the prediction horizon. To this end, a reliable system model is necessary for predicting the plant states in the near future. Therefore, the system model is the core of an MPC (Huusom, Poulsen, Jørgensen, & Jørgensen, 2012; Ławryńczuk, 2009).

Unfortunately, an accurate yet concise mathematical model of the manipulated plant may not always be accessible due to intellectual property protection, safety issues, etc. Under such circumstances, the system model needs to be identified first. Typical identified models for predictive control include auto-regression with exogenous input (ARX) model (Huusom et al., 2012), neural network (Ławryńczuk, 2009), and Fuzzy-logic model (Dovžan & Škrjanc, 2010). However, an effective model identification requires sufficiently rich and representative historical data (Hu, Chen, Wang, Chen, & Ren, 2018) to meet the persistent excitation requirement, and such an input–output dataset may not be always available, especially when the controlled plant is recently built. Besides, the model learning process may necessitate a great deal of time. For instance, ARX models with different orders (Huusom et al., 2012) and neural networks with various topologies (Piche, Sayyar-Rodsari, Johnson, & Gerules, 2000) need to be tested and compared to determine the most appropriate model structure. In addition, the obtained model might be nonlinear and nonconvex, which hinders the online implementation of MPC (Forbes, Patwardhan, Hamadah, & Gopaluni, 2015). Furthermore, if the actual system states deviate far from the referential trajectories contained in the training dataset, the MPC performance will be inevitably deteriorated (Novara, Formentin, Savaresi, & Milanese, 2016). To ease the laborious model identification procedure, a direct data-driven MPC has been recently proposed in Piga, Formentin, and Bemporad (2017). This approach has a hierarchical control structure: a low-level controller converts the unknown plant into a predefined linear model, to which a high-level command governor (Garone, Di Cairano, & Kolmanovsky, 2017) is applied. Like the case of ARX-based MPC, the low-level controller in Piga et al. (2017) is continuously trained online through periodically updated

* Corresponding author.
E-mail addresses: wangzejiang@utexas.edu (Z. Wang), jwang@austin.utexas.edu (J. Wang).

https://doi.org/10.1016/j.conengprac.2020.104482
Received 11 November 2019; Received in revised form 18 March 2020; Accepted 26 May 2020
Available online 1 June 2020

system input and output data. However, nonconvex optimizations, e.g., particle swarm optimization (Selvi, Piga, & Bemporad, 2018) and Bayesian optimization (Piga, Forgione, Formentin, & Bemporad, 2019) are involved to determine the parameters of the low-level controller.

In short, the existing model-free predictive controllers rely on either sufficient and high-quality training data or computationally extensive nonconvex optimizations to identify the underlying system model, which may impede their practical utilization. Moreover, the online iterative optimization of MPC, which entails both a remarkable computational burden and a quite large memory footprint, indeed impedes its real-time implementation. To alleviate the MPC computational burden, several approaches have been proposed, such as the explicit MPC (Naus et al., 2008), control-blocking method (Li, Jia, Li, & Cheng, 2014), and interpolation control (Tuchner & Haddad, 2017), etc.

Inspired by the Model-Free Controller (MFC) (Fliess & Join, 2013), we propose in this paper a novel control strategy: Ultra-local Model Predictive Control (ULMPC). ULMPC represents the manipulated plant as an affine system and applies predictive control to this continuously updated linear model for reference tracking and constraint handling. In contrast to the existing model-free predictive control approaches, no training dataset is involved in ULMPC and only small-scale convex optimization is required for control calculation. Additionally, ULMPC improves MFC by reducing the steady-state tracking error. The proposed ULMPC is applied to an automated vehicle trajectory following problem. Carsim-Simulink joint simulations first manifest its effectiveness and field tests with a scaled car further show its ease of use.

The rest of the paper is organized as follows. Ultra-local model and MFC are briefly reviewed in Section 2. Then, the algorithm of ULMPC is summarized in Section 3. After that, an automated vehicle trajectory-following problem is formulated in Section 4, where three controllers: a model-based linear-time-varying predictive controller (LTVMPC), an MFC, and the proposed ULMPC are elaborated and compared. Carsim-Simulink joint simulations demonstrate the advantages of ULMPC. Afterwards, the proposed ULMPC is validated on a scaled car platform in Section 5 to show its implementability in real time. Finally, Section 6 concludes this paper.

2. Ultra-local Model and Model-Free Control

The ultra-local model and the model-free control are initially proposed in Fliess and Join (2013).

2.1. Ultra-local Model

A single-input-single-out (SISO) system with input u and output y can be expressed as an affine form:

$$y^{(\nu)}(\tau) = F(\tau) + \alpha u(\tau). \tag{1}$$

In (1), $y^{(\nu)}(\tau)$ represents the νth order derivative of the system output at time τ. $u(\tau)$ depicts the system input at time τ. The constant input gain α is chosen to make the magnitude orders of $\alpha u(\tau)$ and $y^{(\nu)}(\tau)$ close. $F(\tau)$ condenses both the unmodeled system dynamics and the external disturbances. For control purposes, $F(\tau)$ is regarded as a piecewise constant and needs to be updated at each sampling moment. To determine $F(\tau)$, $y^{(\nu)}(\tau)$ needs to be reconstructed a priori. Directly differentiating system output $y(\tau)$ is ill-posed due to the measurement noises. Instead, the arbitrary-order derivative of a noisy signal is derived from the Algebraic Differentiation Estimation (ADE) (Fliess, Join, & Sira-Ramirez, 2008). ADE approximates the actual signal $y(\tau)$ as its truncated Taylor expansion with order $N \geq \nu$. Then, the truncated Taylor expansion is converted into the operational domain via the Laplace transform. Therein, linear differential operators:
$\Pi_{\kappa,\varpi}^{N,\nu} = \frac{1}{s^{N+\varpi+1}} \frac{d^{n+\kappa}}{ds^{n+\kappa}} \frac{1}{s} \frac{d^{N-\nu}}{ds^{N-\nu}} s^{N+1}, \kappa \in \mathbb{N}, \varpi \in \mathbb{N}$, where s indicates the Laplace variable, can be applied to isolate the coefficient of the νth order term of the truncated Taylor expansion. Note that the iterative integrals from $\frac{1}{s^{N+\varpi+1}}$ will serve as a low-pass filter to wipe out noises in the time domain.

By assigning $N = \nu, \kappa = 0, \varpi = 0$, the zero-, first-, and second-order derivatives of a noisy signal $y(t)$ can be expressed as sliding-window integral formulations (Wang, Bai, Zha, Wang, & Wang, 2019):

$$\begin{cases} \hat{y}(t) = \dfrac{2}{T_{ADE}^2} \displaystyle\int_0^{T_{ADE}} \left(2T_{ADE} - 3\tau\right) y(t-\tau)\, d\tau, \\ \hat{\dot{y}}(t) = \dfrac{6}{T_{ADE}^3} \displaystyle\int_0^{T_{ADE}} \left(T_{ADE} - 2\tau\right) y(t-\tau)\, d\tau, \\ \hat{\ddot{y}}(t) = \dfrac{60}{T_{ADE}^5} \displaystyle\int_0^{T_{ADE}} \left(T_{ADE}^2 - 6T_{ADE}\tau + 6\tau^2\right) y(t-\tau)\, d\tau. \end{cases} \tag{2}$$

In (2), T_{ADE} is the width of the sliding window, in which the system output measurements within the period $\left[t - T_{ADE}, \quad t\right]$ are registered. Normally, T_{ADE} is chosen as an integral multiple of the sampling period T_s, or $T_{ADE} = KT_s, K \in \mathbb{N}^+$. Tuning of T_{ADE} should balance the Taylor expansion truncation error and the noise contribution: as T_{ADE} extends, the noise-filtering effect is reinforced whereas the accumulated truncation error also increases.

A νth order ($\nu > 1$) differentiator can also be created by cascading ν first-order differentiators in (2). This cascading approach leads to improved attenuation of the high-frequency noises (Mboup, Join, & Fliess, 2009). Therefore, we utilize this strategy throughout this paper.

Once $\hat{y}^{(\nu)}(\tau)$ is determined from ADE, the scalar term $F(\tau)$ in (1) can be approximated as:

$$F(\tau) \approx \hat{F}(\tau) = \hat{y}^{(\nu)}(\tau) - \alpha u\left(\tau - T_s\right), \tag{3}$$

where $u\left(\tau - T_s\right)$ indicates the system input at the last step. Consequently, the practical *ultra-local model* at time τ becomes:

$$y^{(\nu)}(\tau) = \hat{F}(\tau) + \alpha u(\tau). \tag{4}$$

2.2. Model-Free Control

To make the system output $y(\tau)$ follow a given reference trajectory $y_r(\tau)$, system input $u(\tau)$ in (4) can be straightforwardly designed as:

$$u(\tau) = \frac{-\hat{F}(\tau) + y_r^{(\nu)}(\tau) + \Omega\left(y(\tau) - y_r(\tau)\right)}{\alpha}. \tag{5}$$

In (5), the tracking error term $\Omega\left(y(\tau) - y_r(\tau)\right)$ is typically chosen as the canonical Brunovsky's feedback to converge $y(\tau)$ towards $y_r(\tau)$. However, derivative estimation from (2) suffers from a tiny time delay due to its sliding window nature, which in turn introduces an inevitable modeling error in (4). As a result, only practical stability instead of asymptotical stability can be achieved with (5). Alternatively speaking, MFC can merely guarantee that the absolute tracking error $\left|y(\tau) - y_r(\tau)\right|$ rests inside a bounded ball around the origin Wang, Mounier, Niculescu, Geamanu, and Cela (2016) with persisting steady-state error. Typical strategies to resolve this issue include adaptive tuning of the input gain α (Polack, d'Andréa-Novel, Fliess, de La Fortelle, & Menhour, 2017) and combination of MFC with model-based control. For instance, authors in Wang, Ye, Tian, Zheng, and Christov (2016) incorporate a sliding-mode controller with MFC for eliminating the terminal attitude tracking error of a quadrotor.

3. Ultra-local Model Predictive Control

Following the idea of hybridizing MFC with a model-based controller, we propose ULMPC by implementing MPC on the ultra-local model (4). As will be shown, ULMPC enjoys the mutual benefits from both MFC and MPC. Firstly, ULMPC inherits the excellent robustness to system uncertainties and disturbances from MFC. Secondly, expressing the manipulated plant as an ultra-local model results in a simplified MPC formulation with a much reduced problem scale. Thirdly, the MPC framework can reduce the steady-state tracking error in MFC.

Table 1
Ultra-local MPC.

Algorithm: Ultra-local Model Predictive Control
At step $\boldsymbol{k}$ do:
Step 1: Calculate $\hat{\boldsymbol{y}}^{(v)}(\boldsymbol{k})$ from (2)
Step 2: Estimate $\hat{\boldsymbol{F}}(\boldsymbol{k}) = \hat{\boldsymbol{y}}^{(v)}(\boldsymbol{k}) - \alpha \boldsymbol{u}_{k-1\|k}$ from (3)
Step 3: Obtain the linear ultra-local model as (4): $\boldsymbol{y}^{(v)}(\boldsymbol{k}) = \hat{\boldsymbol{F}}(\boldsymbol{k}) + \alpha \boldsymbol{u}_{k\|k}$
Step 4: Solve a constrained optimization problem: $\Delta \boldsymbol{u}^*_{k\|k} = \arg\min \boldsymbol{J}(\boldsymbol{k})$
such that $\boldsymbol{y}^{(v)}_{k+i+1\|k} = \hat{\boldsymbol{F}}(\boldsymbol{k}) + \alpha\left(\boldsymbol{u}_{k-1\|k} + \sum_0^i \Delta \boldsymbol{u}^*_{k+i\|k}\right), i = 0, \ldots, \boldsymbol{H}_p$ $\boldsymbol{u}_{k+i\|k} \le \boldsymbol{u}(\boldsymbol{k})_{\max}, \boldsymbol{u}_{k+i\|k} \ge \boldsymbol{u}(\boldsymbol{k})_{\min}, i = 0, \ldots, \boldsymbol{H}_p + 1$ $\Delta \boldsymbol{u}_{k+i\|k} \le \Delta \boldsymbol{u}(\boldsymbol{k})_{\max}, \Delta \boldsymbol{u}_{k+i\|k} \ge \Delta \boldsymbol{u}(\boldsymbol{k})_{\min}, i = 0, \ldots, \boldsymbol{H}_p$ $\Delta \boldsymbol{u}_{k+i\|k} = 0, i = \boldsymbol{H}_c, \ldots \boldsymbol{H}_p$ $\boldsymbol{y}_{k+i\|k} \le \boldsymbol{y}(\boldsymbol{k})_{\max}, \boldsymbol{y}_{k+i\|k} \ge \boldsymbol{y}(\boldsymbol{k})_{\min}, i = 1, \ldots, \boldsymbol{H}_p + 1$
Step 5: Update $\boldsymbol{u}^*_{k\|k} = \Delta \boldsymbol{u}^*_{k\|k} + \boldsymbol{u}_{k-1\|k}$

Similar to the direct data-driven MPC in Piga et al. (2017), the proposed ULMPC also continuously updates the system model at each sampling step. However, thanks to the ADE, the complicated model-learning process is thoroughly removed, which not only relieves the requirement of sufficiently representative system input–output historical data but also ensures that the training process will not affect the control performance of ULMPC. In addition, instead of employing a hierarchical control structure, ULMPC straightforwardly generates the optimal command.

The algorithm of ULMPC is summarized in Table 1. We assume the sampling period between the step $\boldsymbol{k}$ and $\mathbf{k+1}$ is T_s. Concrete examples will be given in Section 4 to further illustrate the formulation of ULMPC.

In Table 1, $\boldsymbol{u}_{k-1|k}$ represents the system command at the last step. $\Delta \boldsymbol{u}^*_{k|k}$ and $\boldsymbol{u}^*_{k|k}$ are respectively the optimal command increment and the optimal command at the current step. The cost function $\boldsymbol{J}(\boldsymbol{k})$ is problem-dependent but in general balances output tracking errors and drastic command fluctuations. The optimized variable corresponds to the command incremental sequence: $\Delta \boldsymbol{u}_{k+i|k} \triangleq \boldsymbol{u}_{k+i|k} - \boldsymbol{u}_{k+i-1|k}, i = 0, \ldots, \boldsymbol{H}_p$. Here, $\boldsymbol{u}_{k+i|k}$ indicates the future system input at step $\boldsymbol{k}+\boldsymbol{i}$ predicted at the current step $\boldsymbol{k}$, and $\boldsymbol{H}_p$ is the prediction horizon. Command incremental instead of command itself is chosen as the optimized variable in order to include integral action into ULMPC for offset-free tracking (Di Ruscio, 2013). To further mitigate the MPC computational load, we enforce $\Delta \boldsymbol{u}_{k+i|k} = 0, i = \boldsymbol{H}_c, \ldots \boldsymbol{H}_p$ where $\boldsymbol{H}_c$ is the control horizon. System constraints can cover input magnitude thresholds: $\boldsymbol{u}(\boldsymbol{k})_{\max}, \boldsymbol{u}(\boldsymbol{k})_{\min}$, input rate-of-change limits: $\Delta \boldsymbol{u}(\boldsymbol{k})_{\max}, \Delta \boldsymbol{u}(\boldsymbol{k})_{\min}$, and system output bounds: $\boldsymbol{y}(\boldsymbol{k})_{\max}, \boldsymbol{y}(\boldsymbol{k})_{\min}$.The cost function $\boldsymbol{J}(\boldsymbol{k})$ has in general the form $\boldsymbol{J}(\boldsymbol{k}) = \sum_{i=1}^{H_p} \left\| \boldsymbol{x}_{k+i|k} - \boldsymbol{x}^{\boldsymbol{r}}_{k+i|k} \right\|^2_{\boldsymbol{Q}} + \sum_{i=0}^{H_c-1} \left\| \boldsymbol{u}_{k+i|k} \right\|^2_{\boldsymbol{R}}$, where $\boldsymbol{x}^{\boldsymbol{r}}_{k+i|k}$ is the desired system output at the future step $\boldsymbol{k}+\boldsymbol{i}$, and $\boldsymbol{Q}, \boldsymbol{R}$ are the weighting matrices with appropriate dimensions. Therefore, $\boldsymbol{J}(\boldsymbol{k})$ can be readily converted into a convex function with respect to the optimized variable $\Delta \boldsymbol{u}_{k+i|k}, i = 0, \ldots, \boldsymbol{H}_p$ (Wang, Bai, Wang, & Wang, 2019), and ULMPC can be then efficiently solved online with a standard quadratic programming (QP) solver.

4. Problem formulation and simulation study

In this Section, the proposed ULMPC is tested under an automated vehicle trajectory following task. To demonstrate the superior performance of ULMPC, an MFC along with a model-based LTVMPC are also provided for comparison. All the simulations are performed on the Carsim-Simulink joint platform.

4.1. Problem formulation

An automated vehicle needs to simultaneously follow an oval path and a velocity profile. System inputs include the steering wheel angle $\delta_{sw}(t)$ and the combined rear-wheel torque $T_r(t)$. Measured outputs are the minimum distance from vehicle's center of gravity (CG) to the desired path centerline $e_y(t)$ and vehicle longitudinal speed $v(t)$.

Table 2
Vehicle configuration.

Item	Value and unit
Longitudinal tire stiffness	66 900 N
Lateral tire stiffness	62 700 N/rad
Distance between CG and front axle	1.232 m
Distance between CG and rear axle	1.468 m
Height of CG	0.54 m
Vehicle total mass	1723 Kg
Track length	1.539 m
Yaw inertia	1960 kgm^2
Wheel radius	0.31 m

During the simulations, both the tire-road friction coefficient (TRFC) μ and the steering ratio $N_s = \delta_f(t)/\delta_{sw}(t)$, where $\delta_f(t)$ represents the front road wheel angle, are designed as functions of the path station s, which is the arc length along the desired path centerline. N_s is fixed at its nominal value $N_s^o = 1/15.176$ at the beginning. Then, it decreases to $75\%N_s^o$ when $s > 278.5$ m and further decreases to $66\%N_s^o$ when $s > 1093$ m. This process mimics a front steering system operating under system faults. As for the TRFC, it is assigned with a nominal value $\mu^o = 0.9$ at the start. Then, TRFC suddenly reduces to $\mu = 0.6$ when $s > 814$ m. Finally, to create a split friction road, the TRFC of the left side of the track turns back to $\mu = 0.9$ when $s > 1350$ m.

Vehicle configurations in Carsim are summarized in Table 2. Nonlinear coupled brush tire model from Wang, Bai, Wang, and Wang (2019) is used for tire force generation.

Note that neither the MFC nor the ULMPC requires access to the vehicle configurations, TRFC, or the steering ratio. On the contrary, the LTVMPC enjoys the full datasheet in Table 2, and LTVMPC utilizes the nominal TRFC and steering ratio during simulations.

4.2. System modeling

According to Menhour, d'Andréa-Novel, Fliess, Gruyer, and Mounier (2017), the ultra-local models of vehicle longitudinal and lateral dynamics can be expressed as:

$$\begin{cases} \dot{v}(t) = f_v(t) + \alpha_v T_r(t), \\ \ddot{e}_y(t) = f_y(t) + \alpha_y \delta_{sw}(t). \end{cases} \tag{6}$$

Although (6) implies that the longitudinal and lateral dynamics have decoupled forms, the inherent tire force coupling effect is indeed maintained in the scalar terms $f_v(t)$ and $f_y(t)$. For both the MFC and the ULMPC, the constant input gains α_v, α_y are tuned offline through a brutal-force approach. A wide range of α_v, α_y were tested offline, where the tested α_v begun from 0.1 and ended at 2.0 with an increment as 0.1, and the tested α_y begun from 18 and ended at 360, with an increment as 18. The finally tuned ultra-local model gains $\alpha_v = 0.18, \alpha_y = 342$ ensure that *MFC's* Pareto front with respect to the path-tracking error and the speed-tracking error has been reached. In other words, the utilized gains make sure that no *simultaneous* improvements on both root mean square values: $RMS(e_y)$ and $RMS(v - v_r)$ of *MFC* can be further realized. In that sense, the tuned gains somehow make the *MFC* achieve an optimal performance. However, as we will see in Section 4.6, the (non-optimally tuned) ULMPC can still evidently outperform the MFC.

4.3. Model-Free Control design

From (6), the combined rear-wheel torque $T_r(t)$ can be designed as:

$$T_r(t) = \frac{-\hat{f}_v(t) + \dot{v}_r(t) + \eta_T\left(v(t) - v_r(t)\right)}{\alpha_v}, \tag{7}$$

where $v_r(t)$ represents the desired speed and $\eta_T < 0$ is the feedback gain. According to (3), we have:

$$\hat{f}_v(t) = \hat{v}(t) - \alpha_v T_r\left(t - T_s\right), \tag{8}$$

with $\hat{v}(t)$ deduced from (2) and $T_r\left(t - T_s\right)$ as the torque command at the last step. Considering the rear-wheel torque constraint, the actual $T_r(t)$ must lay within its upper- and lower-bounds: $T_r^{down}(t) \le T_r(t) \le T_r^{up}(t)$, with:

$$\begin{cases} T_r^{up}(t) = \min\left(T_r^{\max}, \quad T_r\left(t - T_s\right) + \Delta T_r^{\max}\right), \\ T_r^{down}(t) = \max\left(T_r^{\min}, \quad T_r\left(t - T_s\right) - \Delta T_r^{\max}\right). \end{cases} \tag{9}$$

In (9), $T_r^{\max}$ and $T_r^{\min}$ indicate respectively the maximum and the minimum torque outputs and $\Delta T_r^{\max}$ represents the maximal torque increment, which can be expressed as:

$$\Delta T_r^{\max} = \dot{T}_r^{\max} T_s, \tag{10}$$

where $\dot{T}_r^{\max}$ indicates the maximal rear-wheel torque slew rate.

Similar to (7), the steering wheel angle $\delta_{sw}(t)$ can be deduced as:

$$\delta_{sw}(t) = \frac{-\hat{f}_y(t) + \eta_{sw}^1 \dot{e}_y(t) + \eta_{sw}^0 e_y(t)}{\alpha_y}, \tag{11}$$

where $\hat{f}_y(t)$ can be calculated as:

$$\hat{f}_y(t) = \hat{\ddot{e}}_y(t) - \alpha_y \delta_{sw}\left(t - T_s\right). \tag{12}$$

In (12), $\delta_{sw}\left(t - T_s\right)$ indicates the steering wheel at the last step. $\hat{\ddot{e}}_y(t)$ is determined by successively applying twice the first-order ADE differentiators in (2) to the measured output $e_y(t)$. The position tracking error gains η_{sw}^1 and η_{sw}^0 are selected such that the error dynamics matrix: $\begin{bmatrix} 0 & 1 \\ \eta_{sw}^0 & \eta_{sw}^1 \end{bmatrix}$ is Hurwitz.

Akin to (9), the upper- and lower-bounds of $\delta_{sw}(t)$ are determined as:

$$\begin{cases} \delta_{sw}^{up}(t) = \min\left(\delta_{sw}^{\max}, \quad \delta_{sw}\left(t - T_s\right) + \Delta\delta_{sw}^{\max}\right), \\ \delta_{sw}^{down}(t) = \max\left(\delta_{sw}^{\min}, \quad \delta_{sw}\left(t - T_s\right) - \Delta\delta_{sw}^{\max}\right). \end{cases} \tag{13}$$

In (13), $\delta_{sw}^{\max}$ and $\delta_{sw}^{\min}$ indicate respectively the maximal and the minimal steering wheel angle, and $\Delta\delta_{sw}^{\max}$ is the maximal steering wheel increment:

$$\Delta\delta_{sw}^{\max} = \dot{\delta}_{sw}^{\max} T_s, \tag{14}$$

where $\dot{\delta}_{sw}^{\max}$ is the maximal steering wheel angular velocity.

4.4. Ultra-local Model Predictive Control design

By substituting (8) into (6), the ultra-local model of vehicle longitudinal dynamics at step k can be expressed as:

$$\begin{aligned} \dot{v}(k) &\approx \hat{f}_v(k) + \alpha_v\left(T_r(k-1) + \Delta T_r(k)\right) \\ &= \hat{v}(k) - \alpha_v T_r(k-1) + \alpha_v\left(T_r(k-1) + \Delta T_r(k)\right) \\ &= \hat{v}(k) + \alpha_v \Delta T_r(k). \end{aligned} \tag{15}$$

In (15), the unique system input is $u^v(k) = \Delta T_r(k)$ and the sole system state and output is $x^v(k) = v(k)$.

Discretizing (15) with the sampling period T_s gives us:

$$x_{k+1|k}^v = x_{k|k}^v + \alpha_v T_s u_{k|k}^v + T_s \hat{v}(k). \tag{16}$$

In (16), $x_{k|k}^v$ indicates the currently measured vehicle longitudinal speed. $x_{k+1|k}^v$ is the one-step predicted speed. $\hat{v}(k)$ represents the estimated first-order derivative of vehicle longitudinal speed from ADE, and $u_{k|k}^v$ is the system input (torque increment) at the current step. Based on (16), the cost function for vehicle longitudinal control can be formulated as:

$$J^v = \sum_{i=1}^{H_p} \left\| x_{k+i|k}^v - x_{k+i|k}^{vr} \right\|_{Q_v}^2 + \sum_{i=0}^{H_c-1} \left\| u_{k+i|k}^v \right\|_{R_v}^2. \tag{17}$$

As illustrated in Table 1, H_p and H_c represent individually the prediction horizon and the control horizon. Additionally, $x_{k+i|k}^{vr}, i = 1, \ldots, H_p$ indicates the desired velocity profile within the prediction horizon. In (17), the first item aims to minimize the velocity tracking error while the second term penalizes the drastic torque variation.

In conformity with Table 1, the system constraints for speed control are summarized as:

$$\begin{cases} x_{k+i+1|k}^v = x_{k+i|k}^v + T_s \hat{v}(k) + \alpha_v T_s u_{k+i|k}^v, i = 0 \ldots H_p, \\ -\Delta T_r^{\max} \le u_{k+i|k}^v \le \Delta T_r^{\max}, i = 0 \ldots H_c - 1, \\ u_{k+i|k}^v = 0, i = H_c \ldots H_p, \\ T_r^{\min} \le T_r(k-1) + \sum_{i=0}^{i_+} u_{k+i|k}^v \le T_r^{\max}, i_+ = 0, \\ \vdots \\ T_r^{\min} \le T_r(k-1) + \sum_{i=0}^{i_+} u_{k+i|k}^v \le T_r^{\max}, i_+ = H_c - 1. \end{cases} \tag{18}$$

In (18), $x_{k+i|k}^v, i = 1 \ldots H_p + 1$ are the multiple-step previewed vehicle longitudinal velocities. $u_{k+i|k}^v, i = 0 \ldots H_p$ indicate the system inputs (torque increments) within the prediction horizon. $T_r(k-1)$ is the rear-wheel torque at the last step.

By minimizing (17) under the constraints in (18), we can obtain the optimal torque increment sequence within the prediction horizon: $u_{k+i|k}^{v*}, i = 0 \ldots H_p$. The first element of this sequence is chosen as the optimal torque increment at the current step. Thus, the optimal rear-wheel torque at step k can be calculated as:

$$T_r^*(k) = T_r(k-1) + u_{k|k}^{v*}, \tag{19}$$

which is then equally allocated to the right and left side of the rear axle.

Likewise, by substituting (12) into (6), the ultra-local model of vehicle lateral dynamics at step k becomes:

$$\begin{aligned} \ddot{e}_y(k) &\approx \hat{f}_y(k) + \alpha_y \delta_{sw}(k) \\ &= \hat{\ddot{e}}_y(k) - \alpha_y \delta_{sw}(k-1) + \alpha_y\left(\delta_{sw}(k-1) + \Delta\delta_{sw}(k)\right) \\ &= \hat{\ddot{e}}_y(k) + \alpha_y \Delta\delta_{sw}(k). \end{aligned} \tag{20}$$

We assign $x^e(k) = \left[e_y(k), \quad \dot{e}_y(k)\right]^T, u^e(k) = \Delta\delta_{sw}(k)$ as the lateral dynamics states and input.

By discretizing (20) with the sampling period T_s, we have:

$$\begin{bmatrix} e_y(k+1) \\ \dot{e}_y(k+1) \end{bmatrix} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \begin{bmatrix} e_y(k) \\ \dot{e}_y(k) \end{bmatrix} + \begin{bmatrix} 0 \\ \alpha T_s \end{bmatrix} \Delta\delta_{sw}(k) + \begin{bmatrix} 0 \\ \hat{\ddot{e}}_y(k) T_s \end{bmatrix}, \tag{21}$$

which can be expressed in short as:

$$x_{k+1|k}^e = A x_{k|k}^e + B u_{k|k}^e + d(k), \tag{22}$$

where $x_{k|k}^e$ corresponds to the currently measured lateral dynamic states. From (22), the cost function for the vehicle lateral dynamics control can be designed as:

$$J^e = \sum_{i=1}^{H_p} \left\| x_{k+i|k}^e \right\|_{Q_e}^2 + \sum_{i=0}^{H_c-1} \left\| u_{k+i|k}^e \right\|_{R_e}^2. \tag{23}$$

Similar to (17), cost function (23) balances the vehicle position tracking error and the steering wheel fluctuation within the prediction horizon. System constraints for the steering wheel are summarized as:

$$\begin{cases} x_{k+i+1|k}^e = A x_{k+i|k}^e + B u_{k+i|k}^e + d(k), i = 0 \ldots H_p, \\ -\Delta\delta_{sw}^{\max} \le u_{k+i|k}^e \le \Delta\delta_{sw}^{\max}, i = 0 \ldots H_c - 1, \\ u_{k+i|k}^e = 0, i = H_c \ldots H_p, \\ \delta_{sw}^{\min} \le \delta_{sw}(k-1) + \sum_{i=0}^{i_+} u_{k+i|k}^e \le \delta_{sw}^{\max}, i_+ = 0, \\ \vdots \\ \delta_{sw}^{\min} \le \delta_{sw}(k-1) + \sum_{i=0}^{i_+} u_{k+i|k}^e \le \delta_{sw}^{\max}, i_+ = H_c - 1. \end{cases} \tag{24}$$

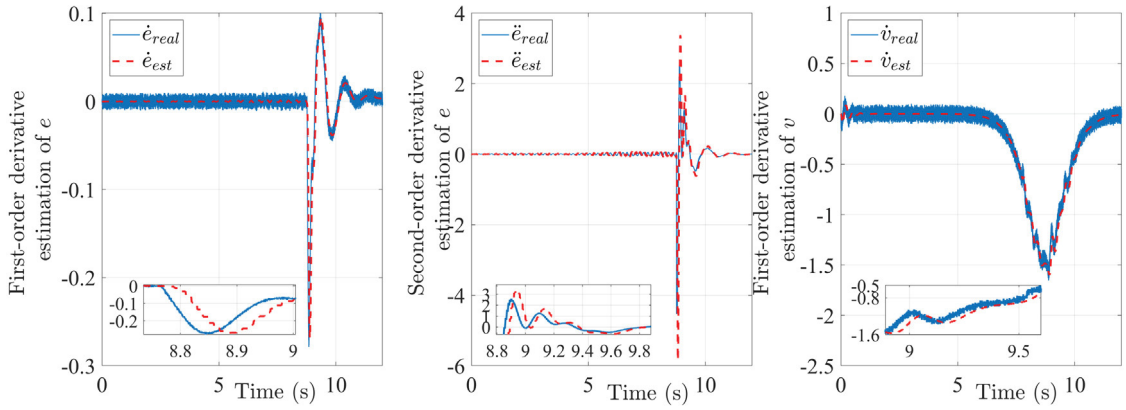

Fig. 1. Algebraic differentiation estimation.

Table 3
MPC problem size comparison.

	ULMPC (Speed)	ULMPC (Steering)	LTVMPC
Non-zero KKT matrix	271	585	3277
MEX file size (KB)	234	392	3025

In (24), $x^e_{k+i|k}, i = 1 \dots H_p+1$ are the multiple-step previewed lateral dynamics states. $u^e_{k+i|k}, i = 0 \dots H_p$ represent the system inputs (steering wheel increments) within the prediction horizon. $\delta_{sw}(k-1)$ is the steering wheel angle at the last step. Minimizing (23) under the constraints in (24) gives us the optimal steering wheel increment sequence within the prediction horizon: $u^{e*}_{k+i|k}, i = 0 \dots H_p$. The first element of this sequence is chosen as the optimal steering wheel increment at the current step. In other words, the optimal steering wheel angle at step k can be calculated as:

$$\delta^*_{sw}(k) = \delta_{sw}(k-1) + u^{e*}_{k|k}. \tag{25}$$

Stability proof of ULMPC is out of the scope of this paper and interested reader is referred to Chen (2010), which illustrates the stability condition of a finite-horizon linear MPC without terminal cost.

4.5. Linear time-varying model predictive control design

The model-based LTVMPC comes from Wang, Zha, and Wang (2019). To be coherent with MFC and ULMPC, the number of actuators is reduced from eight to four by disabling the active rear-wheel steering and the front-wheel independent torques.

4.6. CarSim-Simulink simulation results

All three controllers: MFC, ULMPC, and LTVMPC have the same sampling period $T_s = 0.01$ s. CVXGEN (Mattingley & Boyd, 2012) is employed for MPC formulations and code generations. One advantage of ULMPC is immediately identified during code generation: grounded on the ultra-local model, the ULMPC has a relatively small problem dimension. For instance, the ultra-local longitudinal dynamics in (18) has merely one state and one input while the ultra-local lateral dynamics in (24) has only two states and one input. Instead, the model-based LTVMPC in Wang, Zha, and Wang (2019) has five states and three inputs. By assigning $H_p = 20$, and $H_c = 1$, the number of non-zero KKT matrix entries of the generated codes and the size of the compiled MEX files are summarized in Table 3.

Therefore, the total size of two ULMPCs' MEX files remains less than 21% of the size of LTVMPC's MEX file. As current commercial Electronic Control Unit (ECU) has a quite limited memory for code and data storage, the small problem scale of ULMPC can definitely facilitate its online implementation.

During simulations, white noises are deliberately added on the ultra-local model outputs: $e_y(t)$ and $v(t)$ in (6). The first-order ADE in (2) is utilized for reconstructing $\dot{v}(t)$, $\dot{e}_y(t)$, and $\ddot{e}_y(t)$. By respectively assigning the sliding window size as $\mathrm{T_{ADE}} = 0.05\mathrm{s}, 0.04\mathrm{s}, 0.03\mathrm{s}, (K = 5, 4, 3)$ for the first-order derivative on v, the first-order derivative on e_y, and the first-order derivative on $\hat{e}_y$(cascading), the algebraic differentiation estimation results are obtained and demonstrated in Fig. 1.

In Fig. 1, the red dashed lines indicate the estimated results and the blue solid lines represent the ground truth by directly differentiating the noisy signals. The actual second-order derivative of e is indeed totally covered by the high-frequency noise, hence the middle plot in Fig. 1 is generated after removing the white noise. From Fig. 1, we can conclude that ADE can successfully reconstruct the derivatives of signals from noisy measurements. However, estimation delays do exist and the higher-order estimation suffers further from the Taylor expansion truncation error. In practice, such a delay can be reduced by enhancing the sampling frequency $1/T_s$ (Mboup et al., 2009).

Before exhibiting the control performance of ULMPC, the discretized ultra-local models in (16) and (21) are validated via one-step prediction verification and the validation results are revealed in Fig. 2.

In Fig. 2, the red dashed lines represent the predicted states of the ultra-local models at step $k+1$ on the basis of the measured states and inputs at step k. The blue lines express the actual states at step $k+1$ from Carsim. The normalized maximum estimation errors: $\max\left(x_{k+1} - \hat{x}_{k+1}\right) / \max\left(x_{k+1}\right), x \in \{v, e, \dot{e}\}$ are 0.31%, 0.13%, and 3.97% for respectively v, e, and $\dot{e}$. Therefore, the ultra-local model can sufficiently approximate the dynamics of the vehicle.

We first show the global path-tracking results in Fig. 3. The automated car starts from the origin and follows the oval track counter-clockwise. The MFC is tuned with $\eta_T = -0.001$ in (7) and $\eta^0_{sw} = \eta^1_{sw} = -4$ in (11) and the ULMPC weighting factors in (17) and (23) are tuned with $R_v = 1$, $Q_v = 25\,000$, $R_e = 1$, and $Q_e = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.03 \end{bmatrix}$.

Fig. 3 indicates that all three controllers can effectively make the automated car follow the given track even under front steering fault and varying road conditions, as described in Section 4.1. However, the enlarged plots exhibit that ULMPC achieves the most accurate tracking trajectory. This fact is further revealed in Fig. 4, where the path-tracking error $e_y(t)$ is depicted. Interestingly, MFC gives us a globally negative tracking error whereas the other two controllers lead to a globally positive tracking error. We believe controller-tuning plays an important role in this phenomenon and we will study it in the future. In addition, the RMS of the tracking errors are summarized in Table 4.

As indicated in Fig. 4, the LTVMPC produces a higher tracking error under more challenging road conditions. For instance, LTVMPC results in the highest tracking error $\max\left(e_y\right) = 0.38$ m when the automated car runs on the split-μ road. On the contrary, the model-free nature of ULMPC and MFC ensures that they are insensitive to the modeling error from the changing TRFC. Actually, the road condition variation can be

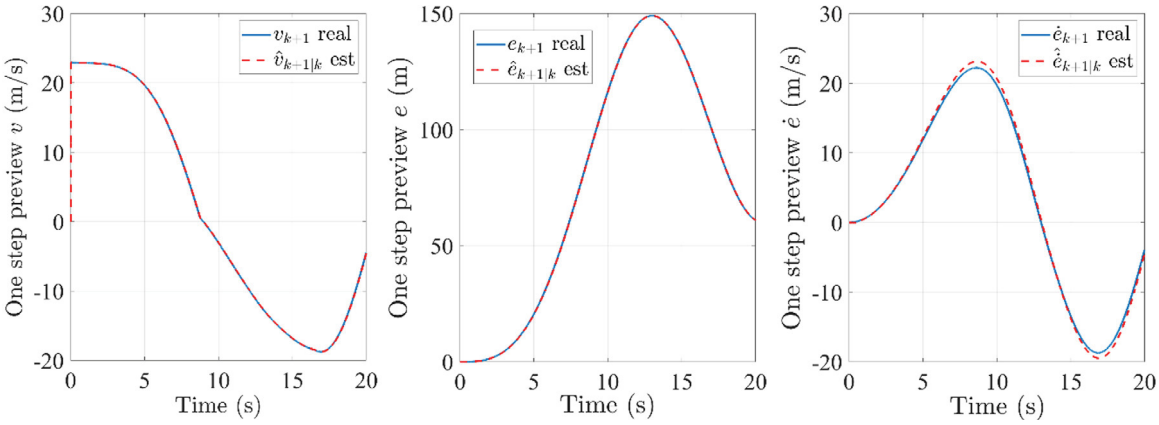

Fig. 2. Ultra-local model validation.

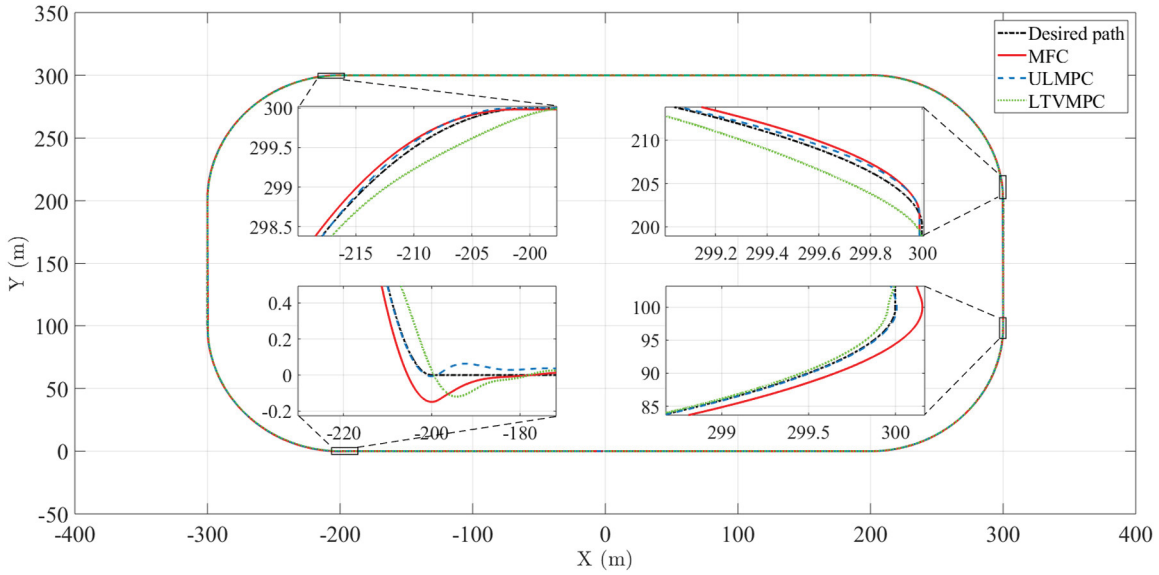

Fig. 3. Global path tracking comparison.

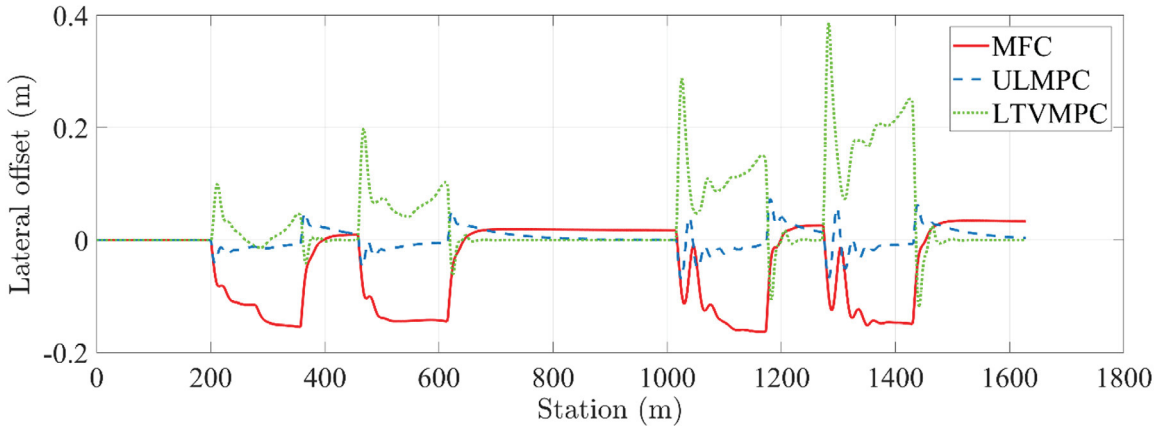

Fig. 4. Path tracking error comparison.

immediately identified in the scalar terms $\hat{f}_v(t), \hat{f}_y(t)$ in (8) and (12), which are then either directly compensated in MFC via (7) and (11) or employed to update the ultra-local models of ULMPC in (16) and (21). However, MFC leads to a steady-state tracking error (red line in Fig. 4) due to the estimation delay of ADE. In contrast, by combining MFC with MPC, the ULMPC inherits the excellent robustness from the ultra-local model while eliminating the steady-state tracking error.

The environment adaption ability of ULMPC is also reflected in the steering wheel angle variation in Fig. 5.

We can firstly conclude that the constraints on the steering wheel angle and angular velocity are well observed. For MFC, input thresholds

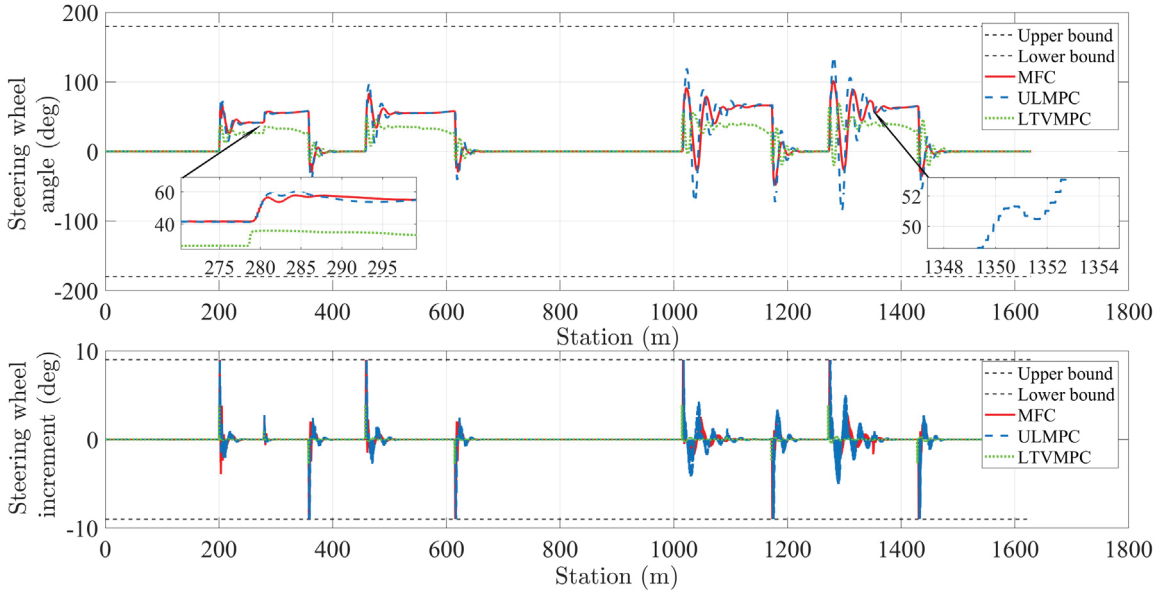

Fig. 5. Steering wheel angle.

Table 4
Root mean square values of path tracking error.

	MFC	ULMPC	LTVMPC
RMS(e_y) (m)	0.0864	0.0194	0.0834

Table 5
Root mean square values of velocity tracking error.

	MFC	ULMPC	LTVMPC
RMS($v - v_r$) (m/s)	0.1375	0.1350	0.2808

are directly imposed in (13) whereas for both LTVMPC and ULMPC, constrained optimization is executed online for satisfying these restrictions. Then, all three controllers adaptively increase the steering angle when the steering ratio suddenly decreases at $s = 278.5$ m. The adaptability of LTVMPC comes from the inherent robustness of predictive control (Bemporad & Morari, 1999). In regards to MFC, the model-free nature grants it robustness to steering system fault. ULMPC, which enjoys both the predictive control architecture and the model-free property, should be more capable at fault-tolerance. This point will be further investigated in the future. Finally, as indicated in the enlarged inset near the bottom right corner of the upper plot, the steering angle from ULMPC temporarily decreases around $s = 1350$ m. At that moment, the left-turning automated car enters into the split friction road. As the left side track suddenly provides a higher TRFC, the reasonable action is to reduce the steering angle for avoiding oversteering. Clearly, ULMPC successfully identifies the road condition change and adjusts the wheel steering accordingly.

In parallel to the position control, velocity-following results are revealed in Fig. 6 and the RMS of velocity tracking errors are summarized in Table 5.

Similar to the results in Fig. 4, the model-based LTVMPC entails a relatively higher velocity tracking error. Moreover, as demonstrated in the enlarged inset, neither LTVMPC nor MFC can reach offset-free velocity tracking: MFC is still impaired by the algebraic estimation delay and the adopted LTVMPC in Wang, Zha, and Wang (2019), which did not utilize the incremental control form in Table 1, was affected by the unmodeled factors, like air drag and tire rolling resistance. In contrast, ULMPC condenses all the disturbances in the ultra-local model and utilizes command increment as the optimized variable to achieve offset-free tracking.

Subsequently, the rear-wheel torques are depicted in Fig. 7.

Obviously, constraints on the torque magnitude and its slew rate are well obeyed by all the three controllers. Indeed, ULMPC produces more chattering, as it endeavors to achieve zero speed tracking error via frequent torque adjustment.

As a final point, the tire usages of the four-wheel are displayed in Fig. 8, where the tire usage is defined as:

$$\theta_{i,j} = \frac{\sqrt{F_{xi,j}^2 + F_{yi,j}^2}}{\mu F_{zi,j}}, i = \{front \quad rear\}, j = \{left \quad right\}, \tag{26}$$

with $F_{xi,j}, F_{yi,j}, F_{zi,j}$ representing respectively the longitudinal, lateral, and vertical tire force.

Therefore, the dynamics of the automated vehicle has fully entered into the nonlinear region as the rear-left tire has its peak usage of more than 90%. This fact further proves the performance of the proposed ULMPC even in nonlinear situations.

4.7. Ultra-local Model Predictive Control with long prediction horizon

The ultra-local model utilizes the estimated first-order derivative of the system output to deduce the evolution of the controlled plant. Therefore, this prediction can remain accurate only within a relatively short preview horizon (Fliess, Join, & Voyant, 2018). In this section, we will illustrate this fundamental weakness of ULMPC through CarSim-Simulink joint simulations.

Fig. 9 demonstrates the path tracking error e_y with the ULMPC prediction horizon H_p in (17), (18), (23), and (24) extended from 20 to 65.

Therefore, the path-tracking performance degradation of ULMPC is clearly revealed. Furthermore, the relationship between the ULMPC prediction horizon H_p and the root mean square path tracking error $RMS(e_y)$ is depicted in Fig. 10.

Hence, as the prediction horizon extends from 20 to 85, the path-tracking error of ULMPC trebles. This fact distinguishes ULMPC from the traditional model-based predictive controller, which in general requires a sufficiently long prediction horizon (Wang, Bai, Wang, & Wang, 2019). Instead, a long preview will indeed excite model-plant mismatch of ULMPC, which can lead to actuator chattering and lower the ultimate control performance. The steering wheel angle $\delta_{sw}^*(k)$ and the real wheel torque $T_r^*(k)$ from ULMPC with H_p equal to either 20 or 65 are compared, respectively, in Fig. 11 and Fig. 12.

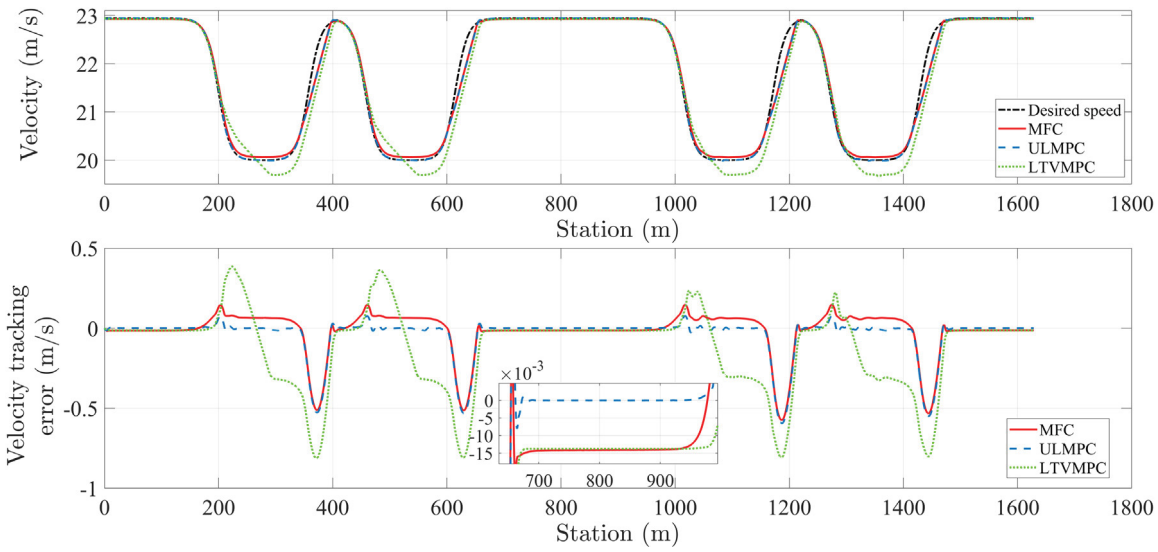

Fig. 6. Velocity tracking error comparison.

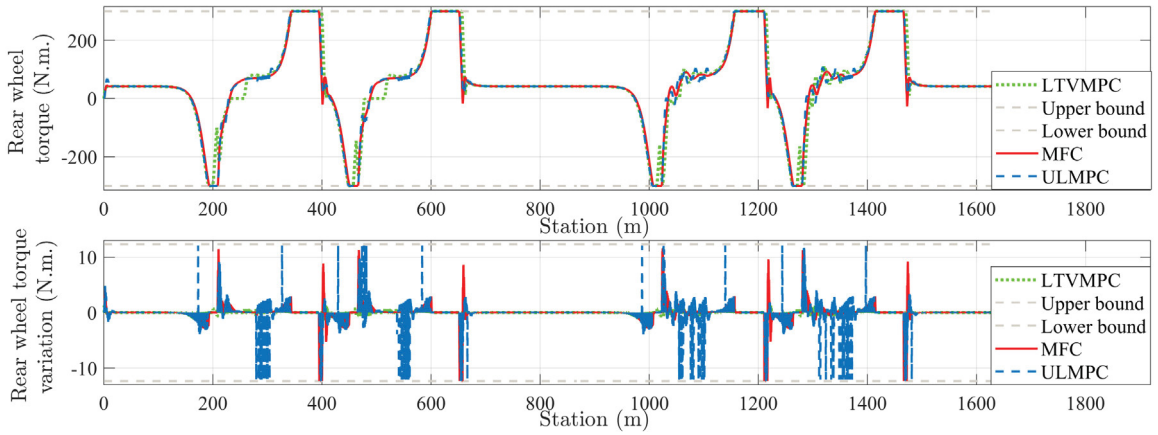

Fig. 7. Rear wheel torque comparison.

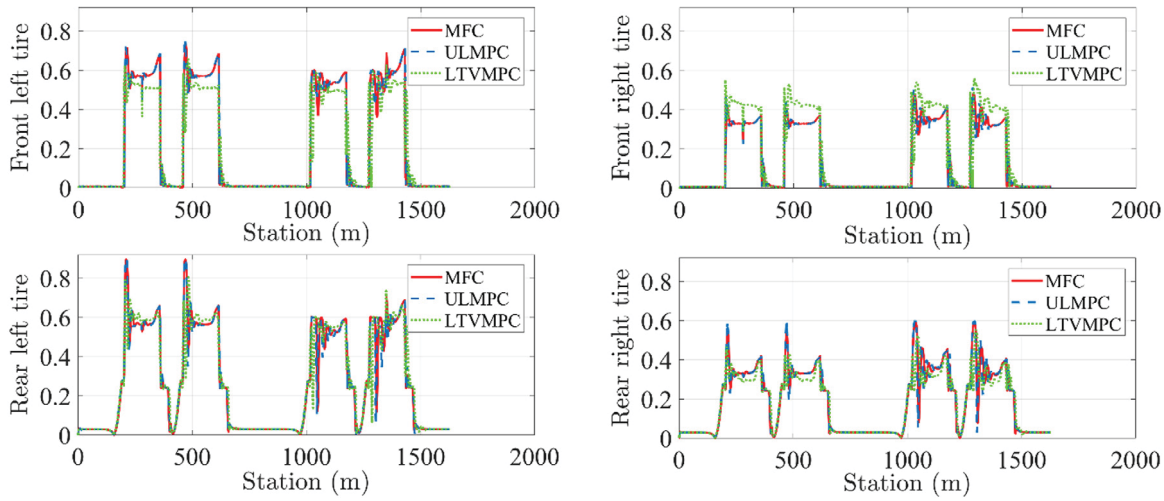

Fig. 8. Tire usage comparison.

Even though the magnitude and the slew rate constraints of the control are always met, such a chattering responses have no practical usage, as the actuators will quickly wear out.

We can therefore draw the conclusion that ULMPC alone is not very appropriate for long-preview applications, such as path-planning for collision/obstacle avoidance. Instead, ULMPC should be combined with

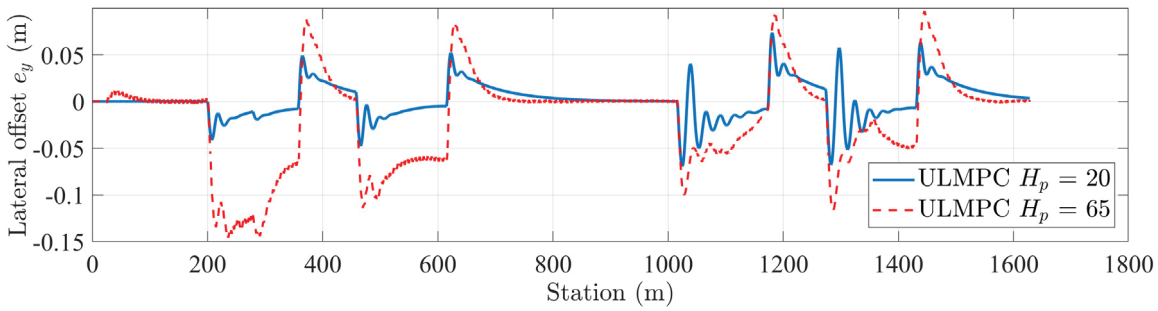

Fig. 9. Path tracking error from ULMPC with different prediction horizon.

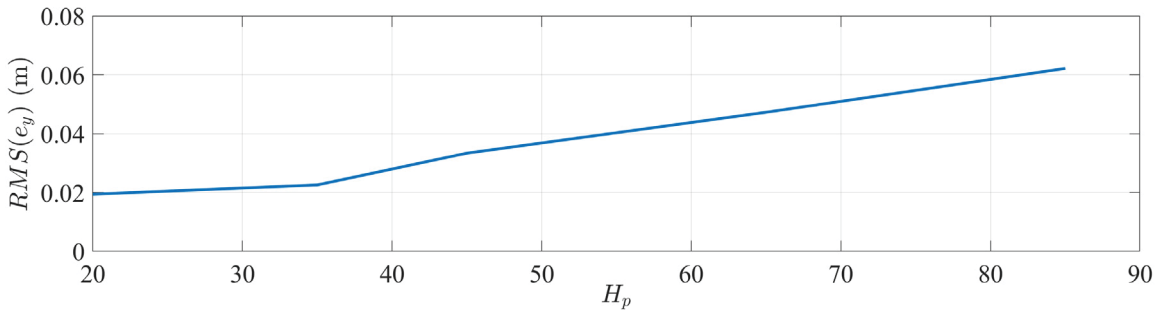

Fig. 10. Relationship between ULMPC prediction horizon and RMS of path tracking error.

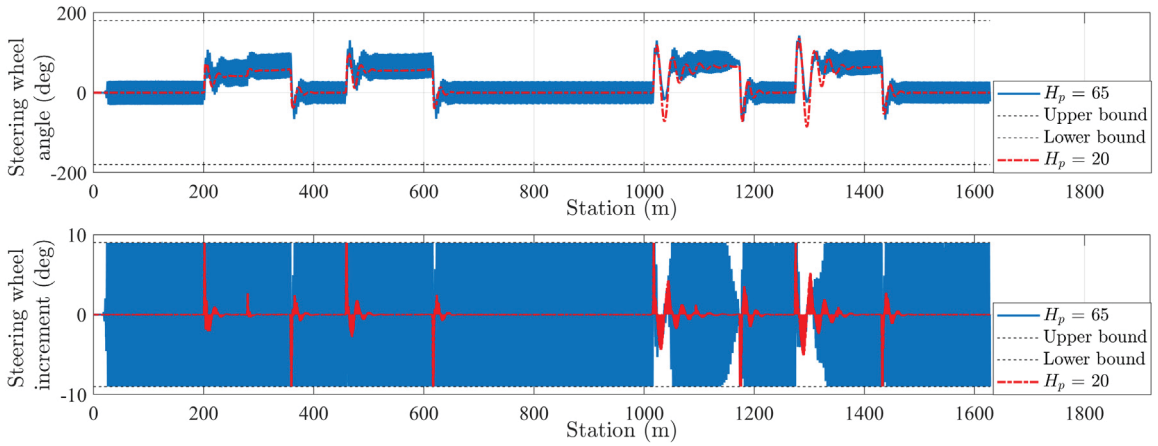

Fig. 11. Steering wheel angle from ULMPC with long and short prediction horizon.

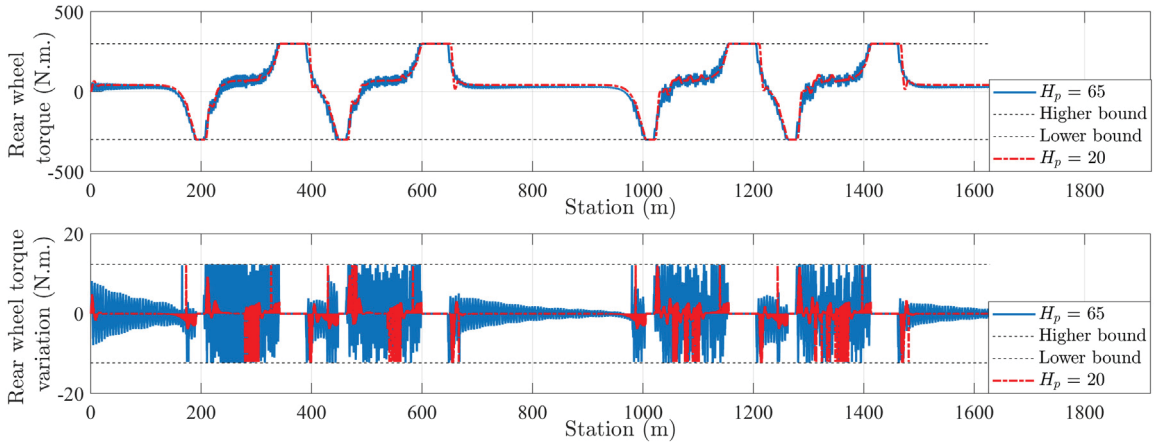

Fig. 12. Real-wheel torque from ULMPC with long and short prediction horizon.

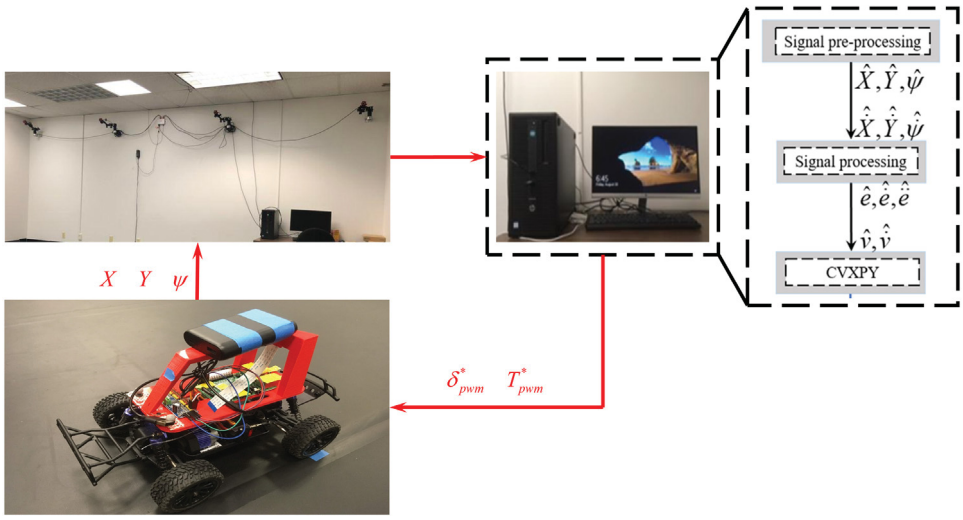

Fig. 13. Scaled car experiment setup.

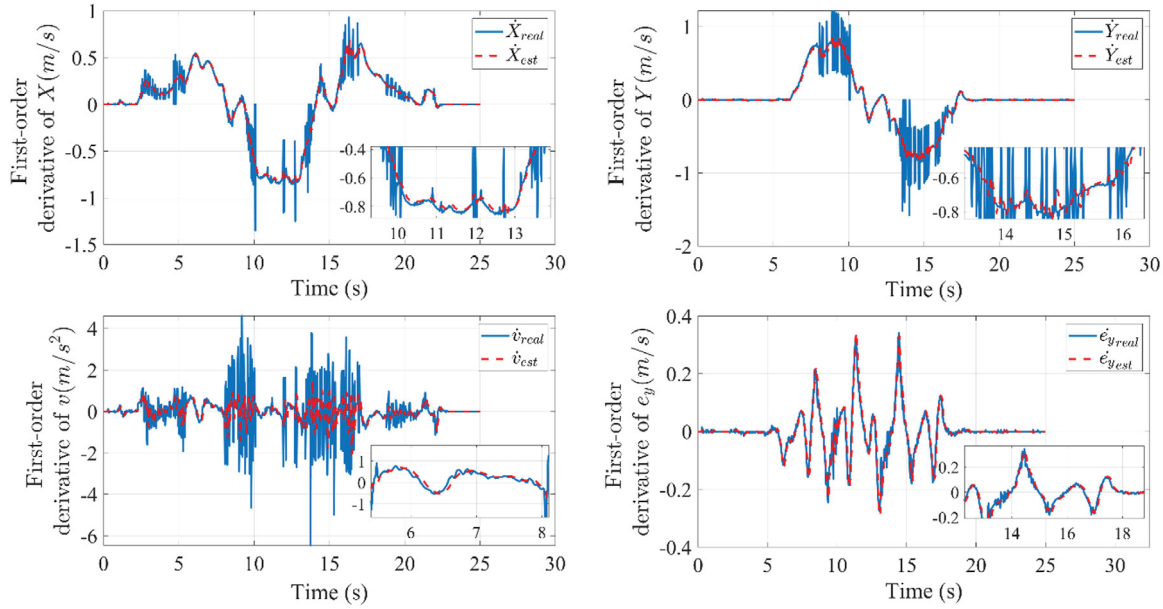

Fig. 14. Algebraic differentiation estimation of field test.

a high-level path planner. The high-level path planner is responsible for receiving obstacle information from the perception system, timely adjusting the reference path for the ULMPC, while considering the system-level delay from perception, decision-making, and controller actuation. ULMPC is then devoted to accurately *tracking* this obstacle-free trajectory under modeling error and external disturbances.

5. Indoor vehicle field test

In addition to the Carsim simulations, the proposed ULMPC is further implemented on a scaled car for experimental investigation of a similar trajectory-following task.

5.1. Experiment setup

The 1/18 scaled car has a DC-servo for front steering and a brushed DC-motor for four-wheel-driving. Both the servo and the motor are connected to a PCA9685, which is a 16-channel, 12-bit Pulse Width Modulation (PWM) driver. A Raspberry Pi works as the ECU for delivering respectively the throttle and the steering PWM commands to the motor and the servo. To capture the position of the scaled car, we utilize a camera-based OptiTrack indoor GPS system. OptiTrack takes multiple 2D images from eight infrared cameras to reconstruct the corresponding 3D coordinates of the scaled car. Specifically for our tests, it provides us the raw data of the global coordinates of the scaled car: $X(t)$, $Y(t)$, as well as its heading angle $\psi(t)$. These (noisy) raw data points are retrieved from OptiTrack in real-time by a local PC.

Inside the local PC, the ADE in (2) is employed to generate the filtered $\hat{X}(t)$, $\hat{Y}(t)$, $\hat{\psi}(t)$ as well as the estimated first-order derivatives: $\dot{\hat{X}}(t)$, $\dot{\hat{Y}}(t)$, and $\dot{\hat{\psi}}(t)$. From $\hat{X}(t)$, $\hat{Y}(t)$, a simple mapping function is utilized to determine the position tracking error $\hat{e}(t)$, and the longitudinal velocity of the scaled car is calculated as:

$$\hat{v}(t) = \dot{\hat{X}}(t)\cos(\hat{\psi}(t)) + \dot{\hat{Y}}(t)\sin(\hat{\psi}(t)). \tag{27}$$

After obtaining $\hat{e}(t)$ and $\hat{v}(t)$, ADE is employed again to produce $\dot{\hat{e}}(t)$, $\ddot{\hat{e}}(t)$, and $\dot{\hat{v}}(t)$. With $\hat{e}(t)$, $\dot{\hat{e}}(t)$, $\ddot{\hat{e}}(t)$, $\hat{v}(t)$, and $\dot{\hat{v}}(t)$, the ULMPCs described in (17)–(18) and (23)–(24) are formulated and converted into standard QP forms via CVXPY (Diamond & Boyd, 2016) and solved with OSQP (Stellato, Banjac, Goulart, Bemporad, & Boyd, 2018). Finally, the

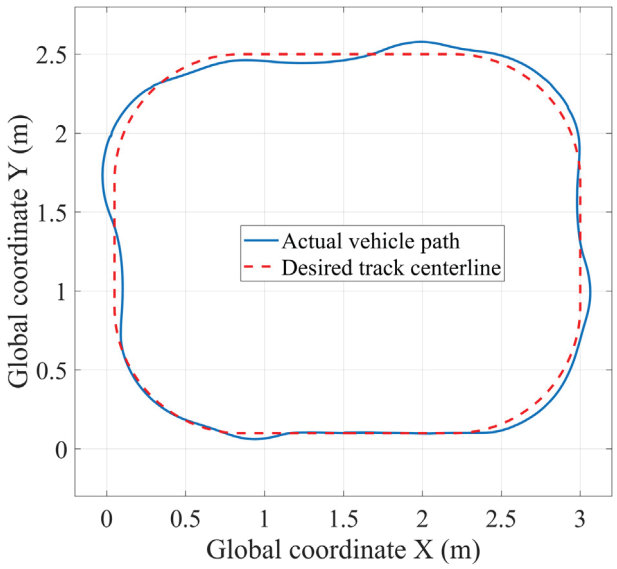

Fig. 15. Scaled car global path tracking.

optimal PWM commands: $\delta^*_{pwm}(t)$ and $T^*_{pwm}(t)$ are wirelessly send back from the local PC to the scaled car to close the loop. The overall architecture is depicted in Fig. 13, which is similar to Liniger, Domahidi, and Morari (2015).

The desired track of the field test remains as an oval. However, the referential velocity profile is modified as a hyperbolic tangent function to account for the fact that the scaled car kicks off from standing still.

Sampling period of OptiTrack and execution periods of both ULMPCs are fixed as $T_s = 0.02$ s, which is determined by the highest possible execution frequency of CVXPY on the available hardware. The PWM commands of the steering and the throttle are individually normalized between: $\delta^*_{pwm}(t) \in [-0.70, \quad 0.25]$, $T^*_{pwm}(t) \in [0.33, \quad 0.41]$. Ultra-local gains in (6) are respectively tuned as: $\alpha_v = 750$, $\alpha_y = -15$. α_y is tuned negative because an increasing PWM command of steering leads to a clockwise-direction steer of the front servo. Prediction horizon $H_p = 10$ and control horizon $H_c = 1$ are used for both speed and steering ULMPCs.

5.2. Field test results

In common with Section 4.6, we first present the signal differentiation results from ADE. The sliding window size is tuned as $T_{ADE} = 0.10s, 0.10s, 0.12s, 0.14s, 0.10s, 0.10s$ $(K = 5, 5, 6, 7, 5, 5)$ for estimating, respectively, $\hat{\dot{X}}$, $\hat{\dot{Y}}$, $\hat{\dot{\psi}}$, $\hat{\dot{v}}$, $\hat{\dot{e}}_y$, and $\hat{\ddot{e}}_y$. Fig. 14 demonstrates $\hat{\dot{X}}$, $\hat{\dot{Y}}$, $\hat{\dot{v}}$, $\hat{\dot{e}}_y$ as examples.

In Fig. 14, the red dashed lines indicate the estimated values and the blue solid lines represent the ground truth by directly differentiating the noisy signals. Evidently, the ADE effectually reconstructs the needed signal derivatives.

The global path tracking result is demonstrated in Fig. 15. The scaled car starts from $X = 1.525, Y = 0.1$, and follows the oval track counterclockwise.

From Fig. 15, we can observe that the scaled car can stay pretty close to the oval track centerline. Path tracking error $e_y(t)$ is then depicted in Fig. 16.

From Fig. 16, we can deduce that the absolute maximal path tracking error is less than 79.1 mm and the RMS of the tracking error is less than 29.4 mm. Considering that the overall length of the desired track is over 9320 mm, the realized path-tracking result is quite accurate.

Then, the optimal PWM steering command $\delta^*_{pwm}(t)$ is demonstrated in Fig. 17.

Hence, both the PWM steering command and its changing-rate satisfy the given constraints. Besides, we have $\delta^*_{pwm}(t) = 0.15$ before 3 s and after 20 s, which corresponds to the desired PWM command leading to the neutral steering of the front DC servo.

Subsequently, the velocity tracking result is depicted in Fig. 18.

The actual speed can practically follow the desired velocity profile with RMS of the speed tracking error less than 0.066 m/s. Note that the highest speed of the scaled car reaches more than 0.95 m/s, which is equivalent to 60 km/h of a full-size car.

The optimal PWM throttle command $T^*_{pwm}(t)$ is next depicted in Fig. 19.

Akin to Fig. 17, both the PWM throttle command and its changing-rate satisfy the given constraints. From Figs. 17 and 19, it is clear that the actual PWM increments of both the steering-servo and the driving-motor are much less than their given thresholds. This hardware constraint and the relatively slow sampling frequency principally accounts for the position and speed tracking errors in Figs. 16 and 18.

Finally, the online execution times of both the speed and the steering ULMPCs are displayed in Fig. 20.

Hence, excluding several control cycles, both the throttle and the steering ULMPCs can be solved within the sampling period $T_s = 0.02$ s. Note that when this paper is written, CVXPY does not support code generation yet. Therefore, both ULMPCs are repeatedly compiled and solved by CVXPY in the local PC (Windows10, Intel i5-6500), which is not a strict real-time platform. This may explain the execution time fluctuations in Fig. 20.

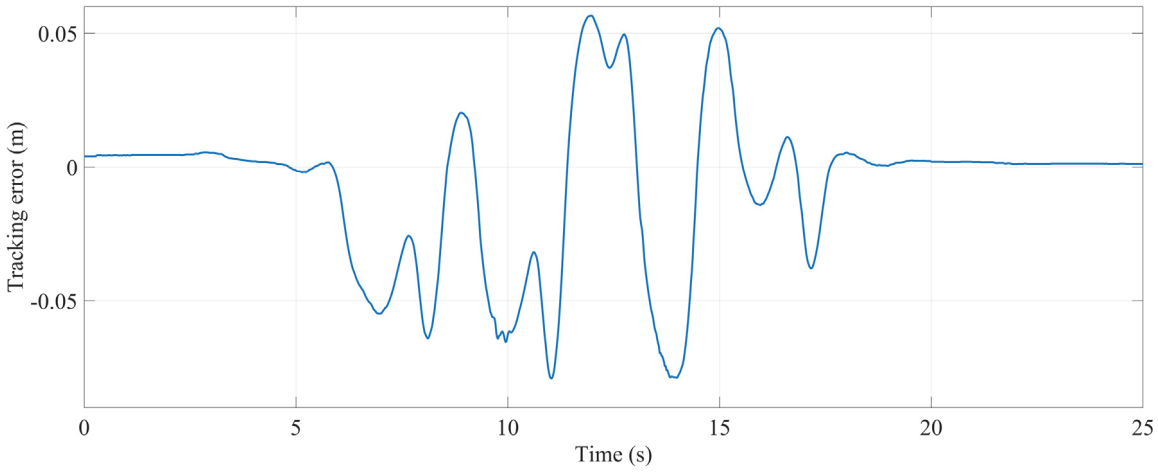

Fig. 16. Scaled car path tracking error.

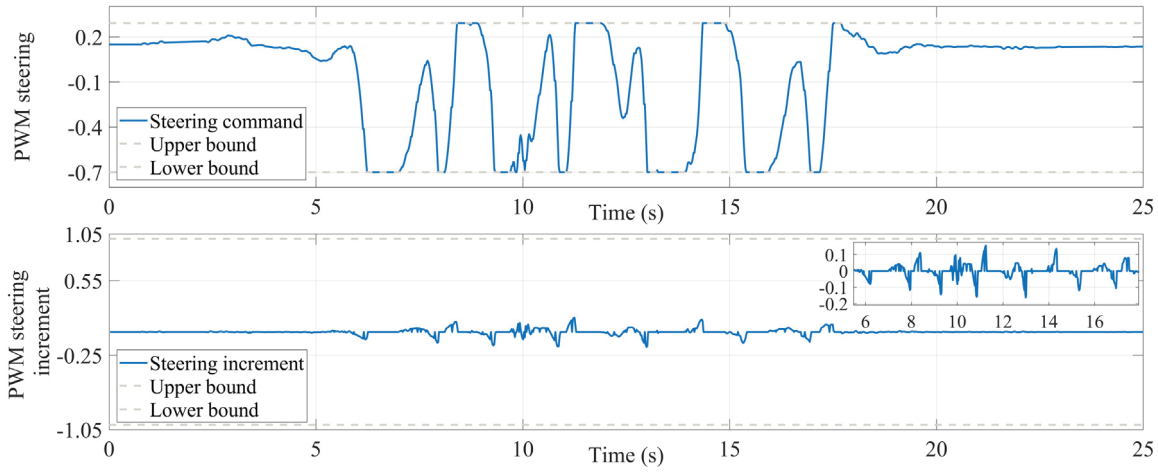

Fig. 17. Scaled car PWM steering command.

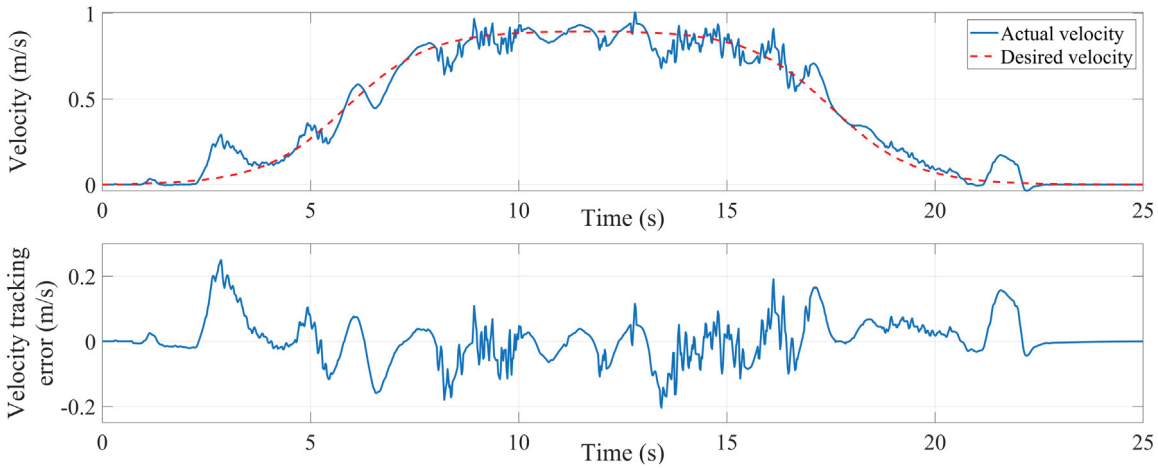

Fig. 18. Scaled car velocity tracking.

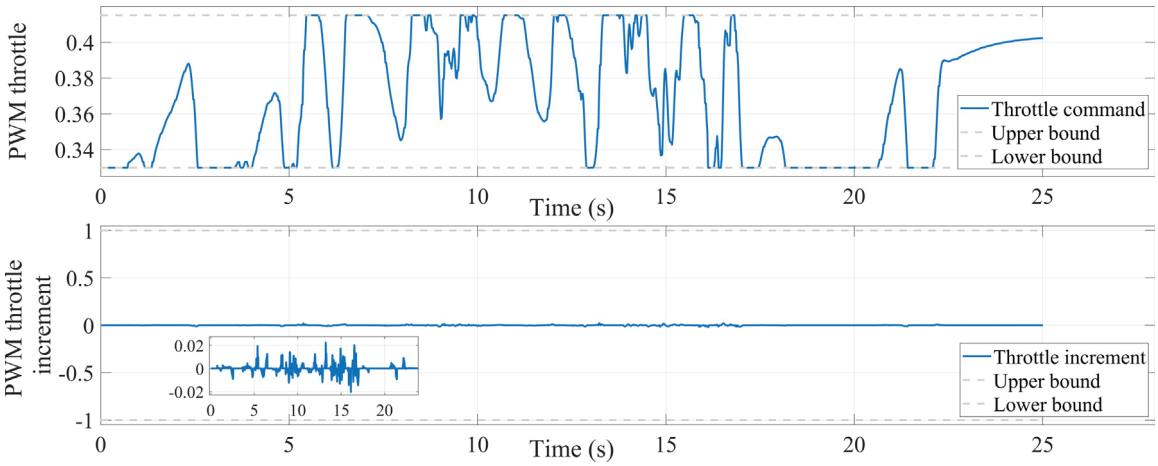

Fig. 19. Scaled car PWM throttle command.

6. Conclusions

This paper presents the ULMPC, which is a novel data-driven, model-free predictive controller, with no need for complex and laborious model learning. The key idea of ULMPC is to apply an MPC on the ultra-local model. ULMPC inherits the robustness to system uncertainties and disturbances from Model-Free Control (MFC), reduces the scale of the constrained optimization problem, and eliminates the steady-state tracking error of MFC. Carsim-Simulink joint simulations and indoor field tests with a scaled car show the effectiveness of ULMPC. However, similar to MFC, the constant gain tuning of the ultra-local model remains as an open problem, which requires further investigation. Moreover, the fundamental weakness of ULMPC is that the continuously updated ultra-local model can only hold within a

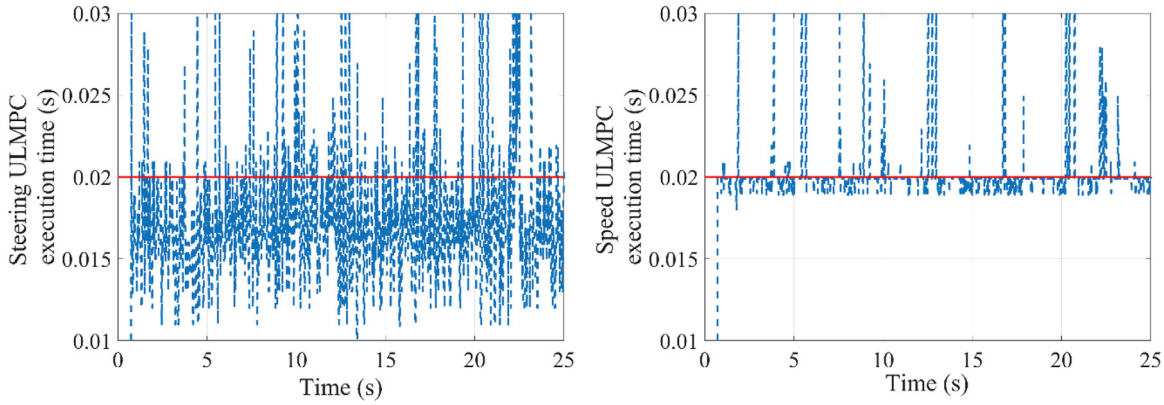

Fig. 20. Online execution times of ULMPCs.

relatively short preview horizon. Therefore, ULMPC alone is not appropriate for long-preview applications, such as path-planning for collision avoidance. Instead, ULMPC shall be combined with a high-level path-planner, which is responsible for generating an obstacle-free reference trajectory, and ULMPC is then devoted to path tracking with both system modeling errors and external disturbances.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by National Science Foundation, USA with project number 1901632.

References

Bemporad, A., & Morari, M. (1999). Robust model predictive control: A survey. In *Robustness in identification and control* (pp. 207–226). London: Springer, http://dx.doi.org/10.1007/BFb0109870.

Brown, M., Funke, J., Erlien, S., & Gerdes, J. C. (2017). Safe driving envelopes for path tracking in autonomous vehicles. *Control Engineering Practice*, *61*, 307–316. http://dx.doi.org/10.1016/j.conengprac.2016.04.013.

Chen, W. H. (2010). Stability analysis of classic finite horizon model predictive control. *International Journal of Control, Automation and Systems*, *8*(2), 187–197. http://dx.doi.org/10.1007/s12555-010-0202-z.

Di Ruscio, D. (2013). Model predictive control with integral action: a simple MPC algorithm. *Modeling Identification and Control*, *34*(3), 119–129. http://dx.doi.org/10.4173/mic.2013.3.2.

Diamond, S., & Boyd, S. (2016). CVXPY: A python-embedded modeling language for convex optimization. *Journal of Machine Learning Research (JMLR)*, *17*(1), 2909–2913.

Dovžan, D., & Škrjanc, I. (2010). Predictive functional control based on an adaptive fuzzy model of a hybrid semi-batch reactor. *Control Engineering Practice*, *18*(8), 979–989. http://dx.doi.org/10.1016/j.conengprac.2010.04.004.

Ercan, Z., Carvalho, A., Tseng, H. E., Gökaşan, M., & Borrelli, F. (2018). A predictive control framework for torque-based steering assistance to improve safety in highway driving. *Vehicle System Dynamics*, *56*(5), 810–831. http://dx.doi.org/10.1080/00423114.2017.1337915.

Fliess, M., & Join, C. (2013). Model-free control. *International Journal of Control*, *86*(12), 2228–2252. http://dx.doi.org/10.1080/00207179.2013.810345.

Fliess, M., Join, C., & Sira-Ramirez, H. (2008). Nonlinear estimation is easy. *International Journal of Modelling, Identification and Control*, *40*(1), 12–27. http://dx.doi.org/10.1504/IJMIC.2008.020996.

Fliess, M., Join, C., & Voyant, C. (2018). Prediction bands for solar energy: New short-term time series forecasting techniques. *Solar Energy*, *166*, 519–528. http://dx.doi.org/10.1016/j.solener.2018.03.049.

Forbes, M. G., Patwardhan, R. S., Hamadah, H., & Gopaluni, R. B. (2015). Model predictive control in industry: Challenges and opportunities. *IFAC-PapersOnLine*, *48*(8), 531–538. http://dx.doi.org/10.1016/j.ifacol.2015.09.022.

Garone, E., Di Cairano, S., & Kolmanovsky, I. (2017). Reference and command governors for systems with constraints: A survey on theory and applications. *Automatica*, *75*, 306–328. http://dx.doi.org/10.1016/j.automatica.2016.08.013.

Hu, Y., Chen, H., Wang, P., Chen, H., & Ren, L. (2018). Nonlinear model predictive controller design based on learning model for turbocharged gasoline engine of passenger vehicle. *Mechanical Systems and Signal Processing*, *109*, 74–88. http://dx.doi.org/10.1016/j.ymssp.2018.02.012.

Huusom, J. K., Poulsen, N. K., Jørgensen, S. B., & Jørgensen, J. B. (2012). Tuning SISO offset-free model predictive control based on ARX models. *Journal of Process Control*, *22*(10), 1997–2007. http://dx.doi.org/10.1016/j.jprocont.2012.08.007.

Ławryńczuk, M. (2009). Neural networks in model predictive control. In N. T. Nguyen, & E. Szczerbicki (Eds.), *Studies in computational intelligence*: *vol. 252*, *Intelligent systems for knowledge management* (pp. 31–63). Berlin, Heidelberg, Germany: Springer, http://dx.doi.org/10.1007/978-3-642-04170-9_2.

Li, S. E., Jia, Z., Li, K., & Cheng, B. (2014). Fast online computation of a model predictive controller and its application to fuel economy–oriented adaptive cruise control. *IEEE Transactions on Intelligent Transportation Systems*, *16*(3), 1199–1209. http://dx.doi.org/10.1109/TITS.2014.2354052.

Liniger, A., Domahidi, A., & Morari, M. (2015). Optimization-based autonomous racing of 1: 43 scale RC cars. *Optimal Control Applications & Methods*, *36*(5), 628–647. http://dx.doi.org/10.1002/oca.2123.

Mattingley, J., & Boyd, S. (2012). CVXGEN: A code generator for embedded convex optimization. *Optimization and Engineering*, *13*(1), 1–27. http://dx.doi.org/10.1007/s11081-011-9176-9.

Mboup, M., Join, C., & Fliess, M. (2009). Numerical differentiation with annihilators in noisy environment. *Numerical Algorithms*, *50*(4), 439–467. http://dx.doi.org/10.1007/s11075-008-9236-1.

Menhour, L., d'Andréa-Novel, B., Fliess, M., Gruyer, D., & Mounier, H. (2017). An efficient model-free setting for longitudinal and lateral vehicle control: Validation through the interconnected pro SiVIC/RTMaps prototyping platform. *IEEE Transactions on Intelligent Transportation Systems*, *19*(2), 461–475. http://dx.doi.org/10.1109/TITS.2017.2699283.

Naus, G., Van Den Bleek, R., Ploeg, J., Scheepers, B., van de Molengraft, R., & Steinbuch, M. (2008). Explicit MPC design and performance evaluation of an ACC stop- and -Go. In *2008 American control conference* (pp. 224–229). http://dx.doi.org/10.1109/ACC.2008.4586495, June.

Novara, C., Formentin, S., Savaresi, S. M., & Milanese, M. (2016). Data-driven design of two degree-of-freedom nonlinear controllers: the D2-IBC approach. *Automatica*, *72*, 19–27. http://dx.doi.org/10.1016/j.automatica.2016.05.010.

Piche, S., Sayyar-Rodsari, B., Johnson, D., & Gerules, M. (2000). Nonlinear model predictive control using neural networks. *IEEE Control Systems Magazine*, *20*(3), 53–62. http://dx.doi.org/10.1109/37.845038.

Piga, D., Forgione, M., Formentin, S., & Bemporad, A. (2019). Performance-oriented model learning for data-driven MPC design. *IEEE Control Systems Letters*, *3*(3), 577–582. http://dx.doi.org/10.1109/LCSYS.2019.2913347.

Piga, D., Formentin, S., & Bemporad, A. (2017). Direct data-driven control of constrained systems. *IEEE Transactions on Control Systems Technology*, *26*(4), 1422–1429. http://dx.doi.org/10.1109/TCST.2017.2702118.

Polack, P., d'Andréa-Novel, B., Fliess, M., de La Fortelle, A., & Menhour, L. (2017). Finite-time stabilization of longitudinal control for autonomous vehicles via a model-free approach. *IFAC-PapersOnLine*, *50*(1), 12533–12538. http://dx.doi.org/10.1016/j.ifacol.2017.08.2191.

Selvi, D., Piga, D., & Bemporad, A. (2018). Towards direct data-driven model-free design of optimal controllers. In *2018 European control conference* (pp. 2836–2841). http://dx.doi.org/10.23919/ECC.2018.8550184, June.

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., & Boyd, S. (2018). OSQP: An operator splitting solver for quadratic programs. In *2018 UKACC 12th international conference on control* (p. 339). http://dx.doi.org/10.1109/CONTROL.2018.8516834, September.

Tuchner, A., & Haddad, J. (2017). Vehicle platoon formation using interpolating control: A laboratory experimental analysis. *Transportation Research Part C: Emerging Technologies*, *84*, 21–47. http://dx.doi.org/10.1016/j.trc.2017.06.019.

Wang, Z., Bai, Y., Wang, J., & Wang, X. (2019). Vehicle path-tracking linear-time-varying model predictive control controller parameter selection considering central process unit computational load. *Journal of Dynamic Systems, Measurement, and Control*, *141*(5), 051004. http://dx.doi.org/10.1115/1.4042196.

Wang, Z., Bai, Y., Zha, J., Wang, J., & Wang, X. (2019). Cooperative adaptive cruise control safety enhancement via dynamic communication channel selection. In *2019 American control conference* (pp. 521–526). http://dx.doi.org/10.23919/ACC.2019.8814653.

Wang, J., Mounier, H., Niculescu, S. I., Geamanu, M. S., & Cela, A. (2016). Event-driven model-free control in motion control with comparisons. *IMA Journal of Mathematical Control and Information*, *34*(4), 1255–1275. http://dx.doi.org/10.1093/imamci/dnw023.

Wang, H., Ye, X., Tian, Y., Zheng, G., & Christov, N. (2016). Model-free-based terminal SMC of quadrotor attitude and position. *IEEE Transactions on Aerospace and Electronic Systems*, *52*(5), 2519–2528. http://dx.doi.org/10.1109/TAES.2016.150303.

Wang, Z., Zha, J., & Wang, J. (2019). Flatness-based model predictive control for autonomous vehicle trajectory tracking. In *IEEE 22th international conference on intelligent transportation systems* (pp. 4146–4151). http://dx.doi.org/10.1109/ITSC.2019.8917260, October.

Weißmann, A., Görges, D., & Lin, X. (2018). Energy-optimal adaptive cruise control combining model predictive control and dynamic programming. *Control Engineering Practice*, *72*, 125–137. http://dx.doi.org/10.1016/j.conengprac.2017.12.001.