Transportation Research Part C 140 (2022) 103697

Contents lists available at ScienceDirect

Transportation Research Part C

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

Significance of low-level control to string stability under adaptive cruise control: Algorithms, theory and experiments

Hao Zhou [a], Anye Zhou [a], Tienan Li [b], Danjue Chen [b], Srinivas Peeta [a], Jorge Laval [a,*]

[a] School of Civil and Environmental Engineering, Georgia Institute of Technology, Atlanta, United States
[b] School of Civil and Environmental Engineering, University of Massachusetts Lowell, Lowell, United States

ARTICLE INFO

Keywords:
Commercial ACC
String stability
Low-level controller
Comma.ai
On-road experiments

ABSTRACT

Commercial adaptive cruise control (ACC) systems are bi-level: an upper-level planner decides the target trajectory and the low-level system executes it. Existing literature on ACCs mostly focus on the planner algorithms or the actuator delay, while the transition process between them, e.g. the low-level control design and its impact are often ignored. This paper tries to fill this gap by digging into the codebase of a recent open-source self-driving system, Openpilot (OP), Comma.ai, from which we extract and formulate the algorithms at both the upper and lower levels. For linear ACCs, the paper extends the transfer function analysis from planners only to full control loops and investigates the impact of slow/fast low-level control on the overall string stability (SS). For MPC ACCs, it studies their planning characteristics based its optimization objectives and approximates the low-level impact using an ODE approach.

We find that low-level control has a significant impact on the overall SS of ACCs: (i) slow low-level control undermines SS under small frequencies and improves SS given large frequencies for linear systems, (ii) MPC features a varying gain throughout an oscillation, where the fast low-level control typically results in a 'fast-slow' changing process of the MPC gain, which benefits the SS, whereas the slow low-level control leads to a 'slow-fast' varying gain which undermines the SS, (iii) slow low-level control are common as they arise from comfort-oriented control gains, from a "weak" actuator performance or both, and (iv) the SS is very sensitive to the integral gain under slow low-level control for both PI and PIF controllers. Overall, the study recommends fast low-level control for ensuring vehicular SS to reduce traffic congestion, considering that large congestion waves usually feature both small frequencies and large amplitudes, although slow controllers could perform even better provided a short and small leader perturbation. The findings of this paper are verified both numerically and experimentally. For the first time in the literature we implement custom ACC algorithms on market cars, and achieve SS on open roads with a random leader by only tuning the low-level controllers. The source code is shared at https://github.com/HaoZhouGT/openpilot to support on-road experiments of arbitrary car-following models, which may be of interest to other studies.

1. Introduction

With the development of vehicle automation, adaptive cruise control (ACC) systems are now widely available on commercial vehicles around the world. From the perspective of traffic flow efficiency, ACC systems are expected to achieve string stability (SS) to ensure that small perturbations do not amplify upstream within a platoon of vehicles (Shaw and Hedrick, 2007; Feng et al.,

* Correspondence to: 790 Atlantic Dr NW, Atlanta, GA, 30313, United States.
E-mail address: jorge.laval@ce.gatech.edu (J. Laval).

https://doi.org/10.1016/j.trc.2022.103697
Received 16 July 2021; Received in revised form 15 February 2022; Accepted 14 April 2022
Available online 19 May 2022
0968-090X/

2019), i.e., speed fluctuations should be dampened rather than amplified by the followers (Naus et al., 2010; Zhou and Peng, 2005; Zhou et al., 2020). In recent years, the SS of factory ACC systems has drawn increasing attention from the traffic flow community. Unfortunately, the factory ACC products can only be treated as "black boxes" due to the proprietary technology. As a result, related studies in the literature are mostly data-driven; i.e., researchers often collect the driving behavior data of factory ACC vehicles and then directly examine the SS features (e.g., Makridis et al. (2021), Li et al. (2021)) or indirectly study the features by calibrating a model (e.g., Gunter et al. (2019, 2020), Shi and Li (2021)) . For example, Gunter et al. (2020) found that most market ACC systems are string unstable. Li et al. (2021) show that factory ACCs can amplify or dampen an oscillation (and overshoot or undershoot after the oscillation), depending on the ACC headway setting, speed level, and leader trajectory. Shi and Li (2021) illustrate similar findings in that a large headway setting produces better SS while a small one usually induces perturbations to grow. Li (2020) indicates that this property of factory ACC suggests a trade-off between SS, mobility and safety. These empirical findings of "black box" factory ACC systems shed light on some of their key features, but lack insights from the perspectives of ACC controller design and execution.

Although the data-driven studies have shed new light on understanding the factory ACC systems, there are some limitations. Notably, in recent commercial car models, ACC functionality is often provided by radar manufacturers, e.g. Bosch, Continental, who sell the units to automakers. Those ACC units usually integrate a built-in ACC algorithm with the radar module. More specifically, an upper-level planner receives the updated information from the radar and plans for the optimal trajectory, and then a low-level controller executes the trajectory by sending low-level commands (gas/brake or acceleration) to the car control interface. Given this design feature of ACC, the data-driven approach is not ideal in uncovering ACC behaviors. First, they use similar GPS devices with a sampling rate of only 0.1 s, which is larger than the typical updating interval for both radar (0.05 s) and the low-level control system (0.01 s) in modern cars. This means that some vehicle maneuver information is potentially lost. Second, the data-drive approach only captures the holistic outcomes of ACC behaviors and cannot decouple the role of the upper-level planner and low-level controller as well as their interactions. Notably, the prevailing approach of modeling ACCs per (Gunter et al., 2019; Li, 2020; Shi and Li, 2021) essentially only calibrates the upper-level planner models against trajectory data while assuming the low-level controllers can achieve perfect performance as desired, which is questionable in real-world driving scenarios. In fact, Li (2020), Shi and Li (2021) calibrated the factory ACC using a parsimonious linear car following (CF) model and found that the model falls short in explaining some empirical ACC features. This highlights the needs of going deeper into the mechanic level of ACC controller design and execution.

Fortunately, a recently open-sourced factory ACC system, Openpilot (Comma.ai, 2021), here provides one way to uncover the details of the black box. Openpilot is developed by an after-market self-driving company, Comma.ai, who aims to be to Tesla what Android is to Apple, but with self-driving technology. Besides the open-source ACC software, Comma.ai also develops an after-factory ACC development kit, Comma Two, which can connect the stock ACC unit and override its control on many regular commercial cars. Note that Openpilot is considered to be factory level because it only overwrites the built-in algorithms in the ACC unit, as it still uses the stock sensor and communicates with existing car control interface. The benefits of such open-source ACC software and after-market hardware are unprecedented. Openpilot enables researchers to obtain full access to all the controller parameters, variables and algorithms of the ACC system. It also opens a new gate for analysis, design and test of our own factory ACC algorithms, which is aligned with the objective of this study.

Prior to the open-source factory ACC algorithms, the SS of ACC systems has already been extensively studied in the control area, e.g. Yanakiev and Kanellakopoulos (1995), Liang and Peng (1999, 2000). The theoretical SS condition for linear CF models have been examined for years since (Wilson and Ward, 2011), and one can notice that it is not difficult to meet the SS condition after tuning the model parameters. For example, Gunter et al. (2019) obtained a wide region of string-stable parameters for the optimal speed velocity speed model and verified the SS through simulation. Wilson and Ward (2011) has derived the string stability region for parameters of some CF models. Yet, two interesting questions arise: can we directly apply those tuned models to a car and achieve string-stable performance? Why are string-stable commercial ACCs so rare in real life?

Unfortunately, the current short answer to the first question is "no", and the second question can be explained by the lack of considerations for low-level controllers. Specifically, this study points out that the gap between SS theory and practice lies in the impact of low-level controller, which has been consistently neglected so far. For the field experiments in literature, researchers used either original equipment manufacturer (OEM) controller or in-house designed controllers. The OEM controller (Eilbert et al., 2020) directly tracks the acceleration/speed command, or torque/brake pressure generated from the upper-level planner but these algorithms are also 'black-boxes'. The in-house designed controllers use specific feedback control law to track the planning trajectories. For instances, Naus et al. (2010), Ploeg et al. (2011) used a 'MOVE' gateway to interact with the vehicle motor to achieve acceleration setpoints. Lu and Shladover (2018) applied the feedback linearization control technique to calculate the desired torque/brake pressure command of truck actuators for realizing desired accelerations. Shladover (2009) utilized the loop-shaping techniques to devise a 'speed servo' to track the planned speed. Although the in-house designed controllers demonstrate desired performance in the experiments, they are only applied to specific field-test scenarios, it remains unclear if they can be generalized to real-world driving. Additionally, most in-house designed controllers are not open-sourced either.

Correspondingly, the paper will focus on the impact of the factory low-level controller, this study first presents the open-source factory ACC algorithms, based on which we further investigate the impact of low-level controllers on SS. The findings are validated through both numerical simulations and real-car tests on a regular commercial car model, a 2019 Honda Civic.

The contribution of this paper is threefold: (i) on the algorithm side, for the first time it formulates the planner models and low-level control algorithms from the codes of an open-source commercial ACC system, Openpilot. Towards this it finds that the commercial planner models (e.g. linear and model predictive controller (MPC)) are significantly different from the traditional models

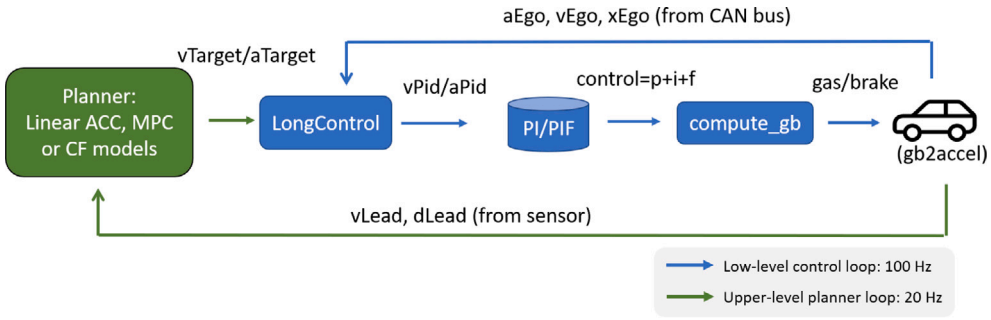


Fig. 1. Pipeline for the longitudinal control in Openpilot ACC.

in the literature. The paper also details the full control thread of the low-level system, including the algorithms to post-process upper-level planning targets, the controllers to track low-level setpoints, as well as the common actuator model used by commercial ACC products. (ii) the theory side, the paper extends the transfer function (TF) analysis for linear ACCs from planners only to bi-level loops including low-level control. That allows us to investigate the impact of slow/fast low-level control analytically. For MPCs, the paper identifies their characteristics in responses to oscillations and approximate the impact of slow/fast control with simplified models. (iii) For the experimental method, the paper contributes to a new data collection method to collect detailed ACC data from regular car models and make adaptions of Comma.ai's original code base to allow easy testing of any CF models on the road. Such data collection methods provide promising opportunities to explain/understand the behaviors of other black-box ACC systems. The new road experiment approach allows traffic flow researchers to test their models on the road, which can be of interest to many other studies.

The remainder of the paper is organized as follows: Section 2 introduces the open-source commercial ACC algorithms for both the planner and the low-level control; Section 3 and Section 4 investigates the impact mechanisms of low-level control on the SS of linear and MPC ACCs respectively; Section 5 presents the numerical and on-road experiments and showcase the results; Section 6 discusses the improvements for low-level control design for the better SS; and Section 7 concludes the paper.

2. ACC algorithms in Openpilot

2.1. Full control pipeline of factory ACCs

Before diving into the detailed algorithms for the upper-level planner and the low-level controller, it is important to first understand the full control pipeline of the ACC system, as shown in Fig. 1.

As mentioned earlier, a typical on-board radar runs at 20 hz. It provides the lead vehicle information (e.g. the lead vehicle speed v_{lead} and the spacing s_{ego}) for the planner, with or without the sensor fusion from cameras. The planner listens to the sensors and responds to the speed and spacing changes by adjusting the target speed, v_{target}, or the target acceleration, a_{target} for the ego vehicle. The planner model is usually a linear controller or an MPC more recently.

The low-level controller operates to achieve the planner targets by producing the low-level commands (gas/brake), which are usually at the rate of 100 hz for recent car models. Due to the higher frequency, the low-level controller forms an inner loop of 5 steps in each planning period (0.05 s). As Fig. 1 shows, the low-level control loop consists of four major steps: (i) first, the low-level control algorithm, 'LongControl', calculates the low-level setpoint, v_{pid} or a_{pid} or both, at each of the five control steps using the most-recent upper-level planner target v_{target} or a_{target}; (ii) then a proportional–integral (PI) or proportional–integral–feedforward (PIF) controller tracks the low-level setpoints and outputs the $a_{control}$ demand; (iii) the $a_{control}$ demand is then fed to a $'compute\_gb'$ function which maps it to the final actuator command, i.e. a gas or brake percent gb for the car; and (iv) finally the gas/brake percentage is applied and moves the vehicle. The true acceleration a_{ego}, speed v_{ego}, position x_{ego} of the vehicle are send back to the loop for the next step.

Here a PI or PIF controller outputs the $a_{control}$ demand, which is further processed by a $compute\_gb$ function to generate the final gb commands. Note that the $control$ can also be understood as the desired acceleration, thus the $compute\_gb$ function is essentially a mapping from the desired acceleration to the necessary gas/brake. Reversely, we refer this mapping from the applied gas/brake gb to the true acceleration a_{ego} as the $gb2accel$ function, which is essentially a model of the engine and brake system.

2.2. Upper-level planner algorithms

For upper-level planners, linear and non-linear models like MPC are common. A linear ACC controller was first used in the earlier versions of OP, and later replaced by a MPC planner. According to the source code in OP, we now formulate the detailed equations for both planners.

2.2.1. The OP linear planner

The OP linear planner adopts a constant time headway policy (CTH), but different from the literature, the desired spacing is calculated using the leader, not the ego vehicle speed. Given a desired time headway τ, and the jam spacing δ, the desired spacing s_{des} from the front bumper of the ego vehicle to the rear bumper of the lead vehicle is calculated as:

$$s_{\text{des}}(t) = \delta + \tau \cdot v_{\text{lead}}(t) \tag{1}$$

Based on the desired spacing, the planner then outputs a target speed v_{target} to reduce the spacing error $\Delta s = s_{ego} - s_{des}$ between the true spacing s_{ego} and the desired spacing s_{des} by the gain k.

$$v_{\text{target}}(t) = k \cdot (s_{ego}(t) - s_{des}) + v_{\text{lead}}(t) = k \cdot \Delta s(t) + v_{\text{lead}}(t) \tag{2}$$

Note that the parameter k physically means how fast the planner tries to adjusts the spacing variability. In OP k is speed-dependent, piecewise linear and decreases with speed values, which is different from the constant assumptions commonly used in existing studies (Gunter et al., 2019; Zhou and Ahn, 2019; Li, 2020; Shi and Li, 2021).

It is also worth noting that here the OP linear ACC model uses the lead vehicle speed v_{lead} to calculate the desired spacing and plan for the target speed, which is different from the tradition CTH in the literature that adjusts the desired spacing based on the ego speed v_{ego}. While the true motivations are unclear yet, we notice that such design enables the planner to run independently from the low-level responses such as v_{ego} or a_{ego}. Similar design is also found in MPC-type planner, which will be introduced shortly. We conjecture that using v_{lead} in the planner can also reduce the hardware communication between the ACC unit and the vehicle CAN (Controller Area Network) bus since no low-level variables need to be retrieved from the CAN bus. The v_{lead} is directly measured from the radar in the ACC unit, which helps the ACC module to be self-contained. From the perspective of SS, the difference between using v_{lead} and v_{ego} is not trivial. We show that the factory linear ACC can be easily string stable in the next section.

2.2.2. MPC planner

MPC planners have become ubiquitous in the literature since they allow the optimization of more general objective functions while considering more refined vehicle dynamics models (Corona and De Schutter, 2008; Naus et al., 2008; Li et al., 2010; Gong et al., 2016; Zheng et al., 2017). This extra computational burden comes at the cost of requiring a professional solver (e.g. Diehl, 2014) to run in real time. The formulation of the optimization objective and the reference trajectory are two key components for an MPC problem, as shown next.

In Openpilot the objective function $C(\hat{t})$ for the longitudinal MPC at the planning time $\hat{t}$ is defined as a weighted sum of four sub-costs, with respect to the time to collision, spacing, acceleration and jerk values:

$$C(\hat{t}) = \sum_{\hat{t} \le t_k \le \hat{t} + T_{\text{MPC}}} w_{ttc} C_{ttc}^2(t_k) + w_{dist} C_{dist}^2(t_k) + w_{accel} C_{accel}^2(t_k) + w_{jerk} C_{jerk}^2(t_k) \tag{3}$$

where t_k denotes the time of each discrete planning step in each planning horizon T_{MPC}; see dotted points in Fig. 2. In OP, the 10-second planning horizon is discretized into 20 intervals, where the first 5 intervals have a step length of 0.2 s, and the remaining 15 intervals are share a step length of 0.6 s. The tunable weights applied here are $w_{ttc} = 5$, $w_{dist} = 0.1$, $w_{accel} = 10$, $w_{jerk} = 20$. Omitting the subscript of t_k, the four sub-costs are defined below:

$$C_{ttc}(t) = \exp\{\frac{s_{\text{des}}(t) - s_{\text{ego}}(t)}{(\sqrt{v_{\text{ego}}(t) + 0.5} + 0.1)/0.3}\} - 1 \tag{4}$$

$$C_{dist}(t) = \frac{s_{\text{des}}(t) - s_{\text{ego}}(t)}{0.05 v_{\text{ego}}(t) + 0.5} \tag{5}$$

$$C_{accel}(t) = a_{\text{ego}}(t)(0.1 v_{\text{ego}}(t) + 1) \tag{6}$$

$$C_{jerk}(t) = j_{ego}(t)(0.1 v_{\text{ego}}(t) + 1) \tag{7}$$

where j_{ego} is the jerk of ego vehicle (i.e., derivative of acceleration). The desired spacing s_{des} for the MPC controller is defined as:

$$s_{\text{des}}(t) = v_{\text{ego}}(t) \cdot \tau - (v_{\text{lead}}(t) - v_{\text{ego}}(t)) \cdot \tau + \frac{v_{\text{ego}}(t)^2 - v_{\text{lead}}(t)^2}{2B} \tag{8}$$

where B is the maximum deceleration rate.

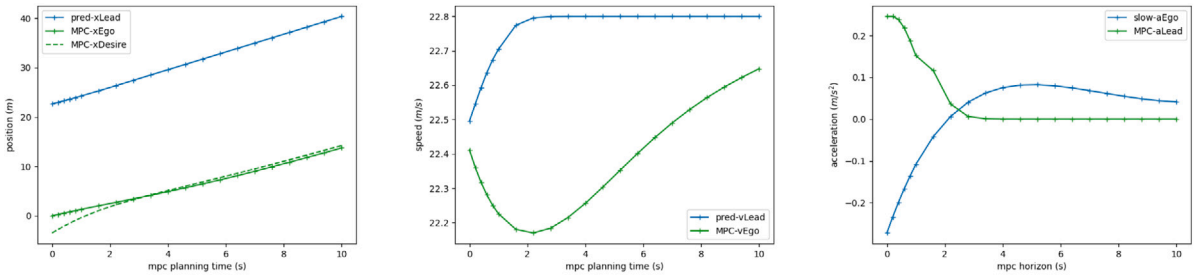
The reference trajectory is the estimated lead vehicle trajectory in the prediction horizon T_{MPC} (10.0 s). Let dt_p denote the varying step length (s) between two consecutive stops in the planning horizon T_{MPC}, the future lead trajectory is estimated using the dynamics model in 5 (a–d), which assumes a decaying acceleration of the lead vehicle with time parameter h (1.5 s). Starting from $t = 0$ to $t = T_{\text{MPC}}$, the lead vehicle states in the planning horizon are predicted and updated as follows:

$$a_{\text{lead}}(t) = a_{\text{lead}}(\hat{t}) \exp(-h \cdot t^2/2) \tag{9a}$$

$$x_{\text{lead}}(t) = x_{\text{lead}}(t) + v_{\text{lead}}(t) \cdot dt_p \tag{9b}$$

$$v_{\text{lead}}(t) = v_{\text{lead}}(t) + a_{\text{lead}}(t) \cdot dt_p \tag{9c}$$

$$t := t + dt_p \tag{9d}$$



(a) prediction of x_{lead} and solution of x_{ego} (b) prediction of v_{lead} and solution of v_{ego} (c) prediction of a_{lead} and solution of a_{ego}

Fig. 2. Example of MPC's prediction and optimization: x_{Desire} is the predicted position of the leader x_{lead} minus the desired spacing s_{des}. The leader acceleration a_{lead} is from model (9d). The solution of a_{ego}, v_{ego} and x_{ego} are subject to the MPC optimization problem (10).

By solving the MPC problem (10) at each time step $\hat{t}$, we can obtain the $a_{\text{target}}(\hat{t}+dt_p)$, i.e. the target acceleration of the MPC at the first step of the optimization solution:

$$\begin{aligned} \min_{a_{\text{target}}} \quad & C(\hat{t}) \\ \text{s.t.} \quad & \text{Discrete point-mass dynamics for vehicle longitudinal movements} \\ & \text{Predicted dynamics for the lead vehicle in (9)} \\ & v_{\text{target}} \geq 0 \end{aligned} \tag{10}$$

2.3. Low-level control system

2.3.1. Algorithms in longitudinal control to update low-level setpoints

The pipeline in Fig. 1 shows that the first step in the low-level control loop is to process the planner targets and calculate the corresponding low-level setpoints. Note that the low-level setpoints are the true references for the car to track, therefore it is important to understand how the algorithms update them using the planning targets.

If the planner gives a target speed v_{target}, the low-level controller usually uses a speed setpoint v_{pid} and a PI controller to track it. Let $\hat{t}$ denote the planning time when the sensor and planner get updated every 0.05 s, and t is the low-level control time with step size $dt = 0.01$ s. The algorithm to compute v_{pid} for the PI controller is shown in Algorithm 1 with initial condition $v_{\text{pid}}(0) = v_{\text{ego}}(0)$.

Algorithm 1 Low-level controller algorithm for v_{pid} with a linear planner

Input: most recent v_{target}; a_{max}, a_{min}; a constant overshoot allowance $oa = 2.0$.
Output: v_{pid}
Initialisation: $v_{pid}(0) = v_{ego}(0)$
Low-level iteration to update v_{pid}
if $v_{\text{pid}} > v_{\text{ego}} + oa$ and $v_{\text{target}} < v_{\text{pid}}$ **then**
  $v_{\text{pid}} \leftarrow \max(v_{\text{target}}, v_{\text{ego}} + oa)$
else if $v_{\text{pid}} < v_{\text{ego}} - oa$ and $v_{\text{target}} > v_{\text{pid}}$ **then**
  $v_{\text{pid}} \leftarrow \min(v_{\text{target}}, v_{\text{ego}} - oa)$
end if
if $v_{\text{target}} > v_{\text{pid}} + a_{max} \cdot dt$ **then**
  $v_{\text{pid}} \leftarrow v_{\text{pid}} + a_{max} \cdot dt$
else if $v_{\text{target}} < v_{\text{pid}} + a_{min} \cdot dt$ **then**
  $v_{\text{pid}} \leftarrow v_{\text{pid}} + a_{min} \cdot dt$
else
  $v_{\text{pid}} \leftarrow v_{\text{target}}$
end if
update $v_{\text{ego}} \leftarrow v_{\text{ego}}$ from CAN bus

In short, the above algorithm 1 shows the current $v_{\text{pid}}(t)$ updates itself by moving towards the given planner target $v_{\text{target}}(\hat{t})$ at a maximum rate but bounded by constraints that include acceleration limits.

If the upper-planner outputs the upper-level desired acceleration a_{target}, the low-level control loop usually outputs the acceleration setpoint a_{pid} along with v_{pid}. The acceleration setpoint is designed for the feedforward term in a PIF controller. It is worth noting that OP introduces two more variables $v_{\text{start}}(\hat{t})$ and $a_{\text{start}}(\hat{t})$ in lieu of v_{target} and a_{target} as the references to calculate low-level setpoints a_{pid} and v_{pid}. The v_{start} and a_{start} are updated using the following rule:

$$a_{\text{start}} \leftarrow a_{\text{start}} + d\hat{t}/dt_p(a_{\text{target}} - a_{\text{start}}) \tag{11}$$

$$v_{\text{start}} \leftarrow v_{\text{start}} + d\hat{t}(a_{\text{target}} + a_{\text{start}})/2 \tag{12}$$

where $d\hat{t}$ denotes the time step for the planner to update (0.05 s). We conjecture those surrogate variables v_{start} and a_{start} are used to avoid potentially large compounding errors when directly incorporating v_{target} or a_{target}. In Algorithm 2, we show the algorithm for computing the low-level setpoints a_{pid} and v_{pid} using the a_{target}, v_{start} and a_{start}. Initially, we reset $v_{\text{start}}(0) = v_{\text{ego}}(0)$ and $a_{\text{start}}(0) = a_{\text{ego}}(0)$.

Algorithm 2 Low-level controller algorithm for a_{pid}, v_{pid} with an MPC planner

Input: most recent a_{target}, a_{start}, v_{start}
Output: a_{pid} and v_{pid}
Low-level loop for $v_{\textbf{pid}}(t), a_{\textbf{pid}}(t)$ **for** $t \in \{\hat{t}, \hat{t} + 0.01 ... \hat{t} + d\hat{t}\}$**:**
1: $dt = t - \hat{t}$
2: $a_{\text{pid}}(t) = a_{\text{start}} + dt(a_{\text{target}} - a_{\text{start}})/dt_p$
3: $v_{\text{pid}}(t) = v_{\text{start}} + dt(a_{\text{pid}}(t) + a_{\text{start}})/2$

2.3.2. Low-level controller

Now we introduce how the PI/PIF controllers track those low-level setpoints. Define the true speed at each control time t as v_{ego}, the speed error as $e(t) = v_{\text{pid}}(t) - v_{\text{ego}}(t)$, the formulation of PI/PIF control input is expressed as:

$$a_{control}(t) = k_p \cdot e(t) + k_i \cdot \int_0^t e(\tau)d\tau + k_f \cdot a_{\text{pid}}(t) \tag{13}$$

where k_p, k_i and k_f correspond to the control gain for the proportional (P), integral (I) and feedforward (F) terms. Note that the default control gains are speed-dependent in Openpilot, where $k_p = k_p(v)$ and $k_i = k_i(v)$; see Fig. 19. Similar to the parameter k in (2), P and I gains are also piecewise linear and smaller at higher speeds. The k_f equals to 1 in theory of the feed-forward control. The first two terms in (13) construct a PI controller, and a feedforward term is added to form a PIF controller if the acceleration setpoint is available. In the PI controller, the control gains are also varying under risky scenarios for safety reasons, where k_p and k_i are increased by 20% for every 1 m/s^2 of the required braking deceleration smaller than -1 m/s^2.

2.3.3. Actuator model and gas/brake estimator

For the $a_{control}$ demand, one has to calculate the gas/brake percentage gb for the actuator to execute it. To do so, we need a good understanding of the actuator performance, i.e. the acceleration produced by a gb value, or the $gb2accel$ function.

$$a_{\text{ego}} = gb2accel(gb, v_{ego}) \tag{14}$$

The $gb2accel$ function represents vehicle actuator model, which is highly nonlinear and dependent on current vehicle speed, acceleration, road grade, wind, vehicle loads, and the control performance of engine/motor, transmission, and brake etc. The detailed model of powertrain dynamics can be found in Rajamani et al. (2000), Li et al. (2016). Note that the torque supply of engine/motor, and the braking pressure are controlled under the factory settings of automakers, which are typically difficult to be accessed. The $gb2accel$ function for a specific car model can be roughly fitted from its driving data. One can test several gb values under different speed levels and record the produced true accelerations. Then a linear function or neural networks can be used to fit the model. We will later show on many recent car models, the $gb2accel$ is approximately a simple scaling function of the gb only, for example, $a_{\text{ego}} = 3gb$, and accordingly the $compute\_gb$ function should share the same scale factor in the denominator, e.g. $gb = a_{\text{ego}}/3$.

Based on the actuator model $gb2accel$, we can estimate the required gb given a desired acceleration using its inverse function, i.e. the estimator function $compute\_gb$, which calculates the desired gas/brake for the $a_{control}$ demand from (13):

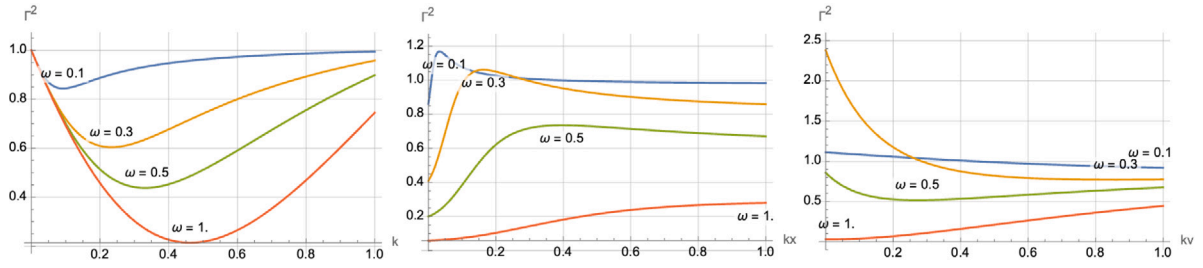
$$gb = compute\_gb(a_{control}, v_{\text{ego}}) \tag{15}$$

where $gb \in [-1, 1]$ because the brake is negative. Recalling that $a_{control}$ is the desired acceleration, to obtain the corresponding gb, the $compute\_gb$ must be in accordance with the $gb2accel$ function. We consider the gas/brake estimator $compute\_gb$ is perfect if the estimated gb always generates the same amount of the true acceleration a_{ego} as the acceleration control demands:

$$a_{\text{ego}} = \beta \cdot a_{control} \tag{16}$$

which holds obviously when $gb2accel$ and $compute\_gb$ are both simple scaling functions with the reciprocal parameters, e.g 3 and 1/3. Since the true dynamics or the simplified scale factor in $gb2accel$ is usually unknown to us, the gas/brake estimator $compute\_gb$ can be designed to have smaller or larger scale factors which causes the true acceleration overshoot or undershoot the $a_{control}$ demand. To further study their impacts, we define the actuator system is "strong" if $|a_{\text{ego}}| > |a_{control}|$, or "weak" if $|a_{\text{ego}}| < |a_{control}|$.

Now we have formulated the upper-level planners and the low-level control systems used in OP. To summarize, OP has introduced two types of planners: a linear speed planner that outputs the target speed v_{target}, and a non-linear MPC planner that gives the target acceleration a_{target}. The respective low-level system first processes v_{target} or a_{target} and transforms it to feasible and continuous low-level setpoints v_{pid} or a_{pid}. Then a speed (PI) or acceleration (PIF) low-level controller is used to track v_{pid}, or a_{pid} or both. The low-level controller calculates the $control$ command, based on which the gas/brake estimator $compute\_gb$ calculates the corresponding gas/brake percentages gb. Finally the actuator plant, i.e. $gb2accel$, executes the gb and generates the true accelerations a_{ego} to move the vehicle.



(a) OP linear planner: $\tau = 1.7$ (b) General linear planner: Γ^2 v.s. k_x when $k_v = 0.25$ (c) General linear planner: Γ^2 v.s. k_v when $k_x = 0.15$

Fig. 3. Comparing SS of OP and general linear planners and the impact of gains and frequencies.

We conjecture the similar low-level controller should widely exist in market ACC systems: (i) the frequency gap between the planning (sensor) and execution (actuator) needs the car to have a separate tracking reference for the low level; (ii) the planner outputs are usually prone to fluctuations due to sensor measurement error or imperfect execution, thus they cannot be directly fed to actuator; (iii) a controller is certainly necessary to help track the setpoints and PI/PIF are typically go-to options if they are not MPCs.

3. Impact mechanism of slow/fast low-level control on SS of linear ACCs

Traditionally, the SS of bi-level ACCs is only captured using planner's model. While the imperfect low-level control inevitably changes the whole dynamics, we conjecture the slow/fast low-level control could cause an effect similar to increased/decreased the planner gains. To investigate the impact of low-level control on the overall SS of ACCs, it would be worthwhile to study the SS of planners first, and see how it could be changed by a small variation of the planner gain. In the above section, we have introduced the OP linear speed planner, which is new to the literature. In this section, we will first study its SS properties and then investigate whether the low-level control can be fully captured by the planner model only. To see this, we will extend the SS analysis from the mere planner to bi-level loops including the low-level control. Notice that OP linear planner is a unique planner that outputs the target speed and pairs with a PI low-level controller, without loss of generality we also investigate the popular linear ACC planner widely adopted in the literature, which outputs target accelerations and is followed by a PIF low-level controller.

3.1. SS characteristics of pure linear planners

As stated earlier, the OP linear planner is different from the traditional linear CF models in the literature, which features a unique CTH policy that depends on the leader speed v_{lead}, not the ego speed v_{ego}. Further, it is a speed planner that directly outputs a desired target speed, not the acceleration, for the low-level system to track. Therefore, such planner does not fall into the range of traditional linear CF models, whose SS is worth more investigation.

We first convert the time-domain linear speed planner in (2) into frequency domain:

$$v_{\text{target}}(j\omega) = k[x_{\text{lead}}(j\omega) - x_{\text{ego}}(j\omega) - \tau v_{\text{lead}}(j\omega)] + v_{\text{lead}}(j\omega) \tag{17}$$

where ω is the angular frequency, j is the complex number indicator. As here we focus merely on the planner, we assume $v_{\text{ego}} = v_{\text{target}}$. Then substituting in (1), and applying the position as the integrator of corresponding speed: $x_{\text{lead}}(j\omega) = v_{\text{lead}}(j\omega)/j\omega$, and $x_{\text{ego}}(j\omega) = v_{\text{ego}}(j\omega)/j\omega = v_{\text{target}}(j\omega)/j\omega$, we can rearrange (17) into the speed to speed transfer function Γ^{OP} using the SS analysis approach in Wilson and Ward (2011), Montanino and Punzo (2021):

$$\Gamma^{\text{OP}} = \frac{v_{\text{target}}(j\omega)}{v_{\text{lead}}(j\omega)} = \frac{j(1-k\tau)\omega + k}{j\omega + k} \tag{18}$$

The squared 2-norm of the transfer function Γ^{OP} equals to:

$$\|\Gamma^{\text{OP}}\|^2 = \left\| \frac{v_{\text{target}}(j\omega)}{v_{\text{lead}}(j\omega)} \right\|_2^2 = \frac{[(1-k\tau)\omega]^2 + k^2}{\omega^2 + k^2} \leq 1 \tag{19}$$

If $|1 - k\tau| \leq 1$, i.e. $0 \leq k\tau \leq 2$, we have $\Gamma^{\text{OP}} \leq 1$, where the planner is string stable, and $\omega \to 0$ gives the maximum $\Gamma^{\text{OP}} = 1$. When $k\tau > 2$, the planner is string unstable and Γ^{OP} increases with the angular frequency ω, where $\omega \to 0$ gives the minimum transfer ratio.

The impact of gain k and the frequency ω on the SS is shown in Fig. 3(a). Noticeably, the increasing/decreasing interval depends on the angular frequency ω, which means the oscillation wavelength determines the impact of a decreased/increased gain. Interestingly, the convex pattern of the OP linear's TF suggests that if the oscillation frequency ω is known in advance, we can derive the optimal gain k^* for the ACC follower to best dampen the wave:

$$\frac{d\|\Gamma^{\text{OP}}\|^2(k)}{dk} = \frac{2\tau\omega(k^2 + (-1+k\tau)\omega^2)}{(k^2+\omega^2)^2} = 0 \Rightarrow k^* = \frac{1}{2}(-\tau\omega^2 + \sqrt{\omega^2(4+\tau^2\omega^2)}) \tag{20}$$

which suggests that predicting the leader oscillation wavelength can help the follower dynamically adjust the response rate in order to reduce speed perturbations. This feature might be similar to what human drivers are capable of.

The general linear acceleration planners have been widely used in ACC literature (Wilson and Ward, 2011; Gunter et al., 2019), which has control gains k_x and k_v with respect to both spacing and speed errors, and such linear controllers output desired accelerations instead of speeds. The linear ACC planner is typically formulated as follows:

$$a_{target}(t) = k_x(s_{ego}(t) - (\tau \cdot v_{ego}(t) + \delta)) + k_v(v_{lead}(t) - v_{ego}(t)) \tag{21}$$

The squared 2-norm of its TF has already been derived in previous studies as:

$$\|\Gamma^{PD}\|^2 = \frac{\omega^2 k_v^2 + k_x^2}{\omega^2(k_v + k_x\tau)^2 + (k_x - \omega^2)^2} \tag{22}$$

Opposite to the linear speed planner in OP, the TF of the general linear acceleration planner shows a concave pattern; see Fig. 3(b). Also the increase in the spacing gain k_x and the speed gain k_v around their default values can lead to opposite impacts on the SS. The impact of spacing gain changes on TF is more sensitive to the oscillation frequency ω, since the common value $k_x = 0.15$ could fall into either the increasing or the decreasing interval when ω is small. It suggests increasing k_x improves SS under small frequency such as $\omega = 0.1$ or $\omega = 0.3$, but undermines SS when $\omega = 0.5$ or $\omega = 1.0$. By contrast, the TF is less sensitive to the changes in speed gain k_v, whose common value is around 0.25 and seems to always improve the SS unless the frequency is large, e.g. $\omega = 1.0$.

Interestingly, the TF of linear planners all show the convergence trend with gains. Mathematically, we have:

$$\lim_{k\to\infty}\|\Gamma^{\mathrm{OP}}\|^2 = 1 + \tau^2\omega^2 \tag{23}$$

$$\lim_{k_x\to\infty}\|\Gamma^{PD}\|^2 = \frac{1}{1 + \tau^2\omega^2} \tag{24}$$

$$\lim_{k_v\to\infty}\|\Gamma^{PD}\|^2 = 1 \tag{25}$$

although regular values for those gains are around 0.2, (23) and (24) could partially explain why the impact of gain k and k_x are more related to oscillation frequency ω and time headway τ. Eq. (25) suggests controlling the speed difference is an easy solution to the marginal SS but it does not guarantee the safe spacing.

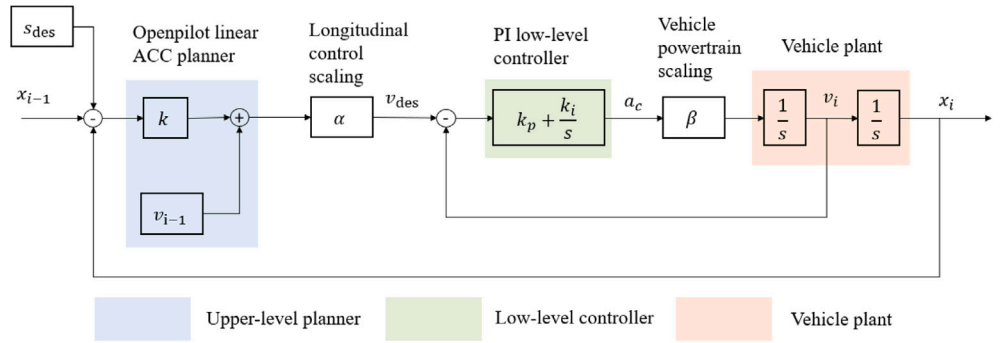
The above SS analysis of linear planners suggest that the OP linear speed planner has unique features against general acceleration planners in following aspects: (i) the string stable condition holds for almost all feasible values of the only control gain k, (ii) there exists an optimal k for best SS provided the oscillation frequency ω, and (iii) increasing the gain k deteriorates the SS under small ω and improves SS under large ω, which has an opposite pattern with general linear ACC planners. The common feature of both the OP and general linear ACC is that they both show the variation in spacing gains, i.e. k and k_x, can significantly affect the SS and such impact is closely related to the angular frequency ω, or equivalently the wavelength, of a leader oscillation.

The above analysis adopts the traditional TF method in the literature to study how linear ACC's SS is affected. Suppose the slow/fast low-level control decreases/increases the upper-level planner gains as an overall effect, the above analysis could offer some insights of the low-level impact mechanism. However, such assumption needs more justification and also fails to capture the full interaction between two levels, which motivates us to study the SS of full bi-level loops of ACC systems.

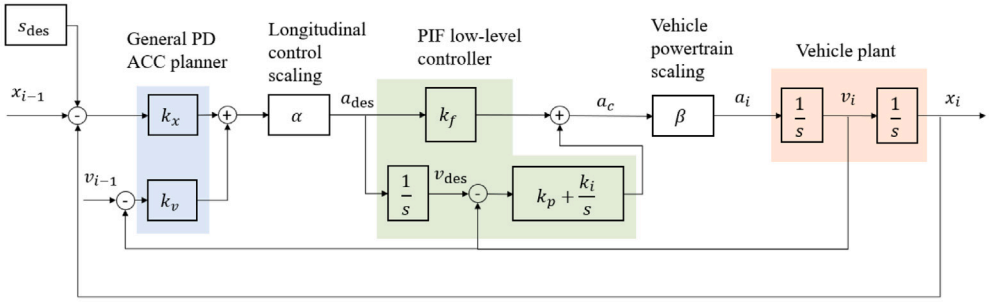
3.2. SS theory for full ACC control loops

The above analysis of SS for planners casts some light into the impact mechanism of slow/fast low-level control by assuming they behave as decreased/increased planner gains. however, the true impact of low-level control could be more complex considering the ACC system is a closed feedback control loop, where the execution of low-level controller delivers signal back to the upper-level planner, while the upper-level planner also sends signal to the low-level controller. The interaction between these two levels makes the overall dynamic difficult to track, and errors become compounding due to the feedback mechanism. Additionally, the intermediate modifications from the factory longitudinal control algorithm, and the imperfect execution of vehicle actuator in the real world also contributes to the difficulty of analysis the string stability of an ACC system. In this section, with some proper simplifications, we try to derive the SS of the bi-level control loop in linear ACC systems, and then investigate how slow/fast low-level control affects their overall SS.

Fig. 4(a) illustrates a simplified control loop of an ACC system consisting of the OP linear planner (2) and the PI low-level controller (labeled as OP-PI ACC system). The α capture the effect of the longitudinal control algorithm transforming upper level planner targets to low-level setpoints, $\alpha = v_{pid}/v_{target}$ for speed planner or $\alpha = a_{pid}/a_{target}$ for the acceleration planner. The β is another factor representing the imperfect control execution of the vehicle actuator possibly due to some real-world disturbances such as grades, i.e. $\beta = a_{ego}/a_{control}$ meanwhile β can also be understood as a multiplier to scale the default controller gains, which makes controller faster when $\beta > 1$. Following the similar style, Fig. 4(b) shows a simplified control loop of an ACC system consists of the general linear acceleration planner (21), i.e. (PD) upper-level planner, and the PIF low-level controller (labeled as PD-PIF ACC system).



(a) Feedback control loop of the OP-PI ACC system



(b) Feedback control loop of the PD-PIF ACC system

Fig. 4. Feedback control loops of linear ACC systems.

Then, following the feedback control loop in Fig. 4(a), we derive the TF of the full OP-PI ACC system following the procedures of TF derivation in Åström and Murray (2008). Following the feedback loop from left to right, we can write the expression of the position of ego vehicle $X_i(s)$ in the Laplace domain as:

$$X_i(s) = \beta(k_p + \frac{k_i}{s})\left[\alpha(k(X_{i-1}(s) - X_i(s) - \tau V_{i-1}(s)) + V_{i-1}(s)) - V_i(s)\right]\frac{1}{s^2} \tag{26}$$

Then, with s as the differential operator and $1/s$ as the integral operator, we can substitute $V_i(s) = sX_i(s)$, and $V_{i-1}(s) = sX_{i-1}(s)$ into (26) to perform some arithmetic simplification. Correspondingly, we can obtain the TF as:

$$\Gamma^{\text{OP-PI}}(s) = \frac{X_i(s)}{X_{i-1}(s)} = \frac{\alpha\beta[(k_p - kk_p\tau)s^2 + (kk_p + k_i - kk_i\tau)s + kk_i]}{s^3 + \beta k_p s^2 + \beta(\alpha k k_p + k_i)s + \alpha\beta k k_i} \tag{27}$$

Substituting in $s = j\omega$, the corresponding squared L2-norm of (27) is:

$$\|\Gamma^{\text{OP-PI}}(j\omega)\|_2^2 = \frac{\alpha^2\beta^2(k_i^2 + k_p^2\omega^2)(\omega^2 + k(k - 2\omega^2\tau + k\omega^2\tau^2))}{(\alpha\beta k k_i - \beta k_p\omega^2)^2 + (-\beta(k_i + \alpha k k_p)\omega - \omega^3)^2} \tag{28}$$

Similarly, following the feedback control loop in Fig. 4(b), we derive the TF as of the PD-PIF bi-level ACC loop:

$$\Gamma^{\text{PD-PIF}}(s) = \frac{\alpha\beta[k_v k_f s^3 + (k_x k_f + k_v k_p)s^2 + (k_x k_p + k_v k_i)s + k_x k_i]}{s^4 + \beta[K_{\text{TF},1}s^3 + K_{\text{TF},2}s^2 + \alpha K_{\text{TF},3}s + \alpha k_x k_i]} \tag{29}$$

where $K_{\text{TF},1} = k_p + \alpha k_x k_f\tau + \alpha k_v k_f$, $K_{\text{TF},2} = k_i + \alpha k_v k_p + \alpha k_x k_p\tau + \alpha k_x k_f$, $K_{\text{TF},3} = k_x k_p + k_x k_i\tau + k_v k_i$. The corresponding squared 2-norm of (29) is expressed as:

$$\|\Gamma^{\text{PD-PIF}}(j\omega)\|_2^2 = \frac{[\alpha\beta k_x k_i - (\alpha\beta k_x k_f + \alpha\beta k_v k_p)\omega^2]^2 + [(\alpha\beta k_x k_p + \alpha\beta k_v k_i)\omega - \alpha\beta k_v k_f\omega^3]^2}{[\alpha\beta K_{\text{TF},3}\omega - \beta K_{\text{TF},1}\omega^3]^2 + [\alpha\beta k_x k_i - \beta K_{\text{TF},2}\omega^2 + \omega^4]^2} \tag{30}$$

Now we have derived the TFs for full ACC loops with both upper-level planners and low-level control system. Interestingly, we find the TF of those pure planners mentioned earlier are just special evaluations of the TFs of the full ACC loops we derived here. To show this, from (28), we first assume the integral gain $k_i = 0$ and $\alpha = 1$, and let $k_p' = k_p\beta$ which reduces the full loop TF to:

$$\|\Gamma^{\text{OP-PI}}(j\omega, k_i = 0, \alpha = 1)\|_2^2 = \frac{(kk_p')^2\omega^2 + \omega^4(k_p' - kk_p'\tau)^2}{k_p'^2\omega^4 + (kk_p'\omega - \omega^3)^2} = \frac{k^2 + \omega^2(1 - k\tau)^2}{\omega^2 + (k - \frac{\omega^2}{k_p'})^2} \tag{31}$$

Taking the limit of k'_p gives us:

$$\lim_{k'_p\to\infty} \|\Gamma^{\text{OP-PI}}(j\omega, k_i = 0, \alpha = 1)\|_2^2 = \frac{[(1-k\tau)\omega]^2 + k^2}{\omega^2 + k^2} \tag{32}$$

which is the exact the TF of the OP linear planner we derived earlier in (19). It also suggests that only fast enough PI low-level control can achieve the desired SS.

Similarly, for general linear ACCs, we assume $\alpha = 1$ and $k_i = 0$, which gives us:

$$\|\Gamma^{\text{PD-PIF}}(k_p = 0)\|_2^2 = \frac{\beta^2 k_f^2 k_x^2 \omega^4 + \beta^2 k_f^2 k_v^2 \omega^6}{\beta^2 k_f^2 (k_v + k_x\tau)^2\omega^6 + (-\beta k_f k_x \omega^2 + \omega^4)^2} = \frac{k_x^2 + \omega^2 k_v^2}{\omega^2(k_v + k_x\tau)^2 + (k_x - \frac{\omega^2}{\beta k_f})^2} \tag{33}$$

$$\|\Gamma^{\text{PD-PIF}}(k_f = 0)\|_2^2 = \frac{\beta^2 k_p^2 k_x^2 \omega^2 + \beta^2 k_p^2 k_v^2 \omega^4}{(-\beta k_p k_x \omega + \beta k_p \omega^3)^2 + (-\beta k_p (k_v + k_x\tau)\omega^2 + \omega^4)^2} = \frac{k_x^2 + \omega^2 k_v^2}{\omega^2(k_v + k_x\tau - \frac{\omega^3}{\beta k_p})^2 + (k_x - \omega^2)^2} \tag{34}$$

Let $k'_f = \beta k_f$ and $k'_p = \beta k_p$, we have:

$$\lim_{k'_p\to\infty} \|\Gamma^{\text{PD-PIF}}(k_f = 0, k_i = 0)\|_2^2 = \|\Gamma^{\text{PD-PIF}}(k_p = 0, k_i = 0, k'_f = 1) = \frac{\omega^2 k_v^2 + k_x^2}{\omega^2(k_v + k_x\tau)^2 + (k_x - \omega^2)^2} \tag{35}$$

which also indicates the TF of the general linear ACC planner in (22) is just a special evaluation of the bi-level ACC system given the perfect low-level control.

3.3. Impact of low-level control on SS of linear ACCs

As mentioned earlier, β can be understood as the scale for the low-level controller gains. Before studying the impact of low-level control gains, we are interested in if a scaled low-level controller gain has the same or similar effect with the scaled planner gains with the same factor β.

To check this, apply the scale factor β to the planner gain k, i.e. $k' = \beta k$, and plug in the original P gain k_p, then Eq. (31) becomes:

$$\frac{(k'k_p)^2\omega^2 + \omega^4(k_p - k'k_p\tau)^2}{k_p\omega^4 + (k'k_p\omega - \omega^3)^2} \tag{36}$$

Unfortunately, this does not equal to (31), suggesting that a scaled low-level controller gain k_p has different impacts on the SS compared to a scaled planner gain k with the same β.

To check if the scaled planner gains k'_x, k'_v have the same effect with the scaled low-level controller gains k'_p and k'_f for the general linear ACCs, apply the replacing rules $k'_f = \beta k_f$, $k'_p = \beta k_p$, $k'_x = \beta k_x$ and $k'_v = \beta k_v$ into (33) and (34). After substitutions we have:

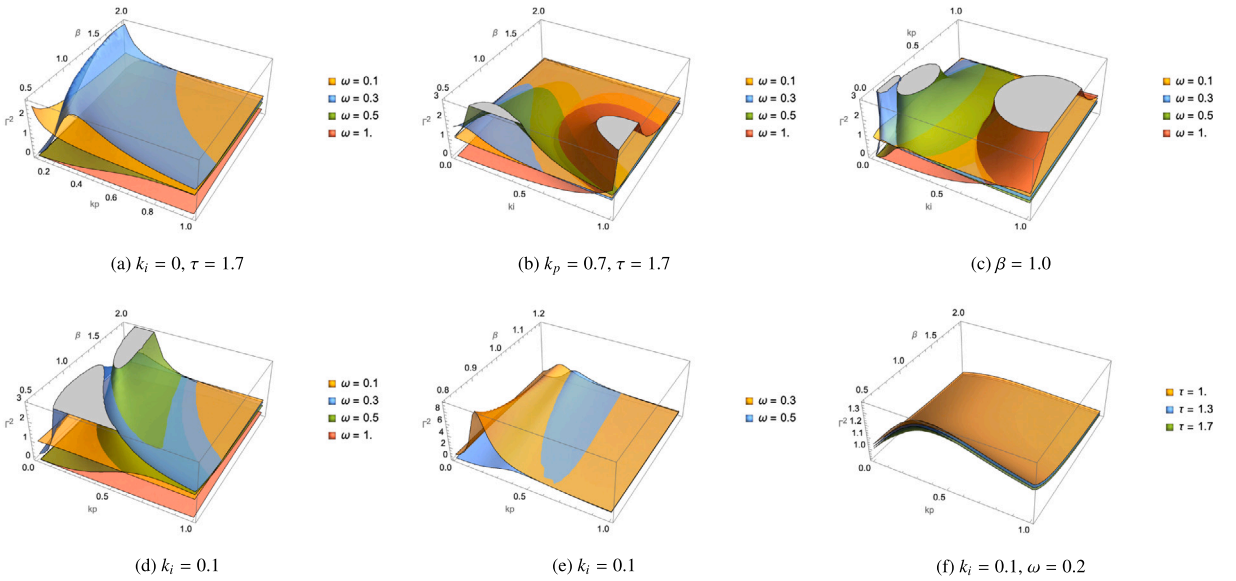
$$\frac{k_f'^2 k_x^2 \omega^4 + k_f'^2 k_v^2 \omega^6}{k_f'^2(k_v + k_x\tau)^2\omega^6 + (-k'_f k_x\omega^2 + \omega^4)^2} = \frac{k_x'^2 k_f^2 \omega^4 + k_f^2 k_v'^2 \omega^6}{k_f^2(k'_v + k'_x\tau)^2\omega^6 + (-k_f k'_x\omega^2 + \omega^4)^2} \tag{37}$$

$$\begin{aligned}\frac{k_p'^2 k_x^2 \omega^2 + k_p'^2 k_v^2 \omega^4}{(-k'_p k_x\omega + k'_p\omega^3)^2 + (-k'_p(k_v + k_x\tau)\omega^2 + \omega^4)^2} &= \frac{k_p^2 k_x'^2 \omega^2 + k_p^2 k_v'^2 \omega^4}{(-k_p k'_x\omega + \beta k_p\omega^3)^2 + (-k_p(k'_v + k'_x\tau)\omega^2 + \omega^4)^2} \\ &\approx \frac{k_p^2 k_x'^2 \omega^2 + k_p^2 k_v'^2 \omega^4}{(-k_p k'_x\omega + k_p\omega^3)^2 + (-k_p(k'_v + k'_x\tau)\omega^2 + \omega^4)^2}\end{aligned} \tag{38}$$

where (37) suggests that when only F gain exists, the scale factor β on k_f has the exactly same effect of scaled planner gains k_x and k_v with the same β. The (38) suggests that the scaled P gain βk_p has an approximately equal impact with the scaled planner gains βk_x and βk_v, provided the I gain and F gain is zero.Contrary to OP linear ACC, the general linear ACC with PIF low-level controllers seem to better support the assumption that slow low-level control just means smaller planner gains.

The above analysis also sheds insights of how the low-level control gains, the actuator performance or both can affect the overall SS of the ACC. Using (32), (33), and (34), we are able to derive the increasing/decreasing interval of TFs with respect to βk_p and βk_f. Correspondingly, the decreasing interval for the TF (32) of the OP linear ACC is $[\omega/k, +\infty)$ with respect to βk_p. For general linear ACCs, the decreasing interval of (33) with respect to βk_f, is $[\omega^2/k_x, +\infty)$, and for βk_p, the decreasing interval of (34) is $[\omega^3/(k_v + k_x\tau), +\infty)$. Starting from the default values $\beta = 1$, $k_f = 1$ and $k_p = 0.7$, a small deviation from them can result from control design or external disturbances. Their impact on SS can be determined using those increasing/decreasing intervals derived above. In general, fast low-level control, i.e. larger β, k_p and k_f, will benefit the SS under small frequencies and has the opposite effect at large frequencies if they fall into the increasing intervals of the TF.

The above analysis based on (37) and (38) both require a zero I gain along with a zero P or F gain for maximum simplification. In practice, the impact mechanisms could be more complex when P, I, and F interact with each other. Now we release the assumption of k_i and study the details changes of the TF caused by different combinations of controller gains and actuator performance under



(a) $k_i = 0, \tau = 1.7$ (b) $k_p = 0.7, \tau = 1.7$ (c) $\beta = 1.0$

(d) $k_i = 0.1$ (e) $k_i = 0.1$ (f) $k_i = 0.1, \omega = 0.2$

Fig. 5. Impact of low-level control on SS of the OP linear ACC: $\Gamma^2 > 3$ is truncated from the figures.

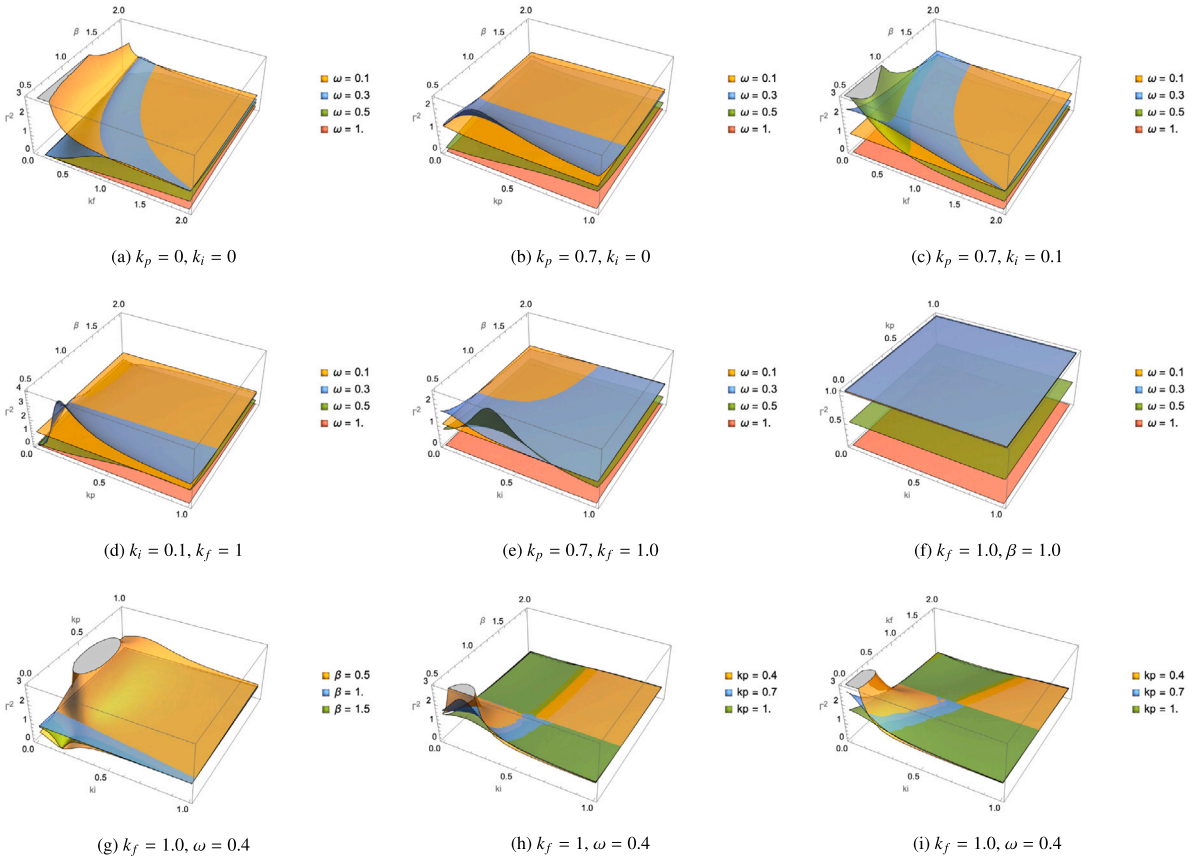
varying oscillations. Based on original TFs (28) and (30) for full bi-level ACCs, Figs. 5 and 6 depicts how the SS, i.e. $|\Gamma|^2$, is affected by low-level control and oscillation frequencies.

For the OP linear ACC, Fig. 5(a) shows that, increasing β (default value 1.0) and k_p (default value 0.7) both improves the SS at small frequencies and undermines it at large frequencies, which is consistent with our theory of its decreasing interval. However, such impact pattern only holds when the integral gain $k_i = 0$, and it is substantially altered after a slight increase in k_i. Fig. 5(b) shows that k_i can lead the TF to rise quickly under large frequencies such as $\omega = 0.5$ or $\omega = 1.0$ provided the actuator is weak, i.e. $\beta < 1$. By contrast Fig. 5(b) shows the OP linear ACC is always string stable for arbitrary k_p or β values under those same oscillations without integral gains, where the SS even gets better with a smaller β. Fig. 5(c) suggests the integral gain also makes the TF to explode when coupled with a small P gain k_p. In the same figure we see a large k_i (around 0.7) even makes large-frequency ($\omega = 1.0$) oscillations become unstable, unless a very strong k_p (close to 1.0) is paired to stabilize the TF. In comparison with Fig. 5(a) where $k_i = 0$, Fig. 5(d) re-depicts the impact of actuator performance and the P gain given the default I gain value $k_i = 0.1$. We noticed that, for k_p the increasing interval $[0, k_p^*]$ becomes narrower with k_p^* getting smaller under all frequencies. Similarly, the increasing interval of β, i.e. $[0, \beta^*]$ gets closer to zero, which makes the TF no longer monotonically decreases with β, i.e. a stronger actuator cannot always ensure better SS. Fig. 5(e) verifies that our estimated decreasing interval for βk_p is $[\omega^2/k, +\infty)$, where $k = 0.5$ and $\omega = 0.5$ lead to $k_p^* = \omega^2/k = 0.5$, and it becomes smaller when β is large. Fig. 5(f) confirms our finding about the decreasing interval is independent of headway τ.

For general linear ACCs, Figs. 6(a) and 6(b) shows that a stronger actuator (β) and a larger P/F gain (k_p/k_f) improve the SS at small frequencies and undermine it at large frequencies, which aligns with the increasing/decreasing analysis from (33) and (34), as well as the earlier analysis on planner gain k_x in Fig. 3(b). Fig. 6(c) and Fig. 6(d) shows the similar impact pattern still holds given the default P and I gains ($k_i = 0.1$ and $k_p = 0.7$). Interestingly, impact of the integral gain k_i becomes less sensitive to small k_p and β values compared to the PI controller in the OP linear ACC. Noticeably, Fig. 6(d) shows the default integral gain $k_i = 0.1$ does not explode the TF even if coupled with small k_p or β, which is in contrast with Fig. 5(d). Fig. 6(e) displays that even super large I gains, e.g. $k_i = 1.0$, do not explode the TF given a weak actuator, showing more stability than the PI low-level controller in Fig. 5(b). Fig. 6(g) further manifests the robustness of the PIF controller against errors, where k_p and k_i seem not affect the SS at all given $k_f = 1$ and $\beta = 1$. The last row of Fig. 6 further shows that the impact of the I gain is only significant given weak actuators (see Fig. 6(g)), smaller P gains (see Fig. 6(h)) and slow feedforward control (see Fig. 6(i)).

From the above analysis, we draw the following remarks:

R-1 Slow/fast low-level control has a similar but not equivalent effect on SS compared to small/large upper-level planner gains. We found that the slow/fast low-level controller in linear ACC systems, i.e. smaller/larger k_p and k_f gains, or β, behave similarly but not equivalently to the decreased/increased planner gains with the same scale factor. Mathematically, only Eq. (37) proves that the discount factor β applied to low-level controller gains is equivalent to β multiplied by upper-level planner gains provided no P or I gain is used. The (36) and (38) both suggest that such equivalence does not apply to k_p even if k_i is zero. Such finding suggests that the ACC design should not only focus on tuning the planner gains, instead the low-level control should be considered as well to guarantee the overall SS.



(a) $k_p = 0, k_i = 0$ (b) $k_p = 0.7, k_i = 0$ (c) $k_p = 0.7, k_i = 0.1$

(d) $k_i = 0.1, k_f = 1$ (e) $k_p = 0.7, k_f = 1.0$ (f) $k_f = 1.0, \beta = 1.0$

(g) $k_f = 1.0, \omega = 0.4$ (h) $k_f = 1, \omega = 0.4$ (i) $k_f = 1.0, \omega = 0.4$

Fig. 6. Impact of slow/fast control loop on SS of general linear ACCs: the captions of subfigures mark the key parameters not shown in the plots.

R-2 Integral gain substantially deteriorates ACC's SS under slow low-level control. The above results suggest that the integral gain can significantly move the increasing/decreasing interval of the TF with respect to k_p and β, which may even cause opposite impact of slow/fast low-level control when it has no integral control, e.g. a stronger actuator can unexpectedly undermine the SS under small frequencies. We also find the integral gain can easily explode the TF if k_i and k_p are not well paired, or the actuator severely undershoots, making the whole ACC less robust in terms of SS. Such issues can happen when I gains are still close to default values, and they are more devastating when I gain gets larger. It suggests that extra precaution is needed to balance the low-level gains and guarantee the actuator performance if the integral control is used.

R-3 The PIF controller is more robust against the impact of slow low-level control than PI controllers. The PI speed controller is shown to be too sensitive to small P gains and actuator undershootings, where the integral gain further deteriorates the SS. By contrast, the PIF controller combines both the feedback and feedforward controller to execute desired accelerations/speeds, which provides extra robustness in case one under-performs. Mathematically, one can verify that the following partial derivatives $\partial \|\Gamma^{\text{PD-PIF}}(k_f = 1, \beta = 1)\|_2^2/\partial k_p = 0$, $\partial \|\Gamma^{\text{PD-PIF}}(k_f = 1, \beta = 1)\|_2^2/\partial k_i = 0$ and $\partial \| \lim_{k_p \to \infty} \Gamma^{\text{PD-PIF}}(\beta = 1)\|_2^2/\partial k_f = 0$, which helps explains why descent control of either F or PI makes the whole controller more robust to the potential failure of the other.

R-4 Low-level impact is more sensitive to oscillation wavelengths than headways. Despite that the headway τ plays a significant role in planner's SS, it barely affects the impact of slow/fast low-level control, as τ has none or little impact on the bounds of the decreasing intervals derived from (31), (33), (34), which are ω^2/k, ω^2/k_x and $\omega^3/(k_v + k_x\tau)$.

The above analysis for linear ACCs is conducted in the frequency domain. We provide more intuition in the time domain in Appendix A. Also notice that for linear ACC systems, the impact of slow/fast low-level controllers does not vary with the amplitude of the signals, indicating a scale-invariant property. We will shortly show, this is not true for non-linear MPC ACCs in Openpilot, as well as the empirical tests in the literature (Li et al., 2021).

4. Impact mechanism of low-level control on SS of MPC ACCs

In the previous section we have shown the impact of low-level control on the SS of linear ACCs based on the TFs of the bi-level feedback loops. Unfortunately, similar analytical methods using TFs cannot be directly used for nonlinear control systems such as MPC ACCs. To circumvent this, we start by investigating the characteristics of the MPC planners experimentally, then analyze the impact of slow/fast low-level control by approximation.

4.1. Characteristics of MPC planners: non-linearity and scenario-dependent gains

Different from most CF models, MPC planners handle the CF task by first predicting the leader movement, and then solving the trajectory of the follower from an optimization problem that balances safety, efficiency and driving comfort. Although no explicit control gains are involved in MPC planners, to understand its characteristics we can approximate the MPC gain as the ratio between accelerations and spacing errors, i.e. $a_{ego}/(s_{ego} - s_{des})$, similarly to the gain k in linear controllers such as (2).

We argue that the MPC gain is scenario-specific due to the prediction capability as well as the safety design according to the open-source MPC model in OP (10). To illustrate, in the look-ahead planning horizon of the MPC, the predicted leader trajectory always impacts the current follower decisions since it is the reference line for estimations of spacing and relative speeds. A simple lead prediction model, e.g. the (9d), often bases its prediction on the current leader acceleration a_{lead}. In the case where the follower needs to brake for a safer spacing, an accelerating leader helps to enlarge the space hence saves the follower's efforts from braking too hard, while the decelerating leader would force the follower to brake harder than the case without prediction. On the other hand, it is common that the safety cost increases nonlinearly with spacing errors. For example, the exponential safety term in (4) becomes more dominant in MPC's total cost when spacing errors are large, which likely results in hard brakes to reduce the safety cost and possibly sacrifices other cost terms with respect to comfort or accelerations. Note that in non-emergency cases the MPC is designed to be comfort-oriented, which always promotes small jerks and small accelerations thanks to the cost functions in (6) and (7). In a nutshell, these design features suggest MPC gains are scale-dependent and scenario-specific, which are supposed to be small given small spacing errors, larger leader accelerations and could increase quickly in risky scenarios when the leader brakes hard and the spacing error is large.

As an example of the varying MPC gain, the OP MPC planner without a low-level controller is tested against a typical deceleration oscillation and the results are displayed in Fig. 7(a). The MPC gain in the deceleration phase is initially small and increases with spacing errors and the leader deceleration for safety reason, later it drops again to comfortably follow the leader when the leader starts to accelerate away. We further show the relationship between the MPC gain in Fig. 7(a) with the spacing error in Fig. 7(b) and the leader acceleration in Fig. 7(c), which support our conjectures above.

In this example, the MPC gain increases first and decreases later when the lead vehicle starts to accelerate again while at the same time the spacing error is not very large. We refer to this as the 'fast-slow' pattern for the varying gain, which benefits traffic congestion since it effectively responds to the leader deceleration at first such that it does not need to brake hard later to cause speed overshootings and undermine the SS. Oppositely, the MPC gain can be 'slow-fast', which reacts to deceleration oscillations slowly at first, accumulates spacing errors, and later has to respond fast to large cumulative spacing errors with larger control gains. The impact of the 'slow-fast' varying gain may have a different impact on SS, which needs further investigation.

4.2. Impact of slow/fast low-level control on the SS of MPC ACCs

The above analysis on MPC's planning characteristics suggests that MPC can have different varying gain patterns, i.e. 'fast-slow' or 'slow-fast' under large oscillations, now we study their impacts on the overall SS.

Fig. 13(a) shows a typical CF scenario in a congestion shockwave, where the ACC follower often overshoots the leader speed v_l and amplifies the congestion wave. At the time T_{os}, the follower reaches the new leader speed v_l, and the spacing error is e_s.

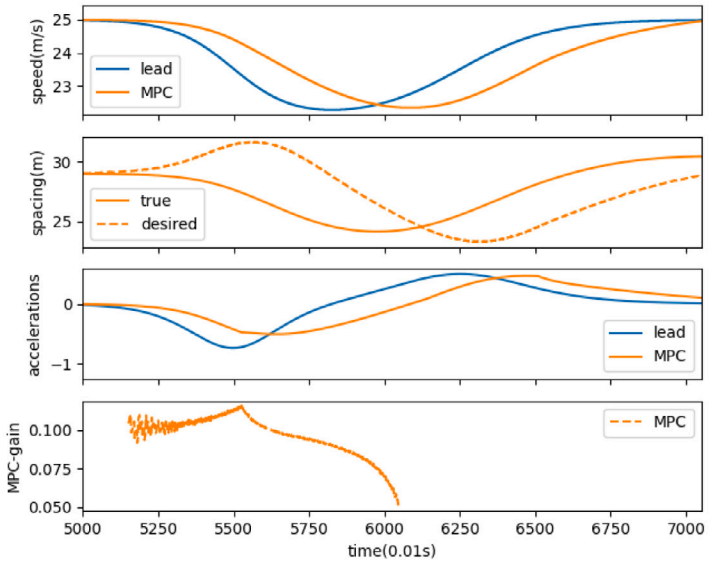
From the beginning to time T_{os}, the spacing error accumulates, which is approximately equal to the area between the shifted leader speed (with time τ) and the true speed of the following MPC. At T_{os}, the planning goal of the MPC is approximately moving its future trajectory towards the desired trajectory safely and comfortably. Considering that the leader has zero acceleration after T_{os}, the desired trajectory of the following MPC can be approximated as the shifted leader, or "Newell trajectory" (Newell, 2002), as shown in Fig. 8(b), where the initial position of the MPC equals the spacing error e_s. Apparently, the MPC with slow low-level control will delay the time T_{os} and consequently result in a larger spacing error e_s.

In the overshooting process after time T_{os}, we approximate the MPC dynamics using the general linear CF model in (21) for mathematical tractability. Using T_{os} and the desired position at T_{os} as the origin point for the axis of the ODE problem; see Fig. 8(b); we have:

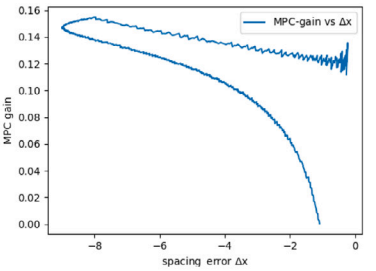
$$\ddot{x}_{ego}(t) = k_x(t)(x_{lead}(t) - x_{ego}(t) - s_{des}(t)) + k_v(t)(\dot{x}_{lead}(t) - \dot{x}_{ego}(t)) \tag{39}$$

$$\approx \bar{k}_x(v_l \cdot t - x_{ego}(t)) + \bar{k}_v(v_l - \dot{x}_{ego}(t)) \tag{40}$$

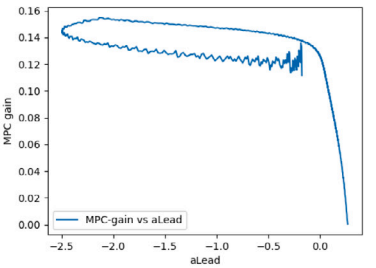
where the initial conditions are $x_{ego}(0) = e_s$ and $\dot{x}_{ego}(0) = v_l$.



(a) Example of varying gains of the MPC: the 'slow-fast' varying gain pattern.

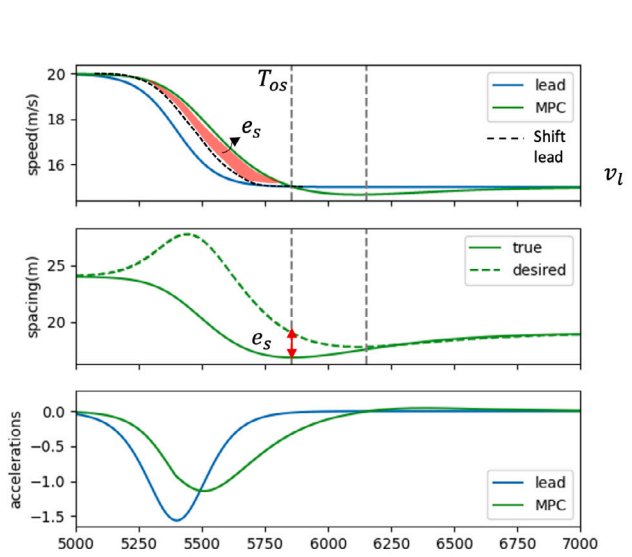


(b) MPC gain v.s. spacing error

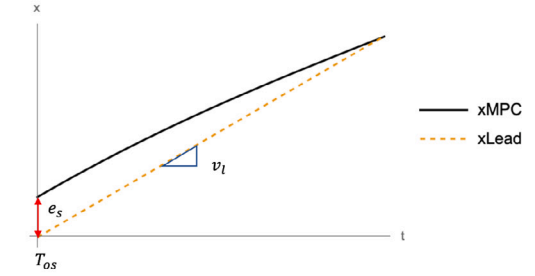


(c) MPC-gain v.s. a_{lead}

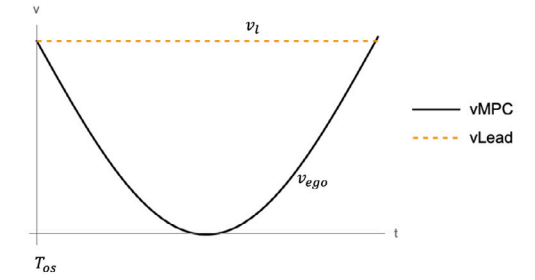
Fig. 7. The varying gain of the MPC and its relationship the leader acceleration and spacing errors.



(a) Example of the MPC that overshoots the speed change in the shockwave



(b) ODE approximation of the planning trajectory



(c) ODE approximation of the planning speed

Fig. 8. Responses of MPCs in a congestion shockwave and the approximation of the speed overshooting.

For simplicity, we assume $k_v = 0$[1] in the overshooting process. By solving the ODE, we can derive the trajectory and the speed of the MPC follower:

$$x_{ego}(t) = t \cdot v_l + e_s \cos(t\sqrt{k}) \tag{41}$$

[1] We have verified numerically that $k_v \neq 0$ does not change the conclusion that larger e_s and larger k_x causes a larger speed overshooting. For mathematical tractability, $k_v = 0$ allows us to have a simple and analytical solution of the overshoot speed.

(a) Installing Comma Dev-kit on Honda Civic (b) Interface of the active-running Openpilot (c) Connecting to the CAN-bus

Fig. 9. On-road experiment set-up Comma Dev-kit.

$$v_{ego}(t) = v_l - \sqrt{\bar{k}} e_s \sin(t\sqrt{\bar{k}}) \tag{42}$$

The ODE solution suggests that, the larger spacing error e_s accumulated prior to the overshooting process, and the larger gain $\bar{k}$ in the overshooting process will both contribute to a larger speed overshooting, which undermines the SS and amplifies the congestion wave. According to this theory, the impact of low-level control becomes how it changes the cumulative spacing error and the control gain in the overshooting process.

As mentioned earlier, the varying MPC gain can have two opposite patterns, i.e. 'slow-fast' or 'fast-slow' depending on how it changes before and after T_{os} in Eq. (40). More specifically, the 'slow-fast' pattern will have a smaller gain before T_{os}, accumulating larger spacing error e_s, and forces itself to have a larger gain $\bar{k}$ after T_{os}. According to our ODE approximation above, the 'slow-fast' varying gain will consequently overshoot the speed more than the 'fast-slow' case. Next we will investigate experimentally how slow/fast low-level control can affect the MPC to switch between 'slow-fast' and 'fast-slow' pattern.

5. Simulation and experiment results

So far we have found that the slow/fast low-level control can affect the SS of linear and MPC ACCs. For linear ACCs, whether it improves or undermines the overall SS largely depends on the control gains, as well as the frequency of the oscillation. For MPCs, we still need more experimental results of the varying control gains to identify the impact of slow/fast low-level control. This section we will study the impact of slow/fast low-level control by numerical simulations, and then validate the findings experimentally on real cars.

5.1. Numerical and experimental methods

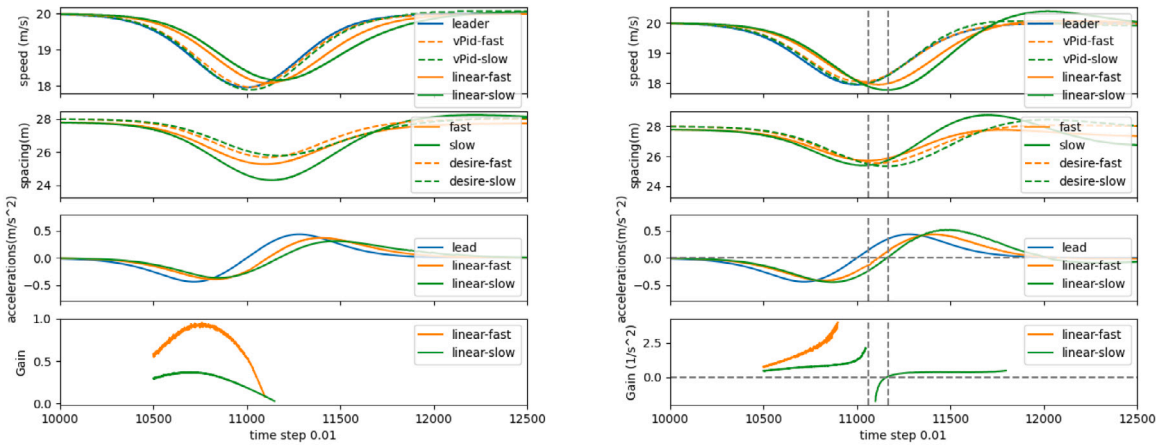
The experimental method is to run a custom Openpilot on a regular commercial vehicle, which builds upon the stock sensors and actuator interfaces of the car. Our numerical method uses the same code base from OP for real cars, but emulates the actuator (gas/brake) with a model and uses the simulation distances in lieu of the radar measurements.

For on-road testing of custom ACCs on a daily car model, the hardware preparation is rather simple. We only need to connect the Comma.ai's after-market device Comma Two, to the interface of the ACC unit of the car, which will be overwriting the stock ACC algorithms and running our custom Openpilot instead. The driving logs from Comma Two include all the ACC-related variables, e.g. the v_{target} and $k_p e(t)$, as well as the CAN bus messages of the car, such as v_{ego} and a_{ego}. The method does not require any modifications of a regular commercial car and the full access of the ACC system allows us to dive into the details of the control algorithm and analyze its impact. We show the setup of the Comma dev-kit in Fig. 9. A more detailed installation tutorial can be found here (Comma.ai, 2020).

To run the same ACC code numerically on computers, we have to create a virtual radar and gas/brake system. For the sensor, we replace the radar estimates v_{lead} and s_{ego} using the free-of-error speed and spacing from simulations. For the virtual actuator model, i.e. the *gb2accel* function, recall that (14) can be approximated as a simple scaling function. To further simulate the impact of a strong or weak actuator, one can simply manipulate the scale factor in *compute_gb* to change the equality in (16). For the other car-specific parameters such as the acceleration bound, or the control gains, we can pick the default values from any car model available in the Openpilot's pool. Here in this paper we use all default values from a Honda Civic in accordance with the field experiments. Since we focus on the longitudinal control, a tangent road is assumed where the steering angle is always zero and no lateral control is needed. We also omit other impact factors such as the grade, and the external speed disturbances. The numerical method can run efficiently without a professional car kinematics software, it is also hazard-free which allows us to test arbitrary control gains and conduct the platoon experiments with little cost.

5.2. Simulation results

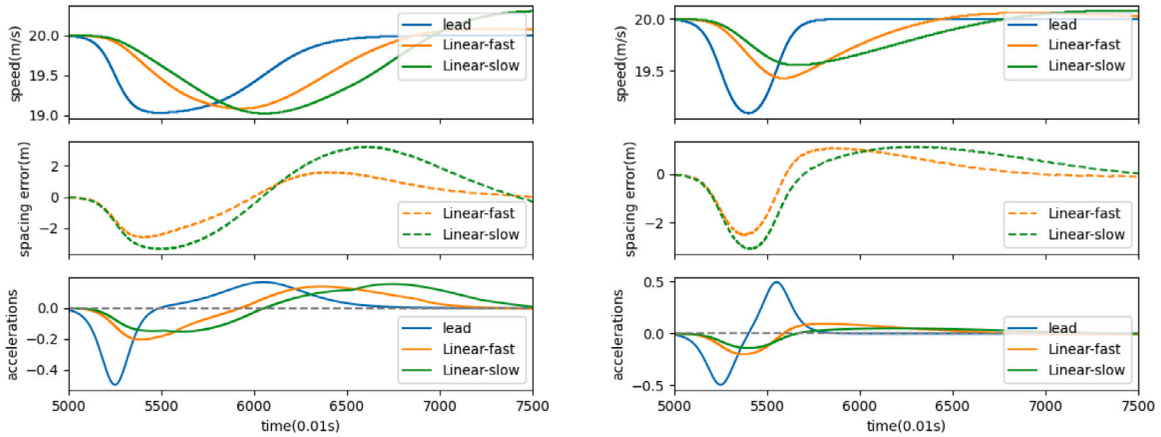
Now we simulate the impact of slow/fast low-level control on the overall SS of linear and MPC ACCs.



(a) linear OP ACC with P only low-level controller

(b) linear OP ACC with PI low-level controller

Fig. 10. The impact of slow/fast low-level control on the OP linear ACC.



(a) the small oscillation frequency (ω)

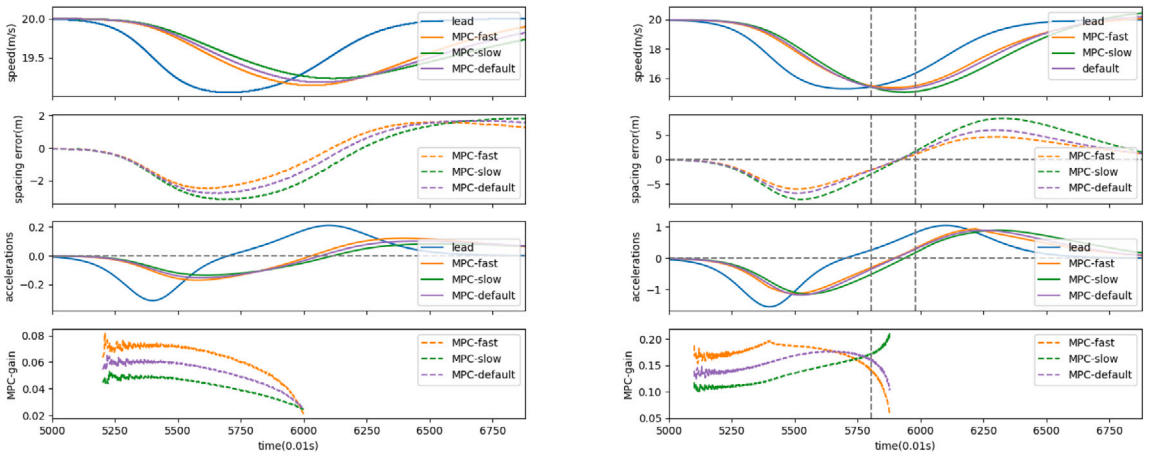
(b) the large oscillation frequency (ω)

Fig. 11. Impact of slow/fast low-level control on SS of general linear ACCs.

5.2.1. Impact of slow/fast low-level control on SS of linear ACCs

We first use numerical simulations to verify the impact of slow/fast low-level control on the OP linear ACC system. We tested the same source code from OP with only emulated radar and actuator. The slow/fast low-level control is simulated by adjusting β, e.g. $\beta = 1.2$ or $\beta = 0.8$ before the default gains. Fig. 10 compares the impact of fast/slow low-level control with a P-only and PI controller. The results in Fig. 10(a) show that the slow low-level P controller improves the SS while adding integral gain reverses such impact, causing the slow low-level control undermine the SS, which is consistent with our earlier analysis and remarks on the integral effect. We also show plot the measured gain, defined as the true acceleration responses over the spacing error, i.e. $a_{ego}/(s_{ego} - s_{des})$. Noticeably, despite the OP ACC uses a linear planner, it outputs significantly varying gains during the whole process, which seems to increase with spacing errors. We conjecture the variance may result from not only the lead speed change, but also the collision-proof design in the low-level control gains we introduced in Section 2, where the low-level PI gains are scaled up when large decelerations are required for the safe braking. In Fig. 10(b), the measured gain in the slow linear ACC becomes negative after the true speed v_{pid} crosses the tracking speed v_{pid}, suggesting integral overshooting exits and deteriorates SS, which is consistent with our analytical findings in Section 3.

In addition to the OP linear ACC system, we also tested the general linear ACC system consisting of the general linear CF model and a PIF low-level controller. As suggested by the theory in Section 3, the impact of slow/fast low-level control on a general linear ACC system largely depends on the oscillation frequency ω, where the slow tracking undermines the SS at small frequencies; see Fig. 11(a); and improves the overall SS at large frequencies; see Fig. 11(b). Notice that they have the same amplitude, the only difference is that the leader accelerates slowly in Fig. 11(a), which is more realistic considering the acceleration limits of vehicles.



(a) Slow low-level control improves the SS under a small scale oscillation

(b) Slow low-level control undermines the SS under a large scale oscillation

Fig. 12. The impact of slow/fast low-level control on the SS of the MPC ACC under small/large oscillations: notice that fast low-level control causes the MPC gain has a 'fast-slow' pattern, whereas the slow low-level control leads the MPC gain to have a 'slow-fast' pattern.

5.2.2. *Impact of slow low-level control on SS of MPC ACCs*

First, we verify our approximation of the MPC under small-scale oscillations in Fig. 12(a), where the results show that slow low-level control helps improve the SS of the MPC ACC. Apparently the slow low-level control causes smaller MPC gain compared to fast low-level control. Their impact on the overall SS can be explained using the theory for general linear ACC systems in Fig. 3(b) where smaller gain k_x in $(0.04, 0.08)$ improves SS under $\omega \approx 0.4$.

Then we study MPC's responses at large oscillations. As stated earlier, our focus here is to experimentally investigate how the slow/fast low-level control decides the 'slow-fast' or 'fast-slow' patterns of the varying MPC gain. Fig. 13(a) shows that the slow low-level control contributes to the 'slow-fast' MPC gain, which is a result of larger spacing errors as shown in the same graph. Oppositely, the fast low-level control; see the orange line; shows a 'fast-slow' varying gain process. At T_{os}, or the step 5850 here; see the left dashed line; the spacing error e_s is larger for the slow low-level control. After T_{os}, the MPC gain is also larger for the one with slow low-level control. According to our earlier explanation of the ODE solution, the slow low-level control causes the MPC to have larger overshootings, which is consistent with the observation here. We also notice that our ODE method in (42) over-estimates the speed overshootings. To see this, the MPC with the slow low-level controller, $e_s\sqrt{k} \approx 3\sqrt{0.25} \approx 1.5$ m/s, where the true speed drops from 15 m/s to around 14.6 m/s in Fig. 13(c). This is probably due to our simplification of the speed gain $k_v = 0$. In practice, the speed gain would also play a role and help close the gap with smaller speed overshootings. We also verified numerically $k_v \neq 0$ does not change the relationship between speed overshooting and k_x/e_s here.

Although the ODE estimation originates from the shockwave case and is not a direct solution for the cyclic speed oscillation in Fig. 12(b), we notice the impact mechanism for both cases seems to be similar, where the slow low-level control also results in a 'slow-fast' varying gain as shown in Fig. 12(b), which leads to more speed overshootings as estimated by the ODE approach.

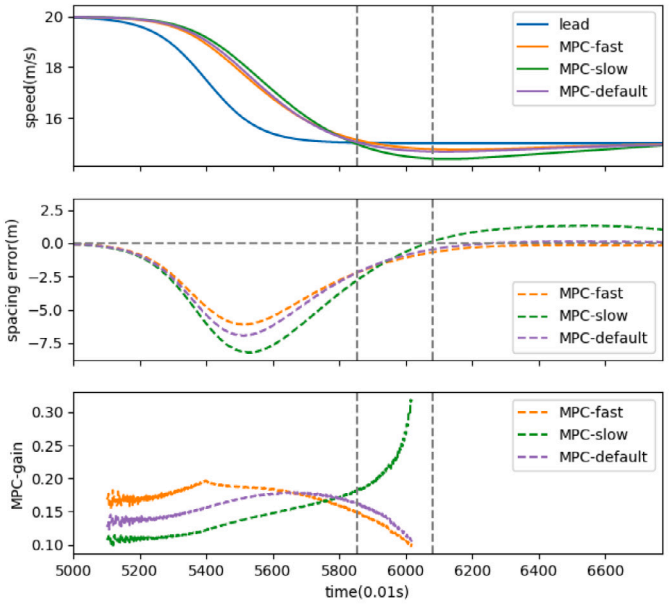
5.3. *Experimental results*

Now we conduct on-road experiments using a daily car model, 2019 Honda Civic, to validate the impact of slow/fast control on the overall SS of the ACC. Thanks to our unique experiment method to run a custom ACC on a real car, we are able to collect detailed data of the actuator and evaluate its performance including the actuator model *gb2accel*, actuator delays, as well as the grade impact.

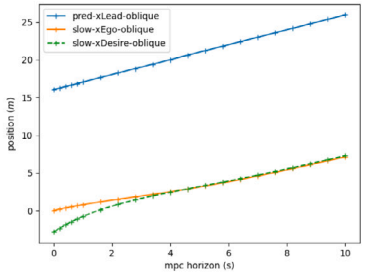
5.3.1. *Real-car validation of the impact of fast/slow low-level control on SS*

To verify the impact of the fast/slow tracking performance, we now test and compare two low-level controllers sharing the same planner (MPC) but with the different control gains and actuator performances. Following the experimental method introduced earlier, two custom MPC+PIF branches are used to overwrite the stock ACC in a 2019 Honda Civic. Specifically, the fast low-level controller uses the control gains $1.0k_p^0$, $0.33k_i^0$, $1.2k_f^0$ and the *compute_gb* is defined as $gb = 1/3 \cdot control$ to make the actuator strong. By contrast, the slow controller adopts $0.5f_p^0$, $0.33k_i^0$, $1.0k_f^0$ and the *compute_gb* is set to $gb = 1/5 \cdot control$ such that the actuator is weak. All the scales are applied to the default control gains used in OP.

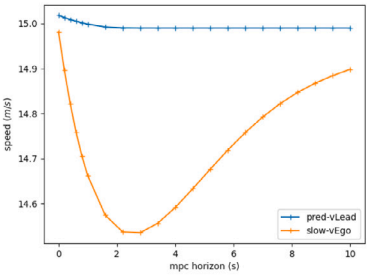
Fig. 14 displayed the results from the two real drives using the fast/slow low-level controller. The ego vehicle was following a human-driven lead vehicle in the natural driving where the lead changes its speed occasionally on a curvy road. It is apparent that the fast low-level controller is able to dampen the lead speed changes while the slow low-level controller amplifies them. The results verified our finding that a fast low-level controller improves the SS and vice versa.



(a) Comparing fast/slow low-level control on shockwave propagation



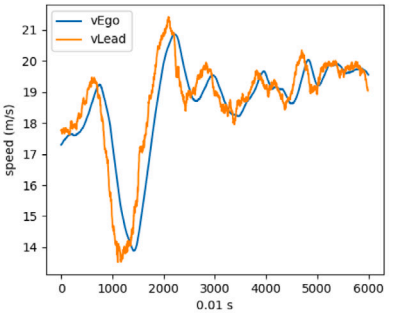
(b) MPC trajectory planning at equal-speed point



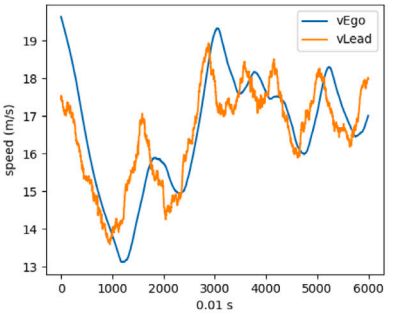
(c) MPC speed planning at equal-speed point

Fig. 13. The impact of slow/fast low-level control on congestion shockwave of MPC ACCs.

(a) testing site: Concord Rd SE, Smyrna, GA



(b) a drive with the fast low-level controller



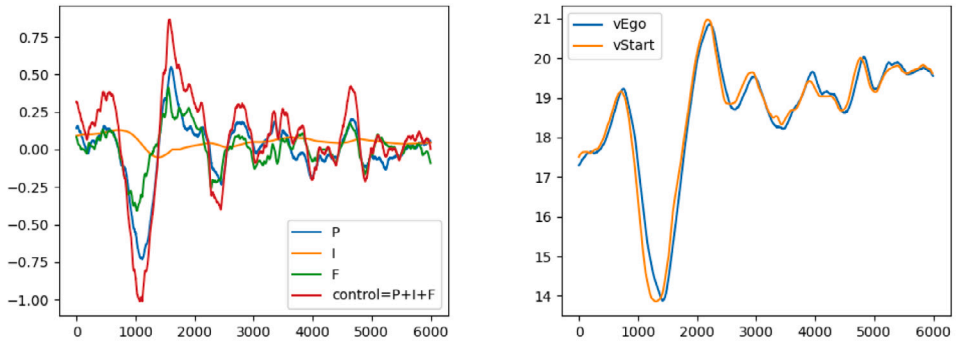
(c) a drive with the slow low-level controller

Fig. 14. Impact of a fast/slow low-level controller on SS in real drives.

We further show more control details in the fast low-level controller and help explain why it improves the SS. Fig. 15(a) displays the detailed P, I, F terms in the *control* input during the 1-minute drive. As we pointed out earlier, the F and P term are expected to be consistent which makes the controller faster, rather than canceling out each other. Also a faster controller helps prevent the overshoot caused by the integral accumulation because the speed tracking error is always small and never dominates the control. Fig. 15(b) displays that v_{start} is close to v_{ego}.

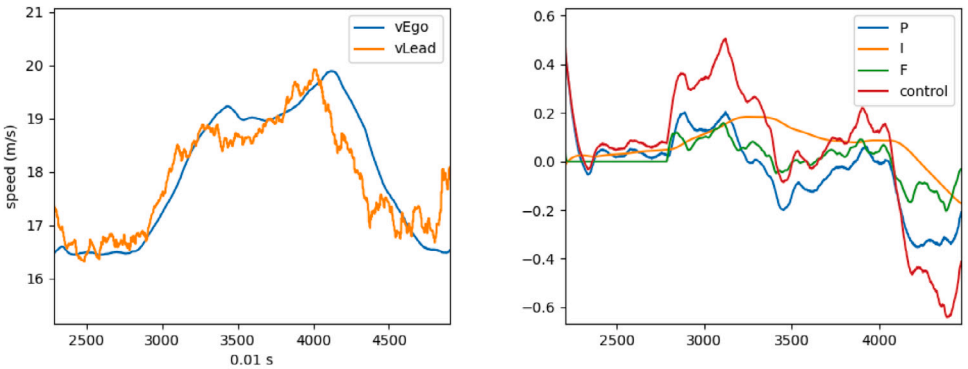
To validate the impact of the integral term, we conducted a real drive and compared the detailed control values from all three sources (P, I and F). Fig. 16(a) showcases a string unstable example where the follower overshoots the lead vehicle around step 3300, meanwhile Fig. 16(b) clearly shows that the overshoot is much due to a large and dominating integral value since both P and F terms are close to zero or even negative. Taken together, the two figures suggest the integral term accumulates when the follower accelerates to catch the leader, similar to what we have explained in Fig. A.22. Then an immediate overshoot takes places where the opposite relationship between the I term and P/F terms.

The similar empirical observations of the overshoots have also been reported in our recent experiments on the commercial ACC systems (Li et al., 2021). Among the three tested car models in the experiments, two show significant overshoots while the other behave much better in preventing them. This indicates the integral overshooting could be common in commercial ACCs, but a fast low-level controller could help alleviate such effect and probably has been applied in some cars models. Since the observed



(a) P, I, F terms in the low-level *control* variable (b) v_{ego} follows the planning variable v_{start} closely

Fig. 15. Details of the low-level and upper-level variables in a real string-stable drive.



(a) integral overshooting (speed) (b) Integral overshooting (control)

Fig. 16. Validation of the integral impact on SS in a real drive.

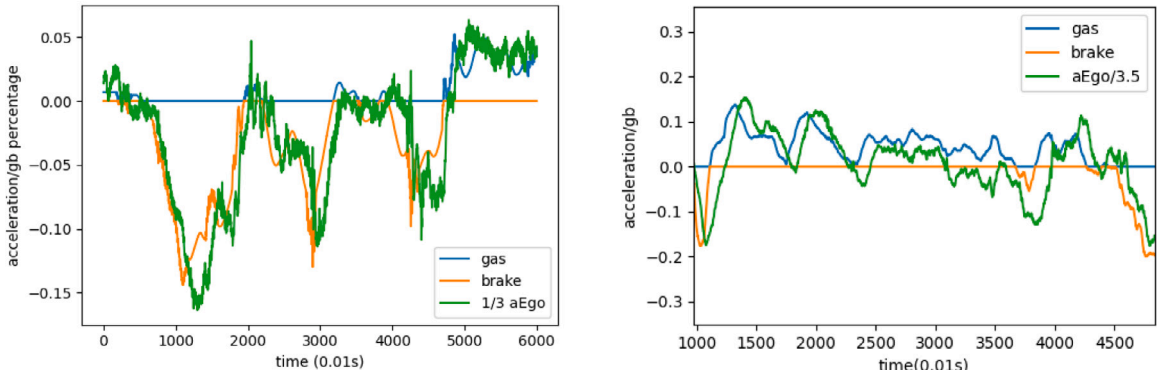
overshoots can result from both the planner and the low-level controller; see Fig. A.22; further analysis is currently conducted to enhance the empirical evidence.

5.3.2. *Evidence of actuator performance: errors, delay and grade impact*

Previously we assume the *compute_gb* and the *gb2accel* as simple scaling functions. Now we show the empirical data collected from a 2020 Toyota Corolla and a 2019 Honda Civic in Fig. 17. We can see a simple scaling function is a good fit to the relationship between gas/brake and true acceleration, though the errors tend to grow when the acceleration changes the direction, possibly due to shift in gas and brake.

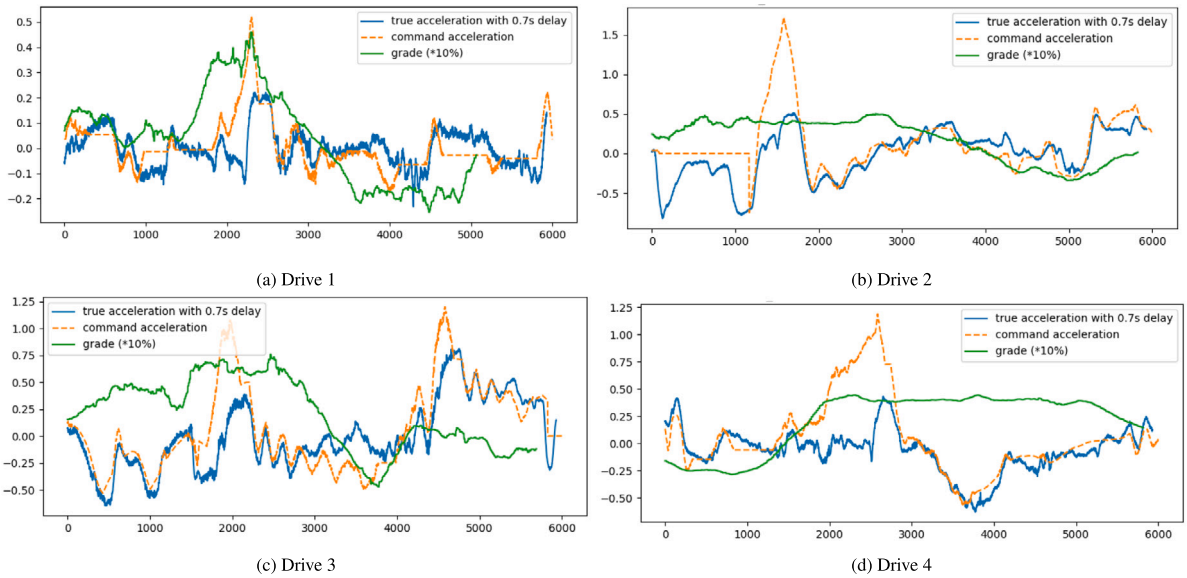
The scaling functions maybe a bit surprising or even counter-intuitive because theoretically the speed should also affect the dynamics as indicated by (14). The simple scaling functions are devised through the special design of the car control interface on recent car models, which can accept the scaled accelerations, or gb equivalently, as the input and execute the true accelerations perfectly at all speed levels. For cars equipped with this control interface, we can view the gb as a scaled acceleration, rather than the true gas/brake percentages. Such scaling factor is predetermined by the car manufactures. According to the discussions in Openpilot community (Smiskol, 2021), at least the latest car models from General Motor and Toyota are found to share this feature.

The earlier analysis has stated the actuator performance, i.e. the factor β, has significant impact on SS if it deviates from 1. Fig. 18 showcases a few examples of the to provide evidence that the weak or strong performance of the actuator can be common in the real world. Those figures show measurement from a Rav 4, including the actuator response errors, delay, and the potential impact from the grade. The data presented here conform that the upgrade leads to insufficient acceleration, while the downgrade could cause the true decelerations fall short, suggesting that $\beta < 1$, or the slow low-level control discussed in this paper may commonly exist in practice. In addition, we manually measure the delay and to our surprise, the actuator delay can be up to 0.7 s, which is larger than the common values (0.2–0.5 s) usually adopted in simulation studies. Notice that those delay measurements are directly from commercial ACC products, not in-house experiment vehicles in the literature.



(a) Toyota Corolla (b) Honda Civic

Fig. 17. Relation between gas/brake and acceleration on real cars.



(a) Drive 1 (b) Drive 2

(c) Drive 3 (d) Drive 4

Fig. 18. Measurements of the actuator performance, delay and grades from a 2019 Toyota Rav: the command acceleration is the *control*, the true acceleration is the output a_{ego} defined in this paper.

6. Guidance on tuning low-level controllers

The Ziegler–Nichols method (Ziegler and Nichols, 1942) is commonly used to obtain control gains for a PID controller. Its procedure is to first set the integral and derivative gain to zero and gradually increase the P gain until the system exhibits oscillatory behavior. Then, a look-up table provides the estimated values of control gains.

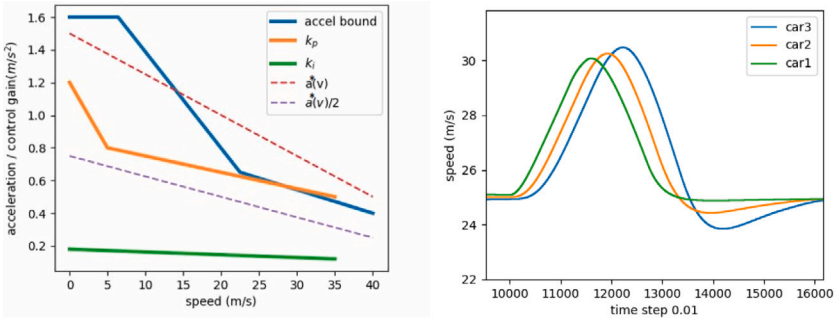
In this study, since our control gains $k_p(v)$ are speed-dependent, we propose the following two-step method as an extension of the Ziegler–Nichols method which only considers constant control gains. The first step is to obtain an initial feasible $k_p(v)$ that tracks the speed reference and drives the car. The second step is to fine tune $k_p(v)$ to achieve a fast low-level controller for better SS.

To derive the initial P gains of the low-level controller, we conjecture that they should be proportional to the maximum acceleration the engine is able to output, namely $a^*(v)$. Rakha et al. (2001) show that a straight line provides a good fit for this function. Note that a straight line fit also matches empirical data of the desired acceleration of human driven vehicles (Laval et al., 2014; Xu and Laval, 2020). Since $a^*(v)$ is the upper bound for the acceleration, it should satisfy:

$$a^*(v) \geq \max a_{ego}(v) \tag{43}$$

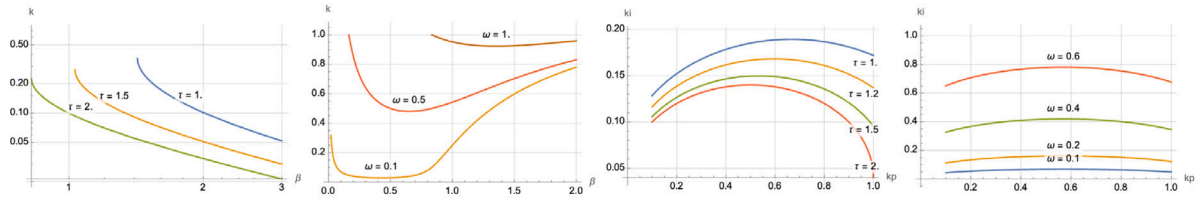
To understand why our conjecture should be robust, we combine (16) and (A.1) to obtain:

$$k_p(v) \cdot e(t) \approx a_{ego}(v(t)) \tag{44}$$



(a) Default acceleration bound and control gains in OP (b) SS of initial gains from linear acceleration model

Fig. 19. Default P and I gains in Openpilot and the initial P gain using linear acceleration model: k_p and k_i are the default control gains, $a^*(v)$ is the linear acceleration bound, and $a^*(v)/2$ is the proposed initial P gain.



(a) $\log k$ v.s. $\log\beta$: $\omega = 0.2$ (b) k v.s. β (c) k_i vs k_p given τs (d) k_i vs k_p given ωs

Fig. 20. Parameter combinations for the marginal SS of the OP linear ACC: default values $\omega = 0.1$, $\tau = 1.5$, $k = 0.2$, $k_p = 0.8$.

Further, from Algorithm 1 we note that the maximum speed error in the low-level controller is $\max e(t) = 2$ m/s. Thus, the maximum acceleration generated by the low-level controller is $\max_t a_{ego}(v) \approx 2k_p(v)$ at speed v, which indicates that the P gain $k_p(v)$ is a multiple of the maximum acceleration.

The maximum acceleration in control design should be approximately equal to the true acceleration bound corresponding to the engine, which gives:

$$k_p(v) \le a^*(v)/2 \tag{45}$$

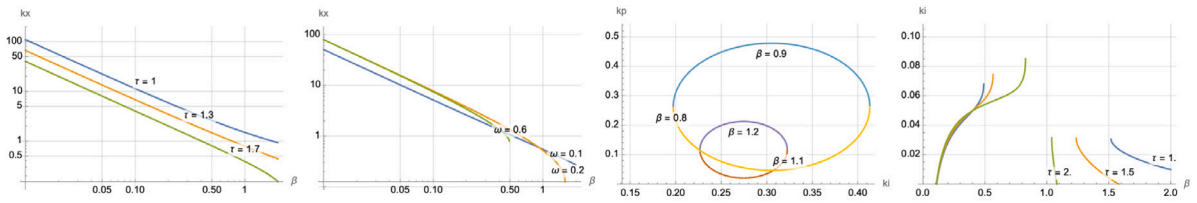
The default acceleration upper bound used in Openpilot shown in Fig. 19(a) is piecewise linear and larger than the linear accelerations of human drivers. If we assume $a^*(v) = 1.0(1 - v/40) + 0.5$ for a regular car model, then using (45) we can derive the initial P gains $k_p(v) = 0.5(1 - v/40) + 0.25$. We test the performance of such initial P gains which are feasible for driving but string unstable, using simulation (see Fig. 19(b)). The results validate our approach to derive the initial P gains.

For a PI controller, we still need the initial value for the I gains. A simple and common method is to determine how long it will take for an integral action to match a proportional action. For example, if the ideal time is 10 steps in the control loop, then it leads to $\Sigma_1^{10} k_i(v)e \approx k_p(v)e$, i.e. $k_i(v) \approx k_p(v)/10$.

Next, we tune the control gains to enable fast tracking performance. To this end, we suggest keeping the shape of $k_p(v)$ and only tuning a scaling factor before it, i.e. $s \cdot k_p(v)$ where s should be gradually increased from 1. For tuning the controller faster, a large body of instructions, handbooks and tools (O'dwyer, 2009; Collins, 2021) can be found online. More advanced tuning methods can also be found in the literature (Wang and Shao, 2000; O'Dwyer, 2006; Kanojiya and Meshram, 2012). The general procedure is to gradually increase the gains while circumventing instability. The most straightforward method is trial-and-error. To verify the new control gains, the SS performance can be evaluated by running the platoon experiments using simulation, as it is efficient and hazard-free. In the simulation experiments, the control gains are increased until the desired SS is achieved. Then, field experiments are recommended for further testing because some other aspects such as driving comfort, impact of grades, or sudden disturbances, also need to be accounted for in the real world. The SS can be evaluated from the field experiments as shown in Fig. 14.

The above procedures are designed for a PI controller, based on which the initial control gains for a PIF low-level controller are easier to determine and one can apply the same method for tuning it fast. To ensure that the gas/brake is strong enough, real driving data needs to be collected for the specific car model (see Fig. 17) and fit to the actuator model in (14), based on which the *compute_gb* function can be designed to ensure that the actuator is strong.

According to the trade-off analysis between SS and safety in Li (2020), the marginal SS where $\Gamma = 1$ appears to be the optimal design for ACCs in order to balance safety and efficiency. While the above trial-and-error method still holds, the TFs of the full ACC loops actually provide us a more efficient way to determine the desired control gains for both low-level controllers and upper-level planners.



(a) $\log k_x$ v.s. $\log \beta$: $\omega = 0.1$ (b) $\log k_x$ v.s. $\log \beta$: $\tau = 1.5$ (c) k_p vs k_i: $\omega = 0.5$ (d) k_i vs β: $\omega = 0.2$, $k_p = 0.1$

Fig. 21. Optimal parameter combination for the marginal SS of general linear ACC systems: $k_x = 0.15$, $k_v = 0.25$, $k_p = 0.1$, $k_i = 0.05$, $\tau = 1.5$.

Now we show the parameter combinations for the marginal SS, i.e. $\|\Gamma\|_2^2 = 1$, based on (28) and (30). We are especially interested in how the planner gain k or k_x can be adjusted to compensate for the slow/fast low-level control captured by the parameter β. Practically it means if the low-level actuator overshoots or undershoots due to some disturbances such as grades, or the low-level controller gains are scaled, how should we adjust the upper planners to maintain the marginal SS. Figs. 20(a) and 20(b) shows the parameter combinations of (k, β) in the OP linear ACC, similarly curves for general linear ACCs are shown in Figs. 21(a) and 21(b). Recall that in Section 3 we reveal that β has a similar effect when applied to the low-level control gains as it is multiplied by the planner gain, which partially explains the linear pattern of the log–log plot in Figs. 21(a) and 21(b). Interestingly, the slope s in each Figs. 20(a), 21(a) and 21(b) has the physical meaning that if β increases by ratio 2, the upper-planner gain needs to increase by 2^s.

The TF of the full loop also casts more insights into balancing low-level control gains. Earlier we found that the integral gain k_i can reverse the impact of slow/fast low-level control on SS. To see how to tune k_i and pair it with k_p, we show the parameter combinations for both the OP linear ACC and the general linear ACC; see Figs. 20(c), 20(d), 21(c) and 21(d). Similarly, the feedforward gain k_f in the PIF low-level controller can be designed to compensate for the actuator performance by simply maintain their product $k_f' = \beta k_f$ according to our earlier analysis in (34).

7. Concluding comments

This study investigates the impact of low-level control on SS, which has previously been ignored in the literature. To summarize, we articulate that the impact of low-level control is significant, which depends on oscillation frequencies for linear ACCs, and is also sensitive to oscillation amplitudes for MPC ACCs. In addition, we point out that for linear ACCs, the slow low-level control undermines SS under perturbations with small frequencies, while it improves SS under perturbations with large frequencies. While the same impacts remain valid for MPC ACCs under small-scale oscillations, we further find that slow/fast low-level control can make MPC gain have opposite evolving pattern under large-scale oscillations. The fast low-level control results in a 'fast-slow' pattern of the varying MPC gain, which effectively prevent speed overshootings, whereas the slow low-level control often contributes to the 'slow-fast' varying MPC gain, which undermines the SS by forcing the MPC to overshoot at the end of the oscillation. Through empirical data we show that slow low-level control is common, which can result from comfortable control gains, environmental disturbances (e.g. grade), and the limited actuator performance. Overall, the study recommends fast low-level control for ensuring vehicular SS to reduce traffic congestion, considering that large congestion waves usually feature both small frequencies and large amplitudes, although slow controllers could perform even better provided a short and small leader perturbation.

The results are based on open-source factory ACC algorithms in Openpilot, Comma.ai. While stock ACC algorithms on commercial car models may vary and still remain proprietary, we conjecture that they follow a bi-level control framework similar to the one introduced in this study, as they share the same stock ACC hardwares (e.g., the radar modules and actuator interfaces from the limited number of manufacturers). As more evidence, we note that the factory ACC algorithms presented here are able to explain some recent empirical findings from market ACC vehicles, such as the varying SS at different speed levels/ headway settings, the observed overshooting/undershooting at the end of an oscillation, and the puzzling gap between many string stable planner designs and string unstable platoons in the real world. The theoretical and experimental methods proposed in this study provide promising new venues to explain more empirical ACC features or approximate the "black-box" ACC products on the market.

Moreover, the study points out that the integral gain, although proved to be useful to reduce steady errors, can substantially deteriorate SS when the low-level control is slow. Notice that although the theoretical analysis in Section 3 does not incorporate anti-windup design for mathematical tractability, all the simulations and road experiments in this paper are using the same codebase in Openpilot, which of course includes the anti-windup design but still reveals the similar impact of integral control on SS. Considering it might be too difficult for PI or PIF low-level controllers to circumvent the integral issue, alternatively we advocate the MPC low-level controllers because: (i) it is also able to combat steady tracking errors but not causing integral overshootings, (ii) an MPC low-level controller can also be leveraged to address the poor actuator performance caused by disturbances such as the grade and air-drag. One can incorporate the grade and air-drag information into the planning horizon of MPC and dynamically increase/decrease the low-level controller gains to compensate for it. Similar methods also apply for other dynamic factors such as the load, rolling resistance, fuel consumption or actuator delays, which can also affect the actuator performance; see Appendix B.

This study also suggests that primarily there are two major types of ACC planners, linear and MPC controllers, both of which originate from the control domain. In this context, it is surprising to note that the large body of the well-established CF models

in the traffic flow domain has neither been applied nor tested. Specifically, the pipeline in Fig. 1 suggests that any model should work if it can generate reasonable target speed or acceleration. This motivates the exploration of the promise of CF models through future research efforts and their feasibility for commercial ACC products. To encourage more experimental testing of ACC algorithms and CF models on real cars, the authors have shared a custom fork of Openpilot at https://github.com/HaoZhouGT/openpilot. This repository includes the implementation of the linear ACC+PI and MPC+PIF frameworks presented in this paper. The low-level controllers are fine-tuned for a 2019 Honda Civic and can be tested in the field by interested readers. We have added a parser to process the raw log files from Comma Two and help retrieve the ACC variables and the CAN bus signals. It is worth noting that the experimental method not only applies to the study of the low-level controllers, any ACC algorithm design (e.g., a new planner CF model) can be tested in the same way. To facilitate such efforts, the authors have implemented the well-known Intelligent Driver Model (IDM) in the same github repository to replace the role of a linear ACC or MPC. Future research efforts can also explore replacing the IDM model with other CF models in our shared repository and testing their performance in the field.

Remarkably, the experiment method in this paper has an unique advantage to obtain detailed ACC control data, including the inputs and outputs of the actuator, which allows us to measure the empirical actuator delay and the impact of the environment such as the grade. Notice that although the actuator delay has been studied for more than 20 years, to the best of our knowledge there do not exist empirical measurements of those delay values in the literature. This is probably due to the limitation of current experiment methods which cannot send commands or record the input signal timing of the actuator. Leveraging the Comma Dev-kit, we are able to obtain the commanded accelerations and the actual vehicle accelerations, which allow us to measure the actuator delay data for the first time in the literature. In addition, from the GNSS (Global navigation satellite system) data we extract the vertical speed of the vehicle, which further helps estimate the grade. The paper did not include delay in our linear analysis, but incorporated a constant delay of 0.5 s in all simulations and the real-car experiments. We also verified the delay does not alter the findings regarding the impact of slow/fast low-level control on the overall SS of ACCs.

CRediT authorship contribution statement

Hao Zhou: Methodology, Experiment design, Data curation, Conceptualization, Writing – original draft, Revision. **Anye Zhou:** Methodology, Conceptualization, Writing – original draft, Revision. **Tienan Li:** Writing – original draft, Revision. **Danjue Chen:** Writing – review & editing, Supervision, Funding acquisition. **Srinivas Peeta:** Writing – review & editing, Supervision, Funding acquisition. **Jorge Laval:** Methodology, Conceptualization, Writing – review & editing, Supervision, Funding acquisition.

Acknowledgments

This study is supported by National Science Foundation (NSF) Cyber-Physical Systems (CPS) grant #1932451 and #1826162. The authors would like to acknowledge the warm help from Joe Pancotti and Chris Souers for their commits in Honda Bosch interface (Pancotti, 2021). We also appreciate the discussions with Shane Smiskol at Openpilot's community.

Appendix A. Intuition of low-level impact in the time domain

A.1. Impact of integral overshootings

The integral term commonly exists in controllers to combat steady errors, however it can cause integral overshootings given large tracking errors of slow controllers. Those overshootings can further deteriorate the SS of ACCs. For completeness we briefly analyze its unique impact here. The role of the integral term in (13) is to accelerate the object towards the setpoint. However, since the integral term responds to accumulated errors from the past, it can cause the present value to overshoot the setpoint value (Willis, 1999). For example, the shaded area in Fig. A.22 is followed by a speed overshoot. Note that the integral overshoot happens after the v_{ego} catches up with v_{pid}. In the real world the observed overshooting (Li et al., 2021) may come from two different sources, planner or low-level controller; see Fig. A.22 for the speed overshooting at the end of an oscillation. We argue the overshooting of v_{pid} compared to the lead speed is mainly due to the planner in response to the extra spacing caused by the slow low-level controller. The additional overshooting of v_{ego} compared to v_{pid} mainly results from the integral accumulation.

Now we investigate the impact of the integral error, namely $\mathcal{I}$, on SS. To this end, for tractability we will derive an upper bound for the accumulated integral error by assuming a P-only controller:

$$control \approx k_p(v_{\text{pid}} - v_{\text{ego}}) \tag{A.1}$$

Recall that a "perfect" pair of *compute_gb* and *gb2accel* functions means $a_{ego} = control$. Then, (A.1) can be written in discrete-time as follows:

$$v_{\text{ego}}(t+1) = v_{\text{ego}}(t) + k_p(v_{\text{pid}}(t+1) - v_{\text{ego}}(t))dt \quad \text{for} \quad t = 1, 2, 3, \dots, \tag{A.2}$$

Combining (A.2) with the initial condition $v_{\text{pid}}(0) = v_{\text{ego}}(0)$, we obtain $v_{\text{ego}}(n)$ at the time step n:

$$v_{\text{ego}}(n) = (1 + k_p dt)(1 - k_p dt)^n v_{\text{ego}}(0) + k_p dt \Sigma_{i=1}^{n}(1 - k_p dt)^{n-i} v_{\text{pid}}(i) \quad \text{for} \quad i = 1, 2, \dots, n-1 \tag{A.3}$$

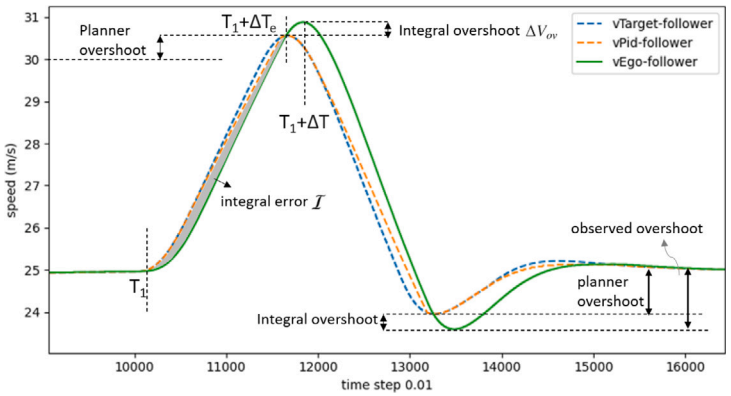


Fig. A.22. Integral accumulation in a PI controller during an oscillation: the lead vehicle changes its speed from 25 m/s to 30 m/s and then reverts to 25 m/s. Starting at time T_1, the follower speed v_{ego} moves towards the setpoint v_{pid} and reaches it for the first time at $T_1 + \Delta T_e$, where $v_{\text{ego}}(T_1 + \Delta T_e) = v_{\text{pid}}(T_1 + \Delta T_e)$ and $v_{\text{ego}}(t) < v_{\text{pid}}(t)$ for $T_1 < t < T_1 + \Delta T_e < T_1 + \Delta T$ in the acceleration case.

Now we can quantify the tracking error $e(n) = v_{\text{pid}}(n) - v_{\text{ego}}(n)$ at time step n:

$$e(n) = (1 - k_p dt)v_{\text{pid}}(n) - (1 + k_p dt)(1 - k_p dt)^n v_{\text{ego}}(0) - k_p dt \Sigma_{i=1}^{n-1}(1 - k_p dt)^{n-i} v_{\text{pid}}(i) \tag{A.4}$$

Typical values for k_p and dt are 1 and 0.01 s, respectively. Thus $(1 - k_p dt)^n$ tends towards zero for large n, making the second term $(1 + k_p dt)(1 - k_p dt)^n v_{\text{ego}}(0) \to 0$. In addition, as $k_p dt \ll 1$, the third term $(k_p dt)\Sigma_{i=1}^{n-1}(1 - k_p dt)^{n-i} v_{\text{pid}}(i)$ is negligible comparable to $(1 - k_p dt)v_{\text{pid}}(n)$. Therefore, the first term is the dominating one, which suggests the following approximation for $\mathcal{I}$:

$$\mathcal{I} = \Sigma_{n=T}^{T+\Delta T_e} e(n) \approx (1 - k_p dt)\Sigma_{n=T}^{T+\Delta T_e} v_{\text{pid}}(n) \tag{A.5}$$

It can be seen the integral error $\mathcal{I}$ decreases with k_p within its sensible range. This indicates that a slow controller may cause large integral error and undermines the SS.

A.2. SS analysis of the low-level impact in time domain

For an ACC system consisting with an acceleration planner and the PIF low-level controller, we track the speed change of v_{start} to study SS since it is the surrogate variable for v_{ego}. Setting $a_{\text{start}} = 0$ in (12) the a_{start} sequence can be simplified using only a_{target}, which leads to:

$$a_{\text{start}}(\hat{t}) = \sum_{n=0}^{\lfloor \hat{t}/d\hat{t} \rfloor - 1} \frac{d\hat{t}}{dt_p}(1 - d\hat{t}/dt_p)^{\lfloor \hat{t}/d\hat{t} \rfloor - 1 - n} a_{\text{target}}(nd\hat{t}) \tag{A.6}$$

where $\lfloor \hat{t}/d\hat{t} \rfloor$ is a floor function to calculate index for planning step of $\hat{t}$ starting from zero. Substitute (A.6) into (12), the changes of v_{start} can be described using a_{target} from the planner:

$$2/d\hat{t} \cdot \Delta v_{\text{start}} = \sum_{T_1 \le \hat{t} \le T_1 + \Delta T} a_{\text{target}}(\hat{t}) + \sum_{n=0}^{\lfloor \hat{t}/d\hat{t} \rfloor} \frac{d\hat{t}}{dt_p}(1 - \frac{d\hat{t}}{dt_p})^{\lfloor \hat{t}/d\hat{t} \rfloor - 1 - n} a_{\text{target}}(nd\hat{t}) \tag{A.7}$$

Correspondingly, the SS index I_{ss} given a MPC planner can be derived as:

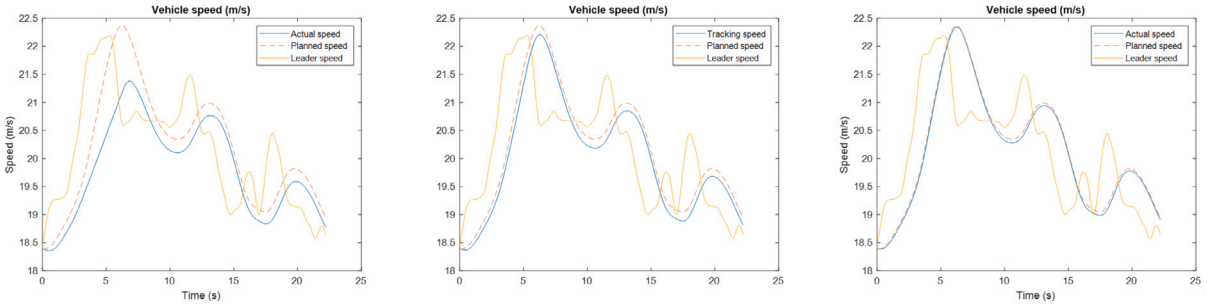
$$\begin{aligned} I_{ss} &= \frac{|\Delta v_{\text{start}}| - |\Delta v_{\text{lead}}|}{|\Delta v_{\text{lead}}|} \\ &= -1 + \sum_{T_1 \le \hat{t} \le T_1 + \Delta T} |a_{\text{target}}(\hat{t})| \frac{d\hat{t}}{2|\Delta v_{\text{lead}}|} + \sum_{n=0}^{\lfloor \hat{t}/d\hat{t} \rfloor} \frac{d\hat{t}(1 - d\hat{t}/dt_p)^{\lfloor \hat{t}/d\hat{t} \rfloor - 1 - n}}{dt_p |\Delta v_{\text{lead}}|} |a_{\text{target}}(nd\hat{t})| \end{aligned} \tag{A.8}$$

where a smaller I_{ss} means better SS.

The impact of slow low-level control can be understood as follows in the time domain: A slow low-level controller makes the true spacing s_{ego} deviate more from the desired spacing s_{des}, which consequently induces larger accelerations $|a_{\text{target}}(nd\hat{t})|$ and longer time ΔT to rectify s_{ego} and v_{ego} to desired values, thus increases the I_{ss} to the detriment of SS.

Appendix B. Impacts of real-world disturbances on actuator performance

During ACC operations, the real-world disturbances (i.e., varying road grade, vehicle loads, rolling resistance, air-drag, etc.) can exert great impact on the tracking performance of the low-level controller, if it is not counteracted by the vehicle powertrain control



(a) Increased load (b) Increased rolling resistance (c) Increased air-drag

Fig. B.23. Increased load, rolling resistance and air-drag contribute to slow low-level control.

(engine/motor and transmission control algorithms designed by the automakers). The paper has already shown the grade impact using empirical data from a Civic and Rav4. For other disturbances which are difficult to capture, here we apply the simplified nonlinear vehicle longitudinal dynamics in Wu et al. (2016) to describe the their impacts on vehicle motions:

$$\dot{x} = v \tag{B.1}$$

$$\dot{v} = \frac{\eta \frac{T}{R} - C_a v^2 - F_f}{m} \tag{B.2}$$

$$\dot{T} = \frac{u - T}{\tau_a} \tag{B.3}$$

where η is the transmission efficiency, R is the wheel radius, T is the engine/brake torque output, C_a is the air drag coefficient, m is the vehicle mass, $F_f = mg(sin(\theta) + f cos(\theta))$ indicates the impacts of rolling resistance and gravity, $g = 9.81$ is the gravity coefficient, f is the friction coefficient, θ is the road grade angle. u is the torque command send to the engine/brake, τ_a is the engine/brake delay factor. Then, to manifest the impacts of disturbances during the real-world operations, the general linear CF model in (21) is used as the upper-level planner. We then apply the feedback linearization technique in Khalil (2008) to devise the low-level actuator control (i.e., determine the desired engine torque of the vehicle to track the planned trajectory). Correspondingly, we transform the vehicle dynamics into the following strict feedback form:

$$\dot{\zeta}_1 = \zeta_2 \tag{B.4}$$

$$\dot{\zeta}_2 = \zeta_3 \tag{B.5}$$

$$\dot{\zeta}_3 = \frac{1}{m}\left[\frac{u - R(m\zeta_3 + C_a\zeta_2^2 + F_f)/\eta}{\tau_a} - 2C_a\zeta_2\zeta_3\right] \tag{B.6}$$

where $\zeta = [x, v, a]$ is the transmission efficiency, $u = R(m\zeta_3 + C_a\zeta_2^2 + F_f)/\eta + 2C_a\zeta_2\zeta_3\tau_a + m\tau_a\iota$ is the desired driving torque, $\iota = k_p e_v + k_i e_x + k_d e_a$ is the PID control input which aims to achieve desired tracking performance, k_p, k_p, and k_p are the proportional gain, integral gain, and derivative gain, respectively. e_x, e_v, and e_a are the spacing error, speed tracking error, and acceleration tracking error, respectively. The simulation results in Fig. B.23 show that, similar to the impacts of road grade and weak actuator, the increased vehicle loads, rolling resistance and air-drag can all contribute to slow tracking of desired speeds in regular CF scenarios. Thereby, factoring real-world disturbances in the low-level controller deign is significant for realizing string stability of an ACC system.

References

Åström, K.J., Murray, R.M., 2008. Feedback Systems: An Introduction for Scientists and Engineers. Princeton university press Princeton, NJ.

Collins, D., 2021. How to tune servo systems for high dynamic response? URL: https://www.motioncontroltips.com/faq-tune-servo-system-high-dynamic-response/.

Comma.ai, 2020. Comma.ai two setup. URL: https://comma.ai/setup.

Comma.ai, 2021. Comma.ai – introducing openpilot. URL: https://comma.ai/.

Corona, D., De Schutter, B., 2008. Adaptive cruise control for a SMART car: A comparison benchmark for MPC-PWA control methods. IEEE Trans. Control Syst. Technol. 16 (2), 365–372.

Diehl, M., 2014. Toolkit for Automatic Control and Dynamic Optimization. URL: https://acado.github.io/.

Eilbert, A., Chouinard, A.M., Tiernan, T., Smith, S., 2020. Performance comparisons of cooperative and adaptive cruise control testing.

Feng, S., Zhang, Y., Li, S., Cao, Z., Liu, H., Li, L., 2019. String stability for vehicular platoon control: Definitions and analysis methods. Annu. Rev. Control. 47, 81–97.

Gong, S., Shen, J., Du, L., 2016. Constrained optimization and distributed computation based car following control of a connected and autonomous vehicle platoon. Transp. Res. B 94, 314–334.

Gunter, G., Gloudemans, D., Stern, R.E., McQuade, S., Bhadani, R., Bunting, M., Delle Monache, M.L., Lysecky, R., Seibold, B., Sprinkle, J., et al., 2020. Are commercially implemented adaptive cruise control systems string stable? IEEE Trans. Intell. Transp. Syst..

Gunter, G., Janssen, C., Barbour, W., Stern, R.E., Work, D.B., 2019. Model-based string stability of adaptive cruise control systems using field data. IEEE Trans. Intell. Veh. 5 (1), 90–99.

Kanojiya, R.G., Meshram, P., 2012. Optimal tuning of PI controller for speed control of DC motor drive using particle swarm optimization. In: 2012 International Conference on Advances in Power Conversion and Energy Technologies. APCET, IEEE, pp. 1–6.

Khalil, H.K., 2008. Nonlinear Systems, third ed..

Laval, J.A., Toth, C.S., Zhou, Y., 2014. A parsimonious model for the formation of oscillations in car-following models. Transp. Res. B 70, 228–238.

Li, X., 2020. Trade-off between safety, mobility and stability in automated vehicle following control: An analytical method.

Li, T., Chen, D., Zhou, H., Laval, J., Xie, Y., 2021. Car-following behavior characteristics of adaptive cruise control vehicles based on empirical experiments. Transp. Res. B 147, 67–91. http://dx.doi.org/10.1016/j.trb.2021.03.003.

Li, S.E., Gao, F., Cao, D., Li, K., 2016. Multiple-model switching control of vehicle longitudinal dynamics for platoon-level automation. IEEE Trans. Veh. Technol. 65 (6), 4480–4492. http://dx.doi.org/10.1109/TVT.2016.2541219.

Li, S., Li, K., Rajamani, R., Wang, J., 2010. Model predictive multi-objective vehicular adaptive cruise control. IEEE Trans. Control Syst. Technol. 19 (3), 556–566.

Liang, C.Y., Peng, H., 1999. Optimal adaptive cruise control with guaranteed string stability. Veh. Syst. Dyn. 32 (4–5), 313–330.

Liang, C., Peng, H., 2000. String stability analysis of adaptive cruise controlled vehicles. Jsme Int. J. Ser. C-Mech. Syst. Mach. Elem. Manuf. 43, 671–677.

Lu, X., Shladover, S., 2018. Truck CACC system designand DSRC messages.

Makridis, M., Mattas, K., Anesiadou, A., Ciuffo, B., 2021. OpenACC. An open database of car-following experiments to study the properties of commercial ACC systems. Transp. Res. C 125, 103047.

Montanino, M., Punzo, V., 2021. On string stability of a mixed and heterogeneous traffic flow: A unifying modelling framework. Transp. Res. B 144, 133–154.

Naus, G., Ploeg, J., Van De Molengraft, R., Steinbuch, M., 2008. Explicit MPC design and performance-based tuning of an adaptive cruise control stop-&-go. In: 2008 IEEE Intelligent Vehicles Symposium. IEEE, pp. 434–439.

Naus, G.J., Vugts, R.P., Ploeg, J., van De Molengraft, M.J., Steinbuch, M., 2010. String-stable CACC design and experimental validation: A frequency-domain approach. IEEE Trans. Veh. Technol. 59 (9), 4268–4279.

Newell, G.F., 2002. A simplified car-following theory: a lower order model. Transp. Res. B 36 (3), 195–205.

O'Dwyer, A., 2006. PI and PID controller tuning rules: An overview and personal perspective.

O'dwyer, A., 2009. Handbook of PI and PID Controller Tuning Rules. World Scientific.

Pancotti, J., 2021. Honda bosch interface for openpilot. URL: https://github.com/jpancotti/openpilot/commit/468b89a.

Ploeg, J., Scheepers, B.T.M., van Nunen, E., van de Wouw, N., Nijmeijer, H., 2011. Design and experimental evaluation of cooperative adaptive cruise control. In: 2011 14th International IEEE Conference on Intelligent Transportation Systems. ITSC, pp. 260–265. http://dx.doi.org/10.1109/ITSC.2011.6082981.

Rajamani, R., Choi, S.B., Law, B.K., Hedrick, J.K., Prohaska, R., Kretz, P., 2000. Design and experimental implementation of longitudinal control for a platoon of automated vehicles. J. Dyn. Syst. Meas. Control-Trans. Asme 122, 470–476.

Rakha, H., Lucic, I., Demarchi, S.H., Setti, J.R., Aerde, M.V., 2001. Vehicle dynamics model for predicting maximum truck acceleration levels. J. Transp. Eng. 127 (5), 418–425.

Shaw, E., Hedrick, J.K., 2007. String stability analysis for heterogeneous vehicle strings. In: 2007 American Control Conference. pp. 3118–3125. http://dx.doi.org/10.1109/ACC.2007.4282789.

Shi, X., Li, X., 2021. Empirical study on car-following characteristics of commercial automated vehicles with different headway settings. Transp. Res. C 128, 103134.

Shladover, S., 2009. Effects of Cooperative Adaptive Cruise Control on Traffic Flow: Testing Drivers' Choices of Following Distances. PATH Research Report.

Smiskol, S., 2021. Discussion on the "compute_gb" function in openpilot community.

Wang, Y.G., Shao, H.H., 2000. Optimal tuning for PI controller. Automatica 36 (1), 147–152.

Willis, M., 1999. Proportional-Integral-Derivative Control. Dept. of Chemical and Process Engineering University of Newcastle.

Wilson, R.E., Ward, J.A., 2011. Car-following modelsfifty years of linear stability analysis a mathematical perspective. Transp. Plan. Technol. 34 (1), 3–18.

Wu, Y., Li, S.E., Zheng, Y., Hedrick, J.K., 2016. Distributed sliding mode control for multi-vehicle systems with positive definite topologies. In: 2016 IEEE 55th Conference on Decision and Control. CDC, pp. 5213–5219. http://dx.doi.org/10.1109/CDC.2016.7799067.

Xu, T., Laval, J., 2020. Statistical inference for two-regime stochastic car-following models. Transp. Res. B 134, 210–228.

Yanakiev, D., Kanellakopoulos, I., 1995. Variable time headway for string stability of automated heavy-duty vehicles. In: Proceedings of 1995 34th IEEE Conference on Decision and Control, Vol. 4. IEEE, pp. 4077–4081.

Zheng, Y., Li, S.E., Li, K., Borrelli, F., Hedrick, J.K., 2017. Distributed model predictive control for heterogeneous vehicle platoons under unidirectional topologies. IEEE Trans. Control Syst. Technol. 25 (3), 899–910. http://dx.doi.org/10.1109/TCST.2016.2594588.

Zhou, Y., Ahn, S., 2019. Robust local and string stability for a decentralized car following control strategy for connected automated vehicles. Transp. Res. B 125, 175–196.

Zhou, A., Gong, S., Wang, C., Peeta, S., 2020. Smooth-switching control-based cooperative adaptive cruise control by considering dynamic information flow topology. Transp. Res. Rec. 2674, 444–458.

Zhou, J., Peng, H., 2005. Range policy of adaptive cruise control vehicles for improved flow stability and string stability. IEEE Trans. Intell. Transp. Syst. 6 (2), 229–237.

Ziegler, J.G., Nichols, N., 1942. Optimum settings for automatic controllers. J. Dyn. Syst. Meas. Control-Trans. Asme 115, 220–222.