POLITECNICO DI TORINO

Master's Degree in AUTOMOTIVE ENGINEERING

Master's Degree Thesis

Energy-Efficient Adaptive Cruise Control for EVs in Urban Scenarios with Traffic Lights Negotiation

Supervisors

Prof. ANDREA TONOLI

PhD Candidate STEFANO FAVELLI

Candidate

CHENGYANG YE

JULY 2024

Abstract

The rapid urbanization and increasing environmental concerns have driven the demand for efficient and sustainable transportation solutions. Small electric vehicles are becoming a popular choice for urban commuting due to their low emissions and cost-effectiveness. However, optimizing energy consumption remains a critical challenge for enhancing the overall efficiency and practicality of these vehicles in complex urban environments. Nowadays, the Advanced Driver Assistance Systems (ADAS) of electric vehicles are flourishing. With the gradual enhancement of the computational power of onboard chips, more complex algorithms, such as Model Predict Control (MPC), can be applied in real-time to ADAS. With the advancement of communication technology, more information such as Signal Phase and Timing (SPaT) can be obtained through Vehicle-to-Infrastructure (V2I) technology, providing the potential for further enhancing the capabilities of ADAS.

The main work of this thesis is developing an advanced vehicle controller based on MPC that seamlessly integrates vehicle following and traffic light information to minimize energy consumption while optimizing driving comfort. The controller utilizes Vehicle-to-Vehicle (V2V) technology or estimation methods to obtain the lead vehicle's trajectory. By dynamically adjusting the headway distance to the preceding vehicle, the controller achieves efficient energy management within the complex and congested urban traffic conditions. A significant aspect of the proposed system is the integration of SPaT information through V2I communication technology. This allows the vehicle to anticipate upcoming traffic signals, proactively adjust its speed, and thereby reduce unnecessary acceleration and braking. In addition, this approach also decreases overall resistance, thereby enhancing energy efficiency. The controller no longer switches between speed tracking and car-following tasks as two separate modes; instead, it integrates them into a single algorithm that balances both tasks in real-time.

The CasADi toolbox coded in MATLAB is used for controller implementation. Firstly, the focus is on designing the non-linear programming architecture, then it is transformed into quadratic programming to deploy it in real-time. The algorithm is compiled with MinGW64 to generate C code and implemented in Simulink with the vehicle model for Software-in-the-Loop (SiL) testing. The driving cycle used is based on a human driver driving recorded in Torino.

We use Monte Carlo simulation-based method to tune the controller and the simulation results demonstrated that the proposed controller performs well in different scenarios, significantly reducing the energy consumption of electric vehicle, improving travel efficiency, and providing a safe and reliable driving experience. Energy consumption can be reduced up to 12%, depending on different scenario and working logic. The inclusion of V2V information is shown to markedly improve performance in terms of energy savings and driving comfort, and it is particularly beneficial for multiple vehicles engaged in cooperative adaptive cruise control.

This study serves as a valuable reference for the future development of intelligent transportation systems. It underscores the potential of integrating advanced communication technologies and predictive control strategies to achieve sustainable and efficient urban mobility solutions for electric vehicles.

Keywords: Model Predictive Control, Adaptive Cruise Control, Vehicle-to-Infrastructure, Energy Efficiency

Table of Contents

Chapter 1

Introduction

1.1 Background

With the accelerating process of global urbanization and the continuous growth of urban populations, urban traffic congestion and air pollution have become increasingly severe issues. Private cars, as a significant component of urban transportation, have a notable impact on environmental pollution. According to statistics from the World Health Organization (WHO) and the International Energy Agency (IEA), emissions from private cars in cities account for over 30% of total urban pollution emissions, making them a primary source of urban air pollution and greenhouse gas emissions [1][2]. Additionally, frequent stops and starts and low-speed driving modes in cities further exacerbate the energy consumption and pollution emissions of private cars. The transportation sector accounts for about 24% of global CO2 emissions, with private cars contributing a significant portion [3].

To address these challenges, many countries and regions have introduced relevant policies to promote the development and adoption of electric vehicles (EVs), reduce greenhouse gas emissions, and achieve sustainable development goals. For example:

- China's "New Energy Vehicle Industry Development Plan (2021-2035)" explicitly states that by 2035, new energy vehicles will become the mainstream, with full electrification in the public sector. The Chinese government supports the development of new energy vehicles through subsidies, tax incentives, and infrastructure construction, aiming for new energy vehicle sales to account for about 20% of total vehicle sales by 2025 [4]. The dual-credit policy requires automakers to produce a certain proportion of new energy vehicles while producing traditional fuel vehicles. If they fail to meet the target, they need to purchase credits from other manufacturers, thus promoting the production and technological advancement of new energy vehicles [5].

- The European Union's "European Green Deal" aims to achieve carbon neutrality by 2050, with the promotion of zero-emission vehicles being a crucial component. According to this agreement, by 2030, the EU plans to reduce greenhouse gas emissions in the transportation sector by at least 55%. To this end, the EU has established a series of strict emission standards and promotes the development and market adoption of electric vehicles through financial support and policy incentives. For example, the EU has set strict CO2 emission limits for new cars, not exceeding 95 grams per kilometer by 2021, and further reducing to 59 grams per kilometer by 2030 [6]. These standards force automakers to accelerate the electrification process.

- The United States' "Clean Future Act" aims to achieve net-zero emissions by 2030, including significant investments in electric vehicle infrastructure such as charging station networks and providing purchase subsidies to encourage consumers to buy electric vehicles. Additionally, the act supports the research and development of electric vehicle technologies, accelerating the commercialization of new technologies [7]. California's Zero-Emission Vehicle Program, as a pioneer in the US, stipulates that all new passenger cars sold by 2035 will be zero-emission, effectively banning the sale of new fuel vehicles. This plan has significantly promoted the adoption of electric vehicles and inspired other states to follow suit [8].

- Japan's "Green Growth Strategy" proposes that by 2035, all new cars sold will be electric vehicles (including hydrogen fuel cell vehicles and hybrid vehicles). By providing financial subsidies, research funds, and infrastructure construction, Japan aims to become a leading global market for new energy vehicles [9]. These policies commonly leverage legislation and economic incentives to promote the production and consumption of electric vehicles, reducing greenhouse gas emissions in the transportation sector and achieving sustainable development goals.

The widespread adoption of electric vehicles in urban travel is crucial for improving air quality and reducing greenhouse gas emissions. Electric vehicles can significantly reduce exhaust emissions and reliance on fossil fuels, thereby lowering urban air pollution levels. This has important implications for enhancing residents' quality of life and promoting public health. Moreover, electric vehicles have notable advantages in terms of energy efficiency, particularly in congested and low-speed urban environments, where their efficient electric drive systems consume less energy than traditional internal combustion engine vehicles under similar conditions.

However, the energy efficiency and range of electric vehicles remain key challenges in their widespread adoption. Limitations in battery technology make it difficult for current electric vehicles to match the range of traditional fuel vehicles. Additionally, the slow rate of battery recharging extends the time needed to replenish energy, especially during peak periods and long-distance travel, where charging time becomes a critical factor affecting user experience. Optimizing the energy consumption management of electric vehicles is thus crucial for their broad application.

In urban traffic environments, improving the energy efficiency of electric vehicles through advanced control strategies and communication technologies can extend their range, reduce operating costs, and decrease energy consumption and environmental pollution, further promoting the adoption and use of electric vehicles.

Model Predictive Control (MPC) has a rich history and wide range of applications in various fields. Initially developed in the late 1970s and early 1980s for the process industries, MPC has evolved into a powerful and versatile control strategy used in various domains, including automotive engineering, aerospace, and robotics. The primary advantage of MPC lies in its ability to handle multi-variable control problems with constraints while optimizing performance criteria. By predicting future system behavior based on a dynamic model, MPC can make real-time adjustments to control inputs, ensuring optimal operation even in the presence of disturbances and uncertainties.

MPC's history dates back to the late 1970s when it was primarily applied in the petrochemical industry for complex process control. With advances in computational power and algorithmic development, MPC's application scope has expanded significantly. In the 21st century, MPC has become widely adopted in the automotive industry, particularly for engine control, chassis control, and advanced driver-assistance systems. The real-time requirements of automotive applications demand that MPC algorithms respond swiftly and accurately to dynamic driving conditions, making it ideal for optimizing performance and safety. In addition to its automotive applications, MPC has also been utilized in aerospace for aircraft attitude control and trajectory optimization, and in robotics for path planning and motion control.

The controller developed in this research utilizes CasADi for coding within MATLAB. CasADi[10] is an open-source software framework for numerical optimization and algorithmic differentiation, created by Joel Andersson, Joris Gillis, Greg Horn, James Rawlings, and Moritz Diehl. It is particularly well-suited for solving dynamic optimization problems and is widely used in the field of control engineering. CasADi provides a powerful and flexible environment for implementing MPC algorithms, allowing for efficient and real-time optimization. Its ability to handle complex mathematical models and perform automatic differentiation makes it an invaluable tool for developing advanced control systems. Using CasADi within MATLAB has significantly enhanced our capability to perform complex optimizations efficiently. We would like to express our gratitude to Joel Andersson and the CasADi development team for their contributions to this powerful tool, which has greatly facilitated the implementation of our MPC-based controller.

1.2 Motivation

Driven by global policies, the electric vehicle market is rapidly expanding, becoming a key solution for achieving environmental goals and improving urban transportation. However, despite the significant environmental benefits of electric vehicles, their range and energy efficiency issues remain major challenges to their widespread adoption. Therefore, optimizing energy consumption in complex urban traffic environments to enhance operational efficiency is a primary research focus.

Frequent stops and starts and low-speed driving modes during peak traffic hours result in significant energy waste and pollution emissions from private cars. Traditional following control systems often fail to fully utilize preceding vehicle trajectory and traffic signal information, leading to frequent acceleration and braking, increasing energy consumption and driving resistance. To overcome these issues, this research proposes a MPC based vehicle controller. MPC is chosen because of its ability to predict future system states and optimize control actions based on these predictions. This allows for real-time adjustments that can account for dynamic changes in the traffic environment and vehicle states.

The proposed MPC-based controller can obtain preceding vehicle trajectories through Vehicle-to-Vehicle (V2V) communication technology or predict the trajectories using known information and assumptions, allowing for dynamic adjustment of headway distance and driving speed to optimize energy management. For example, by obtaining or predicting traffic signal changes in advance, vehicles can decelerate early, avoiding sudden braking and re-acceleration, thereby saving energy.

Optimizing the driving strategy of electric vehicles in urban environments not only helps extend battery range and reduce energy consumption but also improves travel efficiency, reduces traffic congestion, and enhances driving comfort. For instance, by dynamically adjusting headway distance, vehicles can follow the preceding vehicle more smoothly, reducing the discomfort caused by frequent braking. This approach aligns with the principles of sustainable development and provides technical support for the future development of Intelligent Transportation Systems.

Policy-driven initiatives and technological advancements provide a practical background and application scenario for this research. For instance, China's "Smart Transportation" initiative and Europe's "Connected and Automated Mobility" strategy offer broad platforms and opportunities for implementing this research. Therefore, the motivation for this study is to improve the energy efficiency and driving comfort of electric vehicles in urban traffic environments through advanced control algorithms and communication technologies, providing new solutions for the development of Intelligent Transportation Systems and responding to global policies on clean energy and smart transportation.

1.3 Thesis Outline

This thesis work is structured as follows:

- Chapter 2 include the Theoretical Background, introduce the Adaptive cruise control, Model Predictive Control, driving comfort.
- Chapter 3 is dedicated to the design of system architecture, discuss about the reference speed generator and MPC controller base on NLP an QP
- Chapter 4 shows the simulation result, discuss the performance of controller in different scenario, discuss the lead vehicle trajectory estimation, headway policy and string stability
- Chapter 5 is the final chapter, where conclusions and future works are reported

Chapter 2

Theoretical Background

Before discussing the proposed method, this chapter focuses on introducing the theoretical background of this thesis. Firstly, introduce Adaptive Cruise Control (ACC) and its history, explaining the headway distance policy, string stability, and Cooperative Adaptive Cruise Control (CACC). Secondly, discuss the control logic, Model Predictive Control (MPC), detailing its structure, formulation, and execution tools. Third, address issues related to driving comfort.

2.1 Adaptive cruise control

2.1.1 Introduction

Adaptive Cruise Control (ACC) is an advanced driver assistance system designed to enhance the driving experience and improve safety. This system uses radar, cameras, and sensors to detect the speed and distance of the vehicle ahead, automatically adjusting the car's speed to maintain a preset following distance. When the vehicle in front slows down or stops, the ACC system will decelerate or even bring the car to a complete stop; when the road clears or the vehicle in front speeds up, the system will resume the preset speed. This reduces the driver's operational burden and significantly improves comfort and safety during long-distance driving

2.1.2 Head way distance policy

The headway policy is a strategy used to maintain a safe and comfortable following distance between a vehicle and the one ahead. It is a function related to the ego vehicle's speed. There are several types of headway policies. In the paper "Spacing Policies for Adaptive Cruise Control: A Survey"[11] by Cunxue Wu and Zhongming Xu et al. (2020), the existing headway policies are discussed in detail.

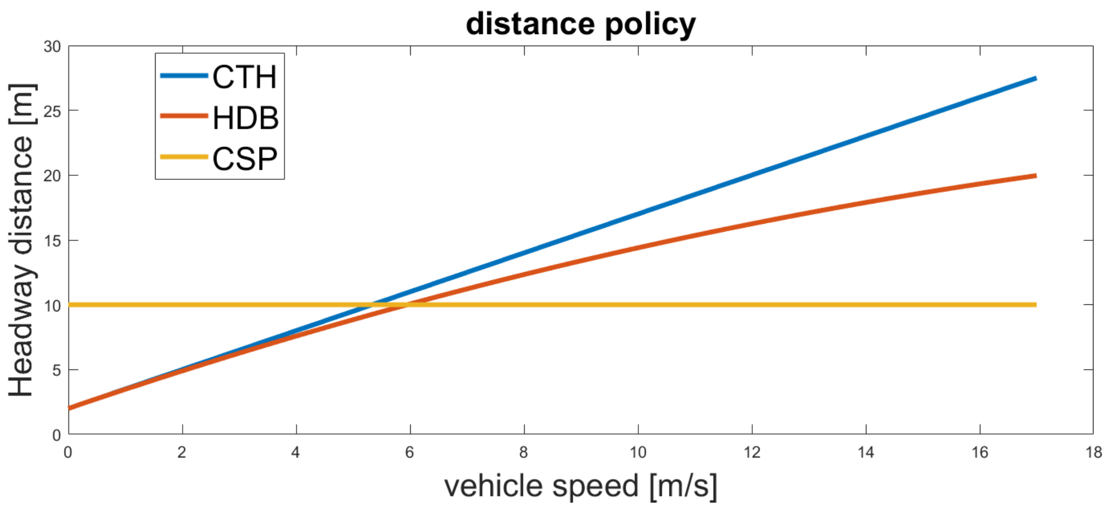


Figure 2.1: Headway distance policy of ACC

The simplest headway distance policy is the Constant Spacing Policy (CSP), which ensures that the vehicle maintains a constant distance from the lead vehicle regardless of the ego vehicle's speed.

$$L_{des} = constant \tag{2.1}$$

A widely used policy is the Constant Time Headway (CTH). In this policy, the headway distance is proportional to the vehicle speed, where τ is the time distance in seconds, and v is the vehicle speed. It is a simple and efficient policy; however, at high vehicle speeds, the headway distance may become too large, which can impact traffic efficiency.

$$L_{des} = \tau v \tag{2.2}$$

Another popular policy is Human Driver Behavior Policy (HDB). This policy uses a polynomial to approximate the human driver's following distance, which can reduce the headway distance at high vehicle speeds compare to CTH. where $A = 2, T = 1.5, G = -0.0246T + 0.010819$.

$$L_{des} = A + Tv + Gv^2 \tag{2.3}$$

Spacing error is the offset between the desired headway distance and the actual headway distance. It is one of the key performance indicators (KPIs) used to evaluate the performance of ACC.

2.1.3 String stability

String stability [12] refers to the ability of a platoon of vehicles (a line of vehicles traveling together) to maintain uniform spacing and speed without amplifying disturbances as they propagate through the platoon. In simpler terms, it means that if the lead vehicle changes speed, the following vehicles can smoothly adjust their speeds without causing increasing oscillations in acceleration and deceleration, thereby preventing instability or traffic jams.

For simple Adaptive Cruise Control, the headway policy strongly affects string stability. The Constant Spacing Policy (CSP) is a typical condition of string instability, which means that the spacing errors of subsequent vehicles are larger than those of preceding vehicles. The figure 2.2, 2.3 shows the concepts of string stability and string instability . In this scenario, the lead vehicle has a sinusoidal speed trajectory, and δ_n represents the spacing error (the difference between the desired headway distance and the actual headway distance) for the nth following vehicle.

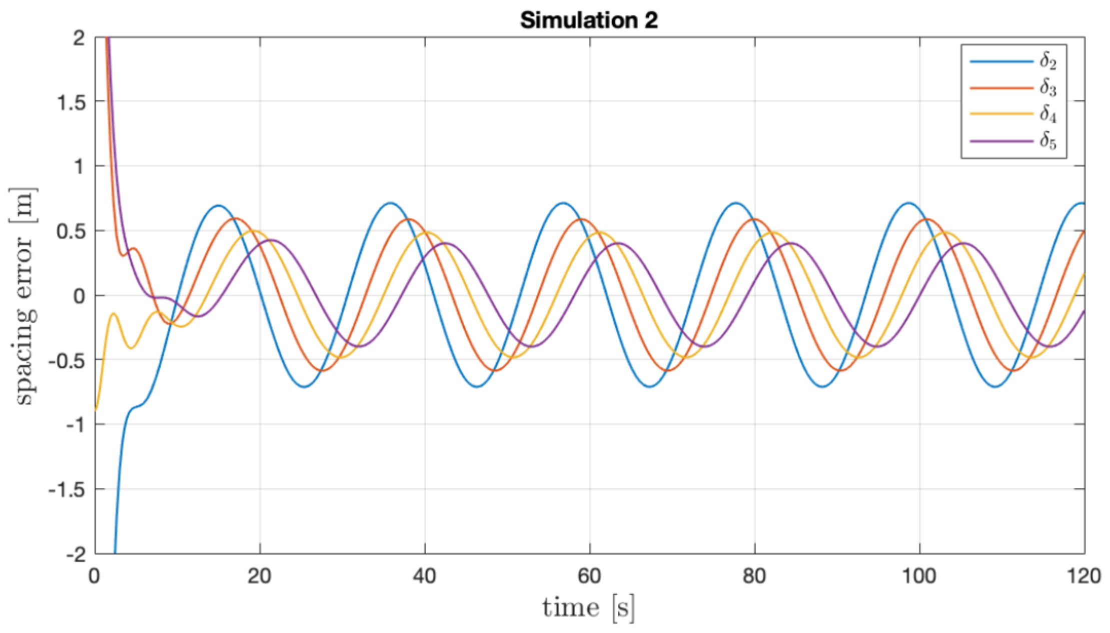


Figure 2.2: String stability convoy

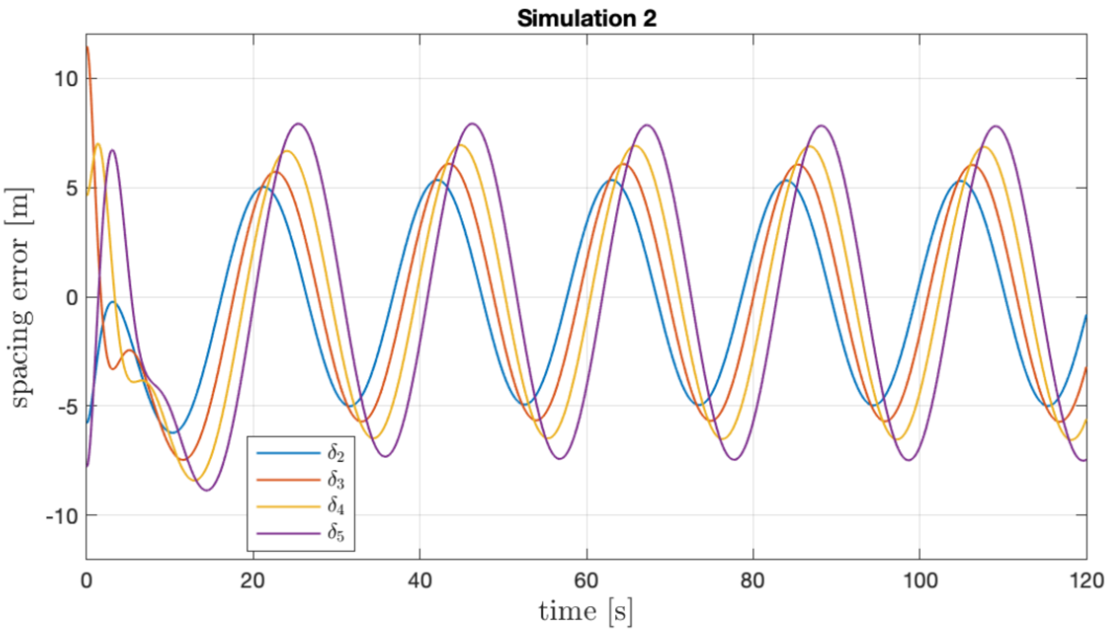


Figure 2.3: String instability convoy

2.1.4 Cooperative Adaptive Cruise Control

Cooperative Adaptive Cruise Control (CACC), is an advanced driver-assistance system that enhances driving safety, efficiency, and fuel economy by enabling communication and coordination among vehicles. Building on traditional Adaptive Cruise Control, CACC incorporates wireless communication between vehicles, allowing them to share information such as speed, acceleration, and emergency braking events. This information sharing enables vehicles to adjust their speed and following distance more precisely, reducing traffic congestion, increasing road capacity, and lowering the risk of collisions. CACC systems rely on Vehicle-to-Everything (V2X) technology, which includes Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication, ensuring real-time data exchange and efficient vehicle coordination. By improving cruise control performance not only for individual vehicles but also for groups of vehicles traveling together, CACC maintains stable following distances and formations, enhancing the overall efficiency and safety of traffic flow. [13] show a example of CACC

2.2 Model Predict Control

Model Predictive Control is an advanced control strategy extensively employed in industrial process control and various dynamic systems. The fundamental concept involves using a mathematical model of the system to forecast future behavior and optimize current control actions accordingly. With advancements in chip processing power, MPC has increasingly found applications in the automotive industry, where it emphasizes real-time performance.

Paper [14] detail introduce Model predict control, Figure 2.4 illustrates the working principle of the Model Predictive Control system. The MPC controller comprises a plant model and an optimizer. The plant model is utilized to predict the future states of the plant over a specified prediction horizon. Based on these predictions and the desired reference trajectory, the optimizer computes the optimal control signals. These control signals are then applied to the plant to achieve the desired performance while satisfying constraints.

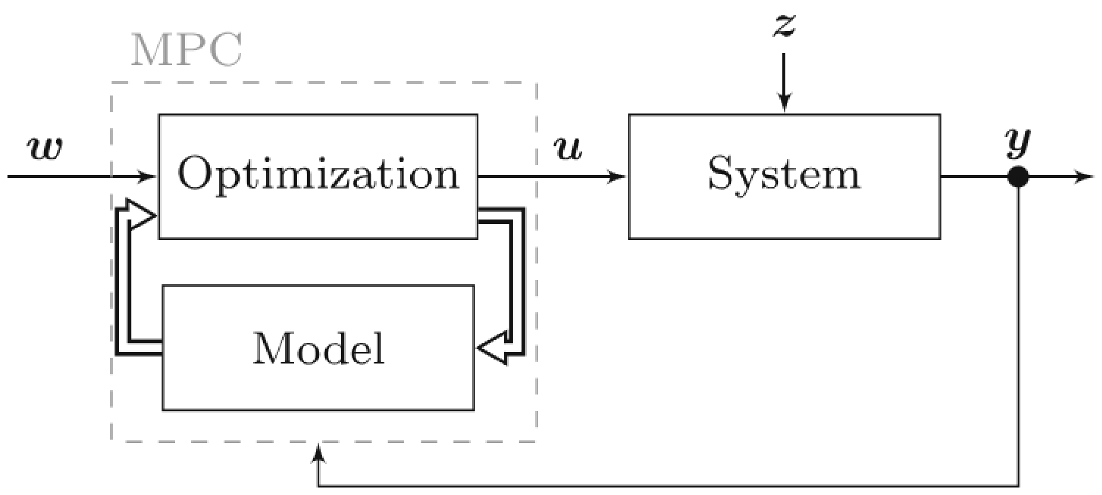


Figure 2.4: MPC loop base on [14]

2.2.1 Predict Horizon

Figure 2.5 shows the predict horizon. In Model Predictive Control, the predict horizon is the length of time or the number of discrete time steps over which the controller predicts the future states and outputs of the system at each control step. Specifically, the predict horizon determines the future time span considered in the optimization process, meaning that at each control interval, the controller uses the current system state and a known model to forecast the system's behavior and state changes over several future time steps. The controller optimizes the future control input sequence within this prediction horizon to minimize a predefined cost function, which typically includes the deviation between the reference state and predicted state and the cost of the control inputs. In practice, although the controller calculates the inputs over the entire prediction horizon, it only applies the control input for the first time step and then re-evaluates the system state, repeating the optimization process. Then, only the first control variable will be executed. The process is then repeated at the next time step.

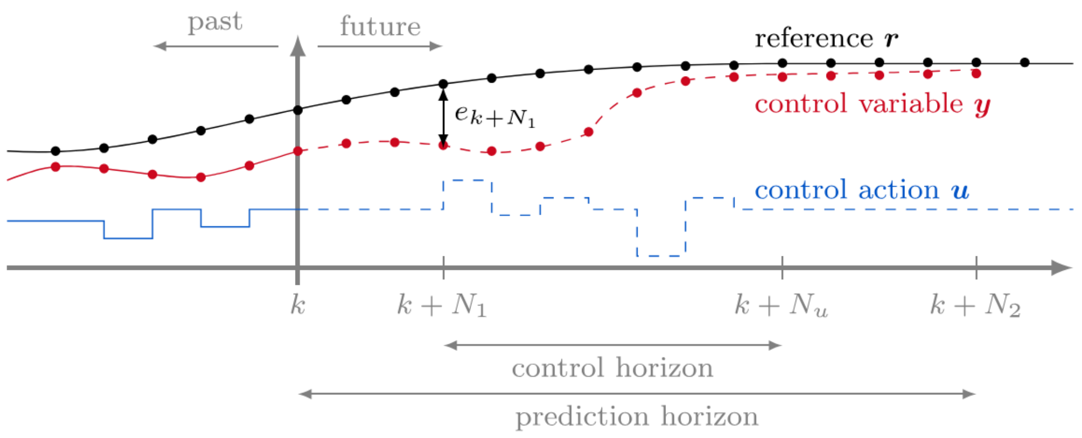

past
future
reference $\boldsymbol{r}$
e_{k+N_1}
control variable $\boldsymbol{y}$
control action $\boldsymbol{u}$
k
$k+N_1$
$k+N_u$
$k+N_2$
control horizon
prediction horizon

Figure 2.5: Predict horizon and control signal base on [14]

2.2.2 Plant Modeling

Establish a mathematical model of the system to be controlled. This model can be linear or nonlinear, typically described by state-space equations or transfer functions. The predictive horizon refers to the future time period over which the system's behavior is predicted and optimized. The model is used to forecast the future system status within this predictive horizon.(x is the system state vector, which includes multiple states,$x = [x_1; x_2; x_3...x_N]$ u is input control variable, y is out put)

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) \\ y_{k+1} &= h(x_k, u_k) \end{aligned} \tag{2.4}$$

2.2.3 Constraint

In Model Predictive Control, constraints are limitations imposed on the system's state variables, control inputs, and output variables to ensure that the system operates within acceptable physical, operational, and safety boundaries. These constraints can include physical limits (e.g., maximum motor speed), operational limits (e.g., reactor temperature range), safety limits (e.g., pressure upper limit), and environmental limits (e.g., emission standards). Constraints are categorized into hard constraints (eq 2.5), which must be strictly adhered to, and soft constraints (eq2.6), which allow for some degree of violation but impose penalties in the objective function though slack variable. By incorporating these constraints into the optimization problem, MPC can generate optimal control strategies that meet both control objectives and practical operational requirements.

$$\begin{gathered} x_{lb} < x_k < x_{ub} \\ u_{lb} < u_k < u_{ub} \end{gathered} \tag{2.5}$$

$$\begin{gathered} x_{lb} - \epsilon_k < x_k < x_{ub} + \epsilon_k \\ u_{lb} - \epsilon_k < u_k < u_{ub} + \epsilon_k \\ \epsilon_k > 0 \end{gathered} \tag{2.6}$$

2.2.4 Cost Function

Also called the objective function, the cost function is used to quantify control objectives in order to optimize control outputs. The optimizer minimizes this function by optimizing the optimized vector, usually control variables. w is weight factor, it is used to balance the importance of different cost terms.

$$\begin{aligned} \min \quad & \sum_{k=1}^{N} w_1 f_1(x_k, u_k) + w_2 f_2(x_k, u_k) + ... \\ \text{s.t.} \quad & \text{constraint (1)} \\ & \text{constraint (2)} \\ & \text{constraint (3)} \end{aligned} \tag{2.7}$$

2.2.5 Format, Solver and Toolbox

Format

Model Predictive Control includes many different formats, such as Nonlinear Programming (NLP), Quadratic Programming (QP), and Distributed MPC. Different formats are well-suited for different fields.

Nonlinear Programming: Used for nonlinear systems where the system model and/or constraints are nonlinear. It typically requires nonlinear optimization algorithms. While the solution is more precise, the high computational complexity makes real-time use challenging

Quadratic Programming: Refers to the optimization problem where the cost function is quadratic and the constraints are linear. This format is computationally efficient, making it suitable for real-time applications.

Distributed MPC : Suitable for large-scale systems, breaking the entire system into multiple subsystems, each independently computing control strategies while coordinating to achieve a global optimization goal. The mathematical form involves local optimization problems for each subsystem, with coordination constraints and information sharing to achieve overall optimization .

Tool box

To solve the MPC problem, a toolbox should be used. There are many open-source toolboxes available, such as CasADi and ACADOS.

CasADi[10] is an open-source software tool for symbolic computation, optimization, and automatic differentiation. It excels in handling complex mathematical models involving dynamic systems and nonlinear optimization problems. CasADi provides an efficient way to compute, allowing for high-performance differentiation and optimization operations through its symbolic expressions. The tool is widely used in fields such as automatic control, robotics, aerospace, and energy, helping users solve large-scale and complex optimization problems.

ACADOS[15] is an open-source software framework focused on solving nonlinear optimization and model predictive control (MPC) problems. It leverages efficient numerical algorithms and automatic differentiation techniques to achieve fast solutions for large-scale optimization problems. acados supports multiple back end solvers and offers flexible interfaces, making it suitable for embedded systems and real-time control applications. Its design aims to deliver high-performance, low-latency control solutions, and it is widely applied in areas such as autonomous driving, industrial automation, and robotics.

Solver

Solvers can be invoked by the above toolbox, the most common solver include

- IPOPT [16] (Interior Point Optimizer) is an efficient nonlinear optimization solver based on the interior point method, capable of handling large-scale optimization problems with nonlinear constraints. It is widely used in scientific research, engineering design, and economics, supporting various programming languages such as C++, Python, and MATLAB. IPOPT is open-source, flexible, and extensible, allowing users to customize objective and constraint functions to meet specific needs.

- qpOASES [17] is an open-source C++ implementation of the online active set strategy for solving quadratic programming (QP) problems. It is designed to handle real-time applications efficiently, making it suitable for model predictive control (MPC) and other embedded optimization tasks. qpOASES excels in scenarios requiring rapid re-solving of QP problems with slightly varying parameters, leveraging warm-start capabilities to enhance performance. Its ease of integration, robustness, and real-time optimization capabilities make qpOASES a popular choice in control systems and other engineering applications.

- QRQP [10] (Quadratic Regularization Quadratic Programming) is an optimization solver designed for efficiently solving quadratic programming problems. It employs quadratic regularization techniques to ensure robustness and convergence, making it well-suited for large-scale and sparse problems. QRQP is particularly effective in handling ill-conditioned and degenerate cases, offering stability and reliability in various applications such as machine learning, finance, and engineering optimization tasks. Its implementation focuses on achieving high performance and accuracy, making QRQP a valuable tool for complex quadratic optimization challenges.

2.2.6 Simple Example

The MPC is explained by a simple ACC example. The ego vehicle is keeping a constant headway distance (L) to the lead vehicle. In the following system, the control variable is vehicle acceleration (a), the state are vehicle distance (s) and speed (v), the reference is lead vehicle position (s_{lead}) in predict time horizon, t_s is iteration time. In the cost function, punish the spacing error, acceleration and also build a soft constraint, use the slack variable ϵ to additionally punish the acceleration higher than 2 m/s^2 and lower than -2 m/s^2.

The state equation is represented as:

$$\begin{aligned} s_{k+1} &= s_k + v_k t_s \\ v_{k+1} &= v_k + a_k t_s \end{aligned} \tag{2.8}$$

Then, the MPC formulation can be written as :

$$\begin{aligned} &\min \textstyle\sum_{k=1}^{N} w_L(s_{lead,k} - s_k - L)^2 + w_a a^2 + w_\epsilon \epsilon_k \\ &\text{s.t.} \\ &-2 - \epsilon_k < a_k < 2 + \epsilon_k \\ &s_k < s_{lead,k} \\ &\epsilon_k > 0 \end{aligned} \tag{2.9}$$

2.2.7 Advantage and Disadvantage

MPC excels in handling complex multi-variable systems, managing multiple input and output variables simultaneously, and explicitly dealing with physical and operational constraints. By utilizing rolling optimization to predict future system behavior and optimize control inputs, it enhances system performance and stability. Additionally, MPC's framework is highly flexible, suitable for various types of systems and objective functions, and employs feed forward control to proactively address disturbances, thereby improving system robustness and response speed.

However, MPC also has some drawbacks. Firstly, it requires solving an optimization problem at each sampling period, resulting in a heavy computational burden, especially for large-scale or complex systems. Secondly, its performance is highly dependent on the accuracy of the system model; inaccuracies in the model can lead to degraded control effectiveness. The tuning of MPC parameters is also complex, necessitating extensive experience and experimentation. Implementing MPC is more complex than traditional control strategies, leading to higher development and maintenance costs, and it imposes high real-time requirements on hardware and software.

2.3 Driving Comfort

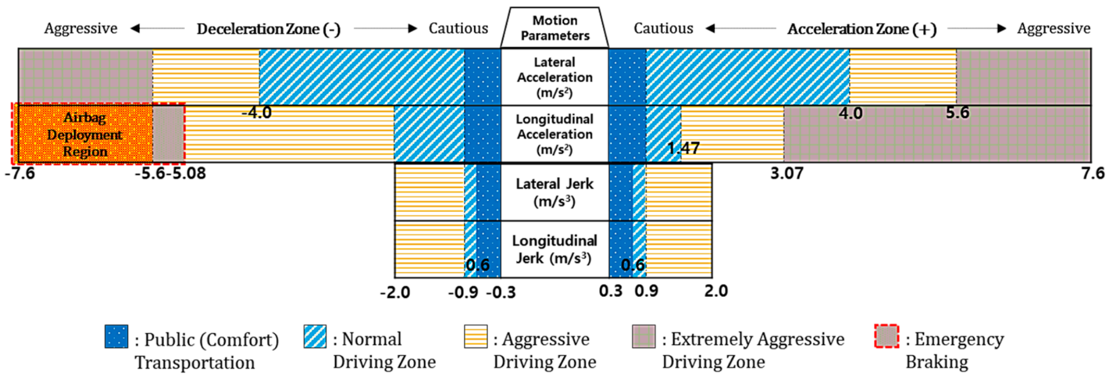


Figure 2.6: Driving comfort standard relative to acceleration and jerk base on [18]

Driving comfort is a very important KPI. Papers [18] and [19] provide detailed explanations of the indicators that affect driving comfort. Figure 2.6 shows the standard.

For lateral acceleration, the range of $\pm 0.9\ m/s^2$ is the most comfortable operating range, while the range of $\pm 4\ m/s^2$ is considered normal driving and does not cause discomfort. Sustained lateral acceleration outside this range may cause discomfort.

For longitudinal acceleration, the range is not symmetric as it is for lateral acceleration. The normal driving zone is from -2 m/s^2 to 1.47 m/s^2.

For jerk, the longitudinal and lateral zones are the same and symmetric. The best zone is within ± 0.9 m/s^3, while values exceeding ± 2 m/s^3 are considered very aggressive.

2.4 SPaT and Actuated Signal

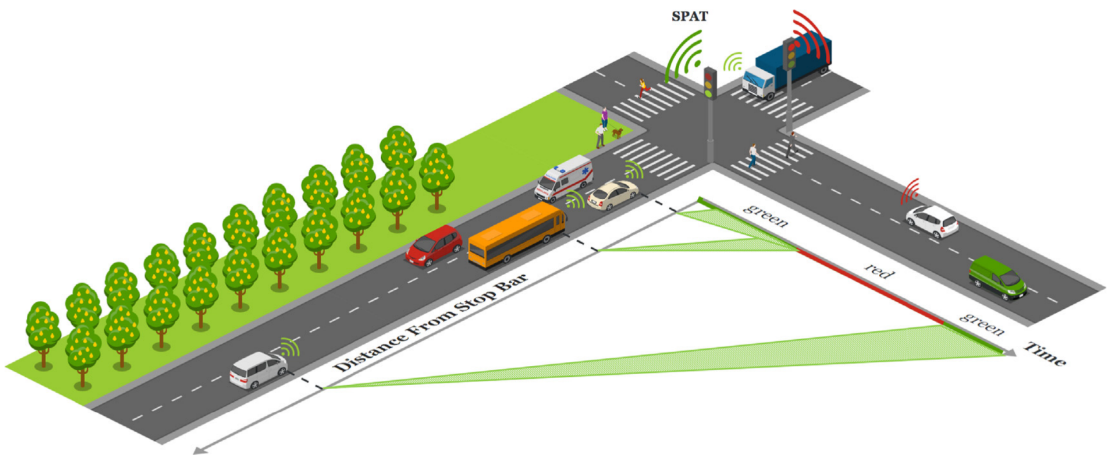


Figure 2.7: SPaT signal description base on [20]

2.4.1 SPaT

Signal Phase and Timing (SPaT)[21][20] is a signal that includes the traffic light phase and countdown timer. It can be broadcast from road site unit to approaching vehicles, so that connected vehicles adjust their speed for a timely arrival at a green light as shown in figure 2.7, allowing them to avoid unnecessary acceleration and stop. Which can improve driving experience and save energy.

2.4.2 Actuated Signal

Actuated Signal is an extension of SPaT, which uses sensors and cameras to monitor real-time traffic flow of vehicles and pedestrians. Based on this data, the system dynamically adjusts the signal phases and timing. The V2I signal is no more a fix countdown timer, but a estimated upper bound countdown timer and a lower bound countdown timer, they can be adjusted at any time base on traffic flow.

Chapter 3

System Architecture Design and Methodology

This chapter focuses on the overall design of our MPC-based ACC controller.

- **The first section**, introduce the overall structure of the controller. The expandability of the system is fully considered, and all parts of the controller are modularly designed to easily upgrade and replace the functions of the controller in future work.

- **From the second to fourth sections**, went into detail the design logic and formula of each sub-block of the controller.

- **The fifth section**, introduce the test environment setup and the controller implementation in Simulink.

3.1 Overall structure

Figure 3.1 shows the overall structure of the controller. The controller operates in a closed loop, with the output being the signals for electric motor torque and mechanical braking pressure for each wheel. The electric motor torque signal can either be positive or negative. Positive torque represents normal driving torque, while negative torque represents regenerative braking.

The MPC controller is implemented by two architecture. The first one is based on Nonlinear Programming (NLP). It uses a high-accuracy model; however, due to computational complexity, the calculation speed is very slow, making it unsuitable for real-time use. it is build as a reference to prove the feasibility of the method.

The second one is based on Quadratic Programming (QP). it is transformed from the NLP structure by simplify the model to achieve real-time usability.

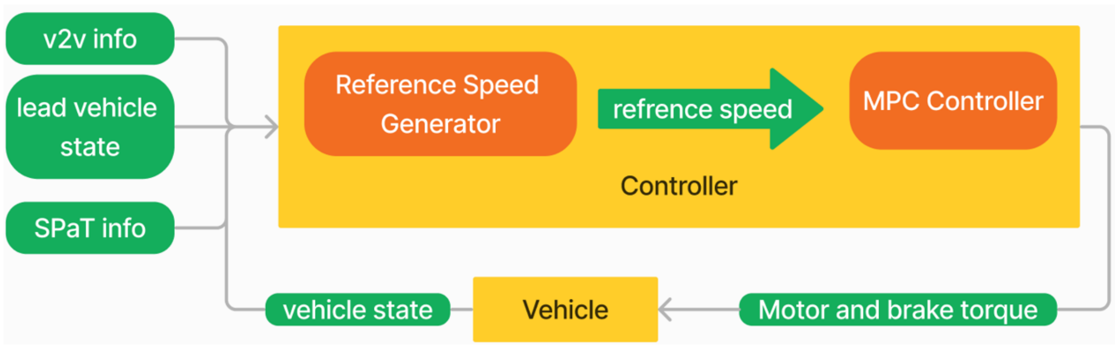


Figure 3.1: Overall structure of the system

The inputs available for implementation are:

- Ego vehicle state, it including the ego vehicle position, current speed, and acceleration.

- SPaT information: This is the information of traffic light, called Signal Phasing and Timing (SPaT), it include the traffic light phase and countdown timer (the time which the traffic light will change the phase) The controller can also be used with actuated signals, which include traffic light phases and the maximum/minimum counter to change the phase.

- V2V information: This includes the lead vehicle's planned speed in the prediction horizon of the MPC control. such information can help the ego vehicle complete more prise tracking. If such information not available, it has to be estimated.

The controller include :

- Reference Speed Generator: This module generates the tracking reference speed according to the lead vehicle and SPaT information and provides it to the MPC controller.

- MPC Controller: This is the main core of the controller. It uses the lead vehicle state, ego vehicle state, and the reference speed generated by the Reference Speed Generator to compute the total required torque at the motor level. Then provide it the the low-level control.

- low-level control: it is used to separate the total required torque into motor torque and brake pressure in each wheel.

3.2 Reference speed generator

The reference speed generator is used to provide the desired speed profile to be tracked by the MPC in the prediction horizon. Figure 3.2 shows the logic of reference speed generator, It include three different working mode with different priority. It first generates the profile according to SPaT information. then, if the SPaT information does not exist, it then tracks the lead vehicle. If both SPaT information and the lead vehicle are absent, it generate the speed profile let the vehicle accelerates to the maximum speed of road limitation. Additionally, there is the possibility to change the priority between vehicle following and SPaT logic to better follow the lead vehicle and avoid other vehicles cutting in.

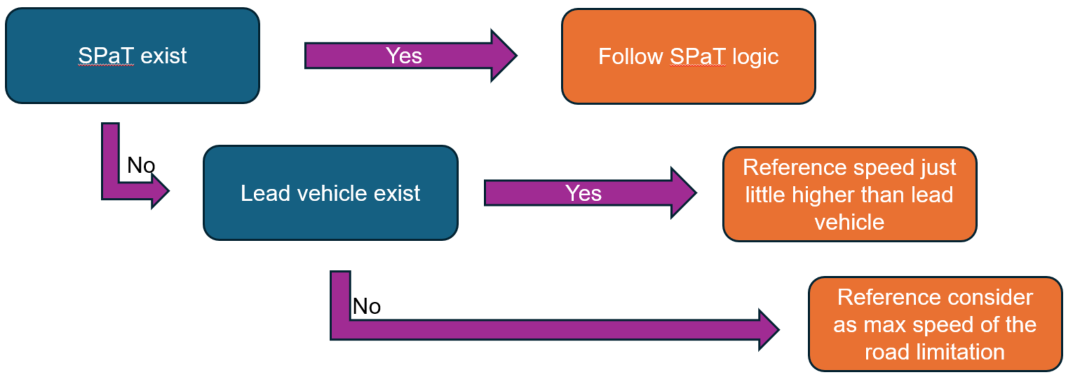

SPaT exist
Yes
Follow SPaT logic
No
Lead vehicle exist
Yes
Reference speed just little higher than lead vehicle
No
Reference consider as max speed of the road limitation

Figure 3.2: Reference speed generator logic and priority

The inputs available for implementation are:

- ego vehicle current speed v_c
- maximum speed of road limitation v_m
- lead vehicle position s_{lead}
- estimated lead vehicle speed in predict horizon v_{lead}
- SPaT signal, include the traddic light phase, upper and lower countdown timer

3.2.1 SPaT logic

Design approach

The primary task of the reference speed generator is to make decisions based on SPaT information. Extensive research has been conducted on energy-efficient vehicle speed profiles. It has been demonstrated in [22] and [23] that for a given travel time and distance (such as passing a traffic light), the most energy-efficient speed profile involves accelerating or decelerating as quickly as possible and maintaining a constant speed in between. In the paper by Peng Hao and Guoyuan Wu, "Developing a Framework of Eco-Approach and Departure Application for Actuated Signal Control" [24], they designed the Eco-Approach and Departure (EAD) application based on a rule-based system to generate speed reminders. We adopted this concept; however, the previous method used a piecewise sinusoidal function to generate the speed profile to avoid high jerk and maintain comfort. This approach is not suitable for MPC control. Therefore, we restructured the rule-based framework and modified the speed profile formula to integrate it into the MPC control. The comfort requirements will be realized by the MPC controller.

Figure 3.3 shows the SPaT reference speed generated logic. table 3.1 shows the symbol meaning

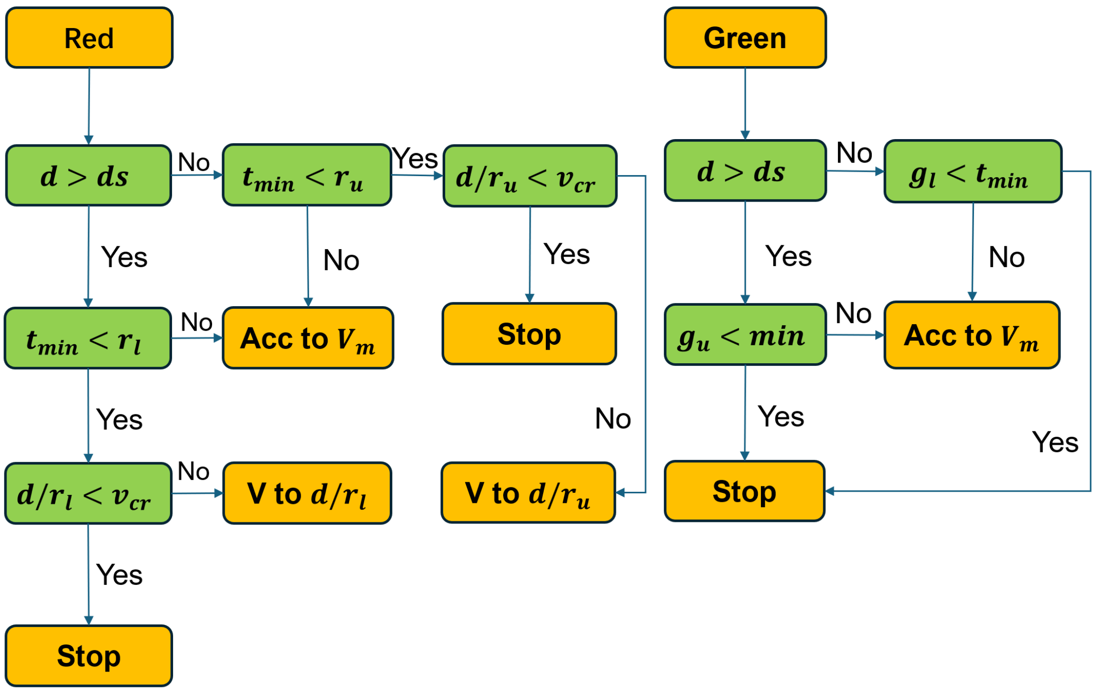


Figure 3.3: SPaT decision-making of traffic lights negotiation

Table 3.1: Parameter symbol and unit of reference speed generator

parameter	symbol	**unit**
distance between vehicle and traffic light	d	m
vehicle current speed	v_c	m/s
maximum speed by road limitation	v_m	m/s
critical speed	v_{cr}	m/s
minimum time needed to pass traffic light	t_{min}	s
maximum comfortable acceleration	a_{max}	m$/s^2$
upper bound SPaT red phase countdown timer	R_u	s
lower bound SPaT red phase countdown timer	R_l	s
upper bound SPaT green phase countdown timer	G_u	s
lower bound SPaT green phase countdown timer	G_l	s

The design target is to avoid stopping, reduce unnecessary acceleration and deceleration, and pass road sections as quickly as possible. The vehicle will accelerate or decelerate to the target speed using a comfortable acceleration and then maintain a constant cruising speed. If the current phase is red, the vehicle will try to pass the traffic light at the beginning of the green phase. However, this may lead the vehicle to maintain a low constant speed, which can affect traffic. Therefore, if the target cruising speed is lower than a threshold (v_{cr}), the vehicle will stop before the red phase. If the current phase is green, the vehicle will try to pass the traffic light at high speed to save travel time. If the remaining time is not sufficient, the vehicle will stop. Even if the ego vehicle has to stop due to resistance from other vehicles, this speed profile still benefits by reducing the total resistance. The power of resistance is proportional to the cube of the vehicle's speed. Early speed reduction can minimize high-speed driving, thereby decreasing total resistance power. More dynamic energy can be regenerated by the electric motor.

Mathematical formulation

t_{min} is the minimum time needed to pass the traffic light if the vehicle accelerates to the maximum speed of the road limitation. a_{max} is the maximum acceleration allowed by the controller, we choose this value equal to 1.47 refer to the content in chapter 2.

$$t_{min} = \begin{cases} \frac{(d+0.5\cdot(v_m-v_c)^2}{v_m a_{max}} & \text{if } d > \frac{v_m^2-v_c^2}{2a_{max}} \\ \frac{-v_c+\sqrt{v_c^2+2a_{max}\cdot d}}{v_m} & \text{if } d < \frac{v_m^2-v_c^2}{2a_{max}} \end{cases} \tag{3.1}$$

In order to make sure ego vehicle will not break traffic regulations and make sure safety, the redundancy of the countdown timer is adopted ($t_b = 1s$). For red countdown timer, add t_b and for green countdown timer, reduce t_b .The non-capital letter is the value after redundancy addition.

$$\begin{aligned} r_u &= R_u + t_b \\ r_l &= R_l + t_b \\ g_u &= G_u - t_b \\ g_l &= G_l - t_b \end{aligned} \tag{3.2}$$

Safety distance refers to the distance required for a vehicle to stop comfortably. If the vehicle is far from the traffic light, meaning the distance to the traffic light is greater than the safety distance, a more aggressive strategy is employed. The reference speed is generated based on the lower bound red phase countdown timer and the upper bound green phase countdown timer. Conversely, if the vehicle is closer to the traffic light, a opposite strategy is used.

$$d_{safe} = max(\frac{V_c^2}{2a_{max}},20) \tag{3.3}$$

The equations (3.4)-(3.6) present the formulas for speed profile generation.
Case 1 Acceleration to maximum speed (acc to v_{max}):

$$v_{ref} = \begin{cases} v_c + a_{max}t & \text{if } t > \frac{v_m - v_c}{a_{max}} \\ v_m & \text{if } t > \frac{v_m - v_c}{a_{max}} \end{cases} \tag{3.4}$$

Case 2 Tracing a specific speed (v to $v_h = \frac{d}{t_{tl}}$):
if the traffic light can not turn green when the ego vehicle reach it by accelerate to maximum speed of road limitation, ego vehicle has to tracing a specific speed in order to pass the traffic light when the phase just change into green with a specific cruising speed to avoid stop

$$v_h = \frac{dis2tl}{t_{tl}}$$
$$v_{ref} = \begin{cases} v_c + a_{max}t \cdot sign(v_h) & \text{if } t < \frac{abs(v_h - v_c)}{a_{max}} \\ v_h & \text{if } t > \frac{abs(v_h - v_c)}{a_{max}} \end{cases} \tag{3.5}$$

Case 3 Stopping before the traffic light (Stop):

$$v_{ref} = \begin{cases} v_c - \frac{v_c^2}{2d \cdot t} & \text{if } t < \frac{v_c^2}{2d \cdot t} \\ 0 & \text{if } t > \frac{v_c^2}{2d \cdot t} \end{cases} \tag{3.6}$$

3.2.2 Vehicle following

When a vehicle in front is detected, the reference speed is generated based on the prediction of the lead vehicle's speed, obtained either from V2V communication or by estimation. Since the SPaT logic and vehicle-following logic share the same weight factor, the speed-tracking error should remain similar to achieve good performance while maintaining the desired headway distance. In SPaT logic, the speed tracking error is low in the early steps of the prediction because acceleration and deceleration are smooth. The reference speed in the vehicle-following logic is set to be just 1 m/s higher than the lead vehicle's speed in the prediction horizon. However, this approach can result in longer times for the ego vehicle to reach the desired headway distance if the current relative distance is too large. Therefore, if the headway distance h exceeds 1.1 times the desired value h_{des}, the reference speed will be increased accordingly.

$$v_{ref} = \begin{cases} v_{lead} + 1 & \text{if } h < 1.1h_{des} \\ 1.2v_{lead} + 1 & \text{if } h > 1.1h_{des} \end{cases} \tag{3.7}$$

3.2.3 No SPaT and no vehicle following

If both lead vehicle and SPaT do not exist, ego vehicle will just travel in the highest speed to save the travel time.

$$v_{ref} = \begin{cases} v_c + a_{max}t & \text{if } t > \frac{v_m - v_c}{a_{max}} \\ v_m & \text{if } t > \frac{v_m - v_c}{a_{max}} \end{cases} \tag{3.8}$$

How to judge if a lead vehicle exists is a problem. Normally, this can be determined by whether a vehicle is detected by the sensor. However, modern radar detection ranges can exceed 300 meters, which is too far for initiating vehicle following. In this paper, we suggest using a threshold. When the lead vehicle is farther than this threshold, we consider the lead vehicle as non-existent. Conversely, if the lead vehicle is within this threshold, we consider it to exist. This threshold is determined by the distance at which the ego vehicle can comfortably stop from the maximum speed of the road limitation.($a_{hard} = -2m/s^2$)

$$d_{threshold} = -\frac{V_m^2}{2a_{hard}} \tag{3.9}$$

3.3 MPC Formulation with NLP

In this section, we introduce the Nonlinear Programming MPC controller. We choose a sampling time of 0.3 seconds and a prediction horizon of 20 steps, resulting in a prediction time of 6 seconds. The controller is written using the CasADi toolbox in MATLAB. CasADi is an efficient symbolic framework for solving nonlinear optimization problems.

The inputs available for implementation are:

- ego vehicle initial state
- lead vehicle trajectory
- reference speed
- traffic light distance base limitation

The output available for implementation are:

- control total torque in motor level

3.3.1 Preliminary knowledge

This section introduces some preliminary knowledge of the CasADi toolbox for solving NLP problems.

NLP in CasADi

In CasADi, the NLP is written as the following form.

$$\begin{aligned} &\min f(z,p) \\ &\text{s.t.} \\ &lbz < z < ubz \\ &lbg < G(z,p) < ubg \end{aligned} \tag{3.10}$$

Where z is the optimized vector, p is input parameter, $f(z,p)$ is the cost function, lbz and ubz are the lower and upper limitation of optimized vector. $g(z,p)$ is constraint. lbg and ubg are the lower and upper limitation of constraint.

Data format

The MPC state and control variables are implemented though MX data format of CasADi. MX allows for elementary operations that are not restricted to scalar unary or binary operations. The elementary operations used to form MX expressions can involve general multiple sparse-matrix valued input and multiple sparse-matrix valued output functions. Consequently, MX can be more economical when working with operations that are naturally vector or matrix-valued with many elements.n_x and n_u are the number of state and control variable, they are defined by $n \times 1$ vector.

$$\begin{aligned} x &= \mathrm{MX.sym}('x', n_x, 1) \\ u &= \mathrm{MX.sym}('u', n_u, 1) \end{aligned} \tag{3.11}$$

Look up table

The MPC formulation is written in the time domain; however, some variables are not in the time domain. For example, road slope, road curvature, and the judgment of SPaT existence are in the distance domain, while maximum electric motor torque and ISO limitations of acceleration and jerk are in the speed domain. In other words, the main variables such as state and control variables are written as $x(t)$ and $u(t)$ (t is time). Other variables, such as road angle, are written as $\theta(s)$ (s is vehicle travel distance, also representing current position). Maximum electric motor torque is written as $T_{max}(v)$ (v is vehicle speed). These variables depend on travel distance or speed, not on time.

In order to use these variable, the lookup table of CasADi should be use. $d_1 ... d_M$ are look up table domain, $Y_1 ... Y_M$ are look up table output, x is input, y is output

$$\begin{gathered} \text{interp\_function} = \text{casadi.interpolant('LUT','linear',}\{[d_1, ..., d_M]\}, [Y_1, ..., Y_M]) \\ y = \text{interp\_function}(x) \end{gathered} \tag{3.12}$$

Build solver

CasADi makes an important distinction between initialisation and evaluation steps, Only construct the solver once, and then evaluate it repeatedly using slightly different numerical inputs. As shown in equation 3.13, The solver should be constructed by optimized vector z, cost function f and constraint formula g, So these content should not change in every repeatedly using. Equation 3.14, the input value at each repeatedly using are ubz, lbz, ubz, lbz (upper and lower boundary of state and constraint), also $z0$ is the iterative starting point, it should be set as close as possible to the final result, a good estimation of iterative starting point can reduce the solving time however it is not mandatory.

$$\begin{gathered} \text{nlp = struct('x',z,'f',J,'g',g);} \\ \text{NLPsolver = nlpsol('solver','ipopt',nlp,opts)} \end{gathered} \tag{3.13}$$

$$\text{result = NLPsolver('x0',z0,'ubx',ubz,'lbx',lbz,'lbg', lbg, 'ubg', ubg);} \tag{3.14}$$

Single shooting method (dense formulation)

In CasADi, the optimized vector can use both single shooting method and multiple shooting method, we firstly introduce the single shooting method in this section. For single shooting, all the state are be written as a function of control variable and inital state, idealy the optimized vector is only include control variable. Equation 3.15 show the logic. every term in Nth sampling time can written as a function of previous $(N-1)th$ sampling time, by parity of reasoning.

$$\begin{aligned}
x_1 &= x_0 + F(x_0, u_0) \\
x_2 &= x_1 + F(x_1, u_1) \\
&= \overbrace{x_0 + F(x_0, u_0)}^{x_1} + F(\overbrace{x_0 + F(x_0, u_0)}^{x_1}, u_1) \\
&\vdots \\
x_k &= x_k(x_0, u_0, u_1, \ldots, u_{k-1}) \quad \forall k \in \{1, 2, \ldots, N\}. \\
Z &= \Big[u_0, u_1, \ldots, u_{N-1}\Big]^\top
\end{aligned} \tag{3.15}$$

3.3.2 Lead vehicle trajectory

Because MPC works based on future predictions, it need the future trajectory of the lead vehicle within the prediction horizon. This information can be obtained from the lead vehicle using V2V technology, such as when the lead vehicle also uses an MPC-based ACC system or has other speed reminders. However, in many real-world scenarios, V2V communication might be unavailable due to technical limitations, lack of infrastructure, or non-cooperative lead vehicles. In such cases, the ego vehicle must rely on estimation methods to predict the future trajectory of the lead vehicle. Two types of estimation are considered: lead vehicle is driving in constant speed or constant acceleration, where t_s is iteration time. k is the number of step in prediction horizon.

- **v2v**: the lead vehicle trajectory can be precise
- **constant speed**: hypothesize that the lead vehicle will maintain a constant speed within the prediction horizon.

$$s_{lead,k} = s_{lead,0} + kt_s v_{lead,0} \tag{3.16}$$

- **constant acceleration**: hypothesize that the lead vehicle will maintain a constant acceleration within the prediction horizon.

$$s_{lead,k} = s_{lead,0} + kt_s v_{lead,0} + \frac{1}{2} a_{lead,0} (kt_s)^2 \tag{3.17}$$

3.3.3 Optimized vector

The NLP-based MPC controller is implemented using the single shooting method. The optimized vector includes only the control variable and slack variable. The initial state (s_0, v_0) should also be included in the optimized vector to iterate. Theoretically, the initial state can be written as a parameter and not included in the optimized vector; however, in CasADi, the constraints do not allow free variables. The specification shows an option to allow free variables, but in actual operation, we did not find this option. Initial acceleration should also be included in order to penalize the initial jerk.

$$z = [x_0, a_0, u_0, \epsilon_0, u_1, \epsilon_1 ...., u_{N-1}, \epsilon_{N-1}] \tag{3.18}$$

3.3.4 Vehicle model

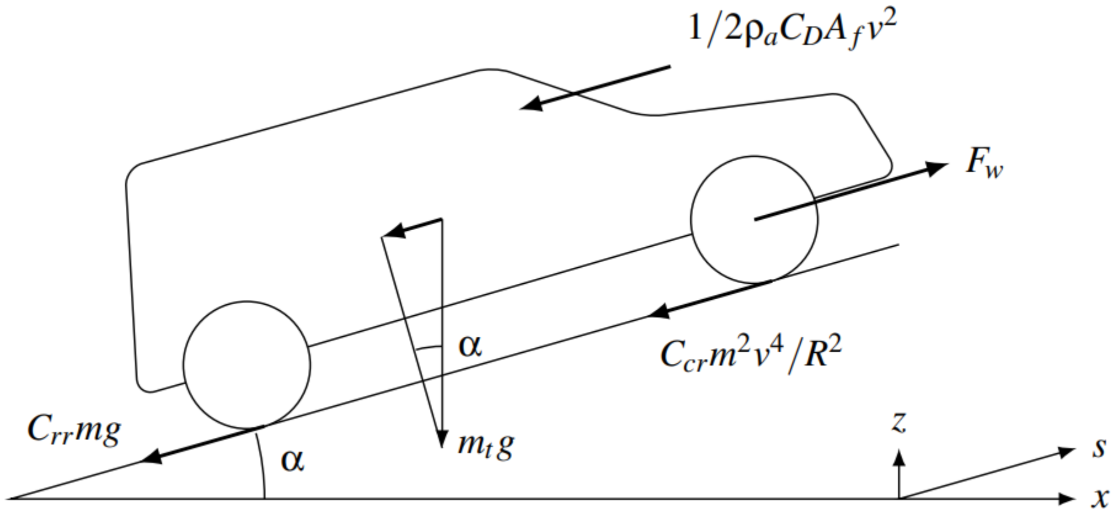


Figure 3.4: Vehicle longitudinal dynamic model base on [25]

In this paper, Only longitudinal driving is involved. Therefore, the internal model is a discretized simple longitudinal model. Table 3.2 shows the symbol meanings and units. Equation 3.19 shows the discretized function, where vehicle travel distance is iterated by speed and speed is iterated by acceleration. We use the total torque at the motor level. A positive value means the motor is providing traction, while a negative value means the vehicle is braking. The braking torque is composed of electric motor regenerative braking and mechanical braking. The mechanical braking torque is divided by the gearbox ratio and equivalently reflected in the motor torque. The content in the bracket of the second equation refers to acceleration. The first term is the traction/braking force at the wheel. The motor torque propagating to the wheel side should account for the gearbox ratio, gearbox efficiency, and wheel radius. The gearbox efficiency should be inverted for traction and regenerative braking. The second to fourth terms are resistance forces, including aerodynamic drag, gravitational forces, and rolling resistance.

$$\begin{aligned} s_{k+1} &= s_k + v_k t_s \\ v_{k+1} &= (\tfrac{1}{r} T_{mk} i_{gb} \eta_{gb}^{sign(T_{mk})} - \tfrac{1}{2}\rho C_x A_x V_k^2 - mgsin(\theta_k) - f_0 mgcos(\theta_k))\tfrac{1}{\lambda m} t_s + v_k \end{aligned} \tag{3.19}$$

Table 3.2: Parameter symbol and unit of NLP controller

parameter	symbol	**unit**
vehicle mass	m	kg
inertial coefficient	λ	-
vehicle front area	A_x	m^2
areo-dynamic coefficient	C_x	-
wheel radius	r_w	m
total torque at motor level	T_m	Nm
gear box efficiency	$i_g b$	-
air density	ρ	kg/m^3
gravitational acceleration	g	N/m^2
road angle	θ	rad
rolling resistance coefficient	f_0	-
sampling time	t_s	s
vehicle travel distance	s	m
vehicle speed	v	m/s
friction coefficient	μ	-
electric motor maximum power	P_{max}	w
electric motor maximum toque	T_{max}	Nm

3.3.5 Constraints

As shown in the preliminary knowledge, in CasADi, The soft constraints is realized by slack variables. The weight factor of the slack variable varies in order depending on the requirement. Some soft constraints can be obeyed simultaneously; the optimized result will be generated through competition. Soft constraints that should not be violated frequently will use a high-order weight factor. Other constraints are hard constraints that should never be violated. the constraints can be written as either optimized vector constraints or formulas of the optimized vector. Both of these include three parts: the formula/optimized vector (G/z), upper bound (*ubg*/*ubz*), and lower bound (*lbg*/*lbz*). The formula (G) is set during initialization and should not change with every repeated use, every term include the optimized vector should written in it. The upper bound and lower bound can be changed, all the input value should be written in it. So the equation written in casadi is not always the same as the normal ordinary logic. In each of the following sections, the formula is first introduced and written in ordinary logic form, then followed by its representation in CasADi using matrix notation.

Optimized vector constraint

The optimized vector includes the initial state as discussed before. In the optimized vector, constraints should be used to force it to be equal to the input initial state.

$$\begin{bmatrix} s_{in} \\ v_{in} \\ a_{in} \end{bmatrix} \leq \begin{bmatrix} s_0 \\ v_0 \\ a_0 \end{bmatrix} \leq \begin{bmatrix} s_{in} \\ v_{in} \\ a_{in} \end{bmatrix} \tag{3.20}$$

The others should be bounded in basic logic, The motor torque do not be constrain, all the slack variable should be positive.

$$\begin{bmatrix} -\infty \\ 0 \end{bmatrix} \leq \begin{bmatrix} F_{m,k} \\ \xi_{i,k} \end{bmatrix} \leq \begin{bmatrix} \infty \\ \infty \end{bmatrix} \tag{3.21}$$

Vehicle speed

The vehicle speed limitation should consider both the road speed limit and safety requirements, and it should always be positive. The road speed limitation depends on the vehicle's position and should be obtained through a lookup table. This speed limitation represents a hard constraint that must not be violated under any circumstances. This limitation defined by traffic regulations and road design parameters.

$$
\begin{gathered}
0 \leq v_k \leq v_{max}(s_k) \\
v_{max} = \text{casadi.interpolant('LUT','linear',\{[distance]\}, [speed limitation])} \\
\begin{bmatrix} -\infty \\ 0 \end{bmatrix} \leq \begin{bmatrix} v_k - v_{max}(s_k) \\ v_k \end{bmatrix} \leq \begin{bmatrix} 0 \\ \infty \end{bmatrix}
\end{gathered} \tag{3.22}
$$

The road curvature affects driving safety. The road curvature depends on the vehicle's position and should be obtained through a lookup table. To ensure safety, the vehicle must drive below a certain speed when drive in curved roads. This safe speed is determined by the road curvature. The equation 3.23 shows the function that limits the safe speed based on road curvature. This constraint of speed due to road curvature can be violated occasionally; it is a soft constraint but should be avoided as much as possible to maintain safety. To manage this, A slack variable is used with a high-value weight factor in the optimization process. The slack variable allows for some flexibility in the constraint, but the high-value weight factor penalizes deviations, ensuring that the vehicle adheres to the safe speed limit under normal conditions. This approach balances safety and flexibility, allowing the vehicle to adapt to unexpected situations while prioritizing safe driving on curved roads.

$$
\begin{gathered}
v_{safe} = 3.34 \cdot |\theta(s_k)|^{-\frac{1}{3}} \\
\theta(s_k) = \text{casadi.interpolant('LUT','linear',\{[distance]\}, [road curvature])}
\end{gathered} \tag{3.23}
$$

$$
\begin{gathered}
0 \leq v_k \leq v_{safe} \\
\begin{bmatrix} -\infty \end{bmatrix} \leq \begin{bmatrix} v_k - v_{safe}(s_k) \end{bmatrix} \leq \begin{bmatrix} 0 \end{bmatrix}
\end{gathered} \tag{3.24}
$$

Speed tracking error

The vehicle should track the reference speed. However, due to other constraints such as traffic lights, maintaining the desired headway distance, and ensuring comfort, the reference speed cannot always be strictly followed. When these constraints interfere, deviations from the reference speed will occur, and such offsets will be penalized to maintain optimal performance. This penalty is realized by introducing a slack variable. Both real speeds higher or lower than the reference should be penalized, The slack variable allows for flexibility in meeting these constraints while still aiming to follow the reference speed as closely as possible. By assigning a penalty to the slack variable, The trade-offs between different constraints can be managed.The weight factor associated with the slack variable determines the severity of the penalty, ensuring that significant deviations are minimized.

$$\begin{gathered} v_{ref,k} \leq v_k + \epsilon_{v,k} \\ v_{ref,k} \geq v_k - \epsilon_{v,k} \\ \begin{bmatrix} v_{ref,k} \\ -\infty \end{bmatrix} \leq \begin{bmatrix} v_k + \epsilon_{v,k} \\ v_k - \epsilon_{v,k} \end{bmatrix} \leq \begin{bmatrix} \infty \\ v_{ref,k} \end{bmatrix} \end{gathered} \tag{3.25}$$

Intrusion to lead vehicle desired headway distance

The ego vehicle should maintain the desired headway distance from the lead vehicle, in this section, the chosen policy is HDB (from eq 2.3). When the distance between the ego vehicle and the lead vehicle is less than the desired value, we impose a penalty on the intruded distance to encourage the ego vehicle to restore the desired headway distance. When the reference speed of the ego vehicle is higher than that of the lead vehicle, the ego vehicle will attempt to intrude into the desired headway distance. This intrusion triggers a distance penalty, which acts to decelerate the ego vehicle to maintain desired headway distance. At the same time, the distance penalty also prevents the ego vehicle from reaching its reference speed, leading to a speed offset penalty. These two types of penalties, distance and speed work against each other. The controller optimizes these penalties to find a balance, resulting in an overall optimal driving behavior. Conversely, when the reference speed of the ego vehicle is lower than that of the lead vehicle, the distance penalty is not triggered. In this scenario, the ego vehicle can successfully track the reference speed without the interference of a distance penalty. This allows the vehicle to achieve other objectives set by the reference speed generator, such as planning a more reasonable energy-saving speed profile to pass through traffic lights efficiently.

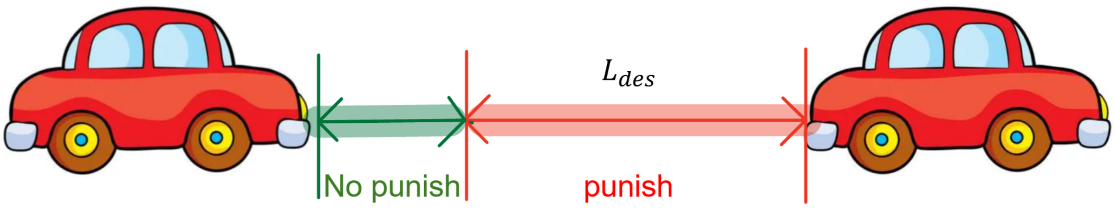


Figure 3.5: Punishment of intrusion to lead vehicle desired headway distance

$$\begin{gathered}\epsilon_{L,k} + s_{lead,k} \geq s_k + A + Tv_k + Gv_k^2 \\ \left[-\infty\right] \leq \left[s_k + A + Tv_k + Gv_k^2 - \epsilon_{L,k}\right] \leq \left[s_{lead,k}\right]\end{gathered} \tag{3.26}$$

Safety distance

The penalty for intruding into the lead vehicle's desired headway distance does not have a high weight factor; it is a soft constraint that can be violated. This does not ensure safety, and the risk of collision still exists. To mitigate this risk and avoid potential danger, another hard constraint is added to ensure that the distance between the lead vehicle and the ego vehicle never falls below one meter. This hard constraint acts as a safety measure to provide an additional layer of protection, ensuring that even if the soft constraint is violated, the vehicles maintain a minimum safe distance.

$$\begin{gathered}s_{lead,k} - s_k \geq 1 \\ \left[-\infty\right] \leq \left[1 + s_k\right] \leq \left[s_{lead,k}\right]\end{gathered} \tag{3.27}$$

Acceleration and jerk should remain within comfortable limits and ISO standard

The ISO standard regulates the maximum acceleration, deceleration, and jerk, which depend on the vehicle speed. These values are piece-wise speed-based functions that should be obtained from a lookup table. These constraints are hard constraints that must not be violated under any circumstances to ensure safety and compliance with regulatory standards. The limitation on jerk is symmetric.

$$\begin{aligned}a_{max}(v_k) &= \text{casadi.interpolant('LUT','linear',\{[0,5,20,25]\}, [5,5,3.5,3.5])} \\ a_{min}(v_k) &= \text{casadi.interpolant('LUT','linear',\{[0,5,20,25]\}, }[-4,-4,-2,-2]) \\ j_{lim}(v_k) &= \text{casadi.interpolant('LUT','linear',\{[0,5,20,25]\}, [5,5,3.5,3.5])}\end{aligned} \tag{3.28}$$

$$\begin{gathered}a_{min}(v_k) \leq a_k \leq a_{max}(v_k) \\ j_{min}(v_k) \leq j_k \leq j_{max}(v_k) \\ \begin{bmatrix}-\infty \\ -\infty \\ -\infty \\ -\infty\end{bmatrix} \leq \begin{bmatrix}a_k - a_{max}(v_k) \\ -a_k + a_{min}(v_k) \\ j_k - j_{lim}(v_k) \\ -j_k - j_{lim}(v_k)\end{bmatrix} \leq \begin{bmatrix}0 \\ 0 \\ 0 \\ 0\end{bmatrix}\end{gathered} \tag{3.29}$$

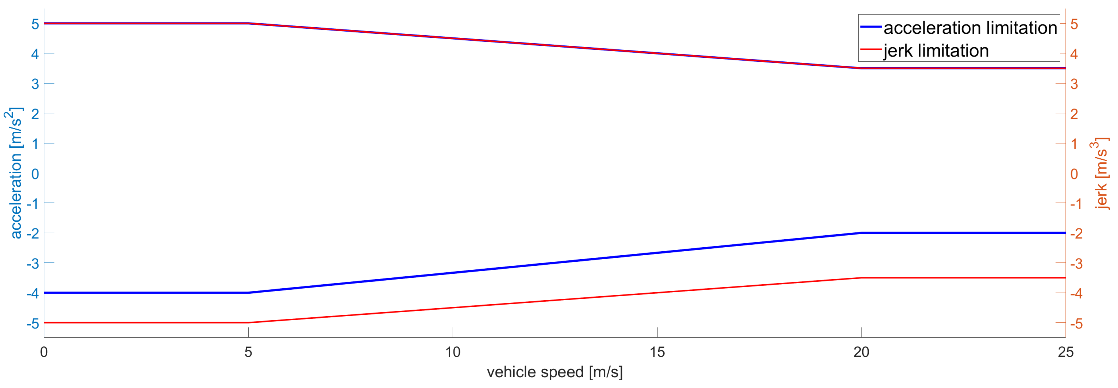


Figure 3.6: ISO standard for acceleration and jerk relative to comfort

The acceleration and jerk should always remain below certain thresholds to ensure comfort. However, in some emergency conditions, some discomfort should be allowed. As discussed in Chapter 2, the acceleration and jerk should always remain within ±2 m/s². This constraint is a soft constraint with a high weight factor, indicating that it should not be violated frequently.

$$\begin{gathered} -2-\epsilon_{s,k} \leq a_k \leq 2+\epsilon_{s,k} \\ -2-\epsilon_{s,k} \leq j_k \leq 2+\epsilon_{s,k} \\ \begin{bmatrix} -\infty \\ 0 \\ -\infty \\ 0 \end{bmatrix} \leq \begin{bmatrix} a_k - 2 - \epsilon_{s,k} \\ a_k + 2 + \epsilon_{s,k} \\ j_k - 2 - \epsilon_{s,k} \\ j_k + 2 + \epsilon_{s,k} \end{bmatrix} \leq \begin{bmatrix} 0 \\ \infty \\ 0 \\ \infty \end{bmatrix} \end{gathered} \tag{3.30}$$

Electric motor limitation

The electric motor has physical limitations; the traction torque and regenerative braking torque must be below a certain value. These values are piecewise functions depending on the motor rotation speed. Because our controller is designed for an electric vehicle that only has a fixed gear ratio, the motor speed is proportional to the vehicle speed. Therefore, it is a speed-based function that can be obtained from a lookup table. Equation 3.31 shows the function of the motor's maximum torque. Since our control variable is the total torque at the motor level, the traction torque will be bounded by the motor's limitations. However, the braking torque is a combination of regenerative braking and mechanical braking, which means it will not be bounded solely by the motor's limitations. Consequently, only the positive motor torque is constrained to ensure it stays within the motor's physical limits.

$$T_{m,max}(\omega) = \begin{cases} T_{max} & \text{if } \omega < \omega^* \\ \frac{P_{max}}{\omega} & \text{if } \omega >= \omega^* \end{cases} \tag{3.31}$$

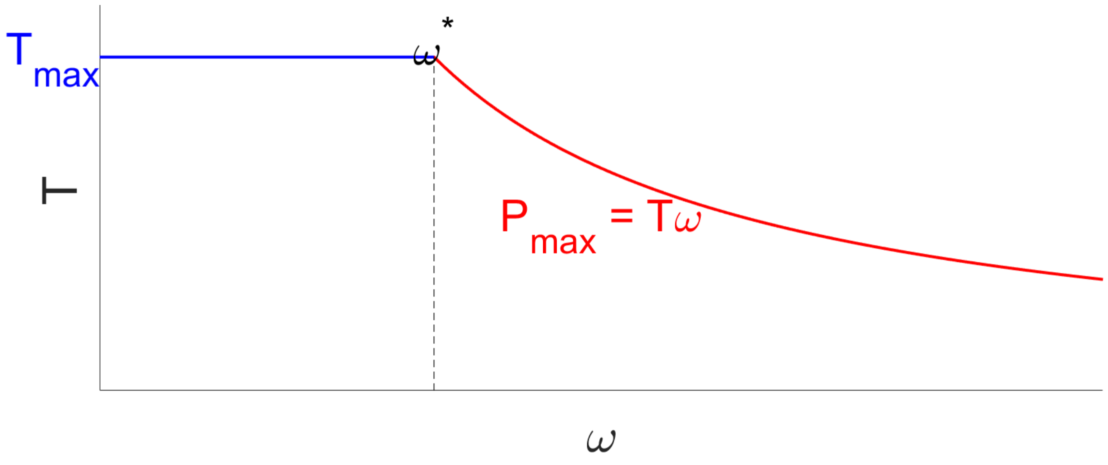


Figure 3.7: Motor torque physical limitation

$$\begin{gathered} T_{mk} \leq T_{m,max}(\omega_k) \\ T_{m,max}(v_k) = \text{casadi.interpolant('LUT','linear',}\{[\omega]\}, [T_m, max]) \\ \begin{bmatrix}-\infty\end{bmatrix} \leq \begin{bmatrix}T_m - T_{m,max}(\omega_k)\end{bmatrix} \leq \begin{bmatrix}0\end{bmatrix} \end{gathered} \tag{3.32}$$

Tire limitation

In this paper, the controller is designed for a rear-wheel-drive vehicle. The friction also limits the drive and braking forces. Excessive motor torque can cause significant slip, decreasing traction force and wasting energy. Therefore, the traction force must be limited. For braking force, mechanical braking can be used on all four wheels. The total friction limitation for both the front and rear wheels is considered

$$\begin{gathered} |T_{mk} i_{gb} \tfrac{1}{r_w} \eta_{gb}^{sign(T_{mk})}| \leq \mu mgcos(\theta(s_k)) \\ T_{mk} i_{gb} \tfrac{1}{r_w} \eta_{gb}^{sign(T_{mk})} \leq \tfrac{\mu_x}{L} [mg(L_f cos(\theta(s_k)) + h_g sin(\theta(s_k))) \\ + h_g(\tfrac{1}{2}\rho C_x A_f v^2 + \lambda am)] \\ \begin{bmatrix}0\\0\\0\end{bmatrix} \leq \begin{bmatrix} \mu mgcos(\theta(s_k)) - T_{mk} i_{gb} \eta_{gb}^{sin(T_{mk})} \frac{1}{r_w} \\ \mu mgcos(\theta(s_k)) + T_{mk} i_{gb} \eta_{gb}^{sin(T_{m,k})} \frac{1}{r_w} \\ \frac{\mu}{L}(L_f cos(\theta(s_k) + h_g sin(\theta(s_k)) + h_g(\frac{1}{2}\rho v_k^2 + \lambda ma) - T_{mk} \eta_{gb}^{sign(T_{mk})} \end{bmatrix} \leq \begin{bmatrix}\infty\\\infty\\\infty\end{bmatrix} \end{gathered} \tag{3.33}$$

Traffic light stop constraint

Even though a reference speed generator is used to create a speed profile that aims to stop the vehicle before the red phase of the traffic light, the jerk must still be limited to maintain comfort. As a result, the reference speed will not be strictly followed, which could lead to the vehicle stopping after the stop line—an unacceptable outcome. Therefore, a constraint ensuring the vehicle stops before the traffic light is necessary. Additionally, if the vehicle is to prioritize car-following behavior, such a constraint is the only way to ensure the vehicle stops when the traffic light turns red

Our overall logic is to use Boolean values to transform real-world problems into mathematical ones. The input value is the phase of the traffic light within the prediction horizon. The green phase is 1, red phase is 0, The future phase is estimated using the current phase and the lower bound of the countdown timer. When the countdown timer finishes, the phase will change. Algorithm 1 show the algorithm

Figure 3.8 shows the logic, We consider a distance-based constraint as a "wall" at the stop line. The position of this wall is obtained from a distance-based lookup table. Before reaching a specific traffic light, the value of this wall remains the same, representing the stop line position. If the vehicle passes the traffic light position in predict horizon, *pass* will equal to 1, otherwise, pass equal to 0. If the phase is green or the vehicle has already pass the traffic light in predict horizon, a high value equal to 10^7 will be added to the wall, effectively "pushing" the wall far away, making it as if the wall has disappeared. The ego vehicle position is estimated using the previous MPC solution, so it can appear in the boundary. We do not use the real position because it would cause numerical issues with the lookup table. Detailed discussion about this estimation will be provided in the next section about QP. After testing, this estimation method has been found to work perfectly without any other effect. the traffic stop limitation term is the upper bound of constraints which can be input as overall.

Algorithm 1 Phase in predict horizon

```
1: for each time step in predict horizon $1 : N$ do
2:     countdown timer = countdown timer - $t_s$
3:     if countdown timer $\leq 0$ then
4:         phase change, different from initial one
5:     else
6:         phase is the same as initial one
7:     end if
8: end for
```

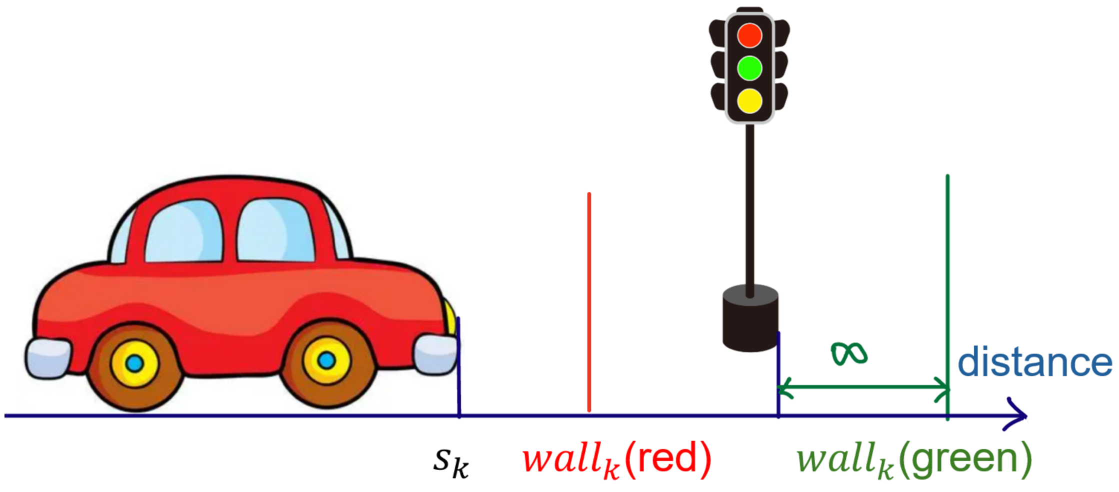


Figure 3.8: Constraint of traffic light schematic

$$\begin{gathered} s_k \leq wall_k(s_k) + phase \cdot 10^7 + pass \cdot 10^7 \\ \left[-\infty\right] \leq \left[s_k\right] \leq \left[wall_k(s_{k,est}) + phase \cdot 10^7 + pass \cdot 10^7\right] \end{gathered} \tag{3.34}$$

3.3.6 Cost function

Power consumption

One of our targets is to reduce power consumption, so including power consumption in our considerations is necessary. The total real power consumption is given by Equation 3.35. The battery power consumption is determined by motor speed, motor torque, and motor efficiency. Due to the opposite directions of energy flow during traction and braking, the motor efficiency should be inverted between these two conditions. Additionally, motor efficiency is not a constant; it depends on the operating point (motor torque and speed).

$$P_b = T_m \omega_m \eta(T_m, \omega_m)^{sign(T_m)} \tag{3.35}$$

Since MPC aims to minimize the cost function formed by a polynomial, to use motor efficiency in the cost function, It should be interpolated as a polynomial. Figure 3.9 first graph shows the efficiency map, due to the structure of the efficiency map being symmetric and having lows in the middle, it is not suitable for polynomial interpolation. Therefore, the power map is used instead, where the input is motor torque and speed, and the output is battery power shown in figure Figure 3.9 second map. The power map monotonically increases when either the torque or speed increases, making this behavior more suitable for polynomial interpolation. the efficiency is already integrated into the battery power map.

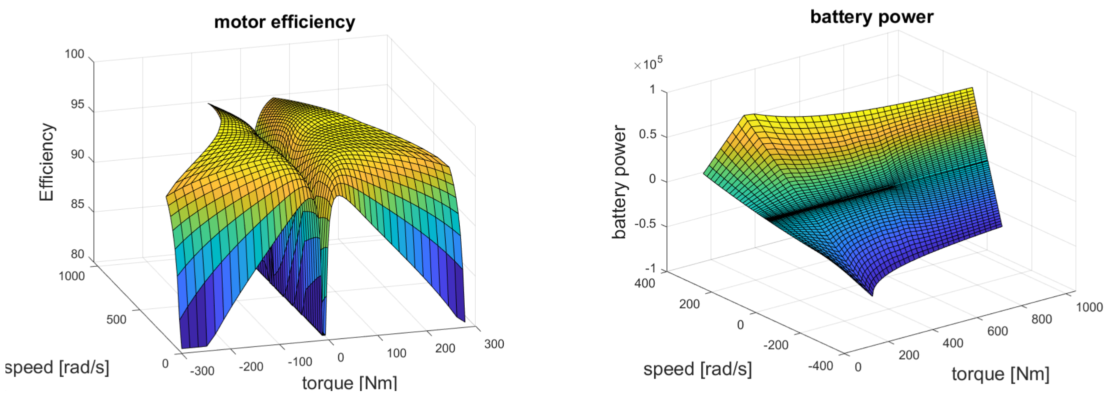


Figure 3.9: Electric motor efficiency map and battery power map

The points on the map are initially sampled, with speed divided into 100 nodes and torque into 200 nodes, creating 20,000 combinations of speed and torque. Subsequently, points exceeding the electric motor's capability are filtered out, resulting in 11,406 remaining points.

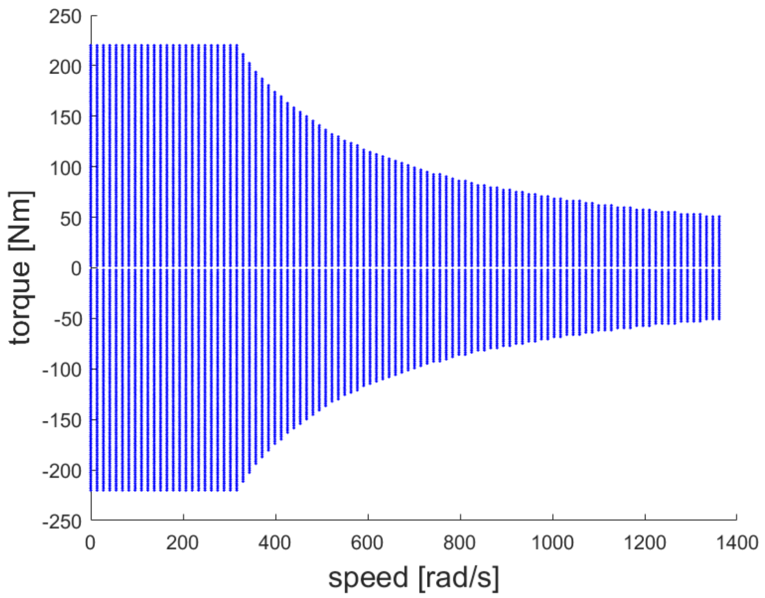


Figure 3.10: Sampling points

Then, a 5×5 polynomial is interpolated with the sampling points. Using another optimization problem created by CasADi to achieve this. For each sampling point, the interpolated battery power is calculated using the equation shown in Equation 3.36. The cost function is the sum of the offsets between the interpolated battery power and the real battery power across all the sampling points. Two constraints are used: 1) the power should increase when the torque increases, and 2) the efficiency should always be lower than zero. These two constraints help reduce local minima in the power map.

$$P_{bat} = b_{00} + b_{10}T_m + b_{01}\omega + ... + b_{50}T_m^5 + ... + b_{14}T_m\omega^4 + b_{05}\omega^5 \tag{3.36}$$

$$\begin{aligned} &\min \textstyle\sum_{k=0}^{N-1} P_{real} - P_{interpolated} \\ &\text{s.t.} \\ &\tfrac{dP}{dT} > 0 \\ &P_{interpolated} > T_m\omega_m \end{aligned} \tag{3.37}$$

Because the most important information on this map is motor efficiency, to prove the reliability, we use the interpolated battery power polynomial to calculate the motor efficiency. The results are shown in the figure: the left one represents the original efficiency map, while the right one is derived from the interpolated data. As observed, the tendencies are similar, indicating that the interpolated polynomial can accurately capture the efficiency characteristics of the motor. This validation step ensures that our interpolation method preserves the essential features of the motor's efficiency map. The resulting goodness of fit among the sampling points is 98.74 %

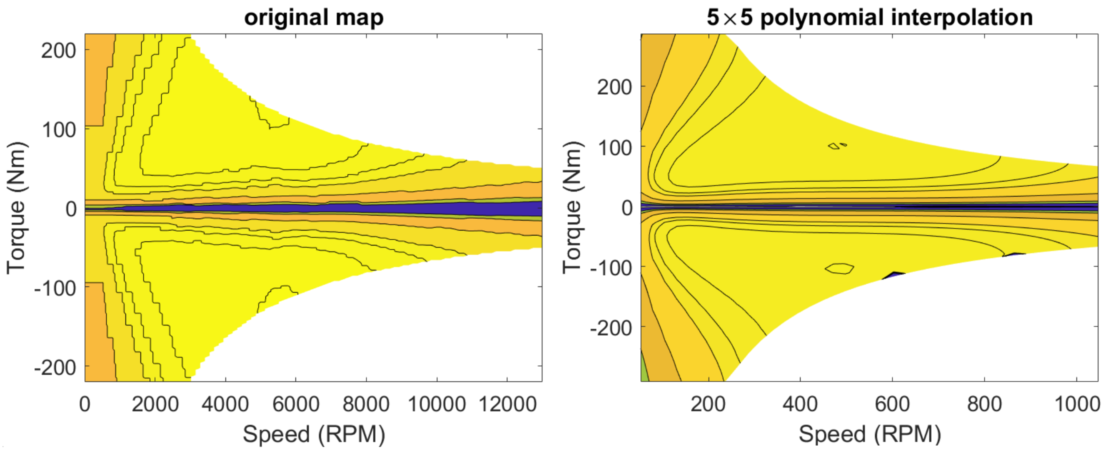


Figure 3.11: Validated original and interpolated efficiency map

Positive power consumption should be penalized to save energy, while negative power consumption, which indicates regenerative energy, should be encouraged. Therefore, the power consumption term in the cost function is added accordingly.

$$J_{p,k} = w_p P_{bat,k}(T_{m,k}, \omega_{m,k}) \tag{3.38}$$

Acceleration and jerk

Acceleration and jerk highly affect comfort. Even though the ISO standard regulates them, the limits can still be too radical. Therefore, it is reasonable to penalize both acceleration and jerk. Both positive and negative values of acceleration and jerk should be penalized, so we use their squares to simultaneously address positive and negative values. The penalties are divided into two sets: the first step penalizes acceleration below ± 2 m/s^2, and the second step addresses values over ± 2 m/s^2 using slack variables, as discussed in the constraints section.

$$J_{aj,k} = w_a a_k^2 + w_j j_k^2 \tag{3.39}$$

Slack variable

the slack variable include :

- speed tracking error ϵ_v
- intruding into the lead vehicle's desired headway distance ϵ_L
- exceeding safety and comfort limitations ϵ_s

The slack variable are always positive, so both linear of quadratic are reasonable. Figure 3.12 show the difference between linear and quadratic. Quadratic has lighter punishment in low offset and harder punishment in high offset. For speed and distance, they are punished by quadratic term because we always desire small deviations in these terms and want to avoid large deviations. For the safety and comfort limitation penalties, they are punished by linear term because these constraints should not be violated frequently, even with small overflows.

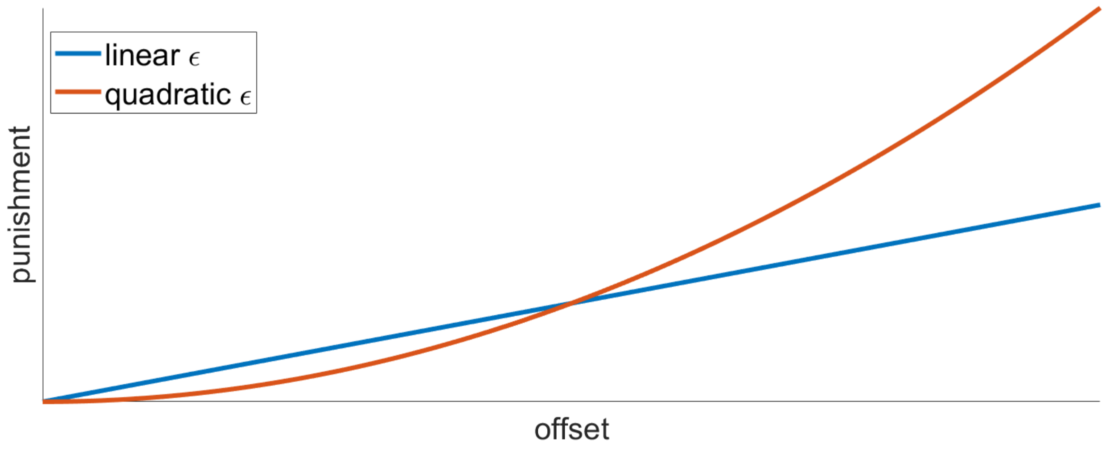


Figure 3.12: Slack variable punishment tendency

$$J_{\epsilon,k} = w_v \epsilon_{v,k}^2 + w_h \epsilon_{L,k}^2 + w_s \epsilon_{s,k} \tag{3.40}$$

Full cost function

the final cost function of NLP is formed by all the terms discussed above

$$J = \sum_{k=0}^{N-1} w_p P_{bat}(T_{m,k}, \omega_{m,k}) + w_a a_k^2 + w_j j_k^2 + w_v \epsilon_{v,k}^2 + w_h \epsilon_{L,k}^2 + w_s \epsilon_{s,k} \tag{3.41}$$

3.3.7 Full formulation

$$
\begin{aligned}
\min_{z} \sum_{k=0}^{N-1} & \Big(w_p P_{bat}(T_{m,k}, \omega_{m,k}) + w_a a_k^2 + w_j j_k^2 + w_v \epsilon_{v,k}^2 + w_h \epsilon_{L,k}^2 + w_s \epsilon_{s,k} \Big) \\
\text{s.t.} \quad & \text{initial state,} && 3.20 \\
& \text{optimized variable,} && 3.21 \\
& \text{speed tracking error,} && 3.25 \\
& \text{intrusion to desired headway distance,} && 3.26 \\
& \text{acceleration and jerk bounds from the ISO standard,} && 3.29 \\
& \text{acceleration and jerk bounds from comfortable,} && 3.30 \\
& \text{road speed limit constraint,} && 3.22 \\
& \text{comfortable curve speed constraint,} && 3.24 \\
& \text{safety distance,} && 3.27 \\
& \text{electric motor constraint,} && 3.32 \\
& \text{tire constraint,} && 3.33 \\
& \text{traffic light stop constraint,} && 3.34
\end{aligned}
\tag{3.42}
$$

3.3.8 Coding

For now, all formula of cost function and constraints are introduced, this part introduce the way to coding them into CasADi in MATLAB. The NLP structure is written in a more understandable structure, the coding is simple.

First, initialize the constraint, cost function. Create the empty list for constraint formula (g), upper and lower bound (ubg,lbg). And also initial cost function (J) is equal to 0.Then use for loop to traversal all time step in predict horizon, for each time step add the new term of cost function and new constraint in the list.

Algorithm 2 NLP coding

```
1: $J \leftarrow 0$
2: $g \leftarrow [\ ]$
3: $ubg \leftarrow [\ ]$
4: $lbg \leftarrow [\ ]$
5: for each time step $t$ in the prediction horizon $1 : N$ do
6:     $J \leftarrow J + \text{new\_cost}(t)$
7:     $g \leftarrow [g; \text{new\_constraint}(t)]$
8:     $ubg \leftarrow [ubg; \text{new upper bound}(t)]$
9:     $lbg \leftarrow [lbg; \text{new lower bound}(t)]$
10: end for
```

3.4 MPC Formulation with QP

In the last section, we introduced the NLP-base MPC controller. While it is accurate, the computation time is too high, making real-time implementation unfeasible. Therefore, NLP is transformed into a QP structure, which can compute much faster and achieve real-time performance. In the QP structure, we simplify the model and use some estimations. Simulations show that the QP structure significantly increases computation speed without greatly decreasing performance.

the input should be :

- ego vehicle initial state
- lead vehicle trajectory
- reference speed
- traffic light distance base limitation

the out put is :

- control total torque in motor level

3.4.1 Preliminary knowledge

In this section, we introduce some preliminary knowledge of the CasADi toolbox for solving QP problems

QP in CasADi

In CasADi, the NLP is written as the following form.

$$\begin{array}{l} \min \frac{1}{2} z^T H z + c^T z \\ \text{s.t.} \\ lbz < z < ubz \\ lbg < Gz < ubg \end{array} \tag{3.43}$$

z is the optimized vector, and H is the Hessian matrix, which is an $n_z \times n_z$ matrix (where n_z is the number of optimized vectors) that includes the quadratic term of the cost function it is a positive semi definite matrix. c is the coefficient vector of the linear term, represented as an $n_z \times 1$ vector that includes the linear term of the cost function. G is the constraint matrix, which is an $n_g \times n_z$ matrix (where n_g is the number of constraints). ubg, lbg, ubz, and lbz are the upper and lower bounds of the constraints and optimized vectors, respectively, similar to NLP.

As shown in the formula, the QP structure only allows up to quadratic terms in the cost function and only linear terms in the constraints. This is why some structures has to be changed.

Data form

SX is another data format in CasADi that can also define the optimized vector in a QP structure. While MX supports more types of operations and is more efficient in handling multiple sparse-matrix valued inputs and outputs, SX is a simpler data format that is more efficient for small-scale problems. After testing, it is proved that SX is more efficient for our QP problem.

$$z = \text{SX.sym}('z', n_z, 1) \tag{3.44}$$

Build solver

In casadi, the QP solver can be build use two kind of interface, they called low level and high level interface.

High-level interface :

$$\begin{gathered}\text{QP} = \text{struct('x',z,'f',}z^T H z^T + c^T z\text{,'g',}Gz\text{);}\\ \text{QPsolver} = \text{qpsol('QPsolver','qpoases',QP,opts)}\\ \text{result} = \text{QPsolver('x0',z0,'ubx',ubz,'lbx',lbz,'lbg', lbg, 'ubg', ubg);}\end{gathered} \tag{3.45}$$

Low-level interface :

$$\begin{gathered}\text{QP} = \text{struct('h', SX(H).sparsity(), 'a', SX(G).sparsity())}\\ \text{QPsolver} = \text{conic('QPsolver','qpoases',QP,opts)}\\ \text{result} = \text{QPsolver('ubx',ubz,'lbx',lbz,'uba',ubg,'lba',lbg,'h',H,'g',C,'a',G);}\end{gathered} \tag{3.46}$$

The low-level interface can only solve QP problems that have the same structure as shown above. However, it also supports changing the cost function and constraint formulas during repeated use, which provides more flexibility. This means that it supports modifying content such as weight factors and terms multiplied by the optimized vector. Additionally, it does not require defining the optimized vector.

The high-level interface is similar to the NLP introduced in the previous section. It has the same limitations as NLP, where only the bounds of constraints and the optimized vector can be changed during repeated use. However, it supports solving any structure of QP problems not limited to the above structures. The optimized vector must be defined.

Multiple shooting method (spare formulation)

For the multiple shooting method, the optimized vector includes not only the control values but also the states. This results in the matrix containing many zero terms, which is why it is also referred to as sparse. The iteration of states is realized through constraints. In this method, the state variables at each time step are treated as independent optimization variables, and the dynamic model of the system is enforced through constraints that link the state variables across time steps.

$$\begin{gathered} 0 \leq x_k - x_{k-1} - f(x_{k-1}, u_{k-1}) \leq 0 \\ z = [x_0, u_0, \epsilon_0, x_1, u_1, \epsilon_1 ......x_{N-1}, u_{N-1}, \epsilon_{N-1}, x_N] \end{gathered} \tag{3.47}$$

3.4.2 Lead and ego vehicle trajectory

For lead vehicle trajectory, the method is the same as NLP :

- **V2V**: the lead vehicle trajectory can be precise
- **Constant speed**: hypothesize that the lead vehicle will maintain a constant speed within the prediction horizon.

$$s_{lead,k} = s_{lead,0} + kt_s v_{lead,0} \tag{3.48}$$

- **Constant acceleration**: hypothesize that the lead vehicle will maintain a constant acceleration within the prediction horizon.

$$s_{lead,k} = s_{lead,0} + kt_s v_{lead,0} + \frac{1}{2} a_{lead,0} (kt_s)^2 \tag{3.49}$$

Because in the QP structure, the optimized vector term cannot be used as the input of a lookup table, we cannot use the same method as NLP to obtain distance and speed-based data such as road slope and motor torque limitation. Additionally, in constraints, only linear terms can be used, so quadratic terms cannot be implemented. To solve these two difficulties, we propose to estimate the ego vehicle trajectory within the prediction horizon and use the estimated values as the input of the lookup table and replace the nonlinear terms in the constraints.

The estimation method has three approaches.

- **Constant speed**: Hypothesize that the ego vehicle will maintain a constant speed within the prediction horizon.

$$\begin{cases} s_k = s_0 + kt_s v_0 \\ v_k = v_0 \end{cases} \tag{3.50}$$

- **Constant acceleration**: Hypothesize that the ego vehicle will maintain a constant acceleration within the prediction horizon.

$$\begin{cases} s_k = s_0 + kt_s v_0 + \frac{1}{2} a_0 (kt_s)^2 \\ v_k = v_0 + kt_s a_0 \end{cases} \tag{3.51}$$

- **Use previous MPC solution**: Hypothesize that the ego vehicle will follow the entire trajectory of the previous MPC solution.

$$s_{ego,k} = \begin{cases} s_{ego,pre,k+1} & \text{if } k = 1,2,3...N-1 \\ s_{ego,pre,k} + v_{ego,k} t_s & \text{if } k = N \end{cases} \tag{3.52}$$

$$v_{ego,k} = \begin{cases} v_{ego,pre,k+1} & \text{if } k = 1,2,3...N-1 \\ v_{ego,pre,k} + a_{ego,k} t_s & \text{if } k = N \end{cases} \tag{3.53}$$

3.4.3 Optimized vector

The QP-based MPC controller is implemented using the multiple shooting method. The optimized vector include both states (s_k, v_k) and control variable (T_{mk}). The acceleration (a_k) is also be constrained as additional state in order to easily punish the jerk.

The resulting optimization vector is :

$$z = [x_0, a_0, u_0, \epsilon_0, x_1, a_1, u_1, \epsilon_1 ......x_{N-1}, a_{N-1}, u_{N-1}, \epsilon_{N-1}, x_N, a_N] \tag{3.54}$$

3.4.4 Vehicle model

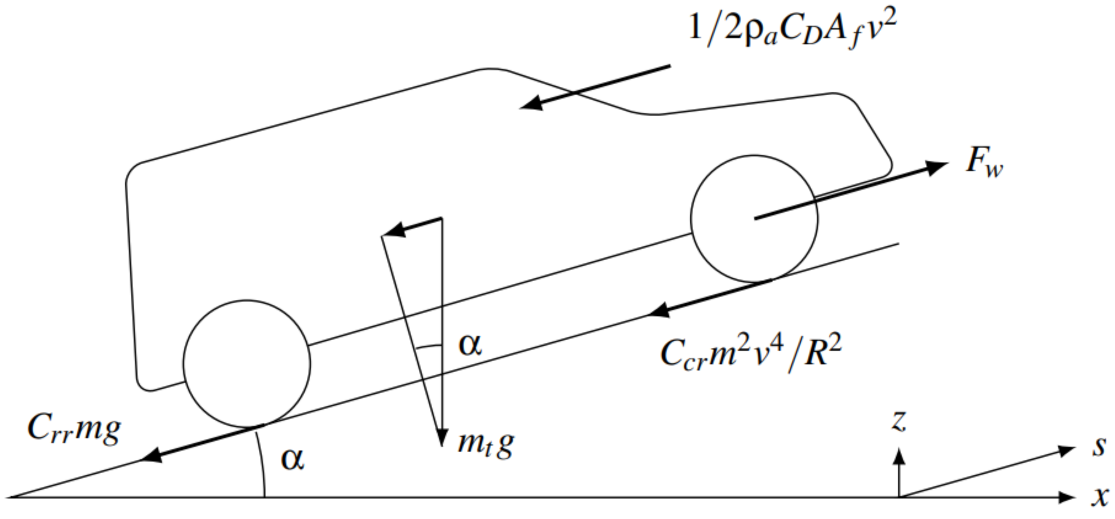


Figure 3.13: Vehicle longitudinal dynamic model base on [25]

The vehicle model remains the same as before, but use the estimated values of speed ($\widetilde{v_k}$) in the quadratic terms in aerodynamic drag and the estimated value

Table 3.3: Parameter symbol and unit of QP controller

parameter	symbol	**unit**
vehicle mass	m	kg
inertial coefficient	λ	-
vehicle front area	A_x	m^2
areo-dynamic coefficient	C_x	-
wheel radius	r_w	m
total torque at motor level	T_m	Nm
gear box efficiency	i_{gb}	-
air density	ρ	kg/m^3
gravitational acceleration	g	N/m^2
road angle	θ	rad
rolling resistance coefficient	f_0	-
sampling time	t_s	s
vehicle travel distance	s	m
vehicle speed	v	m/s
estimated vehicle travel distance	$\widetilde{s}$	m
estimated vehicle speed	$\widetilde{v}$	m/s
friction coefficient	μ	-
electric motor maximum power	P_{max}	w
electric motor maximum toque	T_{max}	Nm

of distance $(\widetilde{\theta_k})$ in road slope. Additionally, the gearbox efficiency is inversed due to different energy flow directions during traction and regenerative braking. Since this cannot be implemented in QP, the gear box efficiency is neglected in QP..

$$\begin{aligned} s_{k+1} &= s_k + v_k T_s \\ v_{k+1} &= (\tfrac{1}{r} T_{m,k} i_{gb} - \tfrac{1}{2}\rho C_x A_x \widetilde{v_k}^2 - mgsin(\widetilde{\theta_k}) - f_0 mgcos(\widetilde{\theta_k}))\tfrac{1}{\lambda m} t_s + v_k \end{aligned} \tag{3.55}$$

Table 3.3 shows the parameters and units use in this chapter.

3.4.5 Constraints

The constraints use the same idea as in NLP, but the variables that are functions of the state and obtained from the lookup table use the estimated state. These estimated states are defined by the initial state and are predefined variables not included in the optimized variables. Therefore, they should not be written in the constraint formula (G) as in NLP, but in the constraint bounds (ubg, lbg). Additionally, the constraint formula should be written as $G \cdot z$, which is different from NLP. In each of the following sections, the formula is first written in ordinary logic form, then followed by its representation in CasADi using matrix notation

Optimized vector constraint

The optimized vector includes the initial state as discussed before. In the optimized vector, constraints should be used to force it to be equal to the input initial state.

$$\begin{bmatrix} s_{in} \\ v_{in} \\ a_{in} \end{bmatrix} \leq \begin{bmatrix} s_0 \\ v_0 \\ a_0 \end{bmatrix} \leq \begin{bmatrix} s_{in} \\ v_{in} \\ a_{in} \end{bmatrix} \tag{3.56}$$

the other states be bounded in base logic, the states are also included. Speed and slack variable should always be positive, travel distance and motor torque are not be constrained

$$\begin{bmatrix} -\infty \\ 0 \\ -\infty \\ 0 \end{bmatrix} \leq \begin{bmatrix} s_k \\ v_k \\ F_{m,k} \\ \xi_{i,k} \end{bmatrix} \leq \begin{bmatrix} \infty \\ \infty \\ \infty \\ \infty \end{bmatrix} \tag{3.57}$$

Iteration constraint

As discussed in preliminary knowledge, the model state iteration should be realize in constraint. the road slop is get from estimated vehicle position in the predict time horizon, the areodynamic resistance also use estimated vehicle speed.

The resulting total resistance is expressed by D_k.

$$D_k = \frac{1}{2}\rho A C_x \widetilde{v_k}^2 + mgsin(\theta(\widetilde{s_k})) + mgf_0cos(\theta(\widetilde{s_k}))(\widetilde{v_k} > 0.1) \tag{3.58}$$

The state evolution becomes

$$\begin{gathered} s_{k+1} = s_k + v_k t_s \\ v_{k+1} = \tfrac{1}{\lambda m} t_s D_k + v_k \end{gathered}$$

$$\begin{bmatrix} 0 \\ \frac{t_s}{\lambda m} D_k \end{bmatrix} \leq \begin{bmatrix} 1 & t_s & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & \frac{i_{gb} t_s}{\lambda m r} \end{bmatrix} \begin{bmatrix} s_{k-1} \\ v_{k-1} \\ s_k \\ v_k \\ T_m \end{bmatrix} \leq \begin{bmatrix} 0 \\ \frac{t_s}{\lambda m} D_k \end{bmatrix} \tag{3.59}$$

Acceleration as additional state for easily punish jerk.

$$\lambda m a_k = \tfrac{i_{gb}}{r} T_{mk} - D_k$$

$$\begin{bmatrix} D_k \end{bmatrix} \leq \begin{bmatrix} \frac{i_{gb}}{r} & -\lambda m \end{bmatrix} \begin{bmatrix} T_{m,k} \\ a_k \end{bmatrix} \leq \begin{bmatrix} D_k \end{bmatrix} \tag{3.60}$$

Vehicle speed

The vehicle speed limitation should consider both the road speed limit and safety requirements, they are get from estimated vehicle position in the predict time horizon.

$$\begin{gathered}0 \leq v_k \leq v_{max}(s_k) \\ 0 \leq v_k \leq 3.34 \cdot |\theta(s_k)|^{-\frac{1}{3}} \\ \begin{bmatrix}0\end{bmatrix} \leq \begin{bmatrix}1\end{bmatrix}\begin{bmatrix}v_k\end{bmatrix} \leq \begin{bmatrix}v_{max}(\widetilde{s_k})\end{bmatrix} \\ \begin{bmatrix}0\end{bmatrix} \leq \begin{bmatrix}1\end{bmatrix}\begin{bmatrix}v_k\end{bmatrix} \leq \begin{bmatrix}3.34 \cdot |\theta(\widetilde{s_k})|^{-\frac{1}{3}}\end{bmatrix}\end{gathered} \tag{3.61}$$

Speed tracing error

The offset of the reference speed in NLP is already linear and can be the same in QP. The reference speed is an input that should be implemented in the constraint bounds.

$$\begin{gathered}v_{ref,k} \leq v_k + \epsilon_{v,k} \\ v_{ref,k} \geq v_k - \epsilon_{v,k} \\ \begin{bmatrix}v_{ref,k} \\ -\infty\end{bmatrix} \leq \begin{bmatrix}1 & 1 \\ 1 & -1\end{bmatrix}\begin{bmatrix}v_k \\ \epsilon_{v,k}\end{bmatrix} \leq \begin{bmatrix}\infty \\ v_{ref,k}\end{bmatrix}\end{gathered} \tag{3.62}$$

Intrusion to lead vehicle desired headway distance

The intrusion into the lead vehicle's desired headway distance also uses three methods. The quadratic term of the headway distance policy should use the estimated ego vehicle speed.

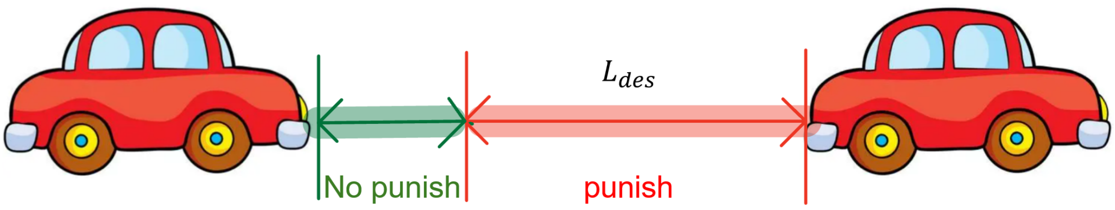


Figure 3.14: Punishment of intrusion to lead vehicle desired headway distance

$$\begin{gathered}\epsilon_{L,k} + s_{lead,k} \geq s_k + A + Tv_k + Gv_k^2 \\ \begin{bmatrix}-\infty\end{bmatrix} \leq \begin{bmatrix}1 & -1 & T\end{bmatrix} \begin{bmatrix} s_k \\ \epsilon_{L,k} \\ v_k \end{bmatrix} \leq \begin{bmatrix} s_{lead,k} - A - G\widetilde{v_k}^2 \end{bmatrix}\end{gathered} \tag{3.63}$$

Safety distance

the ego vehicle also should keep at least one meter to lead vehicle.

$$\begin{gathered} s_{lead,k} - s_k \geq 1 \\ \begin{bmatrix}-\infty\end{bmatrix} \leq \begin{bmatrix}1\end{bmatrix} \begin{bmatrix}s_k\end{bmatrix} \leq \begin{bmatrix}s_{lead,k} - 1\end{bmatrix} \end{gathered} \tag{3.64}$$

Acceleration and jerk should remain within comfortable limits and ISO standard

The acceleration and jerk should also comply with the ISO standard. The ISO standard is a piecewise function of the ego vehicle speed. In QP, these values are obtained from the estimated ego vehicle speed and implemented as inputs in the constraint bounds.

$$\begin{gathered} a_{min}(v_k) \leq a_k \leq a_{max}(v_k) \\ j_{min}(v_k) \leq j_k \leq j_{max}(v_k) \\ \begin{bmatrix}-\infty \\ -\infty\end{bmatrix} \leq \begin{bmatrix}1 \\ -1\end{bmatrix} \begin{bmatrix}a_k\end{bmatrix} \leq \begin{bmatrix} a_{max,k}(\widetilde{v_k}) \\ -a_{min,k}(\widetilde{v_k}) \end{bmatrix} \\ \begin{bmatrix}-\infty \\ -\infty\end{bmatrix} \leq \begin{bmatrix}1 \\ -1\end{bmatrix} \begin{bmatrix}j_k\end{bmatrix} \leq \begin{bmatrix} j_{lim,k}(\widetilde{v_k}) \\ j_{lim,k}(\widetilde{v_k}) \end{bmatrix} \end{gathered} \tag{3.65}$$

The acceleration and jerk should always remain within ±2 m/s^2. Though the slack variable with high weight factor.

$$\begin{gathered} -2 - \epsilon_{s,k} \leq a_k \leq 2 + \epsilon_{s,k} \\ -2 - \epsilon_{s,k} \leq j_k \leq 2 + \epsilon_{s,k} \\ \begin{bmatrix}-\infty \\ -2\end{bmatrix} \leq \begin{bmatrix}1 & -1 \\ 1 & 1\end{bmatrix} \begin{bmatrix}a_k \\ \epsilon_{s,k}\end{bmatrix} \leq \begin{bmatrix}2 \\ \infty\end{bmatrix} \\ \begin{bmatrix}-\infty \\ -2\end{bmatrix} \leq \begin{bmatrix}1 & -1 \\ 1 & 1\end{bmatrix} \begin{bmatrix}j_k \\ \epsilon_{s,k}\end{bmatrix} \leq \begin{bmatrix}2 \\ \infty\end{bmatrix} \end{gathered} \tag{3.66}$$

Electric motor limitation

The electric motor limitation is a piecewise function. This limitation should also be obtained from the estimated ego vehicle speed. The motor limitation is formed by a constant and a hyperbolic function. The boundary should use the minimum value between them.

$$T_{m,max}(\omega) = \begin{cases} T_{max} & \text{if } \omega < \omega^* \\ \frac{P_{max}}{\omega} & \text{if } \omega >= \omega^* \end{cases} \tag{3.67}$$

$$\begin{gathered} T_m \leq T_{m,max}(\omega_k) \\ \begin{bmatrix} -\infty \end{bmatrix} \leq \begin{bmatrix} 1 \end{bmatrix} \begin{bmatrix} T_{mk} \end{bmatrix} \leq \begin{bmatrix} \min(T_{mk,max}, \frac{P_{m,max}}{w_{mk}}) \end{bmatrix} \end{gathered} \tag{3.68}$$

Tire limitation

the friction limitation is similar to NLP, but change the areodynamic term in to estimated ego vehicle, the slop is base on the estimated vehicle position.

$$\begin{gathered} |T_{mk} i_{gb} \tfrac{1}{r_w}| \leq \mu mgcos(\theta(s_k)) \\ T_{mk} i_{gb} \tfrac{1}{r_w} \leq \tfrac{\mu_x}{L} \Big[mg\Big(L_f cos(\theta(s_k)) + h_g sin(\theta(s_k)) \Big) + h_g (\tfrac{1}{2} \rho C_x A_f v_k^2 + \lambda ma) \Big] \\ \begin{bmatrix} -\mu mgcos(\theta(\widetilde{s_k})) \\ -\infty \end{bmatrix} \leq \begin{bmatrix} \frac{i_{gb}}{r_w} \end{bmatrix} \begin{bmatrix} T_{m,k} \end{bmatrix} \leq \begin{bmatrix} \mu mgcos(\theta(\widetilde{s_k}) \\ mg((L_f - f_0)cos(\theta(\widetilde{s_k}) + (h_g - 1)sin(\widetilde{s_k})) \end{bmatrix} \end{gathered} \tag{3.69}$$

Traffic light limitation

the traffic light limitation is still base on the estimated vehicle position in the predict horizon.

Algorithm 3 Phase in predict horizon

```
1: for each time step in predict horizon $1 : N$ do
2:     countdown timer = countdown timer - $t_s$
3:     if countdown timer $\leq 0$ then
4:         phase change, different from initial one
5:     else
6:         phase is the same as initial one
7:     end if
8: end for
```

$$\begin{gathered} s_k \leq wall_k(s_k) + phase \cdot 10^7 + pass \cdot 10^7 \\ \begin{bmatrix} -\infty \end{bmatrix} \leq \begin{bmatrix} 1 \end{bmatrix} \begin{bmatrix} s_k \end{bmatrix} \leq \begin{bmatrix} wall_k(s_{k,est}) + phase \cdot 10^7 + pass \cdot 10^7 \end{bmatrix} \end{gathered} \tag{3.70}$$

3.4.6 Cost function

The cost function in QP is written in the form $\frac{1}{2}z^T Hz + c^T z$. It is separated into two parts. The matrix H includes the quadratic terms. For the variable z_i^2, its term is in $H_{i,i}$. For the variable $z_i z_j$, its term exists in both $H_{i,j}$ and $H_{j,i}$ at the same time. The term should be equally separated into two parts and written in both $H_{i,j}$ and $H_{j,i}$. The vector c includes the linear terms, with the term of variable z_i in c_i. It should be noted that due to the $\frac{1}{2}$ in the formula, the matrix H should be considered as twice the designed cost function's quadratic term. we first introduce and write the formula in ordinary logic form, then introduce the written format in CasADi using matrices.

Power consumption

In QP, the cost function can use up to quadratic terms. Therefore, the 5×5 polynomial can not be used to interpolate the power map; instead, a 2×2 polynomial is used instead. The interpolation method is the same as in NLP, using CasADi. the constant term b_{00} is useless, so it is bot used in the QP. The resulting goodness of fit among the sampling points is 98.13 %

$$
\begin{aligned}
P_{bat} &= b_{00} + b_{10}T_m + b_{01}\omega + b_{11}T_m\omega + b_{20}T_m^2 + b_{02}\omega^2 \\
P_{bat,k} &= \frac{1}{2}\begin{bmatrix} T_{m,k} \\ v_k \end{bmatrix}^T \begin{bmatrix} 2w_p b_{20} & w_p b_{11}\frac{i_{gb}}{r} \\ w_p b_{11}\frac{i_{gb}}{r} & w_p b_{02}(\frac{i_{gb}}{r})^2 \end{bmatrix} \begin{bmatrix} T_{m,k} \\ v_k \end{bmatrix} + \begin{bmatrix} w_p b_{10} \\ w_p b_{01} \end{bmatrix}^T \begin{bmatrix} T_{m,k} \\ v_k \end{bmatrix}
\end{aligned}
\tag{3.71}
$$

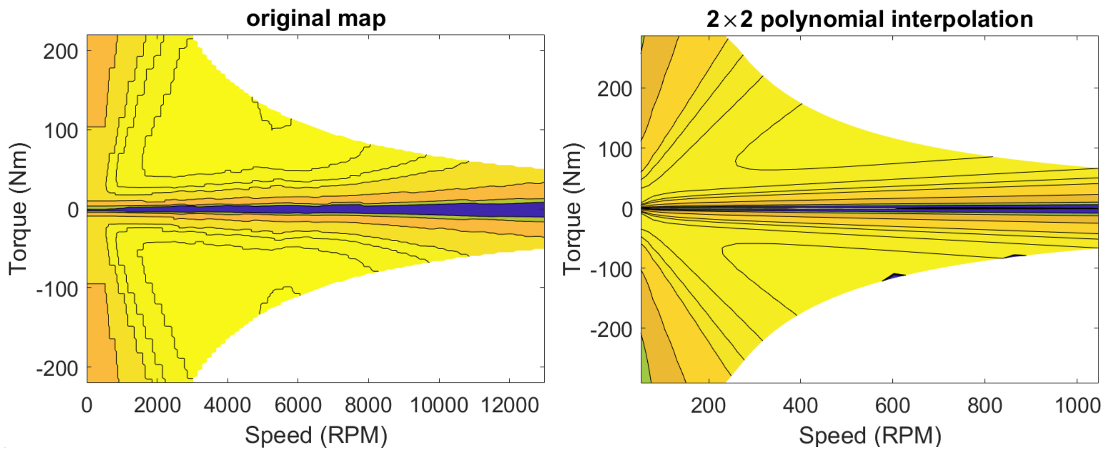


Figure 3.15: Validated original and interpolated efficiency map

Acceleration and jerk

Acceleration and jerk highly affect comfort. The penalties are divided into two sets: the first step penalizes acceleration below ±2 m/s^2, and the second step addresses values over ±2 m/s^2 using slack variables. Jerk is the function of acceleration written in equation 3.72

$$j_k = \frac{a_k - a_{k-1}}{t_s} \tag{3.72}$$

$$\begin{aligned} J_{aj,k} &= w_a a_k^2 + w_j j_k^2 \\ J_{a,k} &= \begin{bmatrix} a_k \end{bmatrix}^T \begin{bmatrix} w_a \end{bmatrix} \begin{bmatrix} a_k \end{bmatrix} \\ J_{j,k} &= \tfrac{1}{2} \begin{bmatrix} a_k \\ a_{k-1} \end{bmatrix}^T \begin{bmatrix} 2w_j(\frac{1}{t_s})^2 & -2w_j(\frac{1}{t_s})^2 \\ -2w_j(\frac{1}{t_s})^2 & 2w_j(\frac{1}{t_s})^2 \end{bmatrix} \begin{bmatrix} a_k \\ a_{k-1} \end{bmatrix} \end{aligned} \tag{3.73}$$

Slack variable

the slack variable include :

- offsetting from the reference speed ϵ_v
- intruding into the lead vehicle's desired headway distance ϵ_L
- exceeding safety and comfort limitations ϵ_s

The slack variable are always positive, as the same as NLP, ϵ_v and ϵ_L use quadratic term, ϵ_s use linear term.

$$\begin{aligned} J_{\epsilon,k} &= w_v \epsilon_{v,k}^2 + w_h \epsilon_{L,k}^2 + w_s \epsilon_{s,k} \\ J_{\epsilon,k} &= \tfrac{1}{2} \begin{bmatrix} \epsilon_{v,k} \\ \epsilon_{L,k} \\ \epsilon_{s,k} \end{bmatrix}^T \begin{bmatrix} 2w_v & 0 & 0 \\ 0 & 2w_L & 0 \\ 0 & 0 & 2w_s \end{bmatrix} \begin{bmatrix} \epsilon_{v,k} \\ \epsilon_{L,k} \\ \epsilon_{s,k} \end{bmatrix} \end{aligned} \tag{3.74}$$

Full cost function

the final cost function of NLP is formed by all the trems discussed above

$$J = \sum_{k=0}^{N-1} w_p P_{bat}(T_{m_k}, \omega_{m,k}) + w_a a_k^2 + w_j j_k^2 + w_v \epsilon_{v,k}^2 + w_h \epsilon_{L,k}^2 + w_s \epsilon_{s,k} \tag{3.75}$$

3.4.7 Full formulation

$$
\begin{aligned}
\min_{z} \sum_{k=0}^{N-1} & \Big(w_p P_{bat}(T_{m,k}, \omega_{m,k}) + w_a a_k^2 + w_j j_k^2 + w_v \epsilon_{v,k}^2 + w_h \epsilon_{L,k}^2 + w_s \epsilon_{s,k} \Big) \\
\text{s.t.} \quad & \text{iteration constraint,} && 3.59 \\
& \text{initial state,} && 3.56 \\
& \text{state,} && 3.57 \\
& \text{offset to reference speed,} && 3.62 \\
& \text{intrusion to desired headway distance,} && 3.63 \\
& \text{acceleration and jerk bounds from the ISO standard,} && 3.65 \\
& \text{acceleration and jerk bounds from comfortable,} && 3.66 \\
& \text{speed limit constraint,} && 3.61 \\
& \text{safety distance,} && 3.64 \\
& \text{electric motor constraint,} && 3.68 \\
& \text{tire constraint,} && 3.69 \\
& \text{traffic light constraint,} && 3.70
\end{aligned}
\tag{3.76}
$$

3.4.8 Coding

Our QP structure has seven optimized variables at each time step, with a prediction horizon of 20 steps, which means we have a total of 143 optimized variables. The Hessian matrix is a 143 × 143 matrix, and the constraint matrix is a 340 × 143 matrix. These huge matrices are very difficult to code. In this section, we introduce a simple coding method.

In the beginning, the empty list is created for the constraint matrix (G) and the upper and lower constraint bounds (ubg, lbg) as in NLP. The Hessian matrix (H) and the coefficient vector of the linear term (c) are initialized with all zeros.

A for loop is employed to iterate through the entire prediction horizon. At each time step, only the current seven optimized variables and three previous states are utilized, constituting partial optimized variables. This approach enables us to independently address constraints for each time step.

Firstly, Identify the useful optimized variables by the time step number k (n_z is the total number of optimized variables, n_x is the number of states). Then, each time a constraint needs to be added, create a $1 \times n_z$ vector g with all zeros, use the identified number of optimized variables to rewrite the specific term of vector g, and then add it to the constraint matrix. The constraint bounds are directly added as new terms.

In handling the Hessian matrix, different part of cost function may share the same term. To prevent overwriting each other, when introducing a new cost, it's added based on the original Hessian matrix. This method is similarly applied to the coefficient vector of the linear term.

Algorithm 4 QP coding constraint and cost

```
 1: $G \leftarrow [\ ]$
 2: $ubg \leftarrow [\ ]$
 3: $lbg \leftarrow [\ ]$
 4: $H \leftarrow \text{zeros}(n_z \times n_z)$
 5: $C \leftarrow \text{zeros}(n_z \times 1)$
 6: for each time step $t$ in the prediction horizon $1 : N$ do
 7:     $N_{Tm} \leftarrow 1 + n_x + (k-1)n_z$
 8:     $N_{\epsilon v} \leftarrow 2 + n_x + (k-1)n_z$
 9:     $N_{\epsilon L} \leftarrow 3 + n_x + (k-1)n_z$
10:     $N_{\epsilon s} \leftarrow 4 + n_x + (k-1)n_z$
11:     $N_s \leftarrow 5 + n_x + (k-1)n_z$
12:     $N_v \leftarrow 6 + n_x + (k-1)n_z$
13:     $N_a \leftarrow 7 + n_x + (k-1)n_z$
14:     $N_{s,previous} \leftarrow N_s - n_z$
15:     $N_{v,previous} \leftarrow N_v - n_z$
16:     $N_{a,previous} \leftarrow N_a - n_z$
17:     $g \leftarrow \text{zeros}(1 \times n_z)$
18:     $g([N_1, N_2, N_3]) \leftarrow [x_1, x_2, x_3]$
19:     $G \leftarrow [G; g]$
20:     $ubg \leftarrow [ubg; \text{new upper bound}(t)]$
21:     $lbg \leftarrow [lbg; \text{new lower bound}(t)]$
22:     $H(N_1, N_1) \leftarrow cost_1$
23:     $H(N_2, N_1) \leftarrow cost_2$
24:     $H(N_1, N_2) \leftarrow cost_2$
25: end for
```

Algorithm 5 QP coding state bounds

```
1: $ubz \leftarrow [\ ]$
2: $lbz \leftarrow [\ ]$
3: $ubz \leftarrow [ubz; \text{initial state}(t)]$
4: $lbz \leftarrow [lbz; \text{initial state}(t)]$
5: for each time step $t$ in the prediction horizon $1 : N$ do
6:     $ubz \leftarrow [ubz; \text{new upper bound}(t)]$
7:     $lbz \leftarrow [lbz; \text{new lower bound}(t)]$
8: end for
```

3.5 Environment Setup and Controller Development

In the previous three sections, we introduced the sub-blocks of the total controller. In this section, we will assemble the controller and integrate it into MATLAB Simulink(R).

3.5.1 Overall structure

The scenario information is formed by two part, SPaT signal and lead vehicle state and aso the sgo vehicle state should be a input of controller.

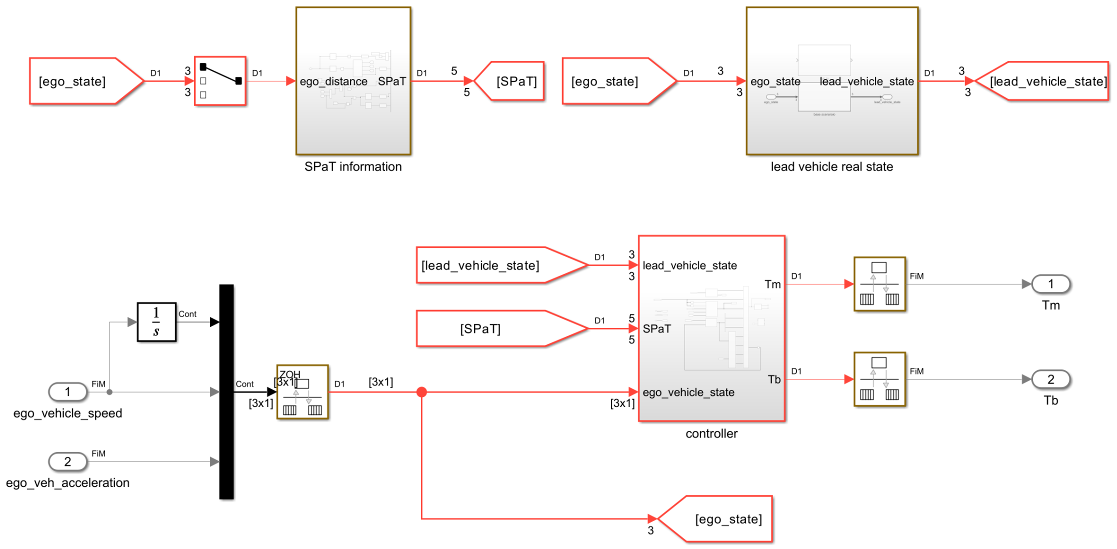


Figure 3.16: Overall structure of the system in MATLAB Simulink

3.5.2 Environment Setup

SPaT signal

In the real use case, the vehicle will directly receive three pieces of information (traffic light phase, upper bound countdown timer, and lower bound countdown timer) from SPaT using V2I technology, and receive the positions of the traffic light and the ego vehicle through GPS. These scenarios should not be integrated into the controller, but rather provided as inputs. To simulate such scenarios, we create a signal called 'SPaT'. This signal not only includes the normal SPaT information, but also integrates the traffic light position and an additional signal representing whether SPaT exists. the signal is get from the following setup, The phase offset time is the time until the first red phase starts in the beginning of test scenario:

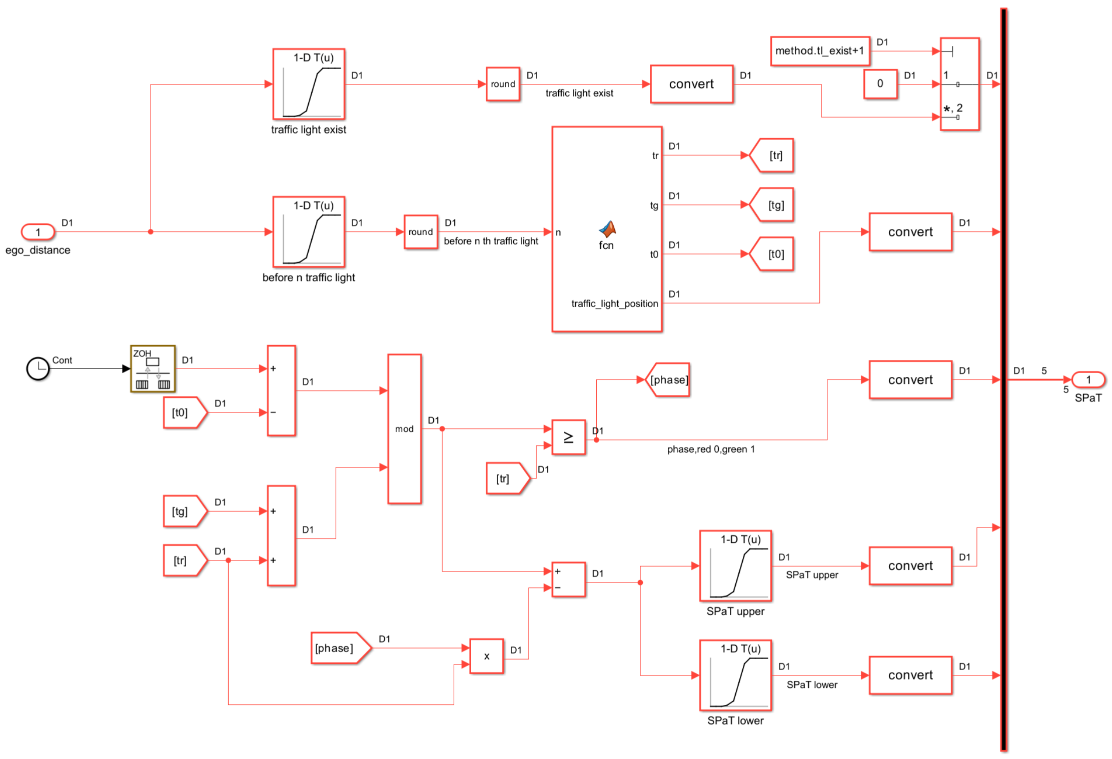


Figure 3.17: Simulated SPaT signal in MATLAB Simulink

Table 3.4: Traffic light setup

parameter	symbol	unit
Green Light Duration	t_r	s
Red Light Duration	t_g	s
phase offset time	t_0	s
current time	t	s
traffic light position	s_{tl}	m

The traffic light exist signal is get from look up table, input is travel distance, out up is logic value, 1 means traffic light signal exist, 0 means it do not exist.

Firstly use the ego vehicle distance to locate which traffic light ego vehicle is ahead by the look up table, secondly get the setup of that traffic light, then adjust the phase by :

$$phase = \begin{cases} 0(\text{red}) & \text{if } (t - t_0) \mod (t_r + t_g) < t_r \\ 1(\text{green}) & \text{if } (t - t_0) \mod (t_r + t_g) > t_r \end{cases} \tag{3.77}$$

To determine the upper bound countdown timer and lower bound countdown timer, it is essential to have knowledge of the duration of the current phase. Then, use the lookup table to get the upper and lower bound countdown timers. The mod operation gives the seconds passed since the beginning of the last red phase, so when the phase is green, the green light duration should additionally subtract t_r.

$$t_{last} = ((t - t_0) \mod (t_r + t_g)) - phase \cdot t_r \tag{3.78}$$

Ego and lead vehicle state

the ego vehicle state are get from the plant model, the plant model is build though the Simscape libraries in MATLAB Simulink.

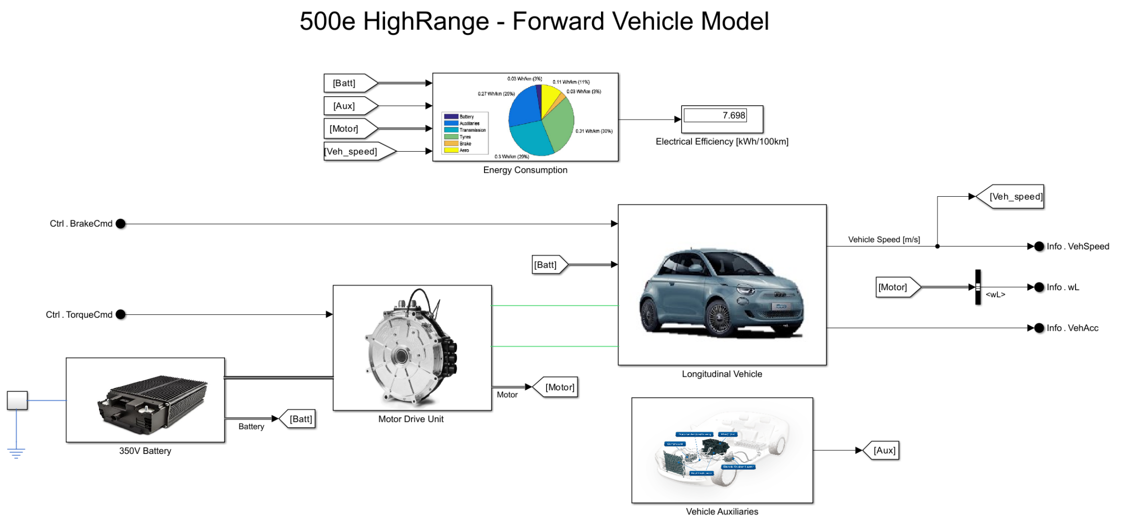


Figure 3.18: Plant model construct by Simscape libraries in MATLAB Simulink.

3.5.3 Controller Development

Different functions are set separately in different blocks, which is helpful for improving specific logic in future work. The following figure 3.19 shows the structure of the QP controller. The NLP controller is similar but does not include 'ego vehicle state estimation' and 'get limitation block'. The blocks for lead/ego vehicle estimation and reference speed generator are implemented using simple

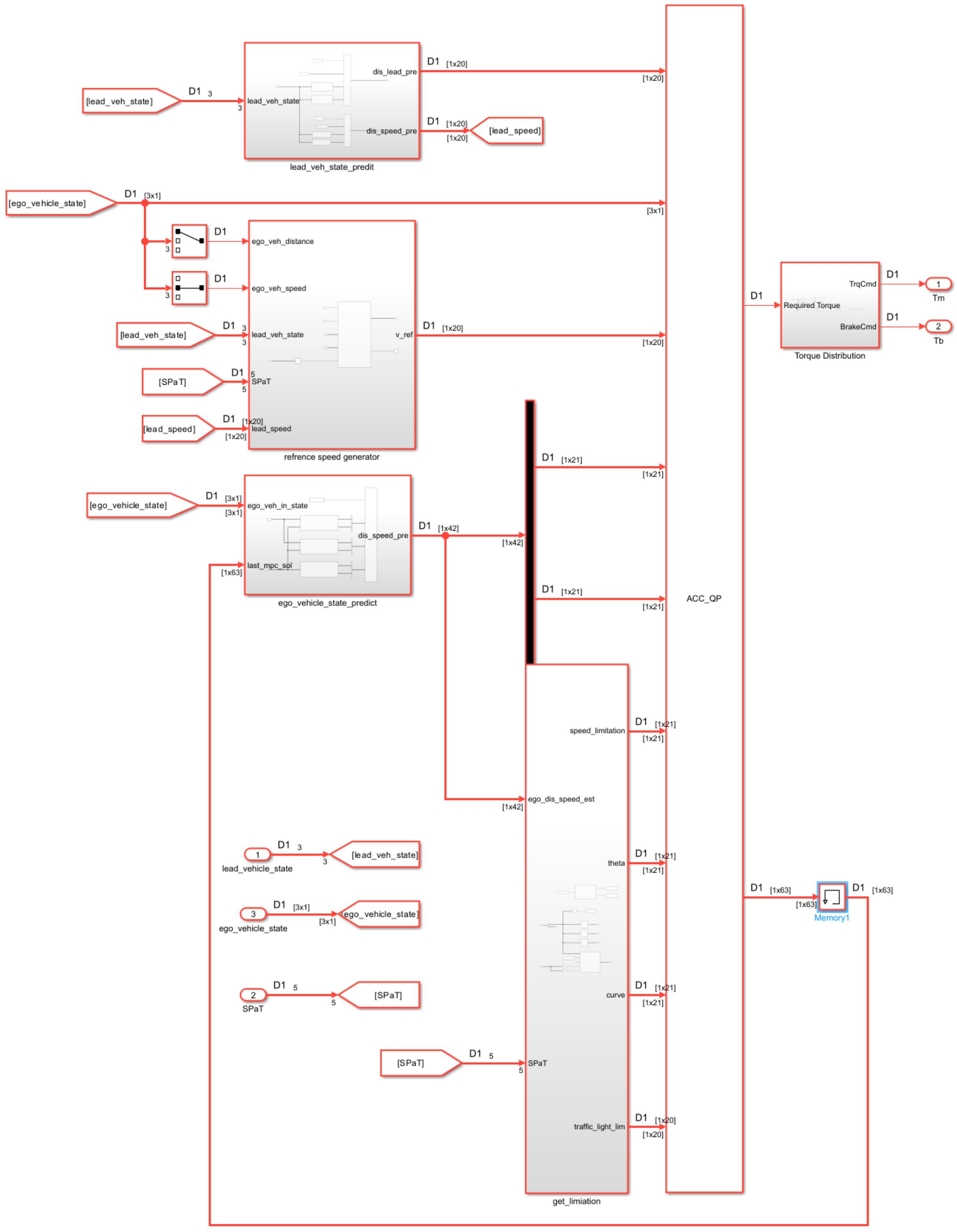


Figure 3.19: MPC controller in MATLAB Simulink.

MATLAB function blocks in Simulink. The block "get limitation" is used to obtain the limitations of speed, traffic lights, and road angle. It is independent of the MPC solver because we want to dynamically change the road information during use, and the solver itself cannot be changed. For NLP controller, it is only a baseline which will not use in real-time, the road information is included in the solver which means a solver can only deal with the specific road section.

The solver includes CasADi code, which cannot be directly implemented in Simulink or on a real vehicle controller. Therefore, we transform the solver into C language by MinGW64, and then implement it into the 's-function' block of Simulink, which can execute C language code. The compilation files are provided by the CasADi official website. Named 'casadi_fun'

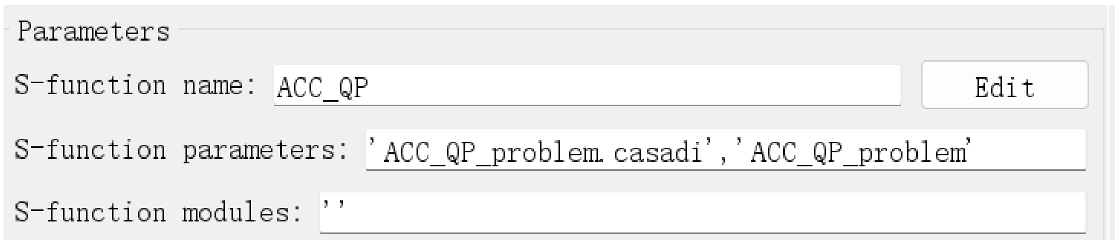
Firstly, write the optimization problem as introduced in the last two sections and generate the function. The inputs should be defined using the MX form in CasADi. In QP-based MPC controller, these inputs include the ego vehicle's initial state, speed limitation, ego vehicle's estimated distance and speed in the prediction horizon, lead vehicle's estimated distance in the prediction horizon, reference speed, and distance-based traffic light limitations. The output is the total torque at the motor level.

$$\text{ACC\_QP\_problem} = \text{Function('ACC\_QP\_problem'}, \{\text{input}\}, \{\text{output}\}) \tag{3.79}$$

Then, generate the CasADi files and compile them into C language files. The 'lib_path' and 'in_path' are the paths for the CasADi toolbox files. These paths should be automatically obtained using the following command. Then we can get a '.mex64' file.

$$\begin{aligned}&\text{ACC\_QP\_problem.save('ACC\_QP\_problem.casadi')}\\&\text{lib\_path} = \text{GlobalOptions.getCasadiPath()}\\&\text{inc\_path} = \text{GlobalOptions.getCasadiIncludePath()}\\&\text{mex('-v',['-I' inc\_path],['-L' lib\_path],'-lcasadi', 'ACC\_QP.c')}\end{aligned} \tag{3.80}$$

At last, put them into s-function block.


Parameters

S-function name: ACC_QP

S-function parameters: 'ACC_QP_problem.casadi','ACC_QP_problem'

S-function modules: ''

Figure 3.20: s-function control interface

Chapter 4

Experiments and Results

This chapter is dedicated to the implementation of the test plan and the subsequent evaluation of the test results. The experiments are executed in MATLAB Simulink, where connect the controller to a vehicle model built using Simscape for testing.

Firstly, tune the weight factor. Secondly, test the reaction to cut-in scenarios and use four different scenarios to test the controller. Thirdly, compare the effects of lead vehicle trajectory estimation and different headway policies. Fourthly, compare the differences between the NLP-based controller and the QP controller. Finally, analyze the multi-vehicle following performance using V2V technology.

4.1 Test scenario

The test driving cycle recorded in Turin, Italy, shown as figure 4.1, with a total length of 3089 meters and travel time of 435 seconds.The lead vehicle is driven by a human, without any optimization. The SPaT information is set by assumption, as shown in Figure 4.3. The x-axis represents the elapsed time of the current phase, while the y-axis represents the maximum and minimum actuated signal back counters, the back counter will suddenly change sometimes. The SPaT information is only available within 300 meters before the traffic light. The signals delivered to the vehicle include only these two back counters and the phase.The test vehicle model is a Fiat 500e, and the data is shown in Table 4.1. The vehicle model is constructed by Simscape libraries in MATLAB Simulink. The controller refresh time is 0.1 seconds, assume that during braking, 80 percent of the brake torque will be contributed by regenerative braking, while the remaining 20 percent will be provided by mechanical braking.

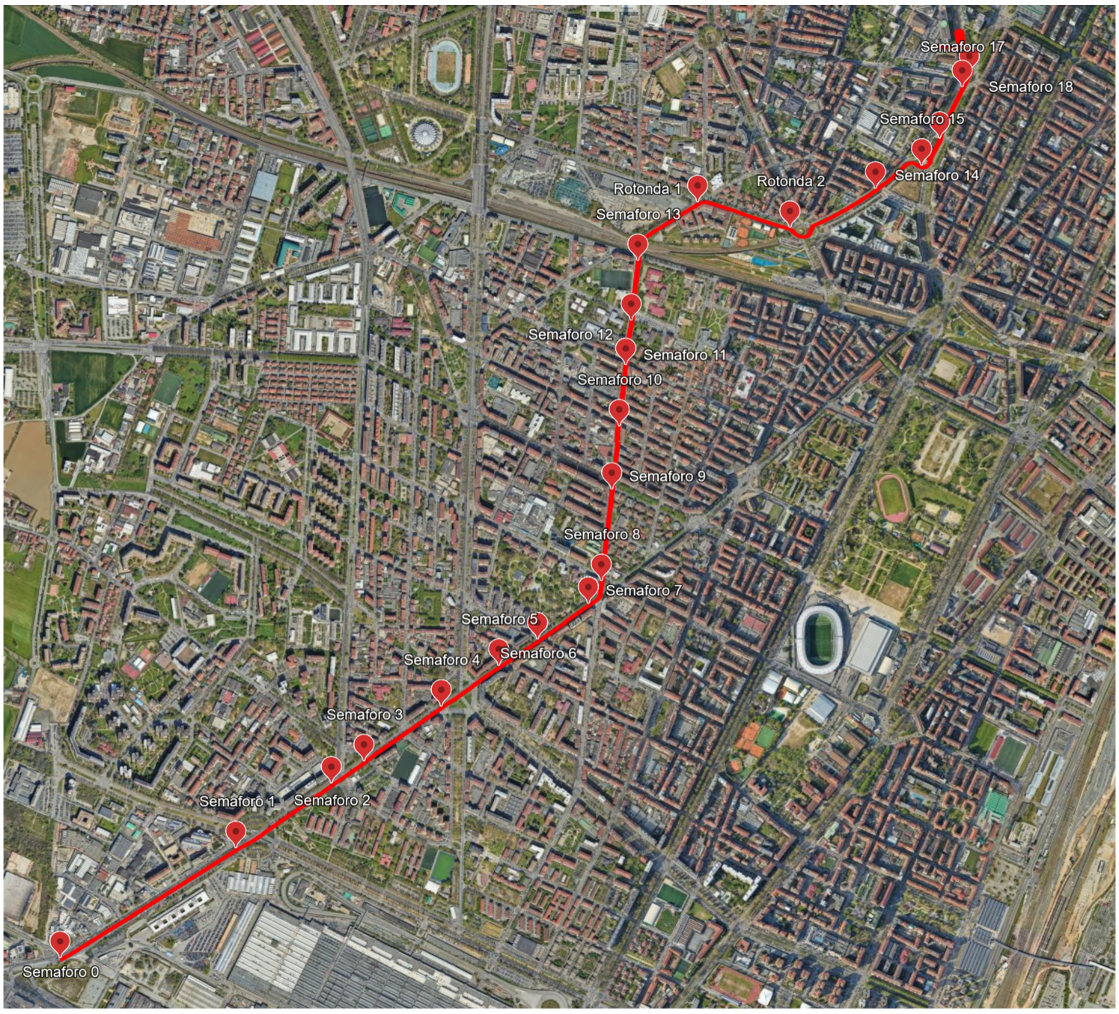


Figure 4.1: Driving cycle in map

Table 4.1: Vehicle data

mass	1443kg	wheelbase	2.322m
inertial factor	1.05	front overhang	1.045m
wheel radius	0.3m	rear overhang	1.277m
front area	2.15m^2	areo-dynamic coefficient	0.33
rolling coefficient	0.006	gear box ratio	9.559
gear box efficiency	0.97	center gravity height	0.5m

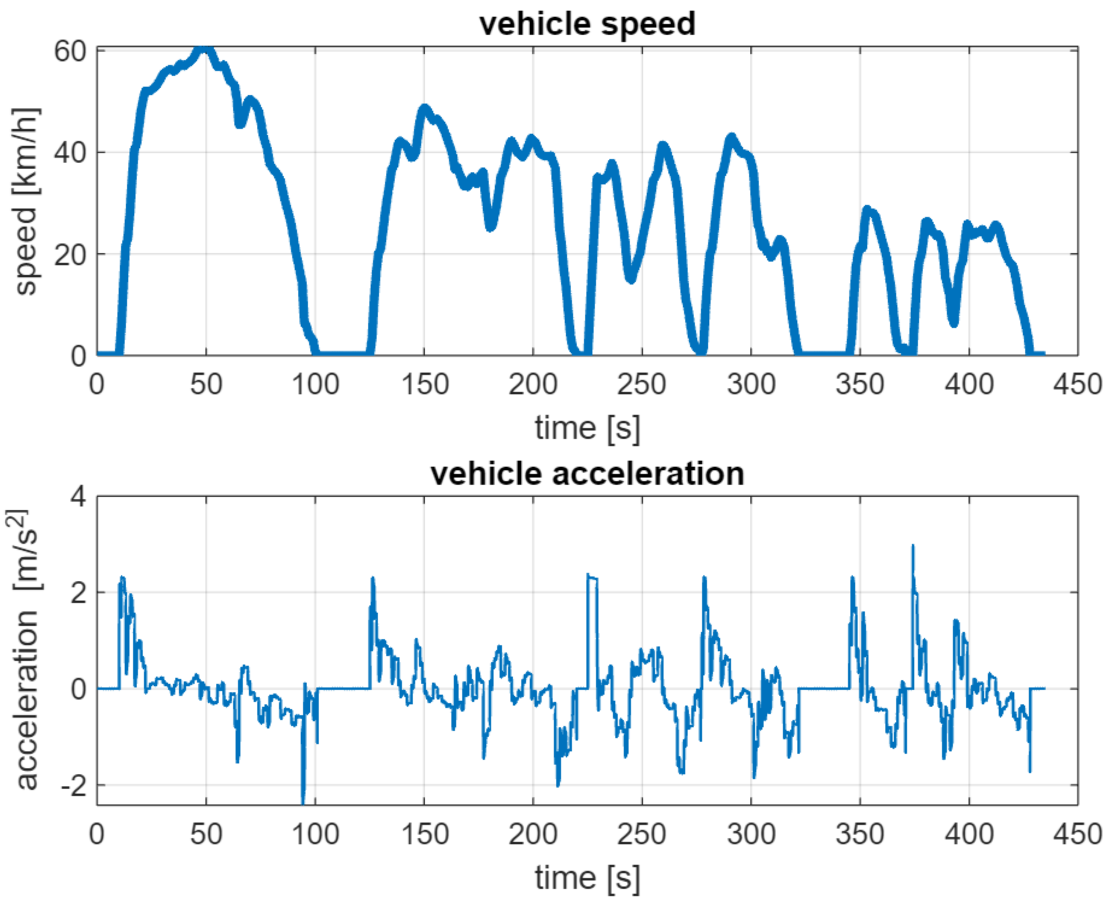


Figure 4.2: Speed and acceleration of driving cycle

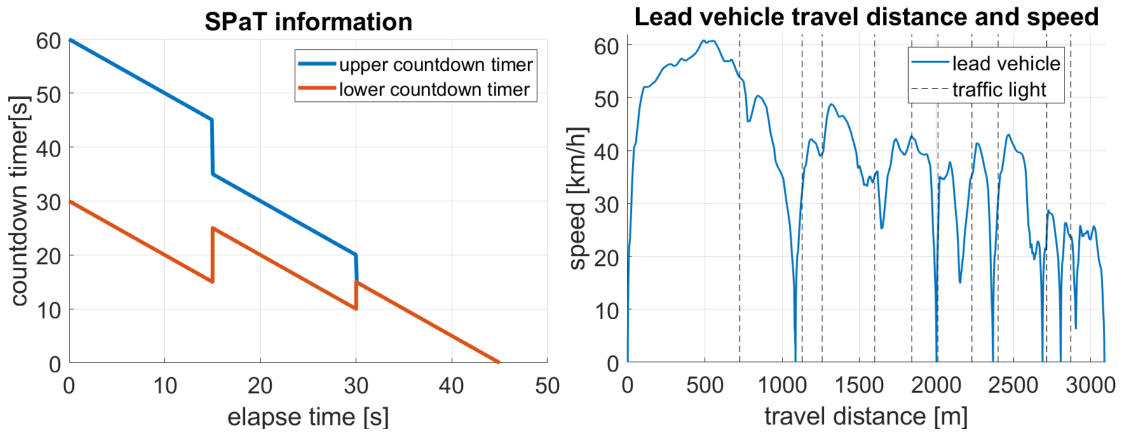


Figure 4.3: SPaT information and driving cycle

4.2 Tuning

The weight factor will significantly affect the performance of the controller. An incorrect combination of weight factors can lead to very poor solutions. The weight factor are tuned by Monte Carlo simulation.

Monte Carlo simulation is a statistical method that uses a large number of random samples to perform numerical calculations, commonly used to solve complex mathematical and physical problems. By generating random numbers to simulate possible states of a system, it then statistically analyzes these states to estimate the behavior and characteristics of the system. This method is widely applied in fields such as financial engineering, physical simulations, and computer graphics, valued for its ability to handle complex and nonlinear problems. However, Monte Carlo simulation has high computational costs, and the accuracy of the results depends on the number of samples.

MATLAB provides the function for the Monte Carlo method. n_s is the number of samples, and n_p is the number of parameters. The vector will be equally divided into n_s intervals between 0 and 1, and a value is randomly generated in each interval. This ensures that the samples are evenly distributed. The initial setting range of these weight factors are shown in equation, the w_s should use much higher order than others, the concrete value is not important. So it is set to 10^8.

$$\begin{gathered} S = \text{lhsdesign}(n_s, n_p) \\ \begin{bmatrix} p_j \\ p \\ \bar{p}_j \end{bmatrix} = \begin{bmatrix} 10^{-5} & 10^{-2} & 10^{-2} & 10^{0} & 10^{0} \\ w_p & w_a & w_j & w_v & w_L \\ 10^{2} & 10^{6} & 10^{6} & 10^{6} & 10^{6} \end{bmatrix} \end{gathered} \tag{4.1}$$

The test scenario is a simple one shown as figure 4.4, then generate 200 combinations of weight factor, run them by QP controller and get the lead vehicle trajectory by v2v, with vehicle following logic in this scenario and get the result.

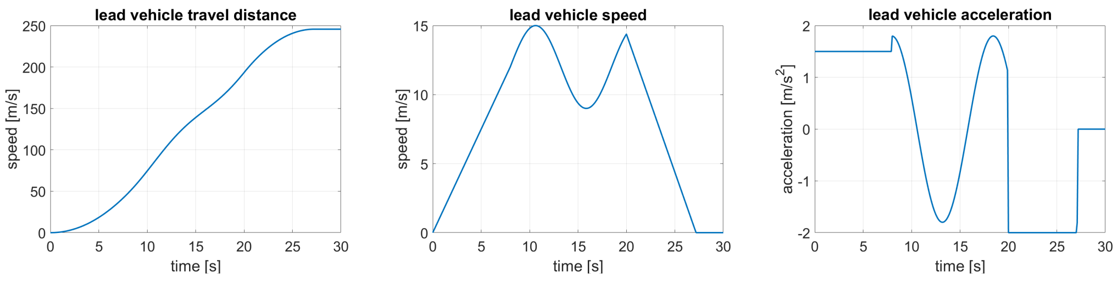


Figure 4.4: Scenario used in tuning

Then, we obtain the KPIs from these results. The KPIs include the RMS of acceleration, RMS of jerk, energy consumption, and RMS headway distance error.

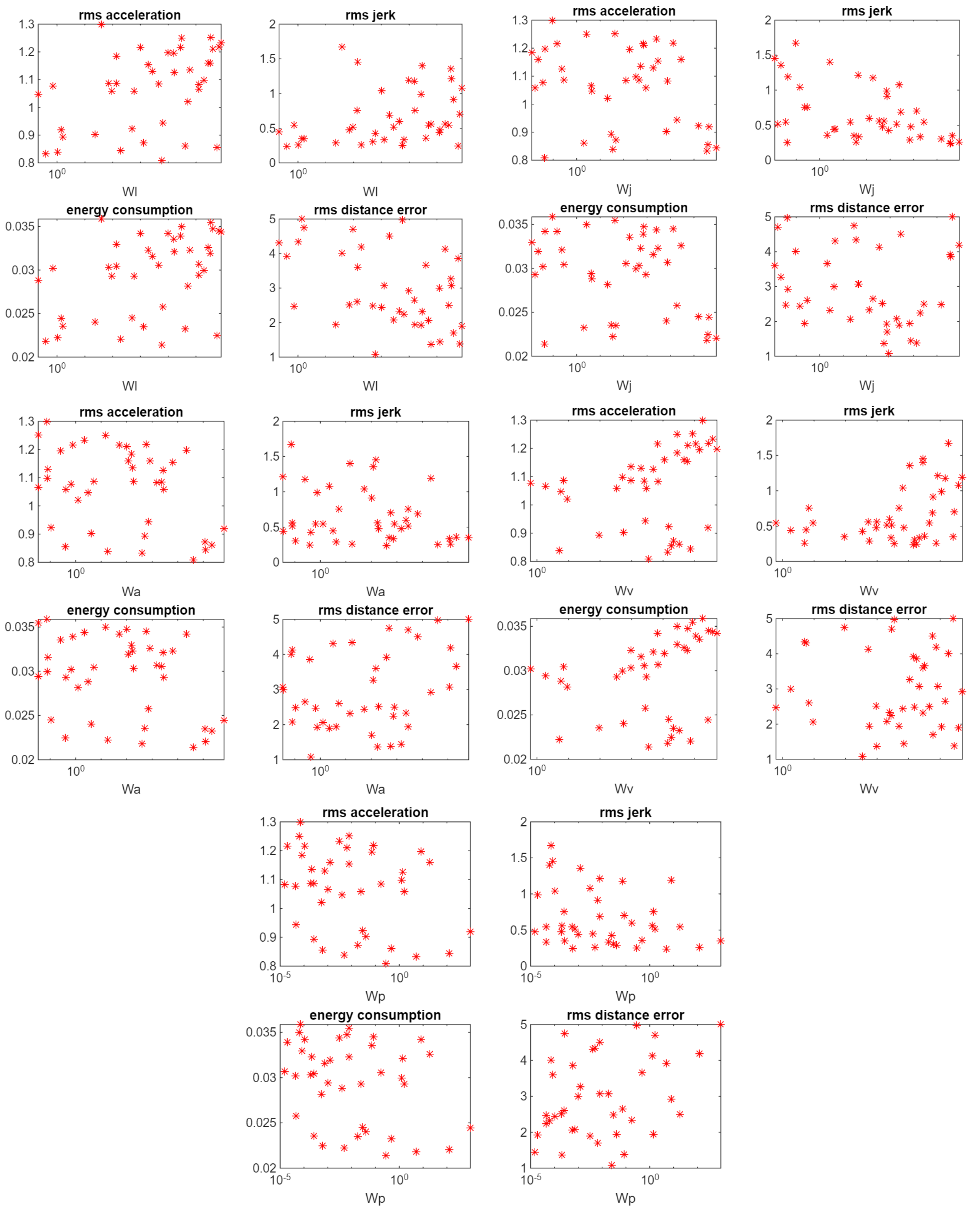


Figure 4.5: Tuning result

Some of these weight factor combinations may completely fail, resulting in a very high RMS distance error. Therefore, a threshold is implemented for the RMS distance error. Any combinations with an RMS distance error higher than this threshold will be considered useless and will be discarded.

After obtaining these results, build another rough cost function as shown in Equation 4.2. This cost function includes all the KPIs and uses weight factors to bring these costs to the same order of magnitude, then choose the combination with the lowest cost function. The results are shown below; this is our first assumption of the weight factors.

$$J = a_{rms} + j_{rms} + d_{rms} + 100P_{bat} \tag{4.2}$$

- $W_p = 0.0253\ (1/w)$ $\quad W_a = 4.2\ (s^2/m)$
- $W_j = 899.21\ (s^3/m)$ $\quad W_v = 337.9\ (s/m)$
- $W_L = 607\ (1/m)$ $\quad W_s = 10^8$

The previous step is just a rough test; the results cannot be directly used. They are only to ensure the order of the weight factors. In the next step, the controller should be test in driving cycle shown in the previous section and manually tune the parameters. Each time, tune just one factor while fixing the others. This process is repeated until obtain the final solution. w_a is not very sensitive; a large weight factor can lead to lower energy consumption. w_p should not be too large or too small. A small value will not make full use of the power map, while a too large value will lead to local minima in each time step calculation. In such cases, the controller will try to decrease speed and relax the headway distance to save energy, but in the next time step calculation, it will have to accelerate to make up for previous decisions. A smaller w_L can give the vehicle more flexibility to change the headway distance and decide on a more efficient working point of the motor to save energy. However, a too low value will reduce the headway distance and result in hard braking when the ego vehicle is approaching the lead vehicle from a far distance at high speed. w_j should be high to avoid high jerk.

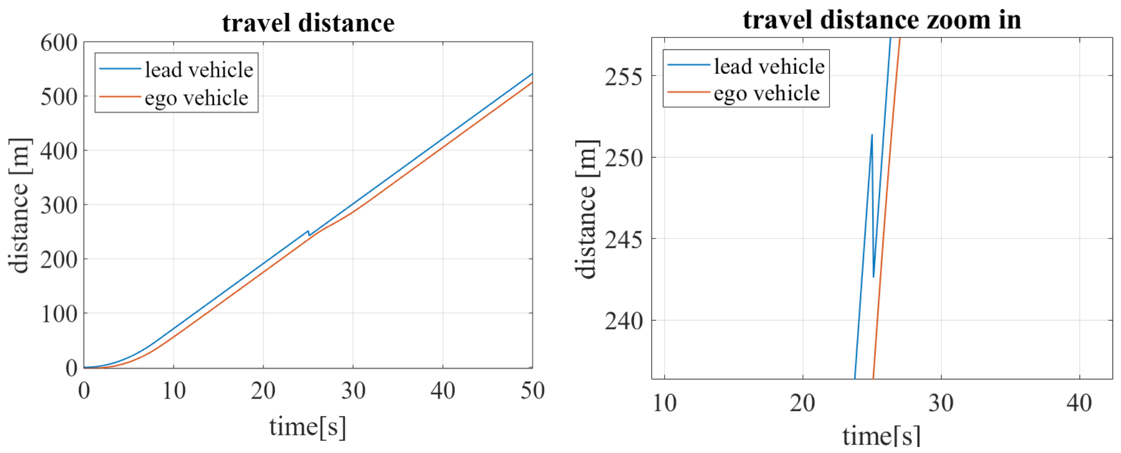
- $W_p = 0.5\ (1/w)$ $\quad W_a = 3000\ (s^2/m)$
- $W_j = 10000\ (s^3/m)$ $\quad W_v = 4000\ (s/m)$
- $W_L = 4000\ (1/m)$ $\quad W_s = 10^8$

At last, open the SPaT signal, and let the vehicle get the lead vehicle trajectory by estimation. The performance is change a lot, We consider that for further prediction steps, the lead vehicle's trajectory information becomes less precise. Additionally, the SPaT logic reference is generated based on the current vehicle state and does not account for conditions within the prediction horizon. Therefore, we decided to decrease the weight factor of speed tracking to 1000 s/m in the second 10 steps. After testing, we found that this adjustment improves the controller's performance.

4.3 Cut-in test

Adaptive Cruise Control needs to evaluate its performance in response to other vehicles cutting in because this situation frequently occurs in real-world driving. When another vehicle suddenly moves into the lane, it alters the driving environment, affecting the current vehicle's speed and following distance. To ensure safety and comfort, ACC must quickly and accurately respond to these sudden changes by adjusting speed and maintaining a safe following distance to avoid collisions or abrupt braking. Thus, assessing ACC's performance in handling cut-in scenarios verifies its reliability and effectiveness in complex driving conditions, ultimately enhancing overall driving safety.

A simple scenario is used to test the reaction to a cut-in. The lead vehicle trajectory are base on estimation of constant acceleration. The lead vehicle accelerates to 13 m/s with a constant acceleration of 1.5 m/s^2, then maintains a constant speed. The ego vehicle keeps a distance of about 16 meters from the lead vehicle. At 25 seconds, a cut-in occurs: another vehicle cuts in 6 meters in front of the ego vehicle and replaces the lead vehicle. The ego vehicle quickly decelerates to increase the headway distance. After that, it accelerates to recover the speed. Throughout the entire process, the acceleration and jerk remain within ± 2 m/s^2, ensuring comfort.



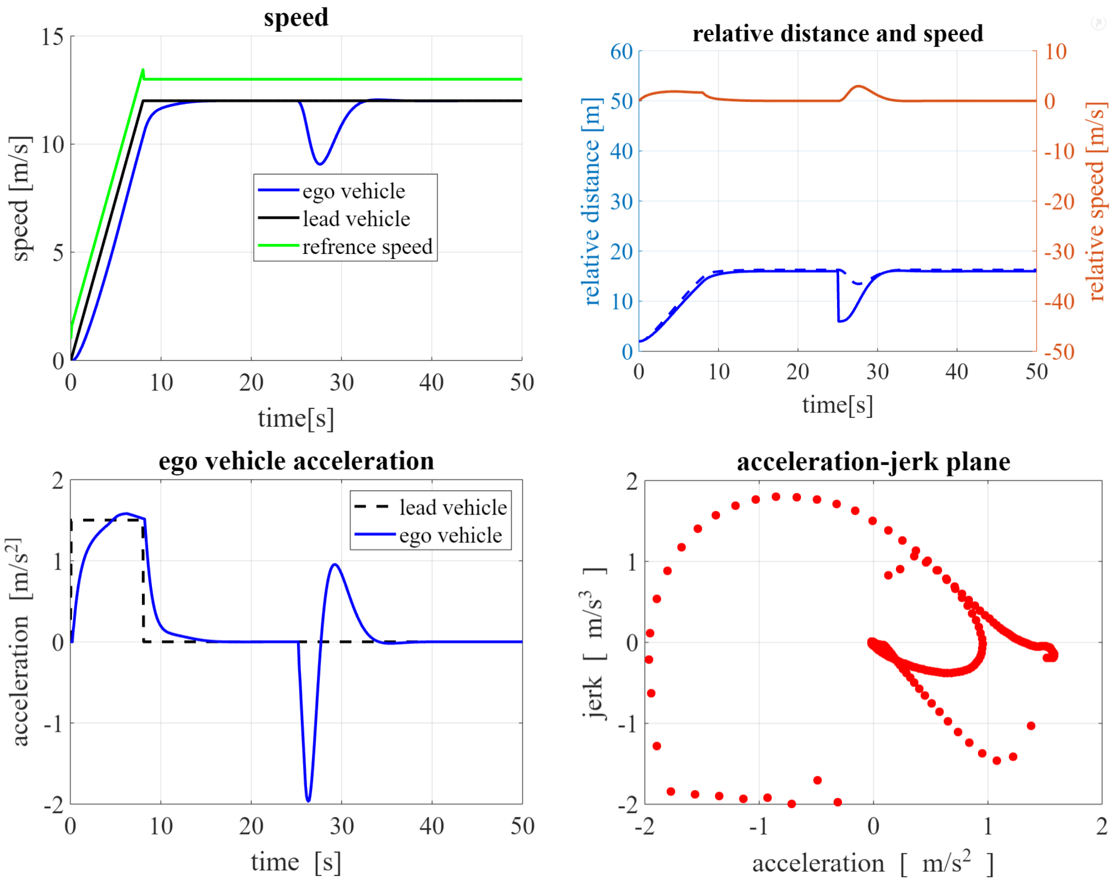


Figure 4.6: Result of cut-in test

4.4 Scenario test

After the simple test, we turn to a more complex test scenario discussed in Section 4.1. In this section, we always use the QP-based controller and obtain the lead vehicle trajectory in the prediction horizon through V2V communication. Our controller can handle different tasks, so we create four different scenarios. The first three use the same lead vehicle trajectory but change the SPaT information, while the last one is a condition where there is no lead vehicle, and the controller only performs traffic light negotiation.

- ACC with SPaT
- ACC with SPaT and stop
- Pure ACC
- Only traffic light negotiation

4.4.1 ACC with SPaT

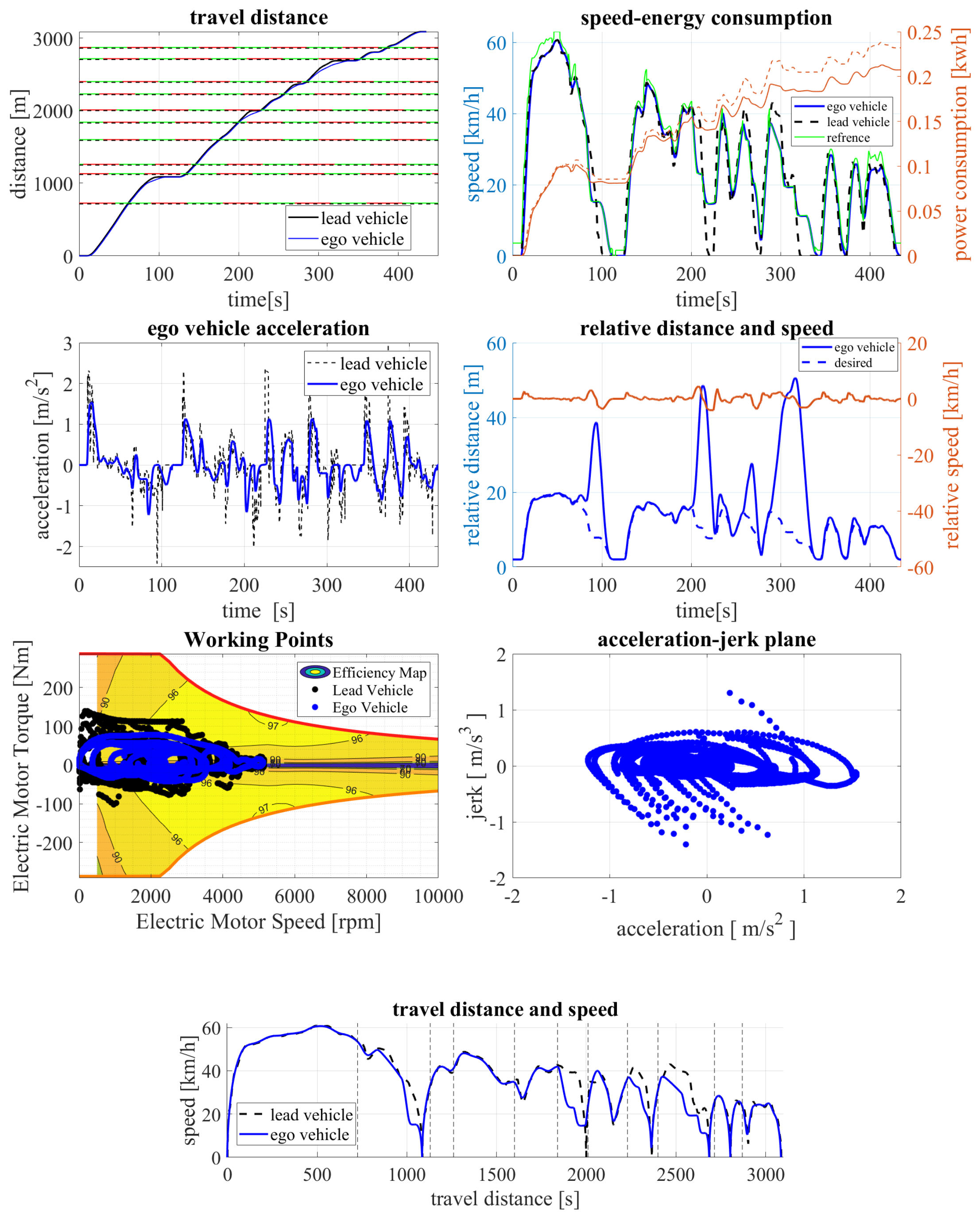


Figure 4.7: Result in scenario 'ACC with SPaT'

The first scenario is shown in Figure 4.7. In this scenario, the vehicle can receive SPaT information 300 meters before the traffic light. The vehicle can maintain the desired headway distance or use the SPaT information to plan a speed profile to avoid stopping and save energy.

From 0 to 43 s, and after 387 s, the vehicle is either far from the traffic light or has already passed it. No SPaT information is available in these periods, and the ego vehicle executes only the ACC task.

From 50 seconds to 120 seconds and from 287 seconds to 340 seconds, the ego vehicle decelerates earlier than the lead vehicle and maintains a constant cruising speed to avoid stopping. The multiple speed steps are due to sudden changes in the SPaT back counter information. However, since the lead vehicle still exists and does not stop exactly at the stop line, the ego vehicle still has to stop. Nonetheless, the ego speed profile benefits from energy savings because it reduces high-speed travel time. If the lead vehicle stops just before the stop line and accelerates quickly when the traffic light turns green, it is possible to avoid stopping.

Table 4.2 shows the KPIs between the lead vehicle and the ego vehicle. Compared to the lead vehicle, the maximum, minimum, and root mean square of acceleration and jerk are significantly decreased, providing a more comfortable driving experience. Some high jerk values beyond 1 m/s^3 occur in the inflection points between the deceleration and the constant cruising phase

Table 4.2: KPIs in scenario 'ACC with SPaT'

	lead vehicle	ego vehicle
Travel time [s]	435	435
Rms acceleration [m/s^2]	0.677	0.494
Max acceleration [m/s^2]	2.701	1.560
Min acceleration [m/s^2]	-2.762	-1.240
Rms jerk [m/s^3]	0.857	0.185
Max jerk [m/s^3]	8.965	1.356
Min jerk [m/s^3]	-5.758	-1.399
Energy consumption(wh)	232.1	215.6
Energy save		10.61%

4.4.2 ACC with SPaT and stop

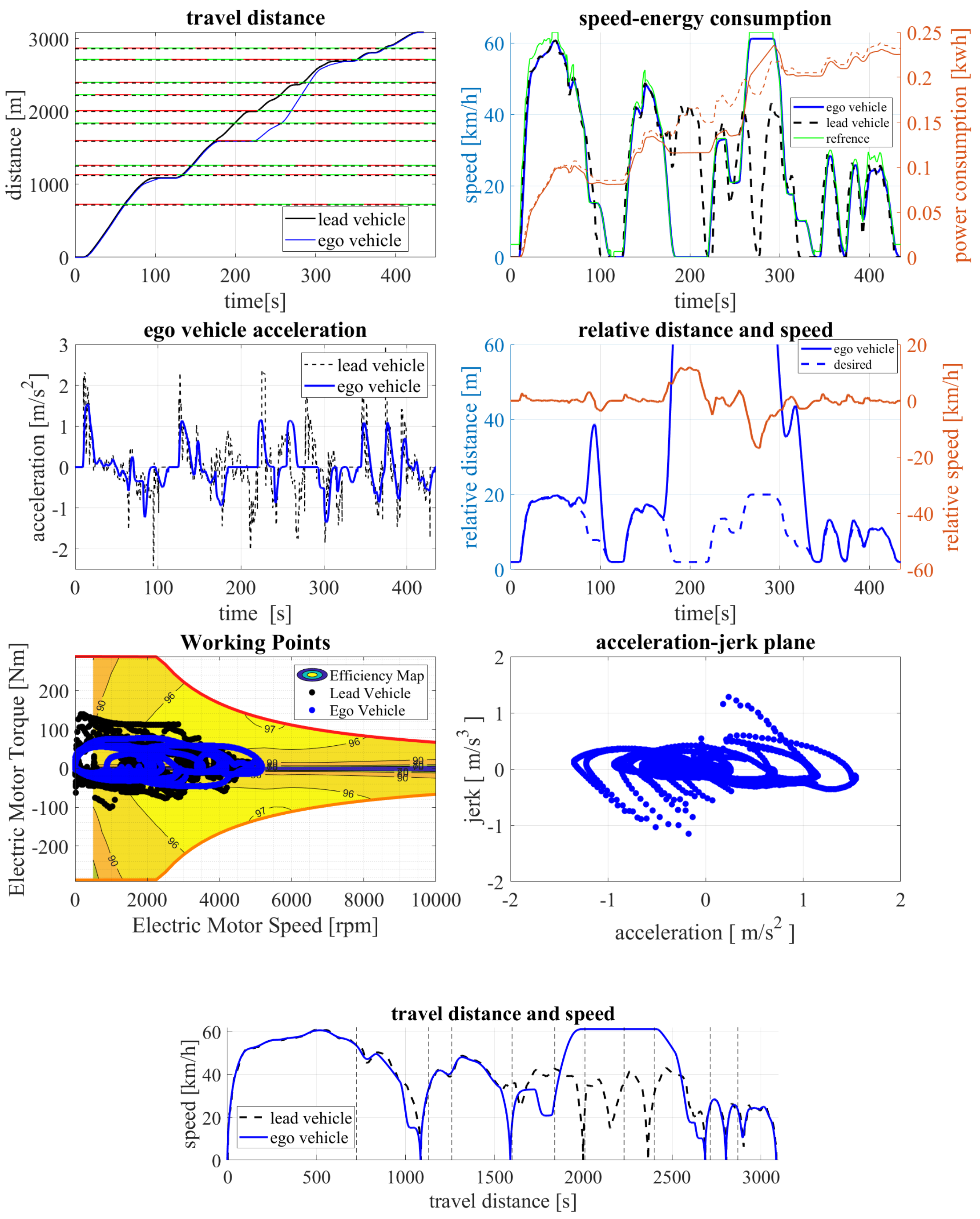


Figure 4.8: Result in scenario 'ACC with SPaT and stop'

The second scenario is similar to the first scenario. only change the 4th traffic light phase offset, creating a condition where the lead vehicle passes the traffic light at the end of the green phase, while the ego vehicle stops before the traffic light because the phase has already switched to red, and it has to wait for the entire red phase.

Figure 4.8 shows the simulation results. From 0 to 43 s, and after 387 s, the vehicle is either far from the traffic light or has already passed it. No SPaT information is available in these periods, and the ego vehicle executes only the ACC task.

At 170 s, the ego vehicle stops due to the traffic light turning red, while the lead vehicle passes in the last second of the green phase. Until 302 s, the lead vehicle is far ahead of the ego vehicle: during this period, the controller has no lead vehicle information. The ego vehicle follows only the SPaT to plan its speed, attempting to pass the intersections as soon as possible. Then, at 302 s, it catches up with the lead vehicle again.

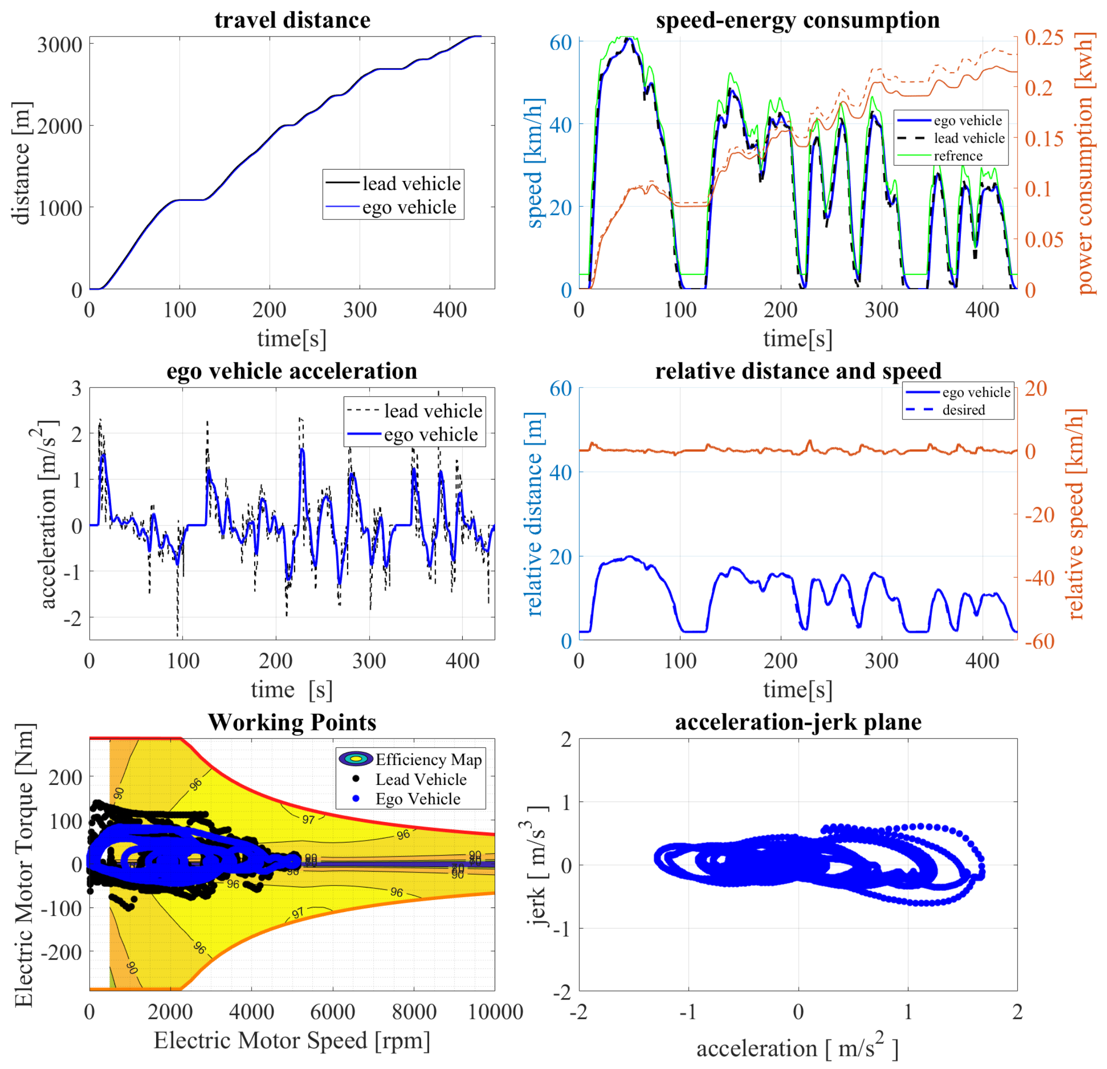
In the other cases, the ego vehicle follows the lead vehicle with SPaT information. From 82 to 112 s, and from 306 to 340 s, the ego vehicle decelerates earlier than the lead vehicle and maintains a constant cruising speed to avoid stopping.

The table 4.3 shows the overall KPIs of the total driving cycle. Compared to the lead vehicle, the maximum, minimum, and root mean square of acceleration and jerk are significantly decreased, providing a more comfortable driving experience. Before stopping due to the traffic light, the ego vehicle saves approximately 6 percent energy compared to the lead vehicle. However, because the ego vehicle maintains a higher speed to catch up with the lead vehicle, this leads to higher energy consumption. overall energy saving is not significant, at about 2.90 percent.

Table 4.3: KPIs in scenario 'ACC with SPaT and stop'

	lead vehicle	ego vehicle
Travel time [s]	435	435
Rms acceleration [m/s^2]	0.677	0.494
Max acceleration [m/s^2]	2.701	1.543
Min acceleration [m/s^2]	-2.762	-1.350
Rms jerk [m/s^2]	0.857	0.158
Max jerk [m/s^2]	8.965	1.287
Min jerk [m/s^2]	-5.758	-1.144
Energy consumption [wh]	232.1	225.4
Energy saving		2.9%

4.4.3 Pure ACC



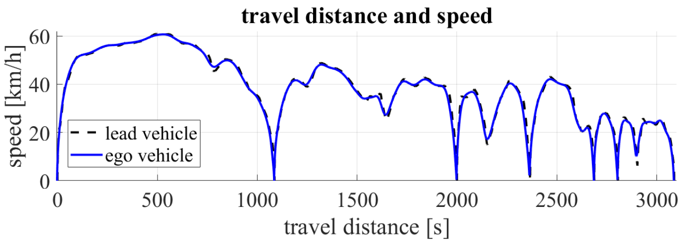


Figure 4.9: Result in scenario 'pure ACC'

The third scenario 4.9 is the condition that there is no SPaT information, the ego vehicle just do the vehicle following.

Table 4.4 shows the KPIs. Again, the maximum, minimum, and root mean square of acceleration and jerk are significantly decreased, providing a more comfortable driving experience. The energy saving becomes 7.43%, which is lower than the scenario with SPaT. SPaT information truly saves more energy. The jerk is always below ±0.6, which is lower than in the previous scenario with SPaT.

Table 4.4: KPIs in scenario 'pure ACC'

	lead vehicle	ego vehicle
Travel time $[s]$	435	435
Rms acceleration $[m/s^2]$	0.677	0.530
Max acceleration $[m/s^2]$	2.701	1.674
Min acceleration $[m/s^2]$	-2.762	-1.288
Rms jerk $[m/s^2]$	0.857	0.158
Max jerk $[m/s^2]$	8.965	0.606
Min jerk $[m/s^2]$	-5.758	-0.611
Energy consumption $[wh]$	232.1	214.8
Energy saving	-	7.43%

4.4.4 Only traffic light negotiation

In the last scenario figure 4.10, there is no lead vehicle, just ego vehicle pass the road section with SPaT information alone.

Table 4.5: KPIs in scenario 'only traffic light negotiation'

	human driver	ego vehicle
Travel time $[s]$	435	289
Rms acceleration $[m/s^2]$	0.677	0.432
Max acceleration $[m/s^2]$	2.701	1.149
Min acceleration $[m/s^2]$	-2.762	-1.352
Rms jerk $[m/s^2]$	0.857	0.167
Max jerk $[m/s^2]$	8.965	1.352
Min jerk $[m/s^2]$	-5.758	-1.348
Energy consumption $[wh]$	232.1	245.4
Energy saving	-	-5.72%

Table 4.5 show KPIs, the ego vehicle totally avoid stoping in such scenario, the travel time is significantly decreased and also keep the comfortable. However, due

to higher average speed, the total resistance is increase, the energy consumption is a little higher than human driver.

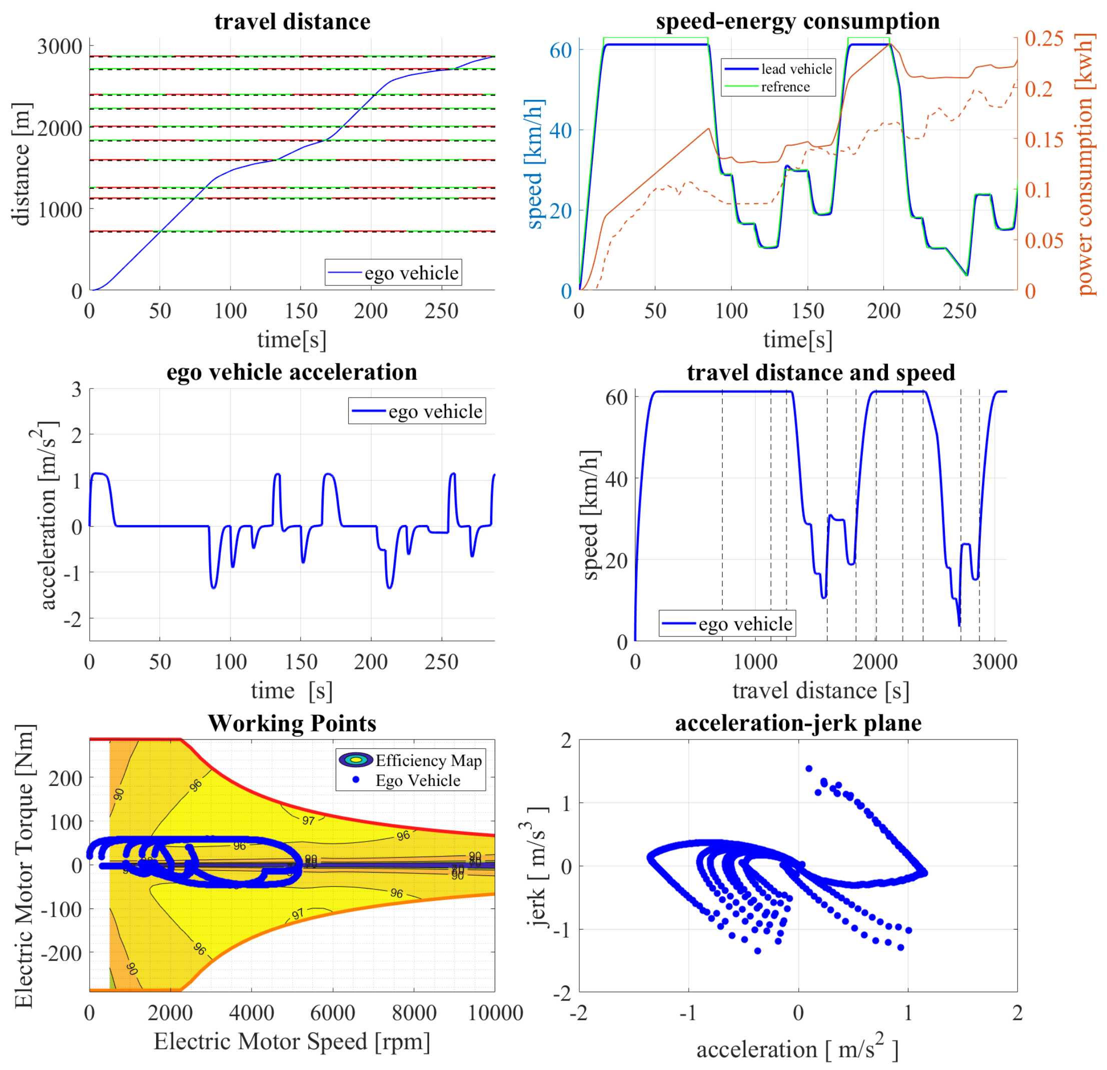


Figure 4.10: Result in scenario 'only traffic light negotiation'

4.5 Lead vehicle trajectory estimation

As discussed in Section 3.4, V2V information is not always available, so it has be estimated. In this section, we discuss how the performance of the controller will change if the lead vehicle trajectory is obtained through estimation. The test scenario is 'ACC with SPaT', with QP base controller.

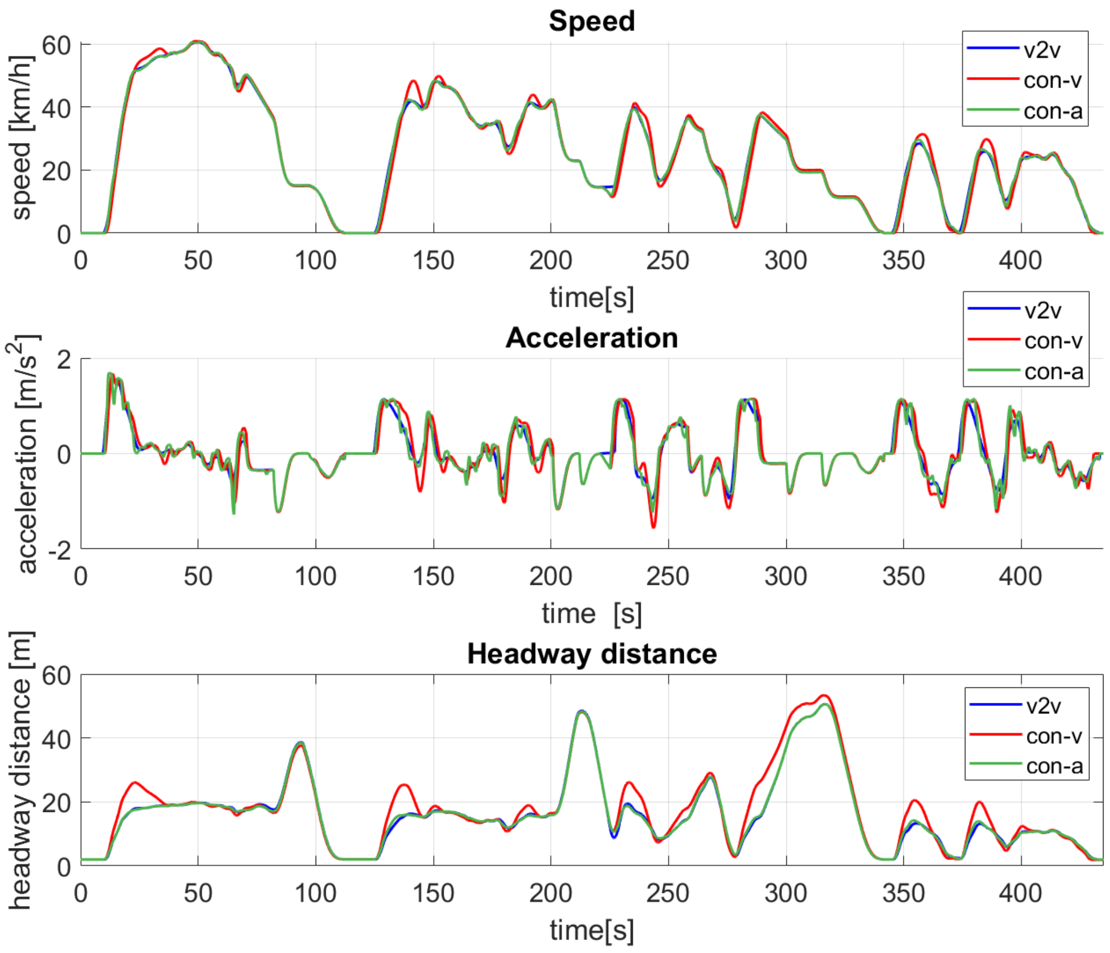


Figure 4.11: Comparison among different lead vehicle trajectory estimation

Figure 4.11 shows the result. For the speed profile, the V2V and con-a are similar, while con-v has some speed overshoot at the end of acceleration. Around 230 s, both con-a and con-v decrease the speed slightly, but this does not happen in V2V. This is because when estimating the lead vehicle trajectory, it is hard to know when the lead vehicle will start to move. In the prediction horizon, the lead vehicle is always seen as stationary, which leads the ego vehicle to decrease speed.

For the acceleration profile, con-a has more oscillation compared to the others. For the headway distance profile, at the end of acceleration, con-v reacts slowly and leads to a higher headway distance than the others.

Table 4.6: KPIs of different lead vehicle trajectory estimation

	v2v	con-v	con-a
Travel time $[s]$	435	435	435
Rms acceleration $[m/s^2]$	0.492	0.561	0.522
Max acceleration $[m/s^2]$	1.543	1.672	1.694
Min acceleration $[m/s^2]$	-1.224	-1.565	-1.278
Rms jerk $[m/s^2]$	0.182	0.230	0.264
Max jerk $[m/s^2]$	1.300	0.906	1.647
Min jerk $[m/s^2]$	-1.403	-1.565	-1.372
Energy consumption $[wh]$	207.5	223.1	213.0
	-10.61%	-3.88%	-8.22%

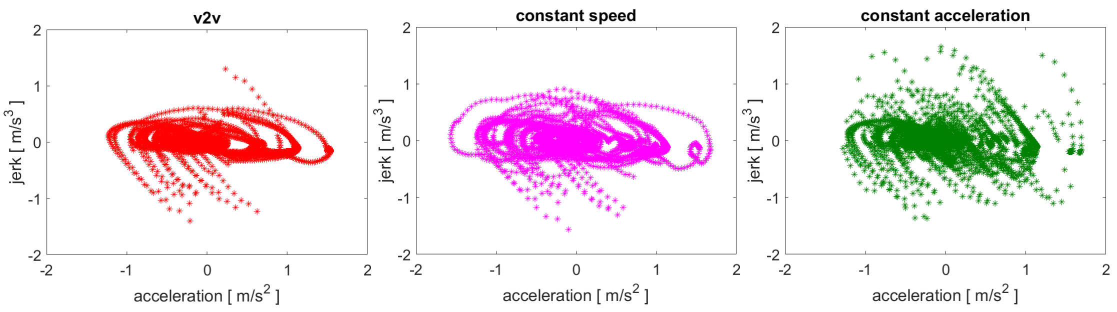


Figure 4.12: Acceleration-jerk plan of different lead vehicle trajectory estimation

Table 4.6 and figure 4.12 shows the KPIs of v2v and two estimation method, Both con-v and con-a yield reduced performance, with higher acceleration, jerk, and energy consumption. The constant speed method significantly increases the energy consumed and tracking error but produces a lower jerk when compared to con-a. Although the root mean square of acceleration is slightly higher than con-a, but the acceleration profile is smoother. Therefore, we consider that con-v is more comfortable than con-a, but con-a can save more energy and provides better tracking, a trade-off between comfort and energy reduction should be made, while the benefits of leveraging V2V remain evident. Allowing a precise prediction of the lead vehicle's speed is a very active area of research, where machine learning algorithms and probabilistic approaches are emerging as concurrents of V2V.

4.6 Headway distance policy

Until now, the controller is always work with human driver behavior policy. In this section, we discuss the performance of different headway distance policies. Figure 4.13 shows these policy.

- Constant time gap(CTG):

$$h = A + \tau v \tag{4.3}$$

  Where h is the headway distance, τ is time distance, it is set as $1.5s$. v is current vehicle speed,A is the constant distance when the vehicle is static

- Human driver behavior(HDB):

$$h = A + Tv + Gv^2 \tag{4.4}$$

  This policy is the rewrite of HDB, increase the headway distance by change the HDB parameter:

  $A = 2$, $T = 1.5$, $G = -0.0246T + 0.010819$.

- Rebuild human driver behavior(rHDB):

$$h = A + Tv + Gv^2 \tag{4.5}$$

  Where the parameters of the rHDB are set as follows:

  $A = 2$, $T = 1.8846$, $G = -0.0226$.

- Saturation (ST) This policy is highly increase headway distance in low speed, then keep the constant headway distance.

$$h = max(3v{,}22) \tag{4.6}$$

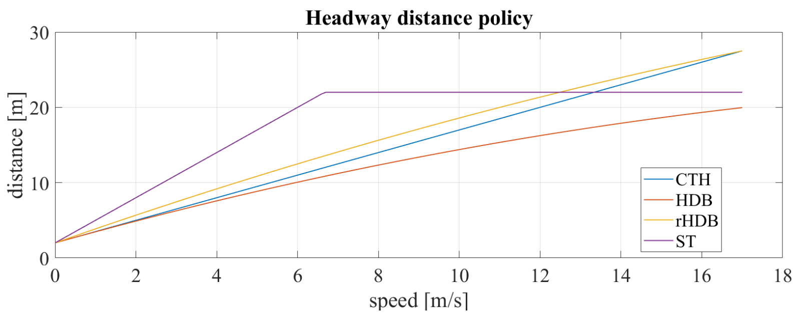


Figure 4.13: Tested distance policy

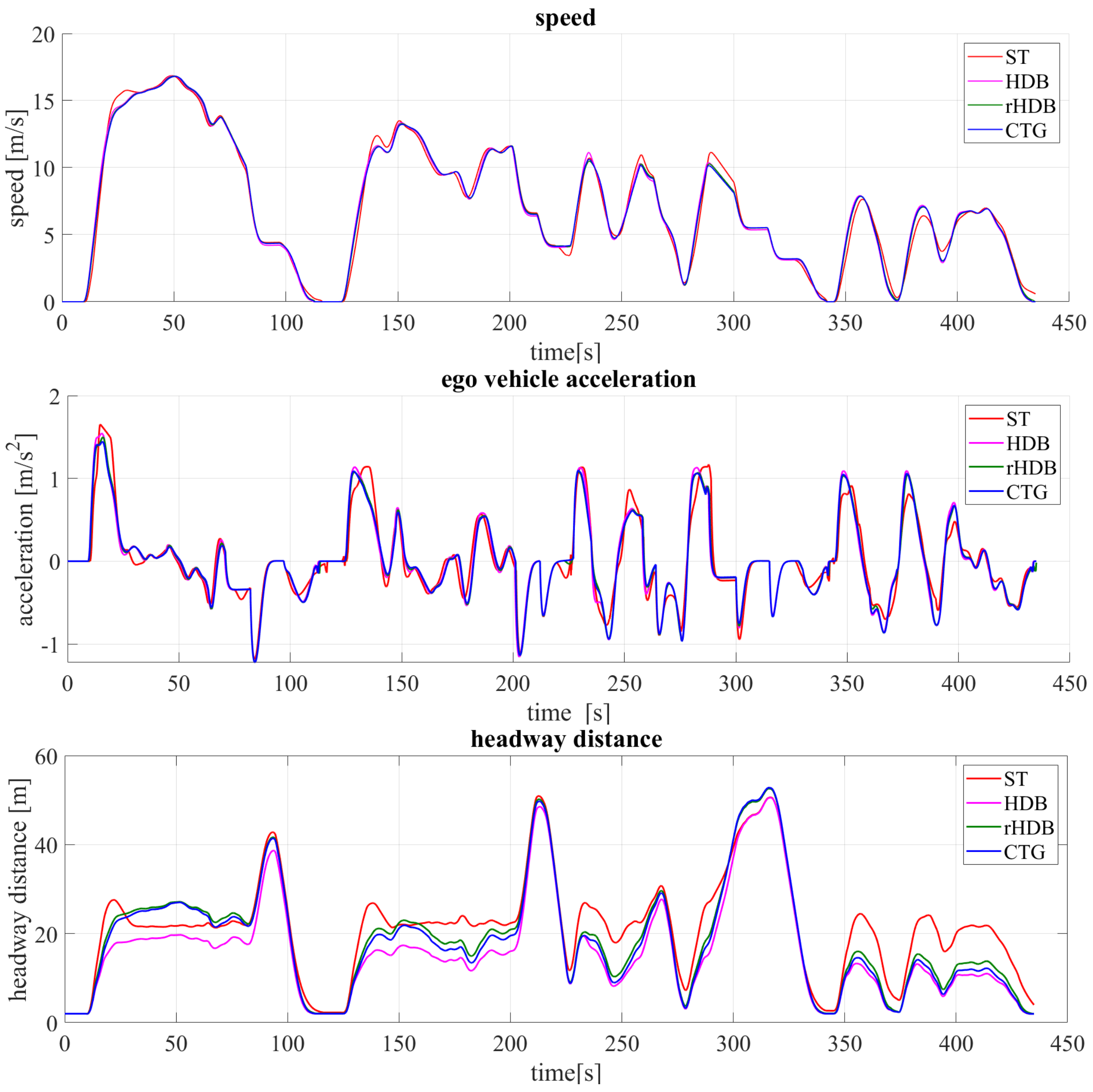


Figure 4.14: Comparison among different distance policy

The results are use QP-base controller, using v2v to get the true lead vehicle trajectory in prediction horizon and test in scenario 'ACC with SPaT', figure 4.6 shows the result and table 4.7 shows the KPIs.

CTG is the most powerful one; it can save the most energy and also has very low acceleration and jerk, which ensure comfort. However, it has a large headway distance during high speeds, increasing the possibility of cut-ins. ST has the highest values in terms of acceleration and the lowest energy savings, and it also shows an overshoot of headway distance at the end of acceleration.

Table 4.7: KPIs of different distance policy

	HDB	rHDB	CTG	ST
Rms acceleration $[m/s^2]$	0.492	0.478	0.477	0.485
Max acceleration $[m/s^2]$	1.543	1.499	1.446	1.652
Min acceleration $[m/s^2]$	-1.224	-1.222	-1.220	-1.186
Rms jerk $[m/s^2]$	0.182	0.174	0.175	0.174
Max jerk $[m/s^2]$	1.300	1.103	1.300	0.825
Min jerk $[m/s^2]$	-1.403	-1.374	-1.388	-1.502
Energy consumption $[wh]$	-10.61%	-11.66%	-11.89%	-10.03%

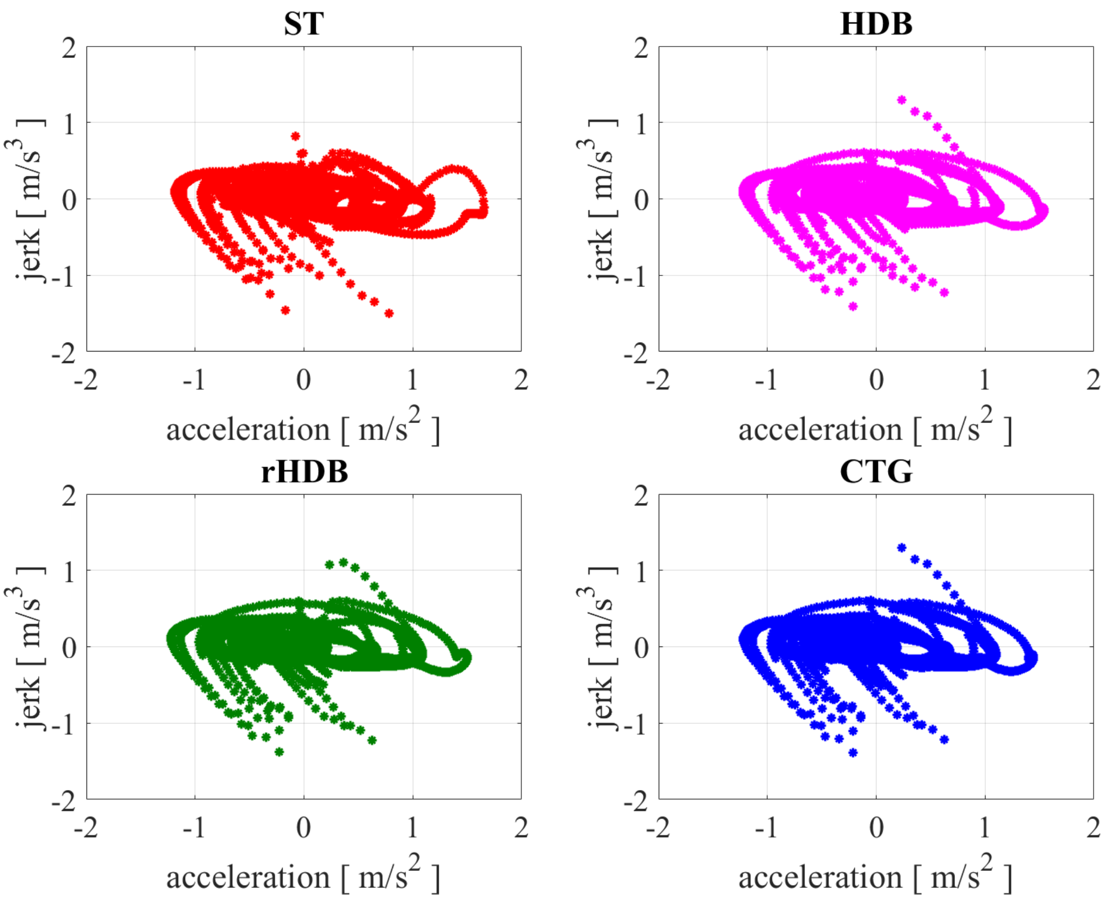


Figure 4.15: Acceleration-jerk plane by different distance policy

Therefore, it not a good choice. rHDB increases the headway distance based on HDB, which leads to less acceleration and jerk and also saves more energy compared to HDB. Thus, increasing the headway distance can provide better performance in both energy saving and comfort. However, rHDB still has more energy consumption than CTG, even though the headway distance of rHDB is always higher than that of CTG. Not only does headway distance affect energy consumption, but the negative slope decrease of the distance policy function also negatively impacts energy consumption.

4.7 NLP and QP

The MPC controller is implenmented in two form : NLP and QP, NLP is more precise and QP can realize real-time. In this section, we compare the different between them, using two scenario (ACC with SPaT and pure ACC) and two method of lead vehicle trajectory estimation (v2v and con-a). the tables shows the KPIs and figure 4.7 shows the headway distance distance error which means the offset between the real headway distance and desired headway distance.

In most condition, NPL can save more energy than QP, however there still exist the condition such as ACC with SPat and v2v that NPL use more energy.

NLP can always do more precise tracking. Lower rms , mean and variance of headway distance error. This is because in QP, the speed quadratic term are use the estimated value, but in NLP they are always true value, no accuracy speed estimation leads to less accuracy of distance tracing.

NLP has higher peak acceleration than QP, however it has lower max deceleration.

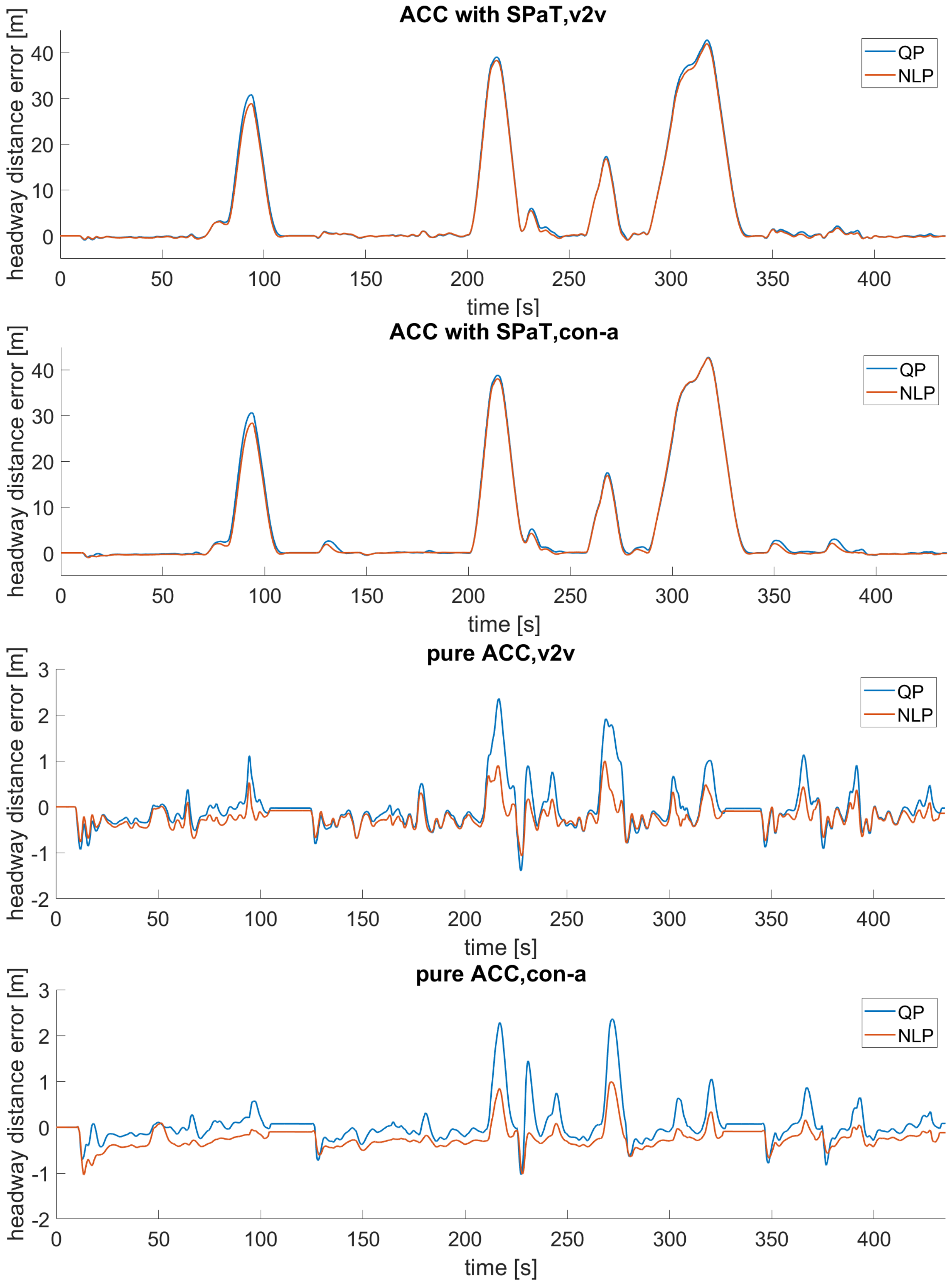
NLP has more benefit when V2V information is not available.

with SPaT,V2V		QP	NLP
headway distance error $[m]$	Var	122.2	115.4
	Rms	12.3	12.0
	Mean	5.6	5.4
Acceleration $[m/s^2]$	Var	0.492	0.496
	Max	1.543	1.551
	Min	-1.224	-1.146
Jerk $[m/s^3]$	Var	0.182	0.182
	Max	1.300	1.354
	Min	-1.403	-1.371
Energy save		10.61%	10.64%

with SPaT,cona		QP	NLP
headway distance error $[m]$	Var	121.1	118.3
	Rms	12.32	12.06
	Mean	5.67	5.42
Acceleration $[m/s^2]$	Var	0.522	0.521
	Max	1.694	1.926
	Min	-1.278	-1.198
Jerk $[m/s^3]$	Var	0.264	0.261
	Max	1.647	1.616
	Min	-1.372	-1.582
Energy save		8.22%	8.81%

pure ACC,v2v		QP	NLP
headway distance error $[m]$	Var	0.236	0.006
	Rms	0.487	0.330
	Mean	0.325	0.276
Acceleration $[m/s^2]$	Var	0.530	0.539
	Max	1.674	1.927
	Min	-1.288	-1.252
Jerk $[m/s^3]$	Var	0.158	0.166
	Max	0.606	0.765
	Min	-0.611	-0.724
Energy save		7.43%	7.13%

pure ACC,cona		QP	NLP
headway distance error $[m]$	Var	0.192	0.057
	Rms	0.441	0.337
	Mean	0.246	0.286
Acceleration $[m/s^2]$	Var	0.568	0.571
	Max	1.695	1.928
	Min	-1.753	-1.666
Jerk $[m/s^3]$	Var	0.313	0.296
	Max	1.840	1.727
	Min	-1.402	-1.582
Energy save		5.24%	5.35%


ACC with SPaT,v2v
headway distance error [m]
time [s]
QP
NLP
ACC with SPaT,con-a
headway distance error [m]
time [s]
QP
NLP
pure ACC,v2v
headway distance error [m]
time [s]
QP
NLP
pure ACC,con-a
headway distance error [m]
time [s]
QP
NLP

4.8 Multiple vehicle following

The multiple vehicle following behavior is a important indicator for ACC controller, in this section, we discuss the string stability of our controller, we use three vehicles follow each other, test the string stability when they use v2v, constant acceleration and constant speed method to predict lead vehicle trajectory.

The controller can benefit significantly when performing the multiple vehicle following task. The first vehicle can estimate the lead vehicle's trajectory within the prediction horizon, and then the MPC solution can be transmitted as V2V information. This information includes the first vehicle's future trajectory and is delivered to the second vehicle, then to the third, fourth vehicles, and so on, forming a data line.

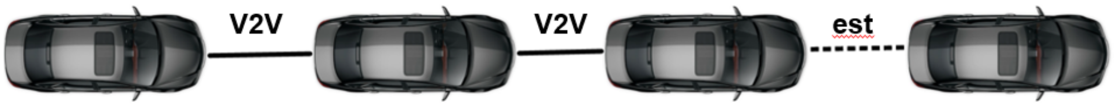


Figure 4.16: Multiple vehicle following by v2v

A simple scenario which the lead vehicle run in a sin wave speed profile show in figure 4.17 is used to test string stability, the ego vehicle only do the car following without SPaT information.

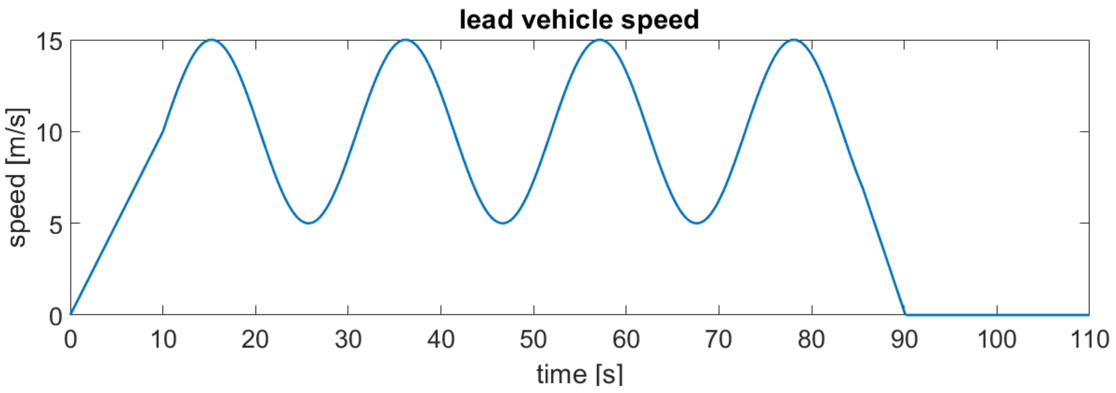


Figure 4.17: Multiple vehicle following tested scenario

4.8.1 All constant speed

The first condition is all the vehicle are estimate the lead vehicle trajectory by constant speed, figure 4.18 shows the graph of speed, acceleration and jerk of each vehicle. table 4.8 shows the KPIs.

This method shows strong string instability. The amplitude of the sinusoidal speed profile increases vehicle by vehicle, and acceleration, jerk, and energy consumption also become higher. This method is not suitable for multiple vehicle following.

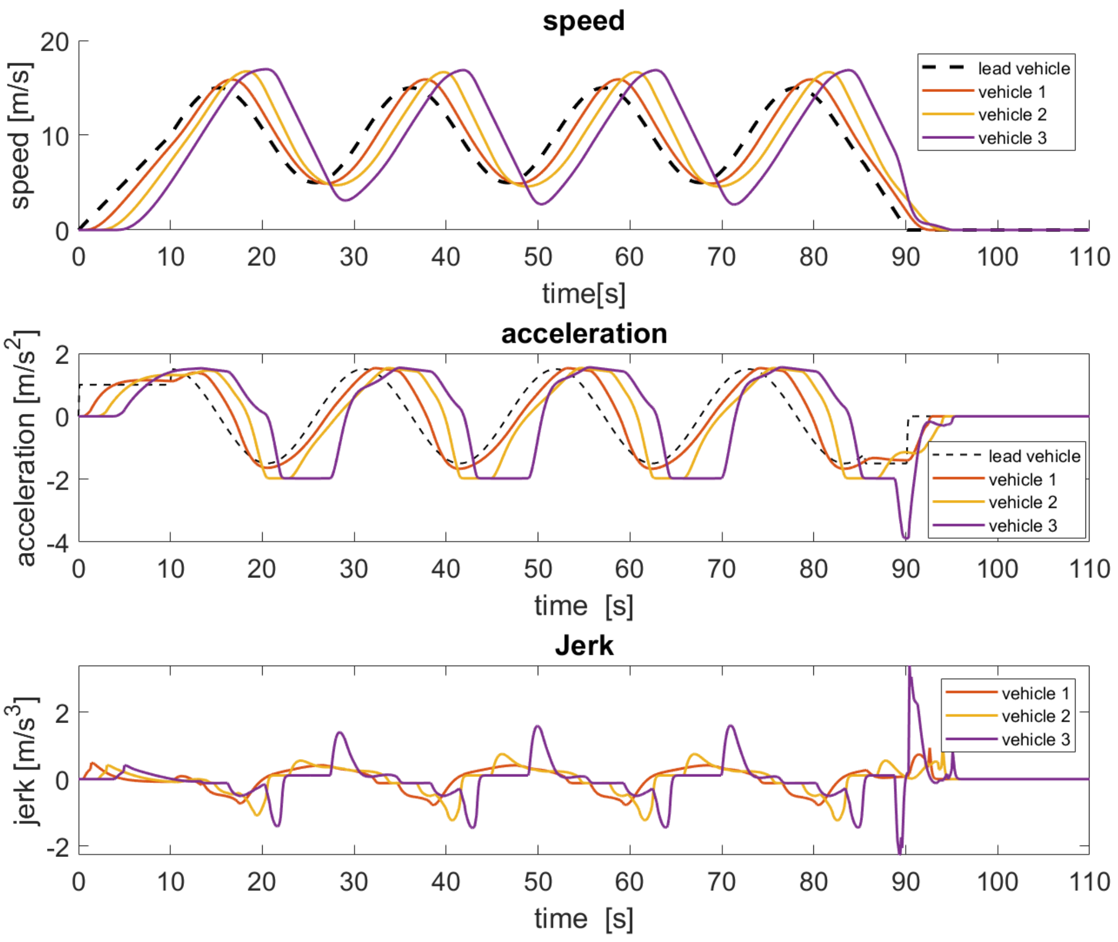


Figure 4.18: Result of method 'all by constant speed'

	vehicle 1	vehicle 2	vehicle 3
Rms acceleration $[m/s^2]$	1.056	1.168	1.318
Rms jerk $[m/s^3]$	0.315	0.385	0.555
Energy consumption $[wh]$	106	116	124

Table 4.8: KPIs of method 'all by constant speed'

4.8.2 All constant acceleration

The second condition is all the vehicle are estimate the lead vehicle trajectory by constant acceleration, figure 4.19 shows the graph of speed, acceleration and jerk of each vehicle. table 4.9 shows the KPIs

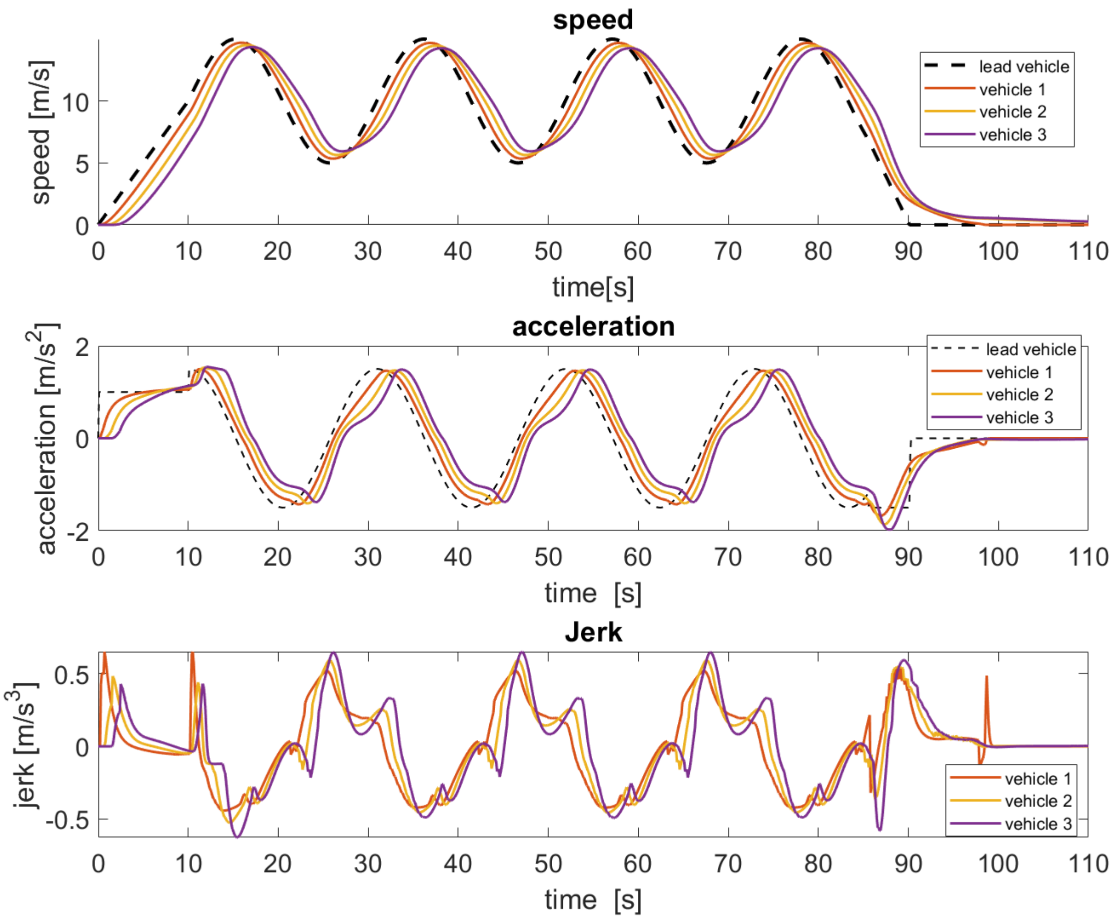


Figure 4.19: Result of method 'all by constant acceleration'

	vehicle 1	vehicle 2	vehicle 3
Rms acceleration $[m/s^2]$	0.924	0.897	0.874
Rms jerk $[m/s^3]$	0.267	0.268	0.277
Energy consumption $[wh]$	92	89	86

Table 4.9: KPIs of method 'all by constant acceleration'

This method demonstrates string stability. The amplitude of the sinusoidal speed profile decreases vehicle by vehicle, and both acceleration and power consumption decrease. However, jerk increases slightly but not significantly. After further testing, when the number of following vehicles increases, it will converge to a certain value.

4.8.3 Constant speed-v2v

The third condition is the first vehicle use constant speed estimation, the other vehicle are use v2v get the previous vehicle MPC solution as the lead vehicle trajectory, figure 4.20 shows the graph of speed, acceleration and jerk of each vehicle. table 4.10 shows the KPIs.

As discussed before, if all the vehicles use constant speed estimation, the vehicle flow will be string instability, but the v2v information can fix such behavior, the amplitude of the sinusoidal speed profile decreases vehicle by vehicle, and all of acceleration , jerk, energy consumption will decrease.

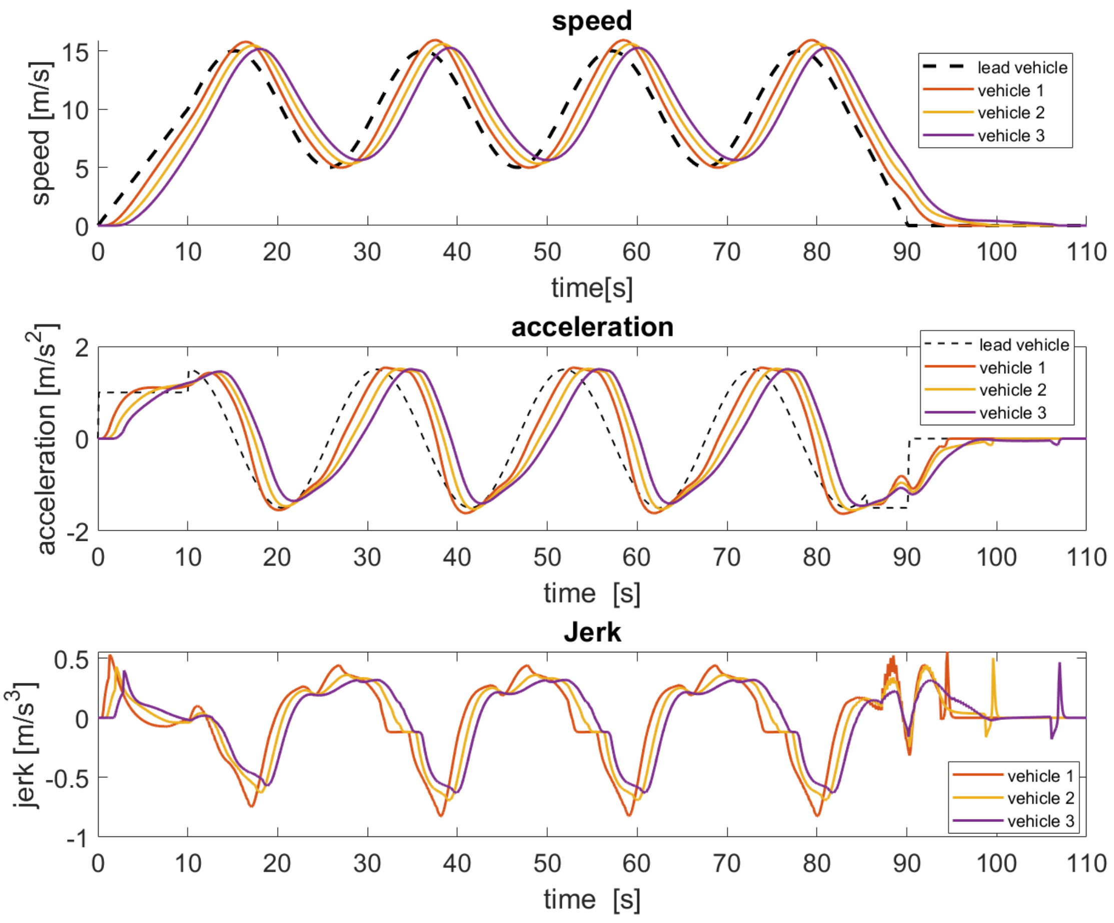


Figure 4.20: Result of method 'constant speed-v2v'

	vehicle 1	vehicle 2	vehicle 3
Rms acceleration $[m/s^2]$	1.038	0.991	0.946
Rms jerk $[m/s^3]$	0.315	0.294	0.275
Energy consumption $[wh]$	106	100	96

Table 4.10: KPIs of method 'constant speed-v2v'

4.8.4 Constant acceleration-v2v

The last condition is the first vehicle use constant acceleration estimation, the other vehicle are use v2v get the previous vehicle MPC solution as the lead vehicle trajectory, figure 4.21 shows the graph of speed, acceleration and jerk of each vehicle. table 4.11 shows the KPIs.

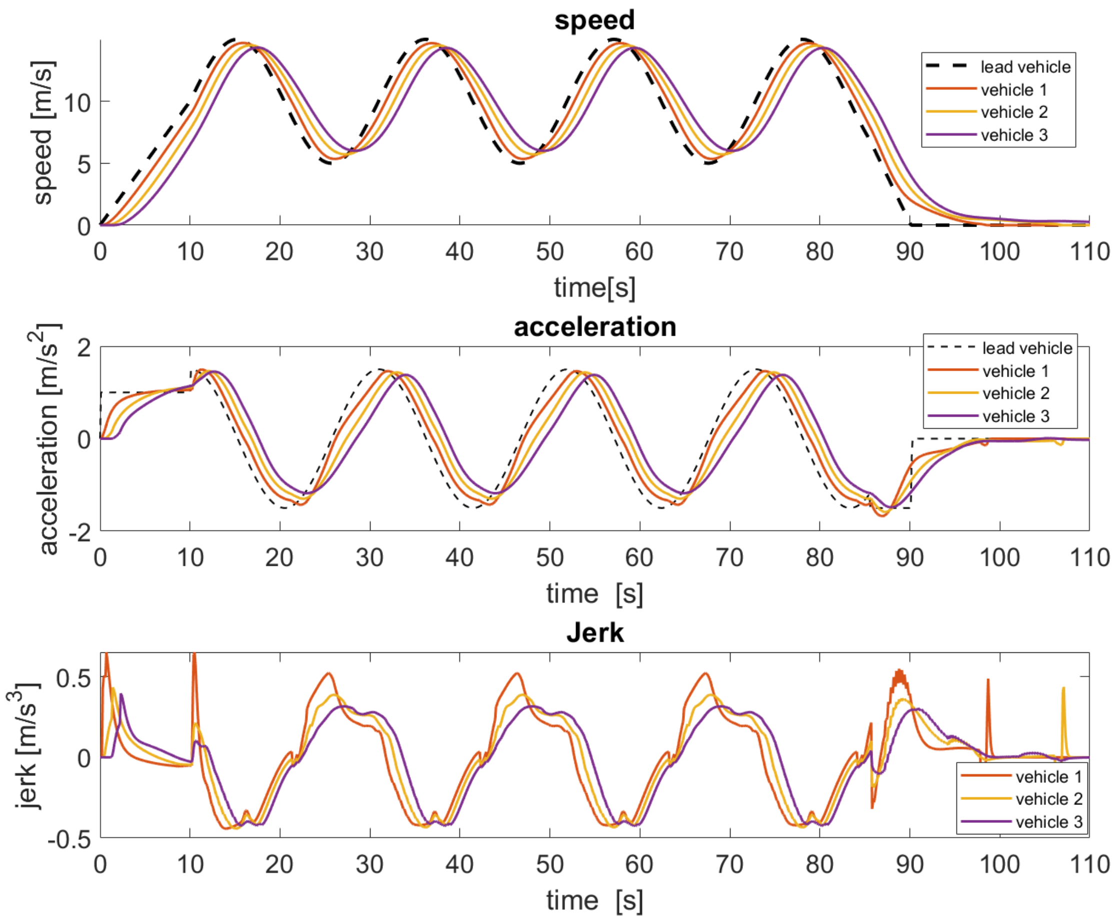


Figure 4.21: Result of method 'constant acceleration-v2v'

	vehicle 1	vehicle 2	vehicle 3
Rms acceleration $[m/s^2]$	0.924	0.883	0.846
Rms jerk $[m/s^3]$	0.268	0.246	0.231
Energy consumption $[wh]$	92	89	86

Table 4.11: KPIs of method 'constant acceleration-v2v'

The contribution of v2v information is similar to before. the vehicle flow is string stability, the amplitude of the sinusoidal speed profile decreases vehicle by vehicle, and all of acceleration , jerk, energy consumption will decrease.

Chapter 5

Conclusions and Future Works

5.1 Conclusion

This study presents an MPC-based vehicle controller that integrates Adaptive Cruise Control (ACC) and traffic light states to optimize energy consumption and enhance the driving experience of connected urban electric vehicles. The ACC dynamically adjusts the vehicle's speed based on the distance and speed of the preceding vehicle, ensuring safety and comfort while minimizing energy consumption. By leveraging V2I communication, the vehicle can anticipate traffic signal changes and adjust its speed proactively, reducing frequent acceleration and deceleration.

The structure of the controller includes a reference speed generator and an MPC controller, which comprises two architectures: Nonlinear Programming (NLP) and Quadratic Programming (QP).

Initially, a rough test using a Monte Carlo simulation was conducted to determine the order of weight factors. These weight factors were refined through manual tuning in a controlled driving cycle to achieve optimal performance. Cut-in reactions were analyzed to assess the controller's robustness and adaptability.

The controller was tested in various scenarios, and the results indicated that it performed well across different conditions. It significantly reduced energy consumption with or without SPaT information, with further reductions achieved when SPaT information was available.

The approach involved comparing the NLP and QP structures within the MPC framework, highlighting differences in performance, particularly in tracking accuracy and computational efficiency. The results demonstrated that the NLP-based controller offered more precise tracking, while QP exhibited significant computational efficiency, making it more suitable for real-time applications.

Different headway distance policies were also explored. Policies like CTG, although energy-efficient and comfortable, might increase the risk of cut-ins due to larger headway distances at high speeds. In contrast, policies like HDB provided a balance by offering better performance in both energy saving and comfort.

Challenges related to V2V communication unavailability were addressed, emphasizing the need for accurate lead vehicle trajectory estimation. The results indicated that while estimation methods could provide satisfactory performance, they were not as reliable as direct V2V information. The application of V2V in multi-vehicle tracking was further discussed, with results indicating that V2V information can enhance string stability.

5.2 Future work

The reference generator block in this paper is rule-based, but there is potential for algorithmic improvement. Future work will explore the use of machine learning techniques to achieve a more precise prediction of the lead vehicle's trajectory. Additionally, machine learning can be employed to make QP's results more similar to NLP's, further enhancing efficiency and accuracy. Further optimization of control algorithms will be pursued to improve the system's real-time performance and robustness. Expanding the scope of simulations to cover more diverse urban traffic conditions and scenarios is also important. Moreover, additional applications of V2I will be investigated to further enhance the intelligence and coordination capabilities of the vehicle controller.

List of Tables

List of Figures

Bibliography

[1] World Health Organization. *Ambient air pollution: A global assessment of exposure and burden of disease*. World Health Organization, 2016 (cit. on p. 1).

[2] International Energy Agency. *CO2 Emissions from Fuel Combustion 2019*. International Energy Agency, 2019 (cit. on p. 1).

[3] International Energy Agency. *Global EV Outlook 2020*. International Energy Agency, 2020 (cit. on p. 1).

[4] The State Council of China. *New Energy Vehicle Industry Development Plan (2021-2035)*. State Council. 2020 (cit. on p. 1).

[5] Ministry of Industry and Information Technology. *Parallel Management of Average Fuel Consumption and New Energy Vehicle Credits for Passenger Car Enterprises*. MIIT. 2017 (cit. on p. 1).

[6] European Commission. *The European Green Deal*. European Commission. 2019 (cit. on p. 2).

[7] U.S. House of Representatives. *The CLEAN Future Act*. House Committee on Energy & Commerce. 2020 (cit. on p. 2).

[8] California Air Resources Board. *Zero-Emission Vehicle Program*. CARB. 2020 (cit. on p. 2).

[9] Cabinet Office of Japan. *Green Growth Strategy*. Government of Japan. 2020 (cit. on p. 2).

[10] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. «CasADi – A software framework for nonlinear optimization and optimal control». In: *Mathematical Programming Computation* (2018) (cit. on pp. 3, 13, 14).

[11] Cunxue Wu, Zhongming Xu, Yang Liu, Chunyun Fu, Kuining Li, and Minghui Hu. «Spacing policies for adaptive cruise control: A survey». In: *IEEE Access* 8 (2020), pp. 50149–50162 (cit. on p. 6).

[12] A. Farnam and G. Crevecoeur. «Guaranteeing String Stability of Multiple Interconnected Vehicles Using Heterogeneous Controllers». In: *Journal of Dynamic Systems, Measurement, and Control* 143.6 (Jan. 2021), p. 061002. ISSN: 0022-0434. DOI: `10.1115/1.4049366`. eprint: `https://asmedigitalcollection.asme.org/dynamicsystems/article-pdf/143/6/061002/6617802/ds\_143\_06\_061002.pdf`. URL: `https://doi.org/10.1115/1.4049366` (cit. on p. 8).

[13] Vicente Milanés, Steven E Shladover, John Spring, Christopher Nowakowski, Hiroshi Kawazoe, and Masahide Nakamura. «Cooperative adaptive cruise control in real traffic situations». In: *IEEE Transactions on intelligent transportation systems* 15.1 (2013), pp. 296–305 (cit. on p. 9).

[14] Max Schwenzer, Muzaffer Ay, Thomas Bergs, and Dirk Abel. «Review on model predictive control: An engineering perspective». In: *The International Journal of Advanced Manufacturing Technology* 117.5 (2021), pp. 1327–1349 (cit. on pp. 10, 11).

[15] Jonathan Frey, Rien Quirynen, Tom van Leeuwen, Andrea Zanelli, Dario Brescianini, Robin Verschueren, Niels van Duijkeren, Andrea Zanelli, and Dario Brescianini. *acados: a modular open-source framework for fast embedded optimal control.* `https://github.com/acados/acados`. 2021 (cit. on p. 13).

[16] Andreas Wächter and Lorenz T Biegler. «On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming». In: *Mathematical programming* 106 (2006), pp. 25–57 (cit. on p. 13).

[17] Hans Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Georg Bock, and Moritz Diehl. «qpOASES: A parametric active-set algorithm for quadratic programming». In: *Mathematical Programming Computation* 6 (2014), pp. 327–363 (cit. on p. 13).

[18] Chang Wang, Xia Zhao, Rui Fu, and Zhen Li. «Research on the comfort of vehicle passengers considering the vehicle motion state and passenger physiological characteristics: Improving the passenger comfort of autonomous vehicles». In: *International journal of environmental research and public health* 17.18 (2020), p. 6821 (cit. on p. 15).

[19] Il Bae, Jaeyoung Moon, and Jeongseok Seo. «Toward a comfortable driving experience for a self-driving shuttle bus». In: *Electronics* 8.9 (2019), p. 943 (cit. on p. 15).

[20] Ardalan Vahidi and Antonio Sciarretta. «Energy saving potentials of connected and automated vehicles». In: *Transportation Research Part C: Emerging Technologies* 95 (2018), pp. 822–843 (cit. on p. 16).

[21] Haitao Xia, Kanok Boriboonsomsin, Friedrich Schweizer, Andreas Winckler, Kun Zhou, Wei-Bin Zhang, and Matthew Barth. «Field operational testing of eco-approach technology at a fixed-time signalized intersection». In: *2012 15th International IEEE Conference on Intelligent Transportation Systems*. IEEE. 2012, pp. 188–193 (cit. on p. 16).

[22] Matthew Barth, Sindhura Mandava, Kanok Boriboonsomsin, and Haitao Xia. «Dynamic ECO-driving for arterial corridors». In: *2011 IEEE forum on integrated and sustainable transportation systems*. IEEE. 2011, pp. 182–188 (cit. on p. 20).

[23] Alexander Koch, Tim Bürchner, Thomas Herrmann, and Markus Lienkamp. «Eco-driving for different electric powertrain topologies considering motor efficiency». In: *World Electric Vehicle Journal* 12.1 (2021), p. 6 (cit. on p. 20).

[24] Peng Hao, Guoyuan Wu, Kanok Boriboonsomsin, and Matthew J Barth. «Eco-approach and departure (EAD) application for actuated signals in real-world traffic». In: *IEEE Transactions on Intelligent Transportation Systems* 20.1 (2018), pp. 30–40 (cit. on p. 20).

[25] Antonio Sciarretta, Ardalan Vahidi, et al. *Energy-efficient driving of road vehicles*. Springer, 2020 (cit. on pp. 27, 45).