Adaptive Cruise Controller Design in Vehicular Applications

by

Yang Zhao
Bachelor of Engineering, Nanjing University of Aeronautics and Astronautics, 2009
Master of Science, Technische Universität Berlin, 2012

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF ENGINEERING

in the Department of Mechanical Engineering

Adaptive Cruise Controller Design in Vehicular Applications

by

Yang Zhao
Bachelor of Engineering, Nanjing University of Aeronautics and Astronautics, 2009
Master of Science, Technische Universität Berlin, 2012

Supervisory Committee

Dr. Yang Shi, Supervisor
(Department of Mechanical Engineering)

Dr. Tao Lu, Outside Member
(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. Yang Shi, Supervisor
(Department of Mechanical Engineering)

Dr. Tao Lu, Outside Member
(Department of Electrical and Computer Engineering)

ABSTRACT

As the most popular active safety driving technology, adaptive cruise control is favored by major car producers and their end-users. This report presents the history, and the functional and regulatory requirements of the ACC (adaptive cruise control) systems, and also introduces three popular controllers and their design approaches. Then a full or partial realization of the ACC function is accomplished by modelling of the ACC system in MATLAB/ Simulink (R2021b) and using the three controllers for simulation respectively. Finally, through a set of designed test scenarios and the simulations accordingly, the designs of the ACC systems are validated, followed by a discussion on the advantages and disadvantages of each control methodology.

Keywords: PID, LQR, Reinforcement Learning, ACC, MATLAB

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

Throughout the writing of this report and my study at University of Victoria (UVic) I have received a great deal of support and assistance.

I would first like to acknowledge the supervision and support of my supervisor, Dr. Yang Shi, a decent and professional scholar, for inspiring my interest in the study of control systems and for the great help and guidance during my master's study at UVic.

Next, I would like to thank the committee members Dr. Tao Lu and Dr. Cheng Lin who have been continuously supporting and helping me. I also would like to thank Tianxiang Lu, Yue Song, and Pengcheng Hu, for their constructive suggestions in improving my project and report.

In addition, I wish to thank my best friend and my wife, Yujin Wang. Without her emotional support, silent dedication to our family, and great faith in me, I would not be the man I am today.

Moreover, sincere thanks go to the communities of Canada, for offering me the chance to study at UVic, for welcoming me and my wife into this beautiful but also strange country, and for giving us generous help. Thank you for their being, making this cold country into a heart-warming place.

Finally, I am indebted to my parents for their continued support and encouragement.

Victoria, B.C., Canada
December 2022

ACRONYMS AND ABBREVIATIONS

2D	Two-Dimensional
3D	Three-Dimensional
ACC	Adaptive Cruise Control
AHS	Automated Highway System
APTS	Advanced Public Transportation System
ARTS	Advanced Rural Transportation System
ATIS	Advanced Traveler Information System
ATMS	Advanced Traffic Management System
AVCSS	Advanced Vehicle Control and Safety System
CAS	Collision Avoidance System
CWS	Collision Warning System
CVOS	Commercial Vehicle Operation System
DDPG	Deep Deterministic Policy Gradient
DQN	Deep Q-Network
GM	Gain Margin
ISO	International Organization for Standardization
ISTEA	Inter-modal Surface Transportation Efficiency Act
ITS	Intelligent Transportation System
LDWS	Lane Departure Warning System
LQR	Linear Quadratic Regulator
LWS	Lane Keeping System
MPC	Model Predictive Control
PM	Phase Margin
PID	Proportional-Intergral-Derivative
RL	Reinforcement Learning
SDKS	Safe Distance Keeping System
TEA-21	Transportation Equity Act for the 21st Century

Chapter 1

Adaptive Cruise Control

Since the first practical automobile was built by German engine designer and automotive engineer Karl Friedrich Benz in 1885 [1], the automotive industry has evolved over the century. Through the development of multiple stages such as mass production, electrification, and intelligentization, automobiles today have played an important role in people's daily lives and economic activities.

Early in 1989, the United States formulated the earliest intelligent transportation system (ITS) development strategy, followed by two ITS acts, ISTEA (Intermodal Surface Transportation Efficiency Act) and TEA-21 (Transportation Equity Act for the 21st Century), to determine the fundamental structure of the intelligent transportation system for vehicle applications. As a complex modern engineering system, ITS consists of several subsystems such as ATIS (advanced traveler information system), ATMS (advanced traffic management system), APTS (advanced public transportation system), AVCSS (advanced vehicle control and safety system), AHS (automated highway system), ARTS (advanced rural transportation system), and CVOS (commercial vehicle operation system [2].

As one of the core subsystems, AVCSS takes the vehicle as the research object and develops various assisted driving technologies that help the driver to partially or entirely control the vehicle, in order to provide a more safe, more efficient, and more comfortable driving experience. AVCSS includes several subsystems: ACC (adaptive cruise control), CAS (collision avoidance system), CWS (collision warning system), LWS (lane keeping system), and LDWS (lane departure warning system), etc. (see Fig. 1.1), among which ACC is currently the most widely used car assisted driving system that researchers are most concerned about, and almost all automotive producers are vigorously developing it.

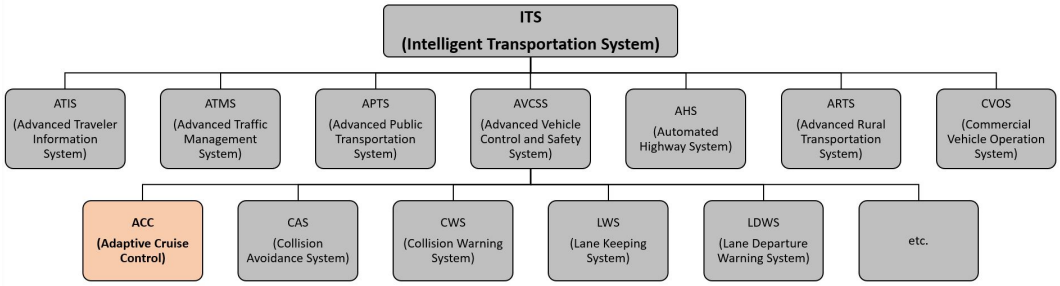


Figure 1.1: ACC in ITS Architecture.

The adaptive cruise control system is initially conceptualized in 1970s. It combines the function of two systems, a cruise control system and a safe distance keeping system (SDKS), to realize a fixed-speed cruising function and to maintain a safe distance from the lead vehicle, by adjusting the throttle valve and braking devices. The control decisions are made according to the target detection, relative distance and speed measurement realized by radar sensors installed at the front part of the car body. The first adaptive cruise control system which is a lidar-based distance detection system is introduced in Japan in 1992. It gives a warning to the driver if it determines that objects are too close by detection. However, it requires the driver to apply the brakes and reduce the vehicle speed, which is the main difference from today's adaptive cruise control system [3].

The advantages of the adaptive cruise control system are listed as follows.

1. The ACC system automatically controls the vehicle to replace a manual operation from the driver during long-distance driving and releases the driver from pressing the accelerator pedal or brake pedal for a long time;

2. The ACC system helps to avoid traffic accidents caused by drivers (drunk driving, fatigue driving, misoperation, etc.) [3];

3. The ACC system improves ride comfort and reduces fuel consumption by smoothly controlling the speed and acceleration of the vehicle accurately, which cannot be achieved by human drivers;

4. The ACC system can avoid mechanical wear caused by frequent acceleration and deceleration of novice and prolong the service life of vehicle parts;

5. The ACC system has the potential to increase the traffic flow by applying a smaller distance between vehicles [3] that can be realized due to a significant

shorter response time of the adaptive cruise control system compared with human drivers.

The vehicle adaptive cruise control system has become a hot research topic in the automotive industry and academia in recent years. It is of great significance to improve the driving experience of vehicle transportation users and to promote traffic intelligentization and modernization.

Chapter 2

Control Methodologies

To realize the ACC system as described in chapter 1, a few methodologies could be implemented to accomplish the system's function. And here in chapter 2, a fundamental introduction to those methods will be introduced.

2.1 PID

Proportional-integral-derivative (PID) controller is the most used controller in many fields, aiming to achieve the regulation of signals such as temperature, pressure, flow, speed, and other process variables. Fig. 2.1 shows the closed-loop system under the PID control framework.

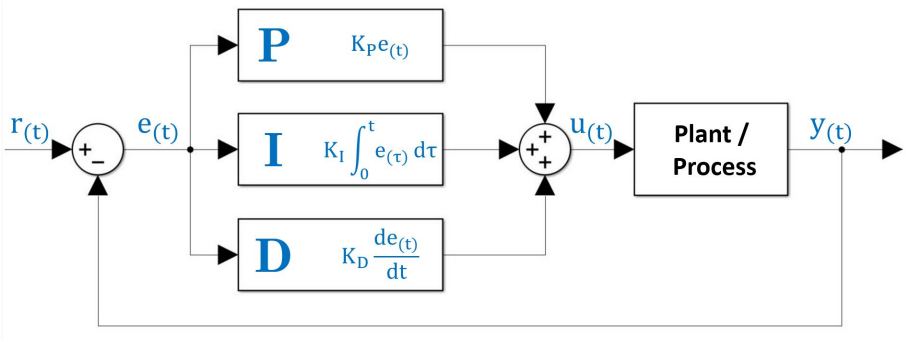


Figure 2.1: A Block Diagram of A PID Controller.

The PID controller uses a feedback control mechanism to let the process variable $y_{(t)}$ track its reference value $r_{(t)}$ by adjusting the congtrol input signal $u_{(t)}$ [4]. A typical PID controller consists of three terms: P (proportional), I (integral), and D (derivative). By adjusting the gains (K_P, K_I, K_D) assigned to each term, a PID

controller can be tuned to adapt to different application requirements. Generally, as formulated in Equation 2.1, K_P captures the difference between the current signal and the reference $e_{(t)}$, K_I accumulates all the differences in the past, and K_D is sensitive to the change rate of the control signal. They are then added together with pre-tuned gains to get the calculated control signal $u_{(t)}$ which satisfies the design requirements of the control system. In some applications, only part of the terms is used. In those cases, a PID controller is modified into a P controller, PI controller, PD controller or I controller, among which PI is more popular, because D term is quite sensitive to noise signals, while a controller without I term cannot guarantee to eliminate the steady-state error. Generally, a PI controller is used to eliminate the system steady state error, and a PD controller is used to improve the system transient response [5].

$$u_{(t)} = K_p e_{(t)} + K_i \int e_\tau \mathrm{d}\tau + K_d \frac{\mathrm{d}}{\mathrm{d}t} e_{(t)} \tag{2.1}$$

2.1.1 Practicle Implementation Issues

There are some practical implementation issues related to designing a PID controller.

1. High sensitivity to noise of the pure derivative controller:
   It is known that the derivative term is sensitive to signals of quick changes, i.e., noises can significantly impact the derivative term in practical applications. To address this issue, normally a filter is used to reject the noise.
   Fig. 2.2 shows a case study where a low-pass filter with the corner frequency being 30 rad/s is applied. The signal used in this case is a sine wave signal with amplitude as 2 and frequency as 20 rad/sec, and the noise added has a mean value of 0.4 with variance 0.02, based on a sample time of 0.001 second.

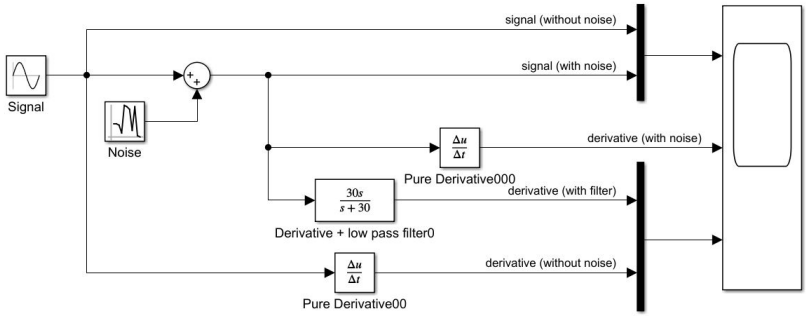


Figure 2.2: Noise Filter for Pure Derivative Controller.

As observed from the scope (see Fig. 2.3), by adding this filter, the derivative signal with a filter ('Derivative of Filtered Signal with Noise') is much improved from the one without a filter ('Derivative of Signal with Noise'), despite the phase lag and attenuation that are also introduced from the filter, which can be seen from the difference between 'Derivative of Filtered Signal with Noise' and 'Derivative of Pure Signal'.

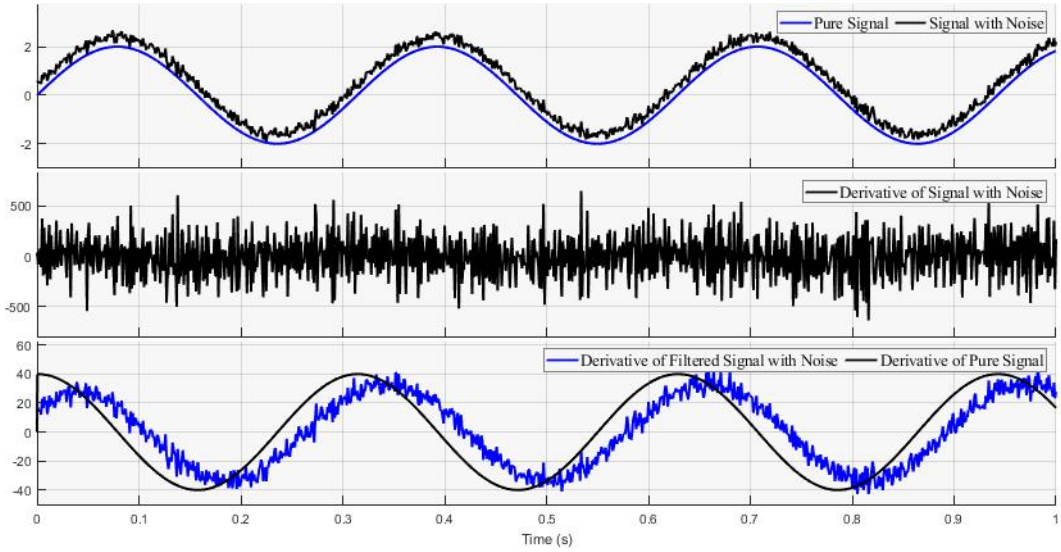


Figure 2.3: Output Comparison of Applying the Noise Filter for Pure Derivative Controller.

2. Step response of the derivative controller:

   Similarly, a step response can be devastating to a derivative term. By introducing a low-pass filter or a small group of them, it is possible to avoid this problem (see Figs. 2.4 and 2.5).

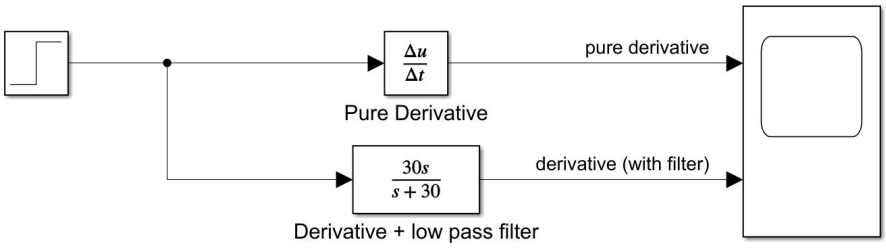


Figure 2.4: Step Response to Derivative Case Study.

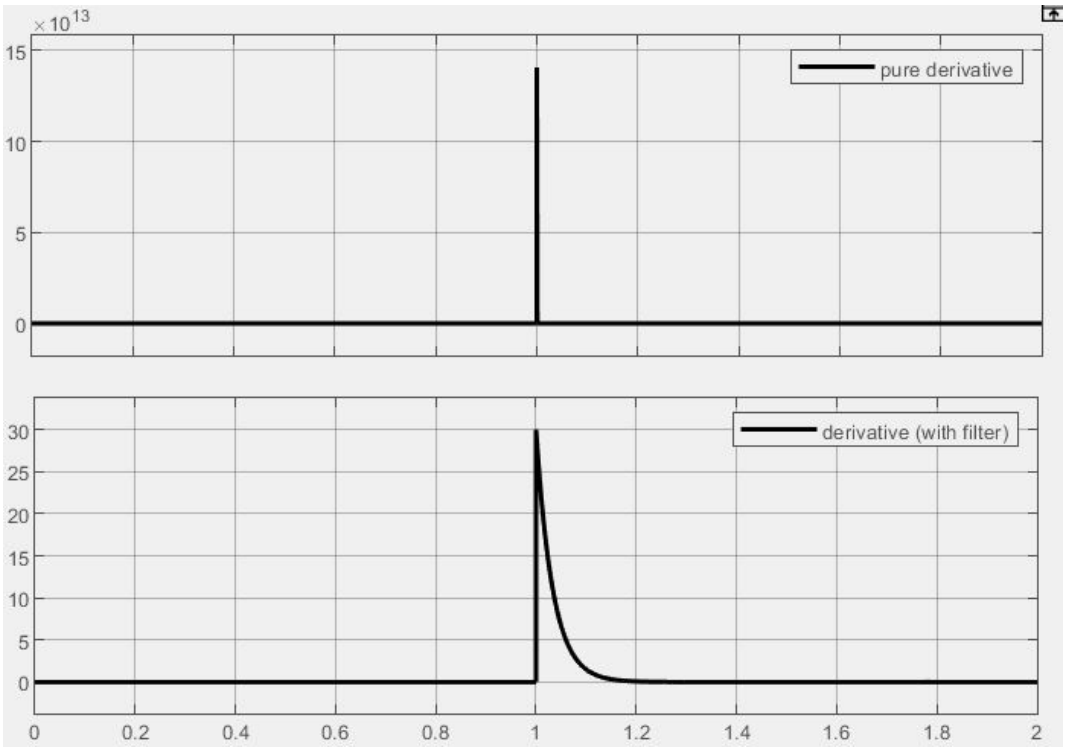


Figure 2.5: Scope Results of Step Response to Derivative Case Study.

3. Integral windup:
   The integrator term of a PID controller is an extremely useful tool to track set points with no steady-state errors. However, it interplays with the saturations of the system to be controlled and leads to severe failures in some cases.
   There are several ways to compensate for the windup [6]:

   - To disable the integrator until the system gets near the set point;
   - To limit the time period over which the integral is calculated;
   - Clmping: to limit the maximum and minimum states of the integrator;

2.1.2 Methods of Tunning Gains

There are a few practical methods to tune the gains (K_P, K_I, K_D).

1. Ziegler-Nicolas Method [7]: Ziegler-Nicolas method is an iterative, online method to choose K_P, K_I, and K_D. This method starts with setting the K_I and K_D to zero, and then raises K_P until $K_{P,stab}$ which makes the system stable. Then the gains can be calculated based on $K_{P,stab}$ according to the type of the control

system.

This method often provides good initial gains without requiring expert knowledge and the model or simulation of the system. But it does need the system to be stable and can be driven unstable by increasing K_P. Besides, validating the gains obtained on a real system sometimes can be costly.

2. Root-locus Method: Root-locus method is a powerful method to tune the gains from the frequency domain perspective. By manipulating the location of poles and zeros of the control system, the system performance can be improved in terms of percentage overshoot, raising time, settling time, gain margin (GM), and phase margin (PM).

   The benefit of using the root-locus method is that it is more intuitive to understand how adjusting the three gain values will affect system performance. But the disadvantage is that the mathematical model of the system needs to be established in advance, and it needs to have certain professional knowledge about the control system.

In MATLAB, tools like Control System Designer and PID Tuner are accessible, which can easily and intuitively observe the performance of the controlled system under different PID gains from the perspective of the time domain and frequency domain. By using these tools, it is quick and easy to obtain the required controller to meet the performance requirements of the control system.

2.2 LQR

LQR (Linear Quadratic Regulator) is a type of optimal control that is based on state-space representation. In a typical LQR structure, the full state vector is feedbacked and multiplied by a gain matrix K to be subtracted from the reference, as the input to the state space model (see Fig. 2.6 and Equation 2.2).

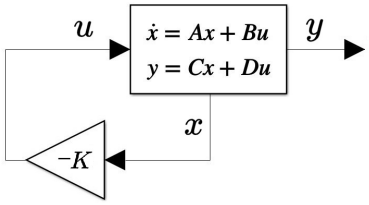


Figure 2.6: A LQR Architecture.

$$u = -Kx \tag{2.2}$$

In the traditional pole placement method, K is chosen by manipulating the locations of the poles. But deciding where to move the poles can be difficult and not intuitive, especially for high-order systems or systems with multiple actuators. While in the LQR methodology, K is calculated by solving an associated Riccati equation which is closely related to the closed-loop stability and optimal control performance. In the LQR method, matrices Q and R are introduced to construct the cost function [8]:

$$J = \int_0^\infty \left[(\bar{x}_{(t)})^T Q \bar{x}_{(t)} + (\bar{u}_{(t)})^T R \bar{u}_{(t)} \right] \mathrm{d}t \tag{2.3}$$

where $\overline{x}_{(t)}$ is a n×1 vector for state error; $\overline{u}_{(t)}$ is a m×1 vector for control inputs; Q is an n×n symmetric diagonal positive semi-definite matrix; and R is a m×m symmetric diagonal positive definite matrix.

Thus, the overall cost can be estimated based on this function, in which Q is to penalize poor performance (error of states), and R is to penalize large actuator effort. By adjusting the values of the elements in those two matrices, the element with a higher value occupies a more important place in the cost equation. This allows critical states or control efforts to be prioritized when calculating the controller output.

The cost function also constructs a quadratic function in the form of $Z = X^2 + Y^2$, which always has a definite minimum [9]. By finding the minimum, the optimized K is also calculated according to the given Q and R.

The solution to this cost function is:

$$\bar{u}_{(t)} = -K\bar{x}_t \tag{2.4}$$

where

$$K = R^{-1}B^T S \tag{2.5}$$

where S is the solution to the Algebraic Ricatti Equation that

$$A^T S + SA - SBR^{-1}B^T S + Q = 0 \tag{2.6}$$

In MATLAB, the calculation processes are integrated into one function **lqr()**, which can be called as: **[K, S, E] = lqr(A, B, Q, R)**, where K and S are defined as above

discussed, and E gives the poles of the closed-loop system.
The LQR method is a powerful tool to develop an optimal controller, and it gives an intuitive understanding of how to tune the K to satisfy the design purpose. While the most drawbacks of this method are that LQR requires a good knowledge about the state of the problem which is not always possible, and LQR is only applicable to linear systems.

2.3 Reinforcement Learning

Reinforcement learning (RL), one of the three basic machine learning paradigms [10], is an area of machine learning aiming at finding the best sequence of actions that will generate the optimal outcome based on a reward basis, rewarding desired behaviors and results and punishing undesired ones.
In reinforcement learning, an agent, a piece of software, explores, interacts with, and learns from the environment. The agent takes actions within the environment, which affects the environment by changing its state, and the environment then produces a reward for that action. By collecting this reward, the agent can adjust which action to take in the future.
The reinforcement learning process can be summarized in the following four steps:

1. The agent observes the current state of the environment;
2. The agent takes an action given the state of the environment;
3. The agent observes the new states and receives a reward for that action;
4. Based on the state observations and corresponding reward, the agent learns whether that action is preferable.

Reinforcement learning is more focused on the interaction between the agent and the environment, which brings a unique challenge – the trade-off between "exploration" and "exploitation" [11], the agent must utilize existing experience to gain benefits, but also to try new actions, which has the potential to bring higher rewards in the future.
The major benefit of using reinforcement learning is its ability to solve complex problems which are difficult or even impossible to be solved by conventional techniques, because of the difficulties to build the mathematical model of the system for example.

In addition, reinforcement learning does not rely on existing data. The training is only dependent on the interaction between the agent and the environment, and the training data is collected during each episode of the training.
Meanwhile, there are also some drawbacks. Reinforcement learning is computationally expensive, time-consuming, and highly dependent on the collection of data. And due to the non-transparency of the neural network, validation and tuning of a reinforcement learning trained system is always difficult and clueless.
To overcome the shortcomings and embrace the advantages of reinforcement learning, a combination of reinforcement learning with other conventional techniques is a good solution.
Nowadays, reinforcement learning has been used in many fields, including games, robotics, production, computer vision, natural language processing, transportation, autonomous driving, etc.

2.4 Others

There are many other control methods showing their possibilities in the field of autonomous driving systems. Model Predictive Control (MPC) is one of the advanced methods for constrained systems (see Fig. 2.7)[12].
MPC is a feedback control algorithm that uses a model to make predictions about the future outputs of a process. It has been used in the process industry since the 1980s. With the increasing computing power of microprocessors, the usage of MPC has spread to other fields as well, such as automotive, aerospace, energy, food processing, industrial manufacturing, metallurgy and mining, robotics, etc [13].
And there are a few benefits to using an MPC controller [13]:

- MPC is able to handle multi-input multi-output (MIMO) systems that have interactions between the inputs and outputs, due to which designing a MIMO system is often a challenging task with a traditional controller such as PID;
- MPC can handle constraints, which should not be violated to avoid undesired consequences;
- MPC has its preview capability, which is similar to feedforward control. This is achieved by optimizing a finite horizon, but only implementing the current timeslot and then optimizing again, repeatedly [12]. MPC can easily incorpo-

rate future reference information into the control problem to improve controller performance.

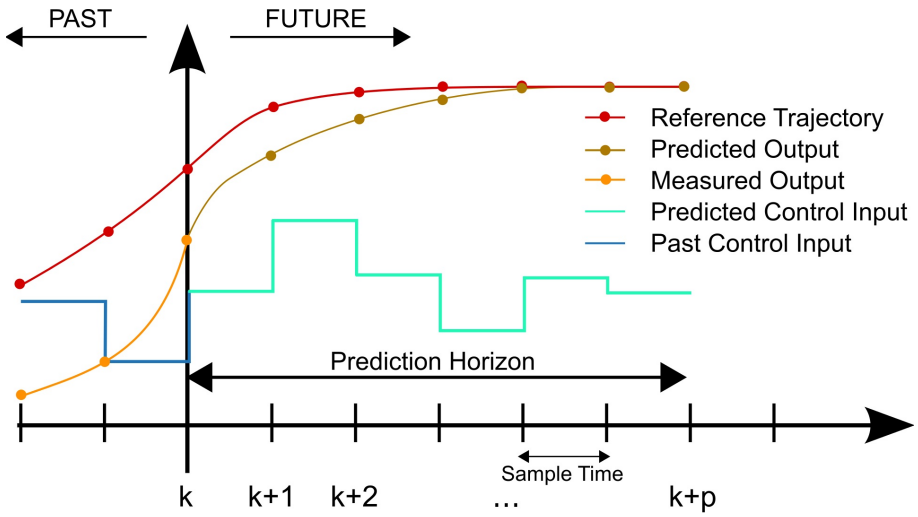


Figure 2.7: A MPC Scheme.

However, designing an ACC system with MPC controller is not included in the scope of this project.

Some examples of ACC systems implemented by MPC are existing. One of the samples similar to the project can be launched in MATLAB using **openExample('mpc/AdaptiveCruiseControlExample')** command. However the vehicle model and driving scenarios definition in this example are different from the one used in the final project, thus a comparison cannot be done using the data from this example directly.

Chapter 3

System Requirement

According to ISO 15622:2018, Intelligent transport systems – Adaptive cruise control systems – Performance requirements and test procedures, the goal of ACC is partial automation of the longitudinal vehicle control and the reduction of the workload of the driver with the aim of supporting and relieving the driver in a convenient manner [14, p. v].
A few terms to be used in this project are listed as follows:

- Clearance: Distance from the lead vehicle's trailing surface to the ego vehicle's leading surface [14, p. 2].
- Time gap: Calculated as clearance divided by ego vehicle speed [14, p. 2]. The time gap shall be greater than or equal to 0.8s, and at least a time gap setting in the range of 1.5 s to 2.2 s shall be provided for speeds higher than 8 m/s [14, p. 6]. In the project, the time gap is set to be 1.5 s.
- Set speed: Desired travel speed of the ego vehicle, set either by the driver or by some control system that is external to the ACC system. And the set speed is the maximum desired speed of the vehicle while under ACC control [14, p. 2]. The minimum set speed shall be equal to or higher than 4.4 m/s [14, p. 10]. In this project, the set speed is set to be 35 m/s.
- Steady state: A condition whereby the value of the described parameter does not change with respect to time, distance, etc. [14, p. 2].
- Ego vehicle: Vehicle equipped with the ACC system in question and related to the topic of discussion [14, p. 3].

- Lead vehicle: Vehicle in front of and moving in the same direction and travelling on the same roadway as the ego vehicle [14, p. 2].
- ACC active state: Phase in which the ACC system controls the speed and/or clearance [14, p. 3].
- ACC speed control state: Phase in which the ACC system controls the speed according to the set speed [14, p. 3].
- ACC following control sub-state: Phase in which the ACC system controls the clearance to the lead vehicle according to the selected time gap [14, p. 3].
- Maximum acceleration: The maximum value allowed for the acceleration of the vehicle, powered by the engine. In this project, the maximum acceleration is set to be 3 m/s^2. There are more detailed requirements on acceleration defined in ISO 15622, which are ignored in this project.
- Maximum deceleration: The maximum value allowed for the deceleration of the vehicle, generated by the brake system. In this project, the maximum deceleration is set to be -4 m/s^2. There are more detailed requirements on deceleration defined in ISO 15622, which are ignored in this project.

ACC system should guarantee that the ego vehicle speed is controlled automatically either to maintain a clearance to a lead vehicle or to maintain the set speed, whichever speed is lower. The change between these two control modes is made automatically by the ACC system [14, p. 5].

Chapter 4

Modelling

Imagine driving a vehicle on a highway without considering the steering of the vehicle, which means the vehicle drives on a vertical plane, the ego vehicle follows a lead vehicle with the help of an adaptive cruise control system onboard. Whereas there are still some other environmental factors that need to be taken into consideration, such as the road slope and wind speed.

As mentioned in the previous section of this report, an ACC system regulates the speed of the ego vehicle and also regulates the distance to the forgoing vehicle. Fig. 4.1 [15] below shows a typical scenario that an adaptive cruise control system needs to deal with on the highway, where θ represents the variations on the road slope and W stands for the wind speed, both are external disturbances. In addition, the main variables include Y, V, V_f, and u. Y measures the distance between the ego vehicle and the lead vehicle, while V and V_f indicate the speed of the ego and lead vehicle individually. And u reflects the control effort of the adaptive cruise control system, which represents the acceleration or braking of the vehicle.

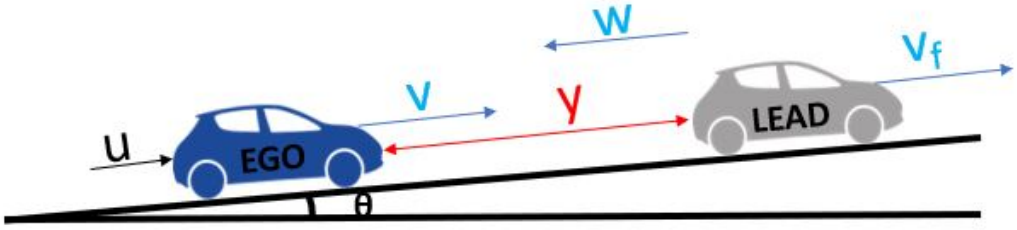


Figure 4.1: ACC Working Scenario.

The ego speed V may be acquired by an onboard vehicle speedometer. While another onboard sensor, such as millimeter-wave radar, camera, or lidar (see Fig. 4.2

[16]), can measure the distance and relative speed between the two vehicles. Combining the ego speed and the relative speed, the speed of the lead vehicle is also acquirable.

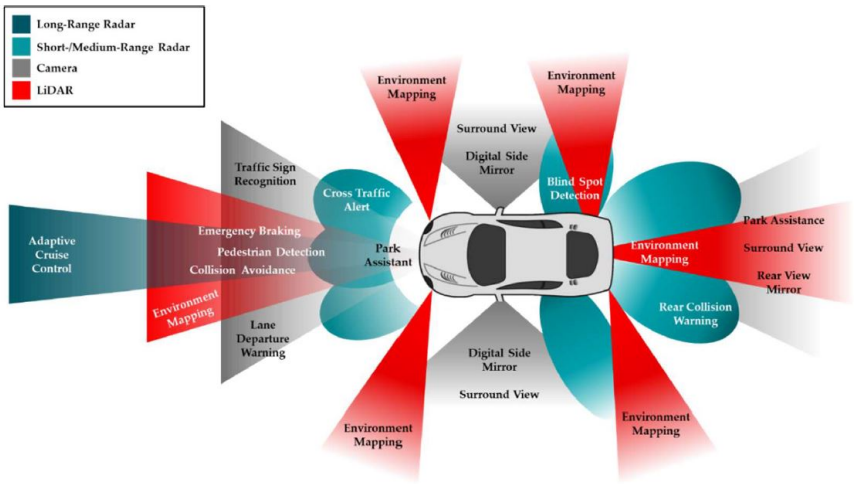


Figure 4.2: Camera, Radar and Lidar.

4.1 State-Space

Based on the physics of the system, the variance of the distance between the two vehicles can be derived from the speed difference between the two vehicle, and the variation of the velocity of the ego vehicle derives from Newton's law.

$$\frac{\mathrm{d}y}{\mathrm{d}t} = v_f - v \tag{4.1}$$

$$\frac{\mathrm{d}v}{\mathrm{d}t} = \frac{1}{m}\left[F_e\left(u\right) - F_g\left(\theta\right) - F_a\left(v, w\right) - F_r\left(v\right)\right] \tag{4.2}$$

In Equation 4.2, a few variables are clarified as follows:

- $F_e(u)$ is the engine force, which comes from the engine output or the braking system of the vehicle.

$$F_e\left(u\right) = T_e U_{(t)} \tag{4.3}$$

  where T_e is the vehicle engine's maximum force, and $U_{(t)}$ is the control effort.

- $F_g(\theta)$ is the gravitational force.

$$F_g\left(\theta\right) = mg\sin\theta_{(t)} \tag{4.4}$$

where m is the vehicle mass, and g is the gravity acceleration.

- $F_a(v,w)$ is the aerodynamic force [17], which gives the air resistance force that is applied on a vehicle and against the moving direction of the vehicle.

$$F_a(v,w) = \frac{1}{2}\rho A C_d \left[v_{(t)} + w_{(t)}\right]^2 \tag{4.5}$$

where ρ is the air density, A is the vehicle cross-sectional area, and C_d is the aerodynamic drag coefficient.

- And $F_r\ (v)$ is the rolling resistance force [18], which represents the resistance force generated on the interface between the road and the tires of the vehicle.

$$F_r(v) = \left(\mu_0 + \mu v_{(t)}^2\right) mg \tag{4.6}$$

where μ_0 and μ are rolling resistance coefficients.

By substituting Equations 4.3 - 4.6 into the Equation 4.2, non-linear model of the system can be derived as below:

$$\dot{Y} = -V(t) + V_f(t) \tag{4.7}$$

$$\dot{V} = \frac{T_e}{m}U_{(t)} - g\sin\theta_{(t)} - \frac{\rho A C_d}{2m}\left[V_{(t)} + W_{(t)}\right]^2 - \left[\mu_0 + \mu V_{(t)}^2\right] g \tag{4.8}$$

Equation 4.7 describes the fact that the change rate in relative distance between two vehicles is the difference between the lead vehicle speed and the ego vehicle speed. While Equation 4.8 illustrates how the variables affect the acceleration of the ego vehicle during adaptive cruising.

To simplify the problem, a few assumptions need to be made to linearize the model. Considering the system works near the linearization point where the road slope and wind speed are zero:

$$\bar{\theta} = 0 \tag{4.9}$$

$$\bar{w} = 0 \tag{4.10}$$

The system model can be then linearized into the following form:

$$\dot{Y} = -V(t) + V_f(t) \tag{4.11}$$

$$\dot{V} = a_{vv}V_{(t)} + a_{vu}U_{(t)} + a_{v\theta}\theta_{(t)} + a_{vw}W_{(t)} \tag{4.12}$$

where

$$a_{vv} = \frac{\partial\left(-\frac{\rho AC_d}{2m}\left[V_{(t)} + W_{(t)}\right]^2 - \left[\mu_0 + \mu V_{(t)}^2\right]g\right)}{\partial V} = -\left(\frac{\rho AC_d}{m} + 2\mu g\right)\bar{v} \tag{4.13}$$

$$a_{v\mu} = \frac{\partial\left(\frac{T_e}{m}U_{(t)}\right)}{\partial U} = \frac{T_e}{m} \tag{4.14}$$

$$a_{v\theta} = \frac{\partial\left(-g\sin\theta_{(t)}\right)}{\partial\theta} = -g\cos\bar{\theta} = -g \tag{4.15}$$

$$a_{vw} = \frac{\partial\left(-\frac{\rho AC_d}{2m}\left[V_{(t)} + W_{(t)}\right]^2\right)}{\partial W} = -\frac{\rho AC_d}{m}\left(\bar{v} + \bar{w}\right) = -\frac{\rho AC_d}{m}\bar{v} \tag{4.16}$$

As a result, the system can be represented as a linearized model as Equation 4.17 below:

$$\begin{bmatrix}\dot{Y}\\ \dot{V}\end{bmatrix} = \begin{bmatrix}0 & -1\\ 0 & -\left(\frac{\rho AC_d}{m} + 2\mu g\right)\bar{v}\end{bmatrix}\begin{bmatrix}Y\\ V\end{bmatrix} + \begin{bmatrix}0\\ \frac{T_e}{m}\end{bmatrix}U + \begin{bmatrix}1 & 0 & 0\\ 0 & -g & -\frac{\rho AC_d}{m}\bar{v}\end{bmatrix}\begin{bmatrix}V_f\\ \theta\\ W\end{bmatrix} \tag{4.17}$$

where Y and V are the states, U is the control, and V_f, θ, and W are the external disturbances.

In addition, in order to be closer to the actual working state, the linearization of the system can be carried out under different road gradient conditions. This will significantly increase the complexity of the system but can achieve more stable performance in practice. Another possible solution is to use MPC controllers because MPC controllers can handle nonlinear systems.

4.2 Transfer Function

In addition, the differential equation model also needs to be rewritten into forms of the transfer function, which will be used later in PID controller design by applying Laplace transformation on both sides of the equation as shown below in Equation 4.18.

$$\begin{cases} sY_{(s)} = -V_{(s)} \\ sV_{(s)} = -\left(\frac{\rho AC_d}{m} + 2\mu g\right)\bar{v}V_{(s)} + \frac{T_e}{m}U_{(s)} \end{cases} \tag{4.18}$$

which can be reshaped as Equation 4.19:

$$\begin{cases} V_{(s)} = \frac{T_e/m}{s+\left(\frac{\rho AC_d}{m}+2\mu g\right)\bar{v}} U_{(s)} \\ Y_{(s)} = -\frac{T_e/m}{s\left[s+\left(\frac{\rho AC_d}{m}+2\mu g\right)\bar{v}\right]} U_{(s)} \end{cases} \tag{4.19}$$

In this project, the following values of parameters are used as in Table 4.1.

Parameter	**Description**	**Value**	**Unit**
m	vehicle mass	1600	Kg
T_e	vehicle engine maximum force	2400	N
A	vehicle cross-sectional area	2.09	m^2
C_d	aerodynamic drag coefficient	0.32 [19]	-
μ	rolling resistance coefficient under 2.6 bar	4.66×10^{-6}	-
μ_0	rolling resistance coefficient under 2.6 bar	0.00885	-
g	gravity acceleration	9.8067 [20]	m^2
ρ	air density	1.2754	Kg/m^3
$\bar{v}$	cruise speed of linearization point	35	m/s

Table 4.1: System Parameters.

By substituting those values into Equations 4.17 and 4.19:

$$\begin{bmatrix} \dot{Y} \\ \dot{V} \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 0 & -0.0219 \end{bmatrix} \begin{bmatrix} Y \\ V \end{bmatrix} + \begin{bmatrix} 0 \\ 1.5 \end{bmatrix} U + \begin{bmatrix} 1 & 0 & 0 \\ 0 & -9.8067 & -0.0187 \end{bmatrix} \begin{bmatrix} V_f \\ \theta \\ W \end{bmatrix} \tag{4.20}$$

$$V_{(s)} = \frac{1.5}{s + 0.02186} U_{(s)} \tag{4.21}$$

$$Y_{(s)} = -\frac{1.5}{s^2 + 0.02186s} U_{(s)} \tag{4.22}$$

And Equations 4.21, 4.22 can be reformed as the transfer functions of the system:

$$G_v\left(s\right) = \frac{1.5}{s + 0.02186} \tag{4.23}$$

$$G_y\left(s\right) = -\frac{1.5}{s^2 + 0.02186s} \tag{4.24}$$

Chapter 5

Simulation System and Validation

In this chapter, the modelling of the simulation system in the SIMULINK environment will be introduced first, then the implementation (parameters tuning) of three different controllers in the ACC system will be discussed. In addition, simulations of the system under different working conditions (with or without external disturbance, whether the speed of the lead vehicle varies with time) will be conducted to analyze the control performance. In the last part, through two groups of designed validation simulations, the reliability of the ACC system under different ego and lead vehicle speeds is estimated.

5.1 Vehicle Model

The state-space model (see Fig. 5.1) and the transfer function model (see Fig. 5.2) of the ego vehicle are developed in Simulink to be used in controller designs to present the dynamic behavior of the vehicles.

In addition, a saturation block, speed limits, is built in both models to limit the speed of the vehicles from 0 to 50 m/s (0 to 180 km/h).

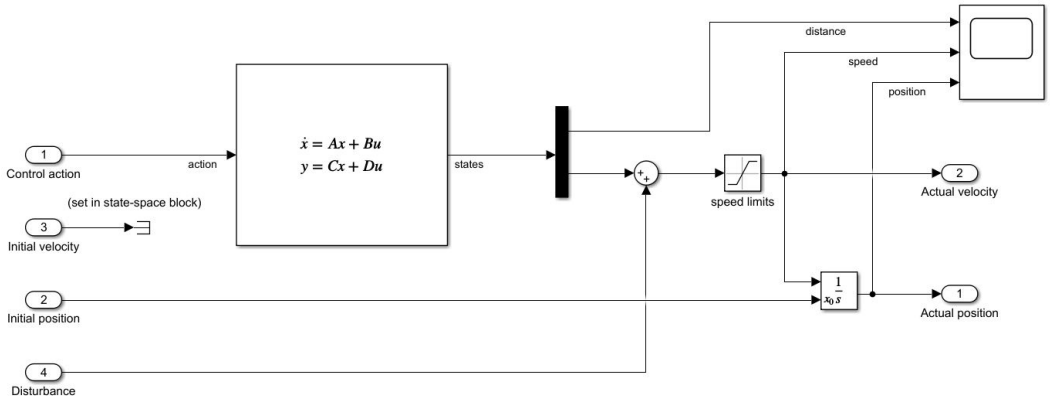


Figure 5.1: Vehicle State Space Model.

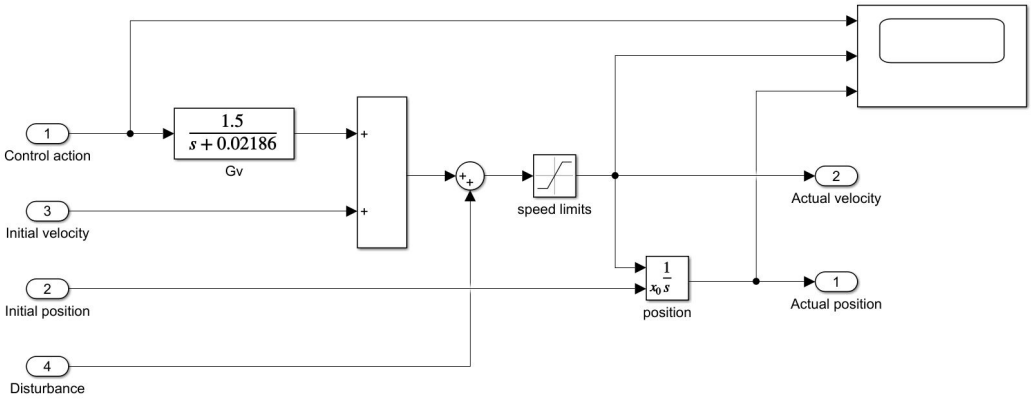


Figure 5.2: Vehicle Transfer Function Model.

5.2 Control for ACC System

To simulate and compare the performance of three different types of controllers, system models are built separately as described below.

5.2.1 PID

The system simulation model using PID control includes the ego vehicle dynamics, control mode decider, PID controller, sensor, ACC system input (driver settings), lead vehicle, external disturbances, animation (2D & 3D) and visualization modules (see Fig. 5.3).

The ego vehicle dynamics, control mode decider, ACC PID controller, sensor and ACC system input together form the ego vehicle. The transfer function model is used

in PID design as the ego vehicle dynamics module, as shown in Fig. 5.3. And the sensor provides the necessary measurements, relative distance and speed, to the ego vehicle as discussed before, while the ACC system input module allows the driver to adjust the set-speed and time gap.

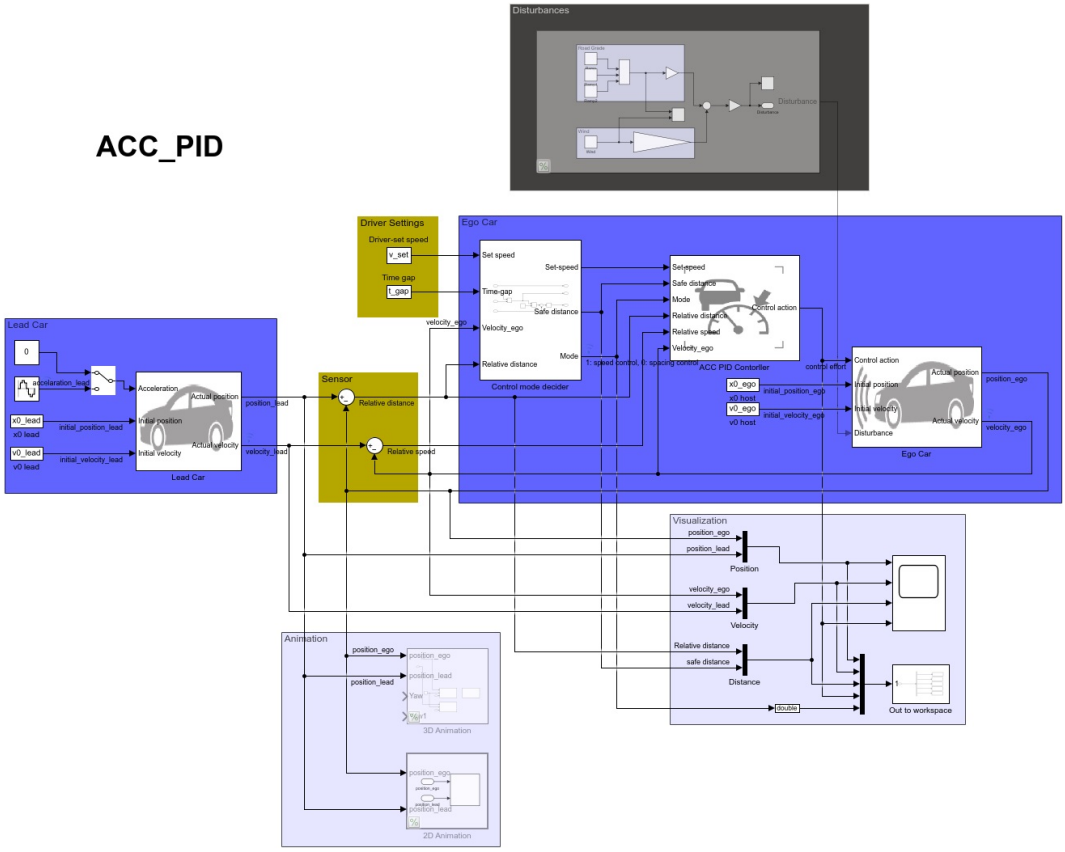


Figure 5.3: ACC PID Simulation.

Based on the inputs from the sensor and the ACC system input module, the control mode decider calculates the safe distance and compares it with the current relative distance (see Fig. 5.4) to decide the working mode which the system is in at each time step.

And it is noticeable that a multiplier of 1.1 is introduced in the module, which will be explained later in section 5.3.1.

In this project, mode 0 means spacing control mode, while mode 1 stands for speed control mode.

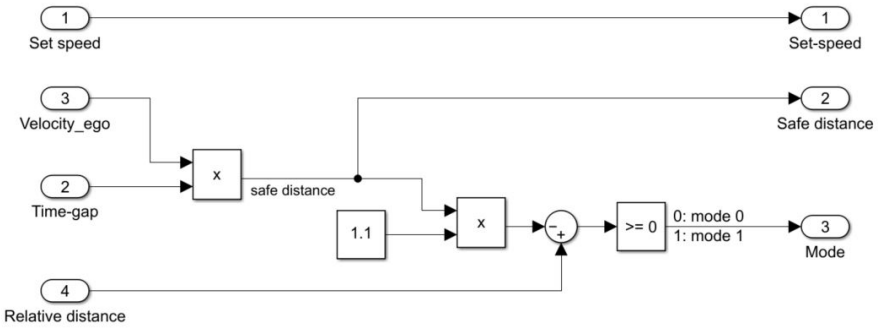


Figure 5.4: Control Mode Decider.

The ACC PID controller (see Fig. 5.5) is another core module in this control system, that it has two separate PID controllers generating control action signals for two different working modes, and the controller module also takes the mode signal sent from the mode decider to judge which control action signal should be dominant at each time step. And the desired control action is regulated by a saturation block that limits the control signal between maximum acceleration and maximum deceleration before sent to the following module.

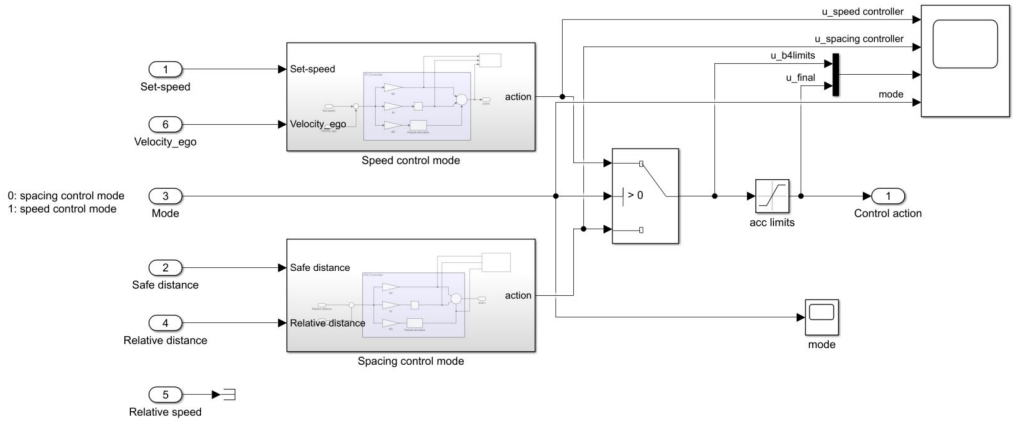


Figure 5.5: Control Mode Decider.

In addition, the limit output option in the integrator block is used, see Fig. 5.6 to limit the upper and lower saturation to avoid the integral windup as discussed in section 2.1.1. Another reasonable way to avoid the windup is to limit the time range of integral prior to the current time instant.

The lead vehicle, disturbance, animation, and visualization are not only introduced in the PID control system but also used when applying LQR and Reinforcement

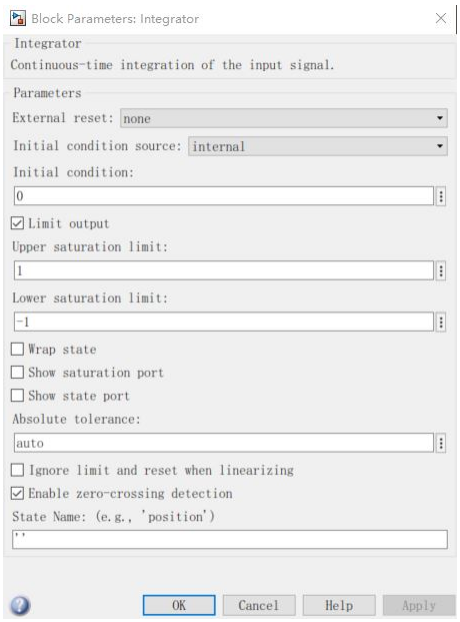


Figure 5.6: Limit Ouput Option in Integrator Block.

Learning.

The lead vehicle uses the same model derived for the ego vehicle to represent the dynamics, and two different speed modes are given to the lead vehicle for further validation.

The disturbance module (see Fig. 5.7) comprises two sub-systems, road grade and wind. And Fig. 5.8 shows how road grade and wind variate regarding time.

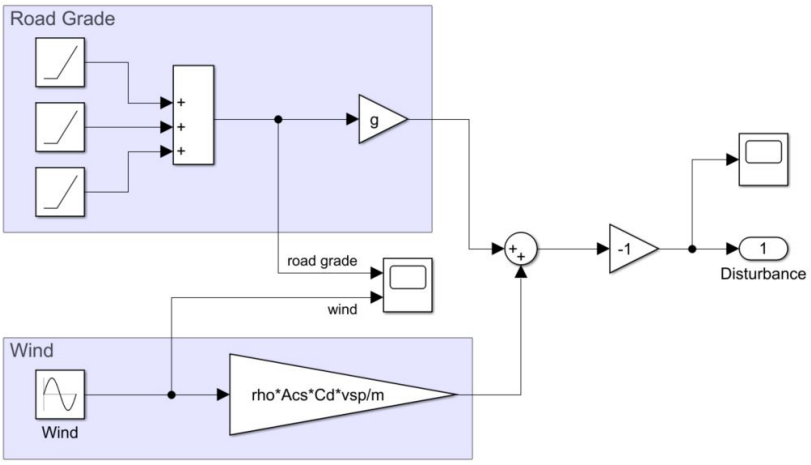


Figure 5.7: Pre-defined Disturbances.

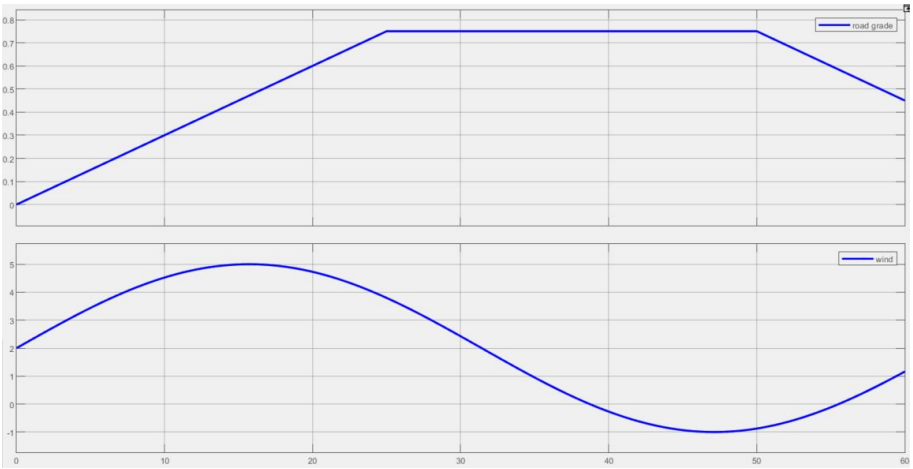


Figure 5.8: Road Slope and Wind.

In addition, the animation module is included as an option in three systems. The 3D animation module (see Fig. 5.9) adopts the built-in 'Simulation 3D Scene Configuration', while the 2D module (see Fig. 5.10) realizes a real-time positioning of the two vehicles from a bird's view through a self-edited MATLAB function.

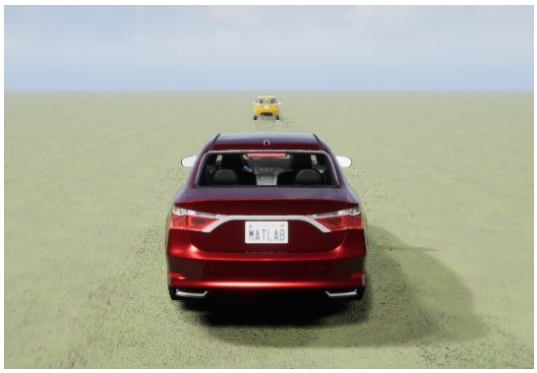


Figure 5.9: 3D Animation.

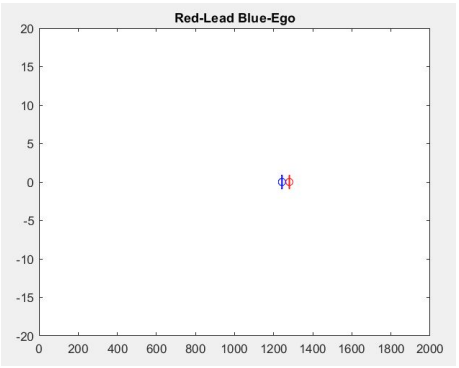


Figure 5.10: 2D Animation.

5.2.2 LQR

The simulation system for the LQR controller design (see Fig. 5.11) is generally consistent with the PID controller simulation system, same control mode decider, same sensor, same driver-setting, and the same lead vehicle. While in LQR, the state space vehicle model (see Fig. 5.1) is used as the ego vehicle dynamics.

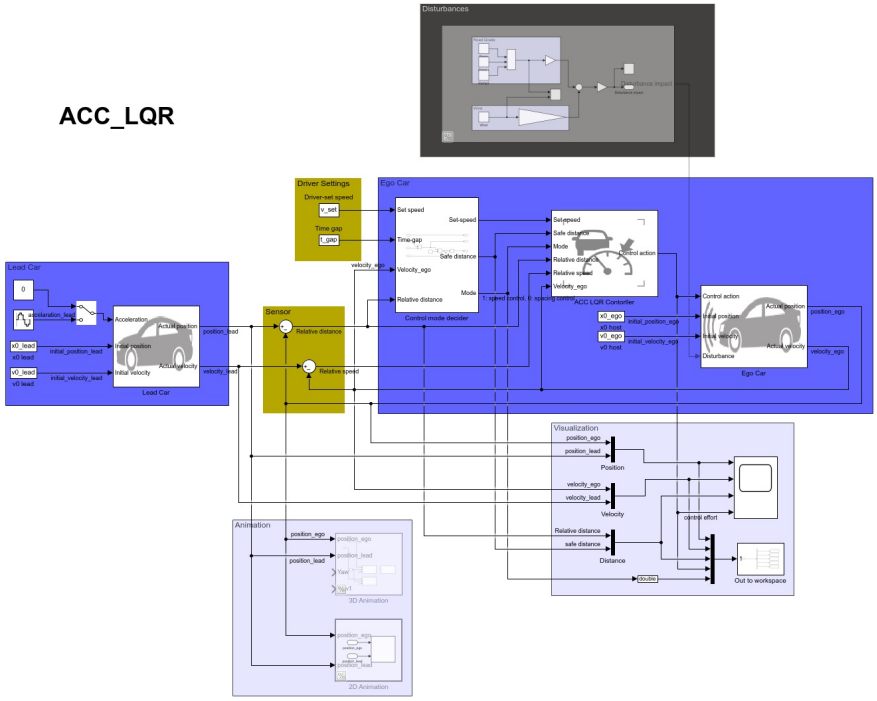


Figure 5.11: ACC LQR Simulation.

And in the ACC LQR controller module (see Fig. 5.12), a feedback design of gain matrix K is applied instead of K_P, K_I and K_D as used in PID.

Additionally, there are two gain blocks K_{r1} and K_{r2} that are used to eliminate the steady tracking error, which will be discussed in Section 5.3.2.

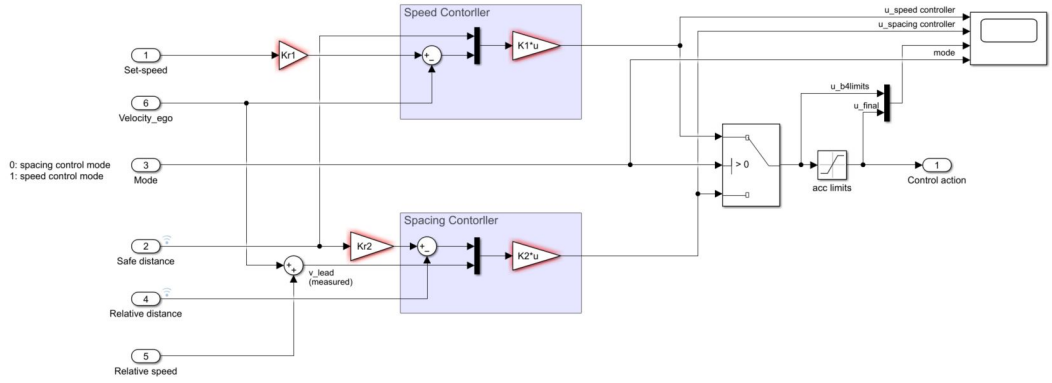


Figure 5.12: ACC LQR Controller.

5.2.3 Reinforcement Learning

The general layout of the system under RL (see Fig. 5.13) is similar to the previous counterparts, PID and LQR. But instead of a control mode decider module and an ACC controller, an agent is introduced here with three input signals which are generated by a pre-processor.

And a DDPG (Deep Deterministic Policy Gradient) algorithm, an actor-critic, model-free, and policy-based reinforcement learning method, is used in this project. In contrast with DQN (Deep Q-Network) which learns indirectly through Q-tables, DDPG learns directly from the observation spaces through the policy gradient method which estimates the weights of an optimal policy through gradient ascent which is similar to gradient descent used in the neural network. Also, the policy-based method is better suited to solving continuous action space environments [21, 22].

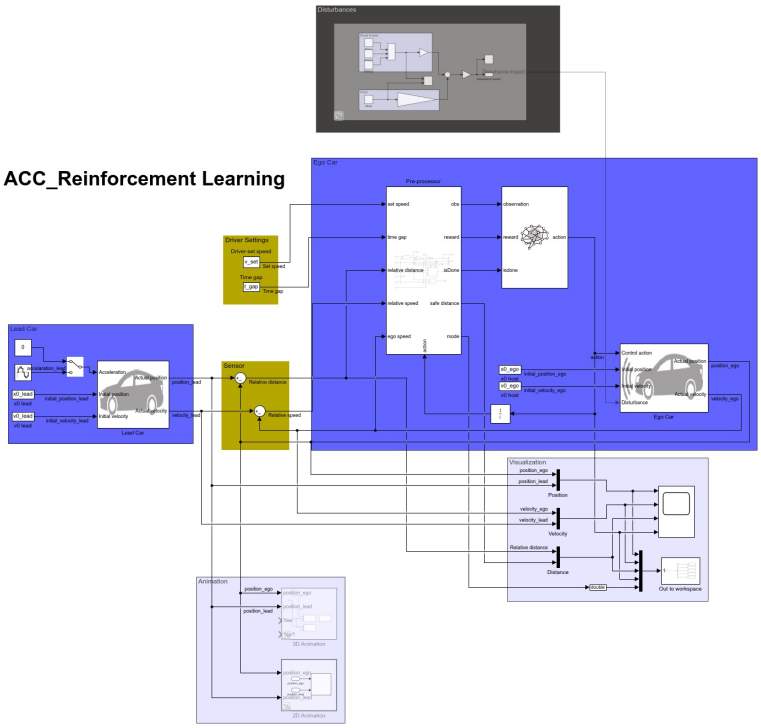


Figure 5.13: ACC RL Simulation.

The actor is a policy network that takes the state as input and outputs the exact action, while the critic is a Q-value network that takes in state and action as input and outputs the Q-value [23], which is aimed to be maximized after numerous episodes in the training. Figs. 5.14 and 5.15 show the layout of the actor and critic networks.

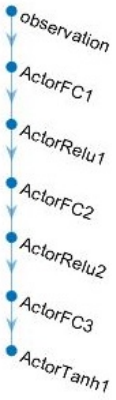


Figure 5.14: Actor Network.

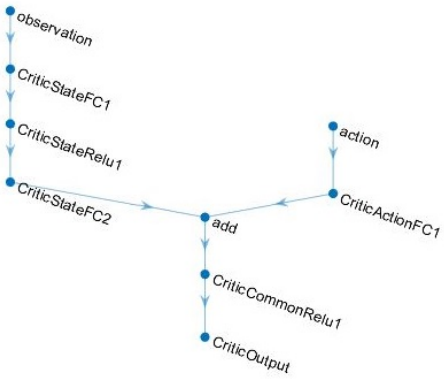


Figure 5.15: Critic Network.

What also needs to be highlighted is that the action signal received by the pre-processor is the signal from the last time step. So later, this signal can be taken into account to calculate the reward.

As illustrated in Fig. 5.16, the pre-processor takes signals from the sensor and the driver-setting, and by a series of calculations, delivers observations (control mode, difference between ego speed and set speed, difference between relative distance and safe distance), reward, and training status (isDone) to the agent.

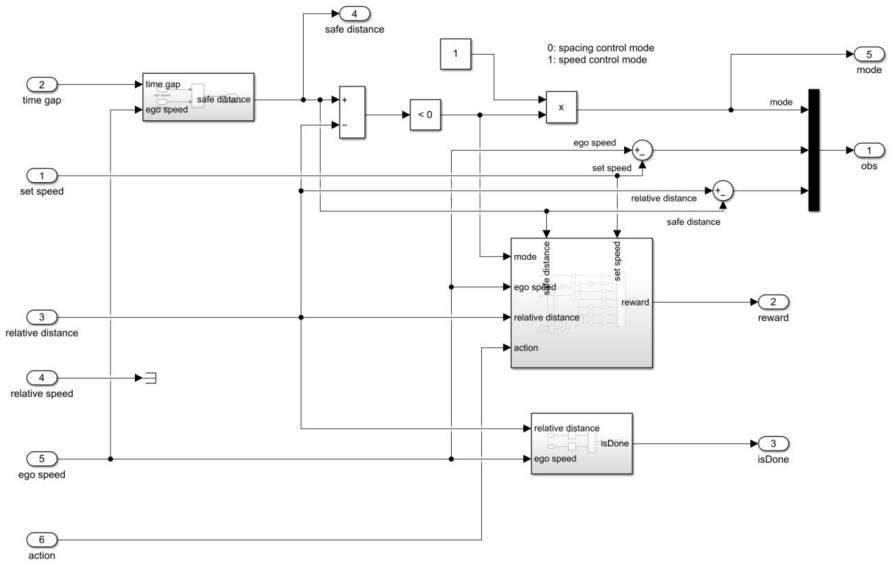


Figure 5.16: Pre-processor.

In reinforcement training, isDone signal is used to specify conditions under which a training episode should be terminated [24]. In this project, when the ego vehicle

speed or relative distance is negative, the isDone signal is set to logical 1 and the current episode is terminated (see Fig. 5.17). A negative ego speed means that the ego vehicle is starting to back up, and a negative relative distance means that the two vehicles have collided. Neither of them is possible nor allowed to happen in an ACC scenario.

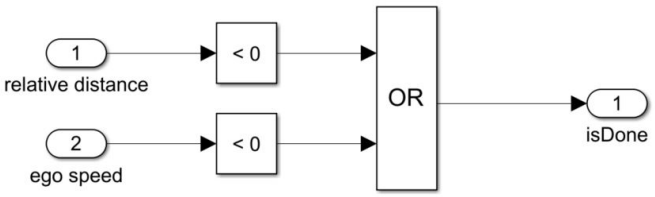


Figure 5.17: IsDone Definition.

5.3 Implementation

5.3.1 PID

For the speed control mode, the DC gain of the plant transfer function G_v (see equation 4.23) is 68.8, which leads to a large steady-state error. Furthermore, the rise time and settling time are too long, over 100 sec (see Fig. 5.18). Thus, a controller needs to be introduced to reduce the rise time, settling time and eliminate the steady-state error.

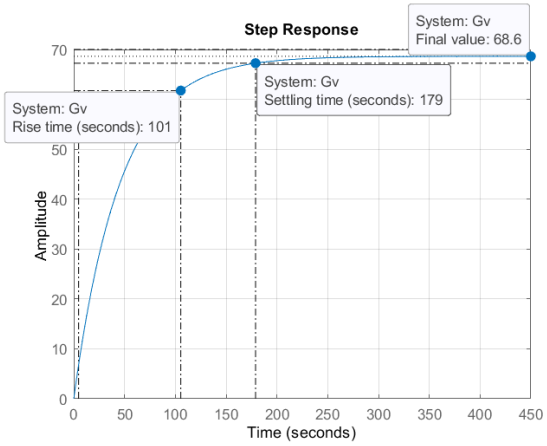


Figure 5.18: Step Response of G_v.

The MATLAB built-in function **controlSystemDesigner()** is used to design the controller. And after importing the plant into the designer, a few system requirements are added as in Fig. 5.19.

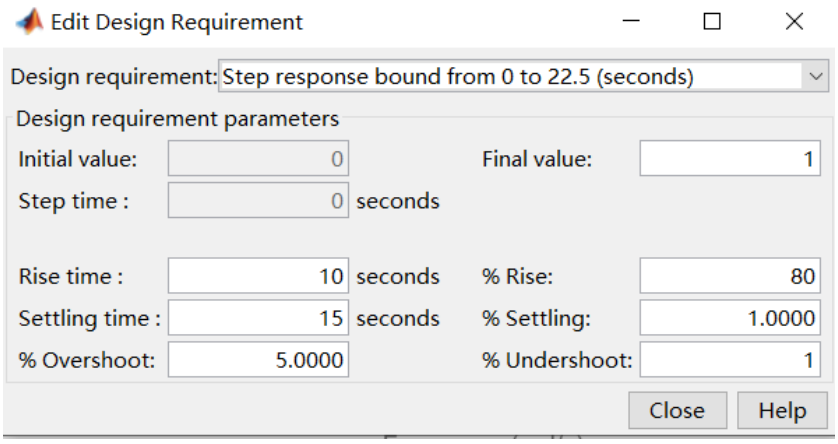


Figure 5.19: System Requirement for PID Speed Controller.

To design the controller, a PI controller is introduced by adding an integrator and a real zero as shown in Fig. 5.20. The phase margin is close to 90 deg, which means the system response is slow. However, considering the application scenario and the response speed requirements of the ACC system, the design can still satisfy the desired performance.

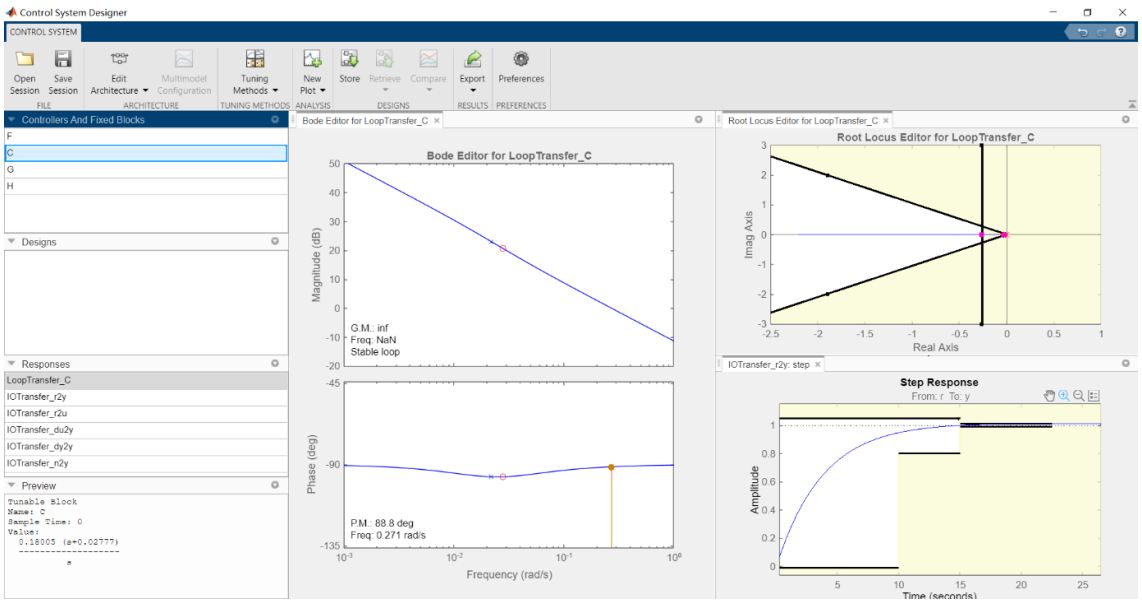


Figure 5.20: Design PID Speed Controller with Control System Designer.

By introducing this PI controller, the steady-state error is eliminated (system response is convergent to 1), and both rising time and settling time are shortened as desired (see Fig. 5.21).

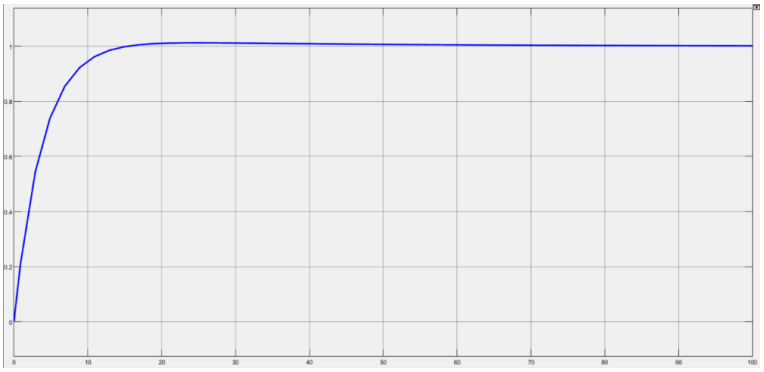


Figure 5.21: Step Response of G_v with PI Controller.

As a result, the control gains in speed and spacing control are computed and listed in Table 5.1.

PID Controller	**Speed Control**	**Spacing Control**
K_P	0.18	0.3
K_I	0.005	0
K_D	0	0

Table 5.1: K_P, K_I and K_D of PID Controller.

Based on the parameters, the PID controllers of two modes (Equation 5.1 for speed control mode and Equation 5.2 for spacing control mode) can be presented as:

$$u_{(t)} = 0.18e_{(t)} + 0.005 \int e_\tau \mathrm{d}\tau \tag{5.1}$$

$$u_{(t)} = 0.3e_{(t)} \tag{5.2}$$

To evaluate the performance of the PID controller for the ACC system under external disturbances, a few simulations are done with/ without disturbances and sinusoidal lead speed (see Figs. 5.22-5.25). And it can be observed that the ACC system functions properly under all situations when applying the PID controller.

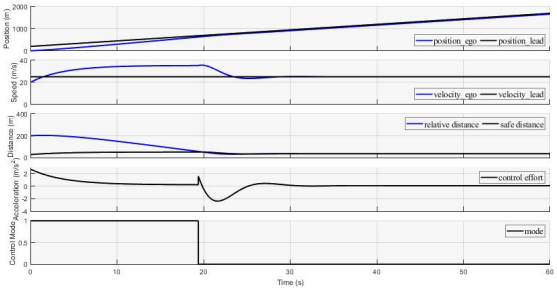


Figure 5.22: PID, Consistent Lead Speed, No Disturbances.

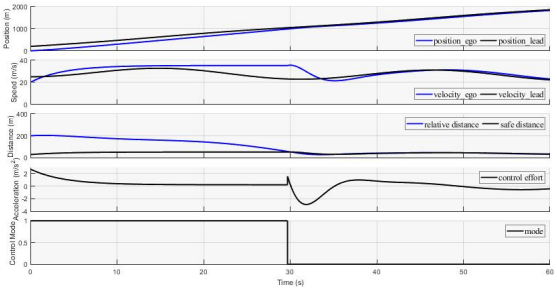


Figure 5.23: PID, Inconsistent Lead Speed, No Disturbances.

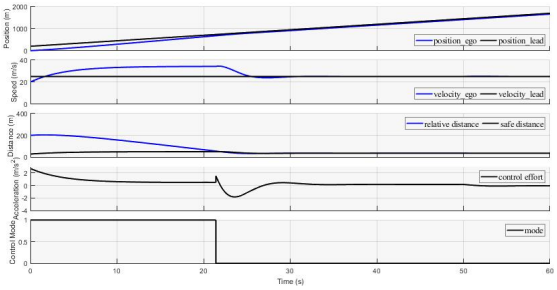


Figure 5.24: PID, Consistent Lead Speed, With Disturbances.

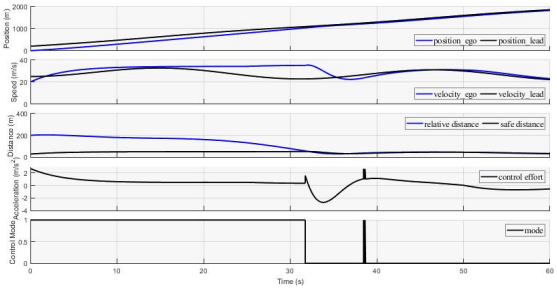


Figure 5.25: PID, Inconsistent Lead Speed, With Disturbances.

One highlight of the design, which is applied in the LQR design as well, is that a multiplier of 1.1 is introduced in the control mode decider. The purpose of this multiplier is to prevent frequent switching between the speed control mode and spacing control mode. The system without this multiplier margin suffers from this frequent switching and generates multiple peaks in control effort (see Fig. 5.26), which gives the occupants a poor driving experience, and significantly increases the energy consumption of the system. And due to the frequent switching, the system will also be much more computational expensive.

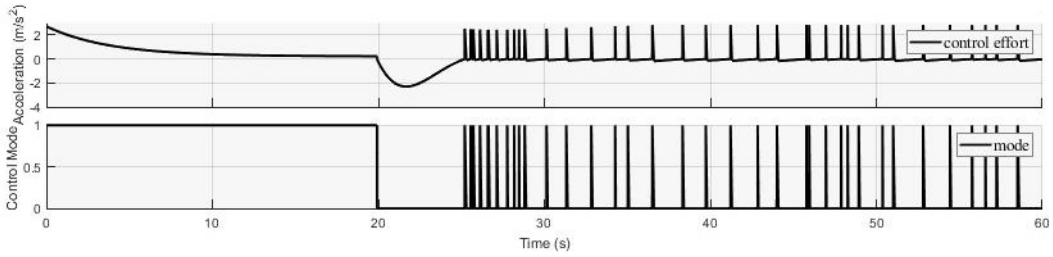


Figure 5.26: PID Controller ACC System without Multiplier Margin.

The selection of this parameter has been conducted by trying different gain values as shown in Fig. 5.27. With the increase of the gain, the peaks of control mode and control effort will be gradually suppressed, but it will bring some negative effects: the control system enters the spacing control mode too ealier, and the control effort at the moment of control mode transition will have a larger peak value, both of which are undesirable in the ACC control system.

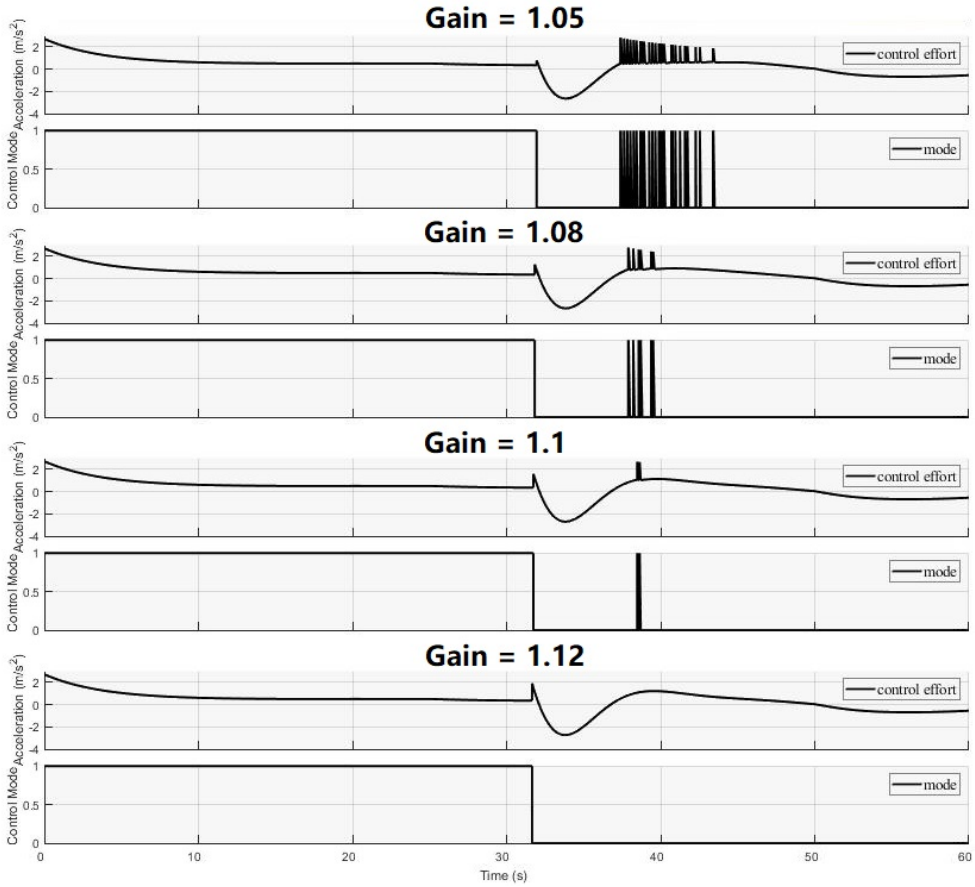


Figure 5.27: Gain Value Selection for Control Mode Decider.

5.3.2 LQR

The controllability of a system answers whether the states of the system can be controlled by the input to transfer the states to any given arbitrary state after a certain period of time. With the derived state space mode equation (25), the controllability of the system can be checked by calculating the rank of the matrix $\begin{bmatrix} B & A \cdot B \end{bmatrix} =$

$\begin{bmatrix} 0 & -1.5 \\ 1.5 & -0.0328 \end{bmatrix}$. The rank of the matrix is 2, which equals to the number of states. So, a conclusion can be made that the addressed ACC system is controllable.

For speed and spacing control modes, different Q and R are assigned to generate the gain K (see Table 5.2). And in addition, as mentioned before, in order to eliminate the steady tracking error, K_r is also assigned for each mode. Moreover, the Q and R can be adjusted to tune the performance of the vehicle, which means to be more aggressive or conservative, by simply modifying the R value. The R value here stands for the cost of the control effort in this case, while the two values in the Q matrix stand for the cost of error in speed (ego speed vs. set speed) and distance (relative distance vs. safe distance) respectively.

	Q	R	**Calculated** K	K_r
Speed Control Mode	$\begin{bmatrix} 1 & 0 \\ 0 & 50 \end{bmatrix}$	$[4000]$	$[-0.0158 \quad 0.1693]$	1.2262
Spacing Control Mode	$\begin{bmatrix} 30 & 0 \\ 0 & 1 \end{bmatrix}$	$[1000]$	$[-0.1732 \quad 0.4672]$	2.72

Table 5.2: Q, R, K and K_r of LQR Controller.

By applying the listed K and K_r into the control system, the performance of the LQR ACC control system under disturbances (road slope and wind) and continuous changing lead speed is shown in Figs. 5.28-5.31. The LQR system well fulfills the requirements of the ACC system for speed tracking and spacing tracking, with even more preferred control effort compared with the PID control system.

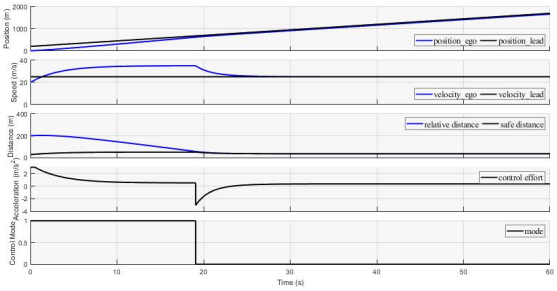


Figure 5.28: LQR, Consistent Lead Speed, No Disturbances.

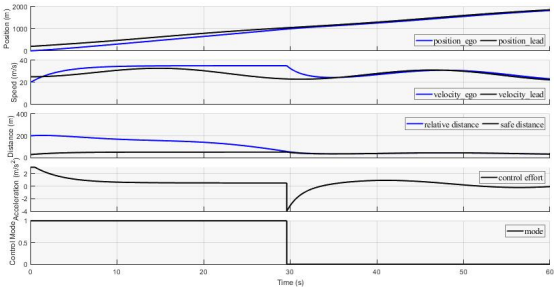


Figure 5.29: LQR, Inconsistent Lead Speed, No Disturbances.

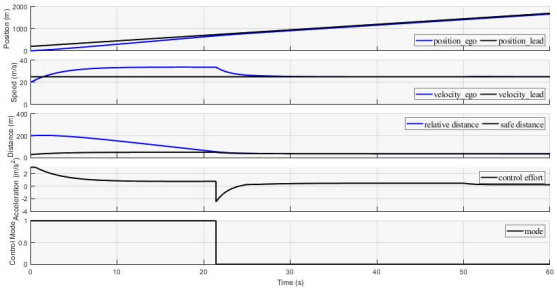


Figure 5.30: LQR, Consistent Lead Speed, With Disturbances.

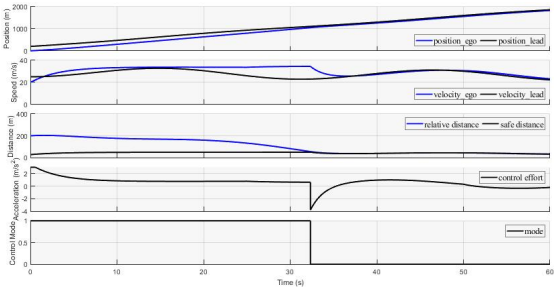


Figure 5.31: LQR, Inconsistent Lead Speed, With Disturbances.

5.3.3 Reinforcement Learning

Reward shaping is always the most challenging, and time-consuming, but also an interesting part of reinforcement learning. Similar to conventional control methods, reward shaping plays the role of a feedback block. But in reinforcement learning, reward shaping is the feedback from the environment to the agent. And by shaping the reward function, it is possible to "guide" the agent to take actions to achieve a preferred result.

In this ACC system, the followings are the main concerns:

1. Action effort that the agent is taken (see 'action effort' in Fig. 5.32);

2. The error between ego speed and set speed when the speed control mode is on (see 'speed control' in Fig. 5.32), and a small error (smaller or equal to 1 m/s) is even higher appreciated (see 'speed accuracy' in Fig. 5.32);

3. The error between relative distance and safe distance when the spacing control mode is on (see 'spacing control' in Fig. 5.32), and a small error (smaller or equal to 3 m) is even higher appreciated (see 'spacing accuracy' in Fig. 5.32);

4. Slow ego speed (smaller or equal to 3 m/s, see 'negative speed' in Fig. 5.32) and dangerous relative distance (smaller or equal to 3m, see 'collision avoidance' in Fig. 5.32) are highly disliked.

Based on these issues, each term of the reward equation is given different weight values after multiple training tests, and the reward function is established as follows (see Equation 5.3 and Fig. 5.32):

$$\begin{aligned} Reward = -1 \cdot \mu^2 - 90 \cdot (v_{ego} - v_{set})^2 + 320 \cdot S1 \\ -180 \cdot (d_{rel} - d_{safe})^2 + 300 \cdot S2 - 50000 \cdot S3 - 100000 \cdot S4 \end{aligned} \tag{5.3}$$

where

$$S1 = \begin{cases} 1, & (v_{ego} - v_{set})^2 \leq 1 \\ 0, & other \end{cases} \tag{5.4}$$

$$S2 = \begin{cases} 1, & (d_{rel} - d_{safe})^2 \leq 9 \\ 0, & other \end{cases} \tag{5.5}$$

$$S3 = \begin{cases} 1, & v_{ego} \leq 3 \\ 0, & other \end{cases} \tag{5.6}$$

$$S4 = \begin{cases} 1, & d_{rel} \leq 3 \\ 0, & other \end{cases} \tag{5.7}$$

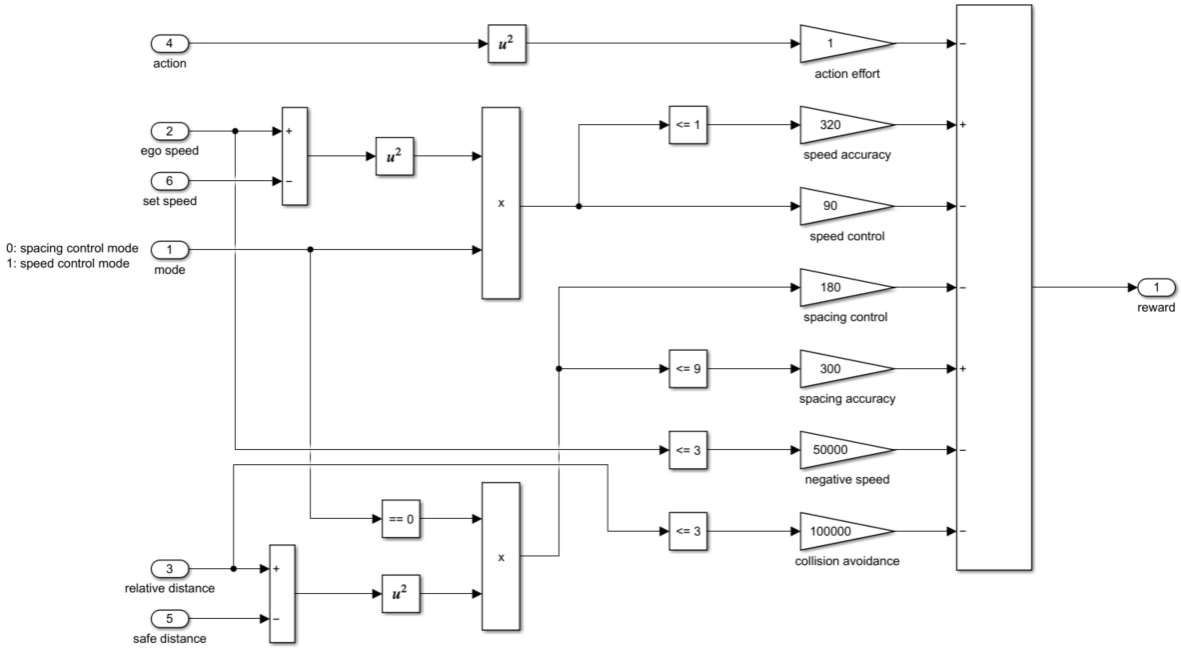


Figure 5.32: Reward Definition.

Parameter	Value
Critic Optimizer	adam
Critic Learn Rate	1e-3
Critic Gradient Threshold	1
Critic L2 Regularization Factor	2e-4
Actor Optimizer	adam
Actor Learn Rate	1e-3
Actor Gradient Threshold	1
Actor L2 Regularization Factor	1e-5
Agent Sample Time	0.1 s
Agent Discount Factor	0.9
Agent Mini Batch Size	200
Agent Experience Buffer Length	1e6
Agent Target Smooth Factor	5e-3
Agent Noise Mean Attraction Constant	1
Agent Noise Variance	0.6
Training Max Episodes	1000
Training Stop Training Criteria	AverageReward
Training Stop Training Value	10000
Training Score Averaging Window Length	5
Random Number Generator	rng(0)

Table 5.3: RL Training Parameters.

The tuning of gains can also be time-consuming and is highly dependent on the simulation result case by case. For example, increasing the gain 'speed control' can help to achieve a better speed control performance of the system, while it may also lead to a poor spacing control result due to the high stickiness to the desired set-speed. Another finding is that the accuracy has not been significantly improved by simply improving the gains of 'speed accuracy' or 'spacing accuracy. Additionally, high values of 'negative speed' and 'collision avoidance' are selected here to maintain a reasonable speed and avoid collision between the two vehicles.

In addition, the parameters in Table 5.3 are used in the agent training. It is worth mentioning that the random number generator is used to address the reproducibility issue of RL when determining the values of parameters. With random actions and random noise exploration, training the same agent with the same parameters can often lead to different results, as the training depends on the model initialization and choice of actions; both are random in the earlier stages. One way to get repeatability is to set the random seed to a specific value. In MATLAB, it can be done using the **rng()** keyword. Often in research, the same agent is run 4 to 5 times with different

seeds and an average result with the upper and lower bounds is to be plotted. Here in this project, a fixed seed is selected so that a repeatable result is guaranteed.

Different combinations of the parameters have been tried. Although most of them were not satisfactory, with the settings and parameters listed above, the training gives a promising result as shown in Figs. 5.33 and 5.34.

Some key parameters to tune are learning rate, mini-batch size, etc. The learning rate defines the step size at each iteration, which represents the speed at which a machine-learning model learns. A too-small learning rate will cause the system to fail to achieve or take too much time to achieve the desired goal, while an excessively large learning rate will cause the system to fail to reach the optimal solution. The idea of mini-batch size is relative to the full batch. It defines the amount of data included in each sub-epoch weight change. A suitable mini-batch size will help to save computational time while maintaining accuracy.

In the reinforcement learning episode manager, it can be seen that although there are several troughs in the process, the overall trend of the reward is gradually increasing, which means that the performance of the agent is gradually improving.

In addition, for training with only 1000 episodes, the training time exceeds 3 hours (see Fig. 5.33), which shows that reinforcement learning is a very time-consuming process.

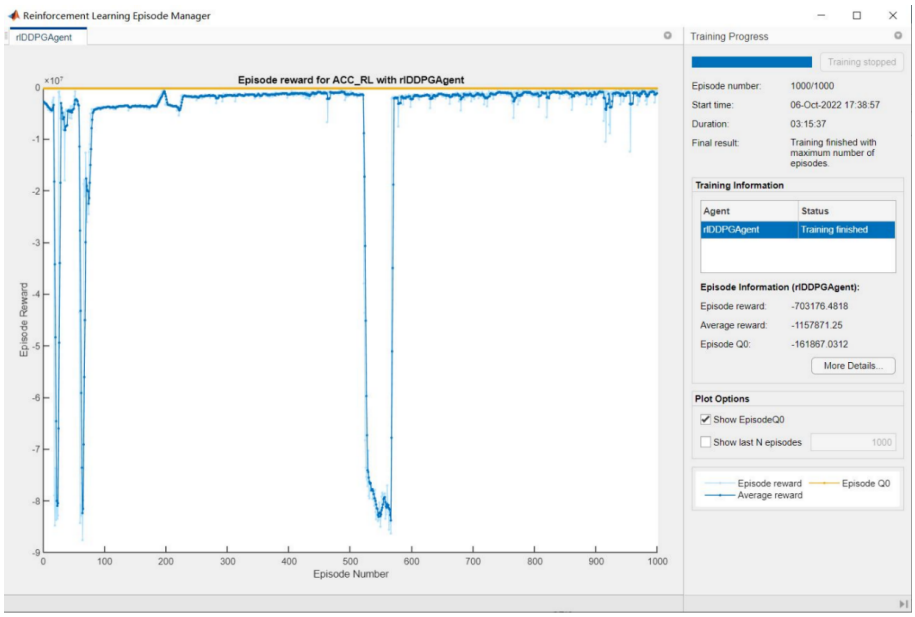


Figure 5.33: RL Training Reward Manager.

From Fig. 5.34, it can be observed that the speed of the ego is driven by the agent from the initial state of 0 m/s to around the target speed of 35 m/s at the beginning phase as desired, and the ego shows a tendency to keep this speed until the relative distance is approaching the safe distance. The ego speed starts to drop significantly since then, and successfully maintains a safe distance when it reaches so. At the same time, despite the fluctuations in the control effort, the overall output of the system is still reasonable.

For the fluctuations, a few tries have been done, for instance by introducing the derivative of the control effort into the reward function, to improve by shaping the reward function, but no satisfactory agent has been trained so far.

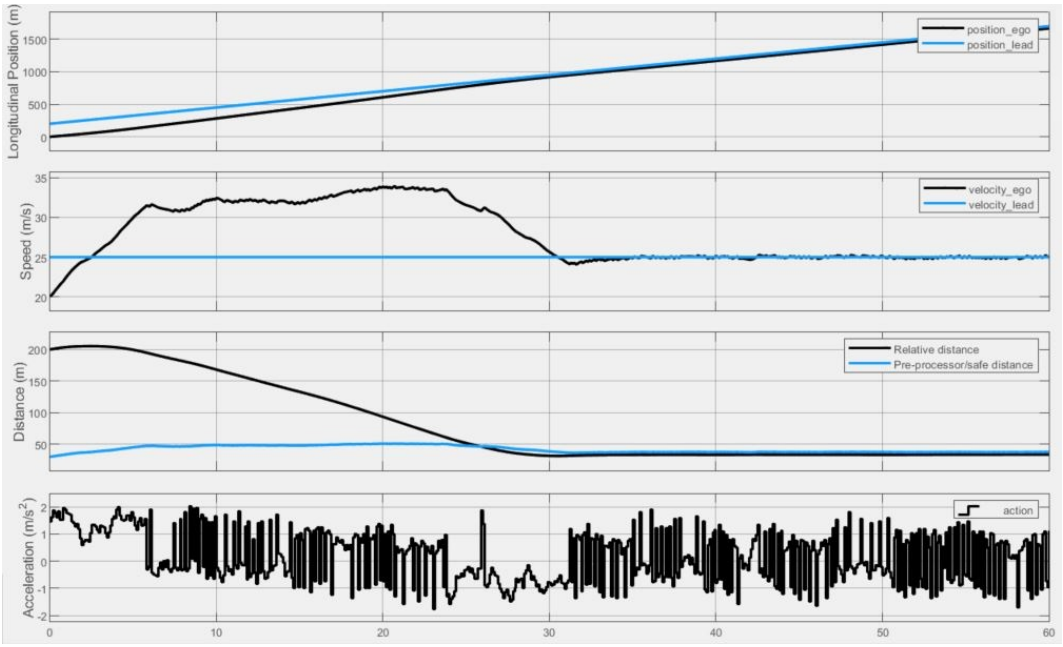


Figure 5.34: RL Training Result.

Same as in PID and LQR design, the performance of the RL controller ACC system is estimated as shown in Figs. 5.35-5.38. It can be observed that for both external disturbances (road slope and wind) and the continuously changing lead speed, the RL controller has certain adaptability and can adjust the output to ensure the speed and distance requirements in the two ACC working modes. But in the spacing control mode, strong oscillation of the control effort has been noticed, and this oscillation has an extremely negative impact on the occupants' experience and the lifetime of the vehicle, which needs to be furtherly optimized in the further design.

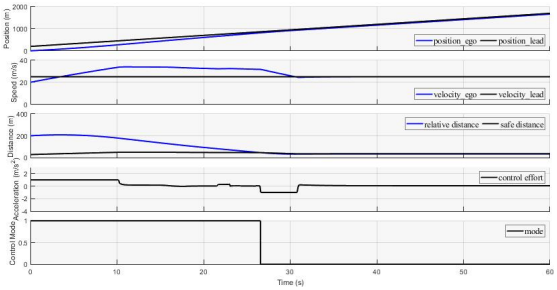


Figure 5.35: RL, Consistent Lead Speed, No Disturbances.

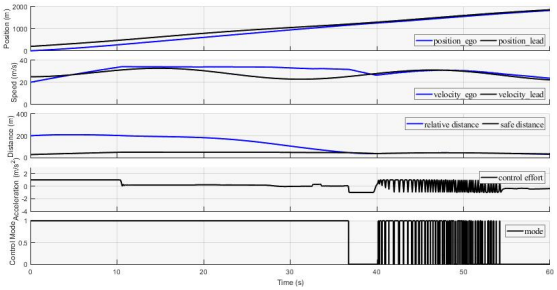


Figure 5.36: RL, Inconsistent Lead Speed, No Disturbances.

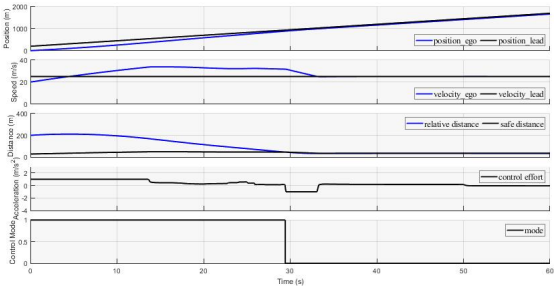


Figure 5.37: RL, Consistent Lead Speed, With Disturbances.

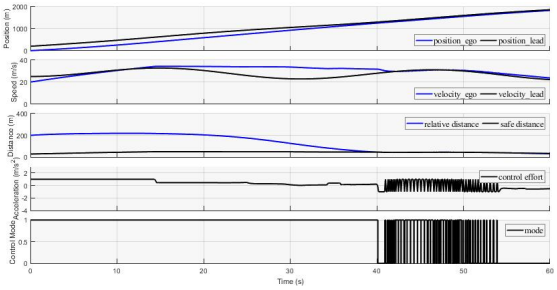


Figure 5.38: RL, Inconsistent Lead Speed, With Disturbances.

5.4 Validation

In order to verify the reliability of these three control systems in different working scenarios, the following two sets of simulations are designed.

Simulation 1: Different set-speeds

Under this simulation condition, the lead speed is constant at 25 m/s, and the set-speed of the ego vehicle is set at 15, 20, 25, 40, 35 and 40 m/s respectively, regardless of the impact from the road slope and wind.

Simulation 2: Different lead speeds

Under this simulation condition, the set-speed of the ego vehicle is constant at 35 m/s, and the lead speed is 15, 20, 25 and 30 m/s respectively, regardless of the impact from the road slope and wind.

From the summarized results (see Tables 5.4 and 5.5), both PID and LQR controllers work sufficiently under all designed working conditions, while the RL controller fails to fully perform properly in the low-speed ranges.

Controller	v_set (m/s)					
	15	20	25	30	35	40
PID	OK	OK	OK	OK	OK	OK
LQR	OK	OK	OK	OK	OK	OK
RL	NOK	NOK	OK	OK	OK	OK

Table 5.4: Set-Speeds Simulations.

Controller	Lead Speed (m/s)			
	15	20	25	30
PID	OK	OK	OK	OK
LQR	OK	OK	OK	OK
RL	NOK	OK	OK	OK

Table 5.5: Lead Speeds Simulations.

For detailed simulation results and further improvements of the design required, see:

- Appendix A: PID Controller, Different Set-speeds
- Appendix B: PID Controller, Different Lead Speeds
- Appendix C: LQR Controller, Different Set-speeds
- Appendix D: LQR Controller, Different Lead Speeds
- Appendix E: RL Controller, Different Set-speeds
- Appendix F: RL Controller, Different Lead Speeds

Chapter 6

Conclusion and Future Work

In general, PID and LQR can successfully accomplish all designed validation tests.

But LQR is preferable due to several reasons listed as follows:

- LQR is more intuitive to tune, from the design perspective;
- LQR has no peaks in action effort, which means more comfortable from drivers' and passengers' perspectives;
- LQR is suitable for more complex control systems, for example, a system that will take steering angle into consideration, in which PID controller designs will be way more complicated.

RL controller fails in 2 set-speed tests and 1 lead-speed test. A more comprehensive RL controller needs to be designed to cover more working conditions. But this design shows the potential of RL in the field of autonomous driving systems.

The results also show that the machine learning methodology may help the engineers with a problem that is really difficult or even impossible to define the problem in a reasonable mathematical way. But for more conventional problems, it is still beneficial to apply the domain knowledge and use the traditional methods from other disciplines to solve the problem, more efficiently and accurately. As the design process of the RL controller is a 'black box', how to design a validation test to cover as many as working conditions could be a real challenge.

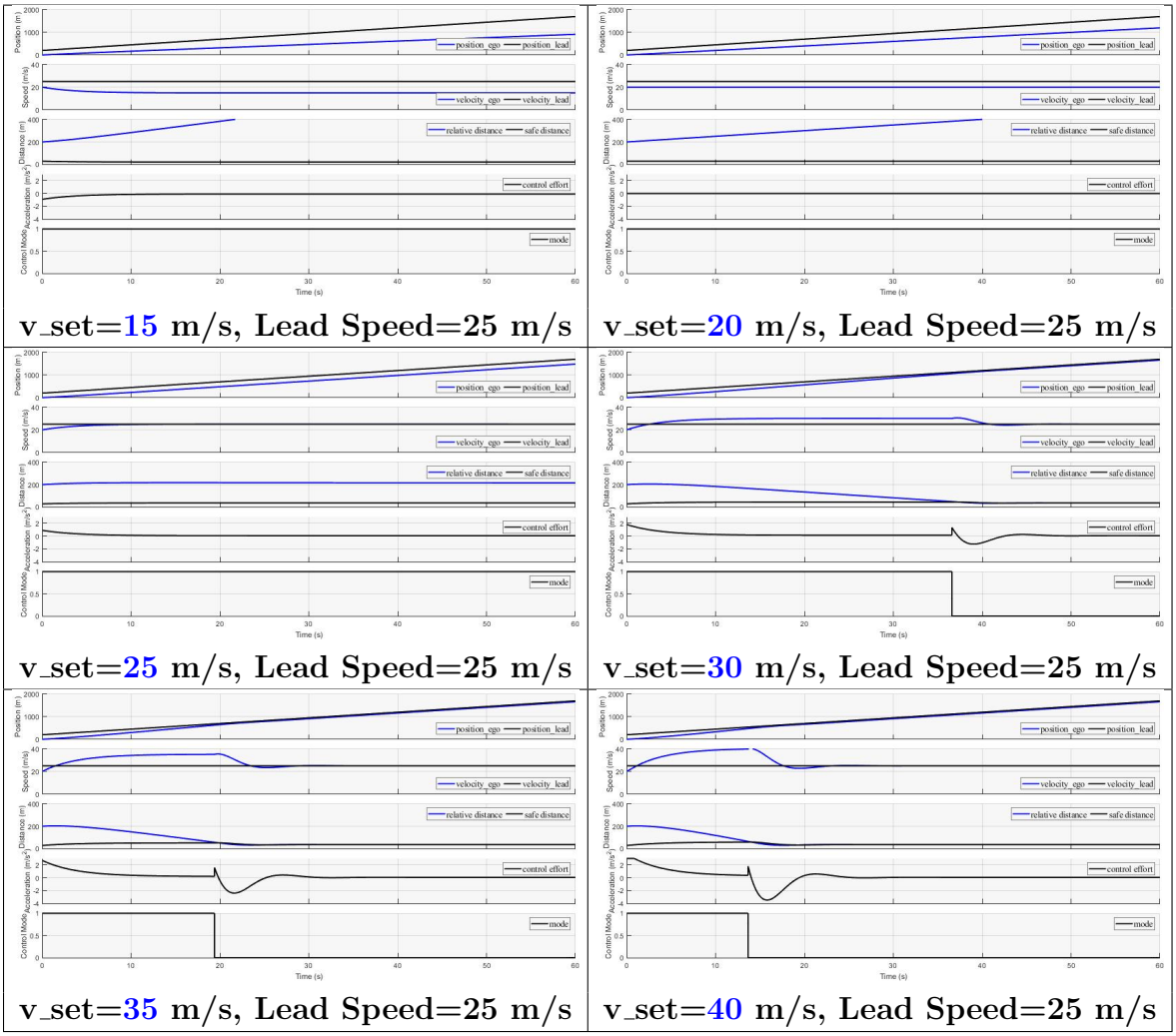
For the further development of the control systems, a few ideas are worth studying:

- Controller should be furtherly tuned for a better balance between the performance, driving experience and fuel consumption;

- Air resistance caused by vehicle speed should be taken into consideration in the simulations which ignore the external disturbances;
- Steering angle should be taken into account, which makes the design more realistic;
- More driving scenarios should be considered, like cut-in and cut-out conditions;
- RL controller needs to be improved to satisfy more working conditions;
- Two RL controllers can be introduced in one control system to work separately for speed control and spacing control mode;
- RL method can be used in PID and LQR controller design to tune the parameters;
- Real road and traffic data can be introduced for training to make the RL ACC system achieve better performance;
- Dual drive modes (performance driven/ fuel consumption driven) for the ACC system can be developed, especially with the help of the LQR controller;
- MPC controllers can be designed to compare with existing designs. Considering that MPC takes into account the system state for a future period of time in the calculation, compared with LQR, MPC has the potential to reduce the control effort peak at the instant when the modes are switched, thereby optimizing the system energy consumption and providing a better driving experience.

Appendix A

PID: Different Set-speeds



v_set=15 m/s, Lead Speed=25 m/s

v_set=20 m/s, Lead Speed=25 m/s

v_set=25 m/s, Lead Speed=25 m/s

v_set=30 m/s, Lead Speed=25 m/s

v_set=35 m/s, Lead Speed=25 m/s

v_set=40 m/s, Lead Speed=25 m/s

v_set = 15 m/s, Lead Speed = 25 m/s:

Ego speed decreases from 20 m/s to v_set speed and maintains the speed properly. The relative distance continues to increase because the lead is faster than the ego. No spacing mode activated. OK.

v_set = 20 m/s, Lead Speed = 25 m/s:

Ego speed maintains at v_set speed properly. OK. The relative distance continues to increase because the lead is faster than the ego. No spacing mode activated. OK.

v_set = 25 m/s, Lead Speed = 25 m/s:

Ego speed increases from 20 m/s to v_set speed and maintains the speed properly. The relative distance does not change because the lead speed is same as the ego speed. No spacing mode activated. OK.

v_set = 30 m/s, Lead Speed = 25 m/s:

Ego speed increases from 20 m/s to v_set speed and maintains the speed properly until the relative distance approaches the safe distance. Then the relative distance tracks the safe distance. Both modes function properly. OK.

v_set = 35 m/s, Lead Speed = 25 m/s:

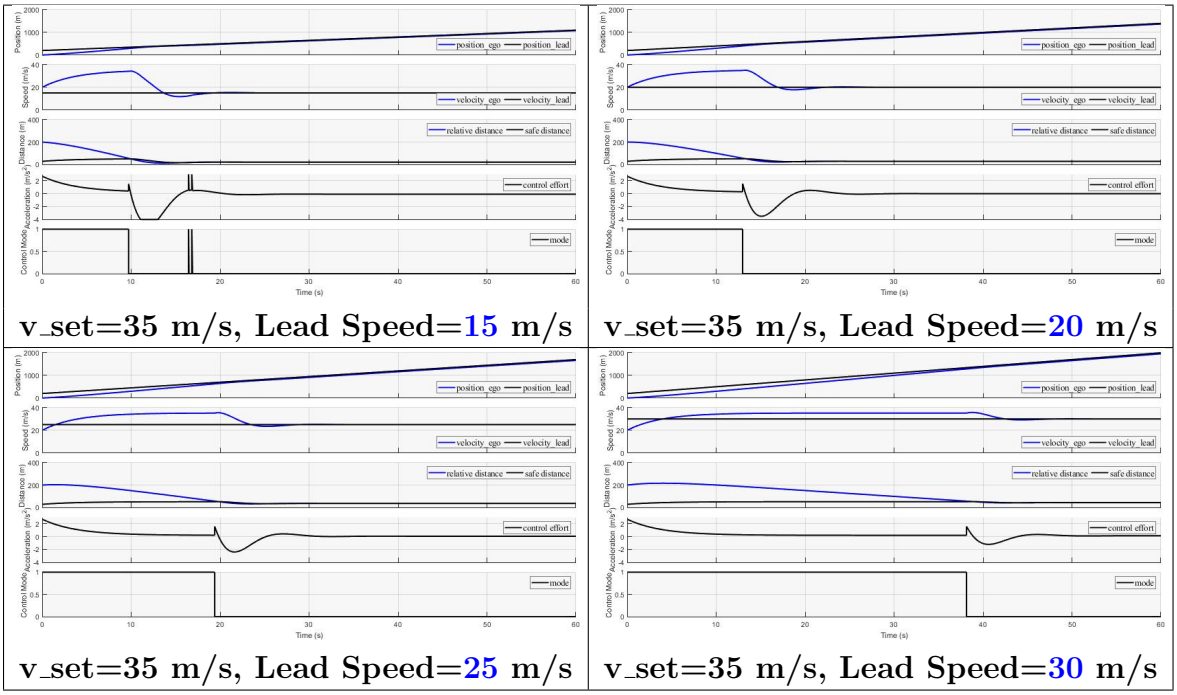
Ego speed increases from 20 m/s to v_set speed and maintains the speed properly until the relative distance approaches the safe distance. Then the relative distance tracks the safe distance. Both modes function properly. OK.

v_set = 40 m/s, Lead Speed = 25 m/s:

Ego speed increases from 20 m/s to v_set speed and maintains the speed shortly until the relative distance approaches the safe distance. Then the relative distance tracks the safe distance with small overshoot at the beginning phase. Both modes function properly. OK.

Appendix B

PID: Different Lead Speeds



v_set=35 m/s, Lead Speed=15 m/s | **v_set=35 m/s, Lead Speed=20 m/s**

v_set=35 m/s, Lead Speed=25 m/s | **v_set=35 m/s, Lead Speed=30 m/s**

v_set = 35 m/s, Lead Speed = 15 m/s:

Ego speed increases from 20 m/s but the spacing mode gets activated before the ego speed can reach v_set speed. Ego maintains the safe distance eventually, but an overshoot of the distance should be of concern. OK.

v_set = 35 m/s, Lead Speed = 20 m/s:

Ego speed increases from 20 m/s but the spacing mode gets activated shortly after the ego speed can reach v_set speed. Ego maintains the safe distance eventually, but an overshoot of the distance should be of concern. OK.

v_set = 35 m/s, Lead Speed = 25 m/s:

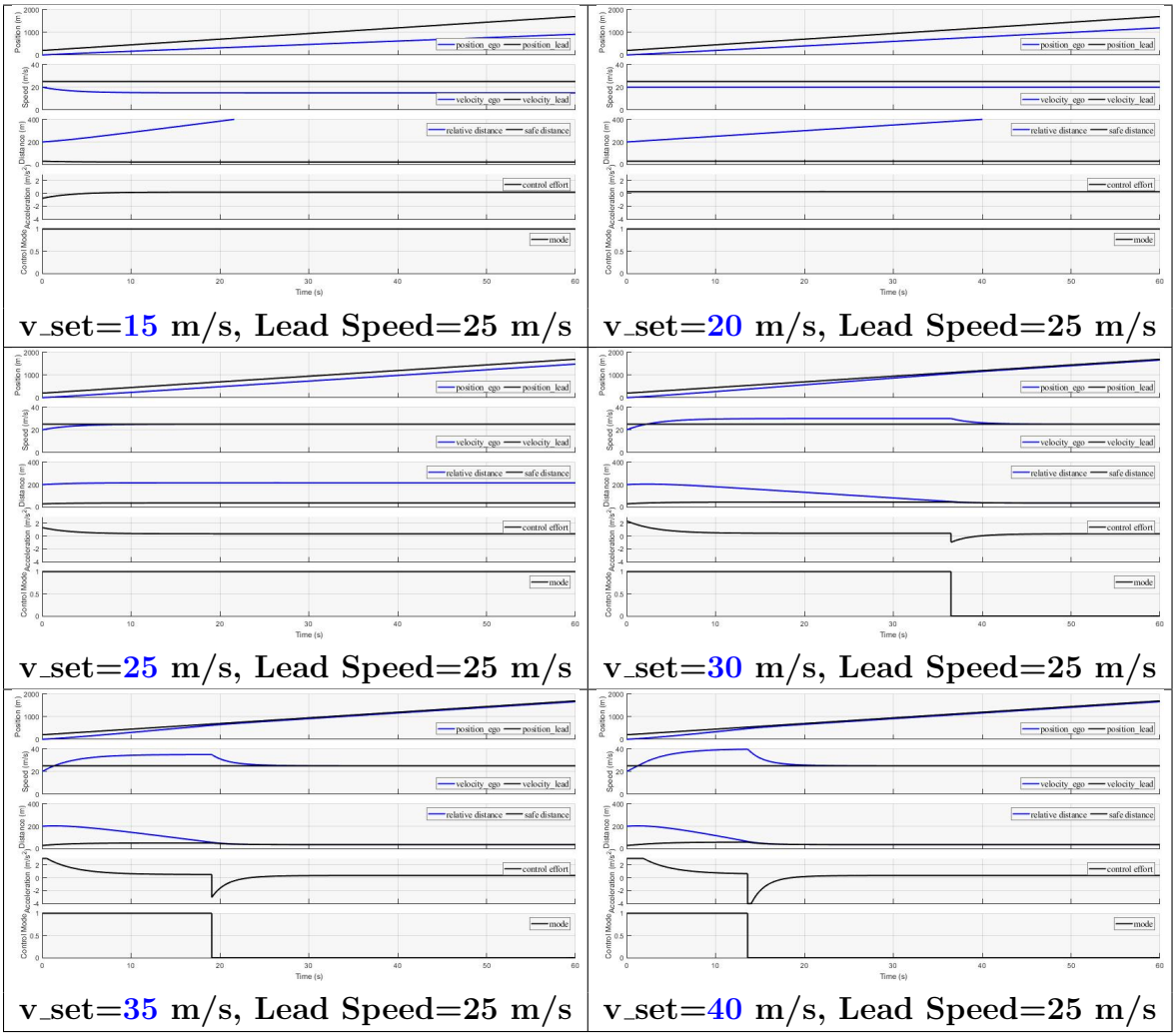
Ego speed increases from 20 m/s and maintains at the v_set speed until the spacing mode gets activated. Ego maintains the safe distance eventually, but an overshoot of the distance should be of concern. OK.

v_set = 35 m/s, Lead Speed = 30 m/s:

Ego speed increases from 20 m/s and maintains at the v_set speed until the spacing mode gets activated. Ego maintains the safe distance eventually with an acceptable overshoot. OK.

Appendix C

LQR: Different Set-speeds



v_set=15 m/s, Lead Speed=25 m/s

v_set=20 m/s, Lead Speed=25 m/s

v_set=25 m/s, Lead Speed=25 m/s

v_set=30 m/s, Lead Speed=25 m/s

v_set=35 m/s, Lead Speed=25 m/s

v_set=40 m/s, Lead Speed=25 m/s

v_set = 15 m/s, Lead Speed = 25 m/s:

Ego speed decreases from 20 m/s to v_set speed and maintains the speed properly. The relative distance continues to increase because the lead is faster than the ego. No spacing mode activated. OK.

v_set = 20 m/s, Lead Speed = 25 m/s:

Ego speed maintains at v_set speed properly. OK. The relative distance continues to increase because the lead is faster than the ego. No spacing mode activated. OK.

v_set = 25 m/s, Lead Speed = 25 m/s:

Ego speed increases from 20 m/s to v_set speed and maintains the speed properly. The relative distance does not change because the lead speed is same as the ego speed. No spacing mode activated. OK.

v_set = 30 m/s, Lead Speed = 25 m/s:

Ego speed increases from 20 m/s to v_set speed and maintains the speed properly until the relative distance approaches the safe distance. Then the relative distance tracks the safe distance. Both modes function properly. OK.

v_set = 35 m/s, Lead Speed = 25 m/s:

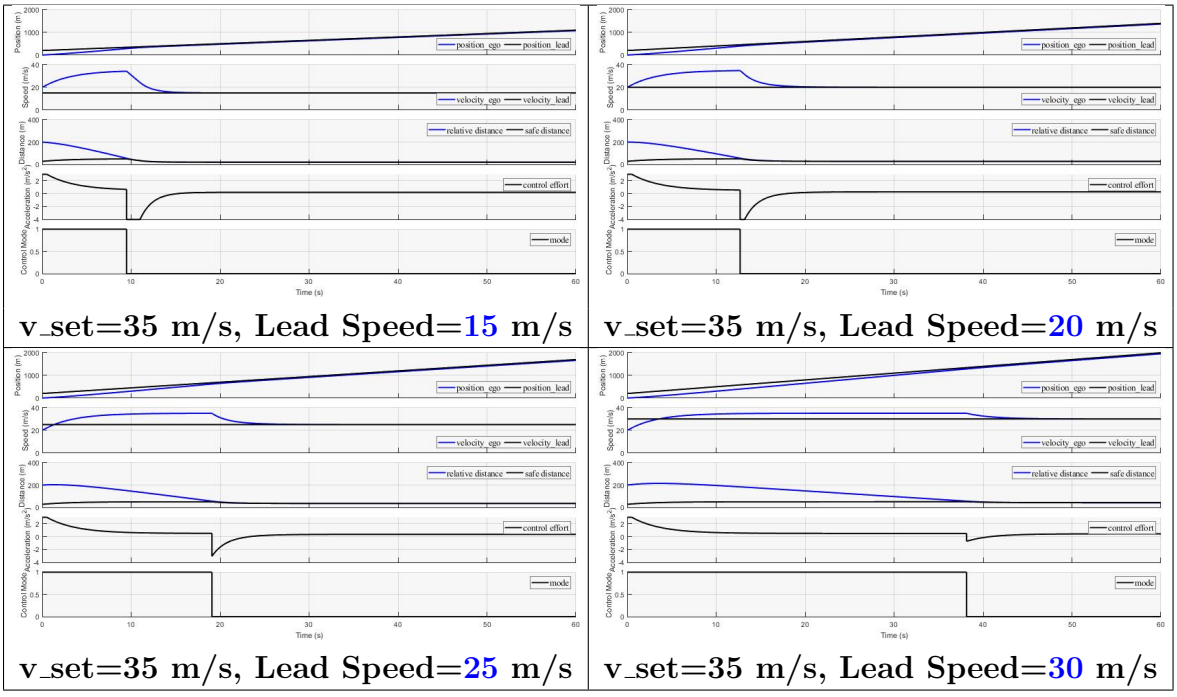
Ego speed increases from 20 m/s to v_set speed and maintains the speed properly until the relative distance approaches the safe distance. Then the relative distance tracks the safe distance. Both modes function properly. OK.

v_set = 40 m/s, Lead Speed = 25 m/s:

Ego speed increases from 20 m/s to v_set speed and maintains the speed shortly until the relative distance approaches the safe distance. Then the relative distance tracks the safe distance without overshoot. Both modes function properly. OK.

Appendix D

LQR: Different Lead Speeds



v_set=35 m/s, Lead Speed=15 m/s **v_set=35 m/s, Lead Speed=20 m/s**

v_set=35 m/s, Lead Speed=25 m/s **v_set=35 m/s, Lead Speed=30 m/s**

v_set = 35 m/s, Lead Speed = 15 m/s:

Ego speed increases from 20 m/s but the spacing mode gets activated before the ego speed can reach v_set speed. The ego maintains a safe distance eventually without overshoot. OK.

v_set = 35 m/s, Lead Speed = 20 m/s:

Ego speed increases from 20 m/s but the spacing mode gets activated shortly after the ego speed can reach v_set speed. The ego maintains a safe distance eventually without overshoot. OK.

v_set = 35 m/s, Lead Speed = 25 m/s:

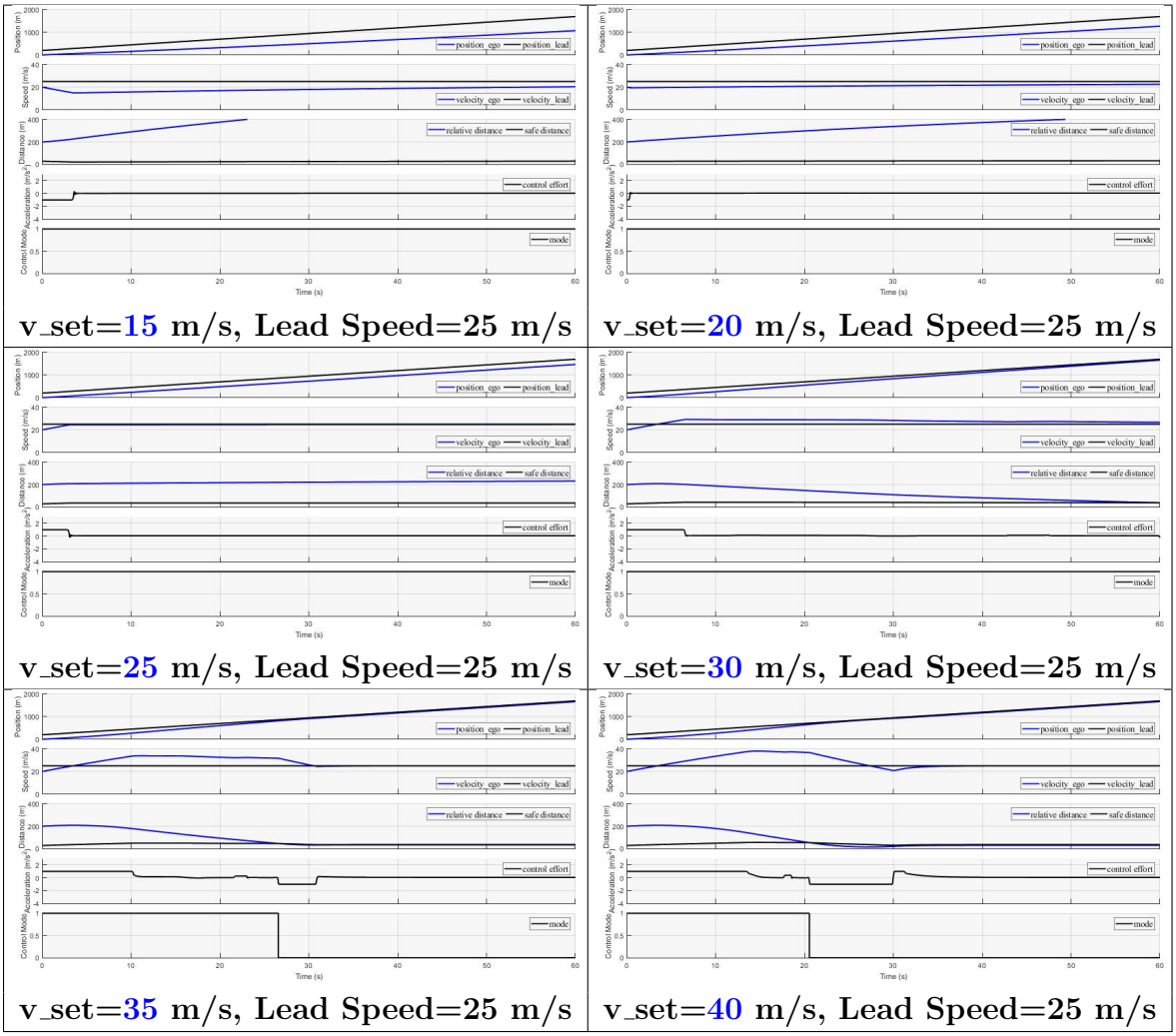
Ego speed increases from 20 m/s and maintains at the v_set speed until the spacing mode gets activated. The ego maintains a safe distance eventually without overshoot. OK.

v_set = 35 m/s, Lead Speed = 30 m/s:

Ego speed increases from 20 m/s and maintains at the v_set speed until the spacing mode gets activated. The ego maintains a safe distance eventually without any overshoot. OK.

Appendix E

RL: Different Set-speeds



v_set=15 m/s, Lead Speed=25 m/s

v_set=20 m/s, Lead Speed=25 m/s

v_set=25 m/s, Lead Speed=25 m/s

v_set=30 m/s, Lead Speed=25 m/s

v_set=35 m/s, Lead Speed=25 m/s

v_set=40 m/s, Lead Speed=25 m/s

v_set = 15 m/s, Lead Speed = 25 m/s:

Ego speed decreases from 20 m/s but fails to maintain at v_set. NOK.

v_set = 20 m/s, Lead Speed = 25 m/s:

Ego speed fails to maintain at v_set. NOK.

v_set = 25 m/s, Lead Speed = 25 m/s:

Ego speed increases from 20 m/s to close to v_set and maintains at the speed. OK.

v_set = 30 m/s, Lead Speed = 25 m/s:

Ego speed increases from 20 m/s to close to v_set speed and maintains the speed until the relative distance approaches the safe distance. Then the relative distance tracks the safe distance. Both modes function properly. Small oscillation appears in acceleration and should get optimized in further designs. In addition, steep change of acceleration should be optimized for a better driving experience. OK.

v_set = 35 m/s, Lead Speed = 25 m/s:

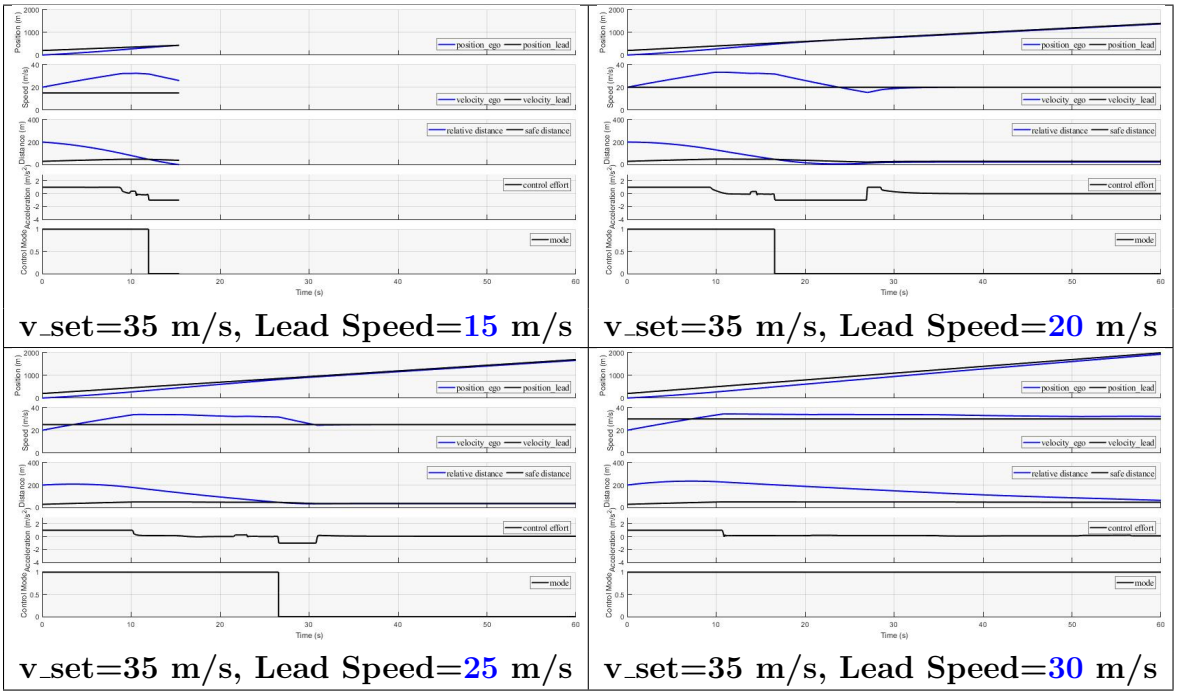
Ego speed increases from 20 m/s to close to v_set speed and maintains the speed until the relative distance approaches the safe distance. Small oscillation appears in acceleration and should get optimized in further designs. In addition, steep change of acceleration should also be optimized for a better driving experience. OK.

v_set = 40 m/s, Lead Speed = 25 m/s:

Ego speed increases from 20 m/s to close to v_set speed and maintains the speed until the relative distance approaches the safe distance. Then the relative distance tracks the safe distance without overshoot. Both modes function properly. Small oscillation appears in acceleration and should get optimized in further designs. OK.

Appendix F

RL: Different Lead Speeds



v_set=35 m/s, Lead Speed=15 m/s **v_set=35 m/s, Lead Speed=20 m/s**

v_set=35 m/s, Lead Speed=25 m/s **v_set=35 m/s, Lead Speed=30 m/s**

v_set = 35 m/s, Lead Speed = 15 m/s:

Ego hits lead. It shows the effort of ego to avoid the collision, but the relative distance still gets negative. NOK.

v_set = 35 m/s, Lead Speed = 20 m/s:

Ego speed increases from 20 m/s and maintains at the v_set shortly until the relative distance approaches the safe distance. The ego maintains a relative distance close to a safe distance eventually, but an overshoot is noticeable, which should be optimized in further designs. In addition, steep change of acceleration should also be

optimized for a better driving experience. OK.

v_set = 35 m/s, Lead Speed = 25 m/s:

Ego speed increases from 20 m/s and maintains at the v_set until the relative distance approaches the safe distance. The ego maintains a relative distance close to a safe distance with a small overshoot. In addition, steep change of acceleration should also be optimized for a better driving experience. OK.

v_set = 35 m/s, Lead Speed = 30 m/s:

Ego speed increases from 20 m/s and maintains at the v_set speed until the spacing mode gets activated. The ego maintains the safe distance eventually without any overshoot. The steep change of acceleration should also be optimized for a better driving experience. OK.

Bibliography

[1] Wikipedia, "Carl benz," https://en.wikipedia.org/wiki/Carl_Benz, Oct. 2022, accessed: 2022-Oct-04.

[2] M. Systems, "Key Finds From The Intelligent Transporation Systems (ITS) Program: What Have We Learned?" U.S. Department of Transportation, Tech. Rep., 1996.

[3] HEARST AUTOS RESEARCH, "What Is Adaptive Cruise Control?" https://www.caranddriver.com/research/a32813983/adaptive-cruise-control/, Jun. 2020, accessed: 2022-Oct-04.

[4] Wikipedia, "PID controller," https://en.wikipedia.org/wiki/PID_controller, Sep. 2022, accessed: 2022-Oct-05.

[5] Sandeep VM, "When and why to use P, PI, PD and PID Controller?" https://medium.com/@svm161265/when-and-why-to-use-p-pi-pd-and-pid-controller-73729a708bb5, Oct. 2021, accessed: 2022-Oct-04.

[6] Christopher Lum, "Practical Implementation Issues with a PID Controller," https://www.youtube.com/watch?v=yr6om0e0oAQ&list=PLxdnSsBqCrrF9KOQRB9ByfB0EUMwnLO9o&index=33&t=6199s, accessed: 2022-Oct-07.

[7] Wikipedia, "Ziegler–Nichols method," https://en.wikipedia.org/wiki/Ziegler%E2%80%93Nichols_method, Dec. 2021, accessed: 2022-Oct-04.

[8] Christopher Lum, "Introduction to Linear Quadratic Regulator (LQR) Control," https://www.youtube.com/watch?v=wEevt2a4SKI&list=PLxdnSsBqCrrF9KOQRB9ByfB0EUMwnLO9o&index=26&t=2641s, Dec. 2018, accessed: 2022-Oct-04.

[9] Brian Douglas, "What Is Linear Quadratic Regulator (LQR) Optimal Control? — State Space, Part 4," https://www.youtube.com/watch?v=E_RDCFOlJx4&t=505s, Jan. 2019, accessed: 2022-Oct-04.

[10] S. Dude, "The 3 Basic Paradigms of Machine Learning," https://somedudesays.com/2020/09/the-3-basic-paradigms-of-machine-learning/, Sep. 2020, accessed: 2022-Oct-04.

[11] Piyush Verma, "The 3 Basic Paradigms of Machine Learning," https://www.synopsys.com/ai/what-is-reinforcement-learning.html, Apr. 2021, accessed: 2022-Oct-04.

[12] Wikipedia, "Model predictive control," https://en.wikipedia.org/wiki/Model_predictive_control, Dec. 2022, accessed: 2022-Oct-05.

[13] MathWorks, "Understanding Model Predictive Control, Part 1: Why Use MPC?" https://www.mathworks.com/videos/understanding-model-predictive-control-part-1-why-use-mpc--1526484715269.html#:~:text=Here%20are%20some%20of%20the,also%20affects%20the%20first%20output., accessed: 2022-Nov-17.

[14] ISO/TC 204/WG 14 - Vehicle/roadway warning and control systems, "ISO 15622 Intelligent transport systems - Adaptive cruise control systems - Performance requirements and test procedures," Technical Committee ISO/TC 204, Intelligent transport systems, Tech. Rep., Sep. 2018.

[15] Rui Neves-Silva, "CC - Lab 0.2 - Development of the Plant Model," https://www.youtube.com/watch?v=g3VkzABqHsI&t=1005s, Mar. 2020, accessed: 2022-Oct-05.

[16] D. J. Yeong, G. Velasco-Hernandez, D. Barry, and P. J. Walsh, "Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review," IMaR Research Centre, Tech. Rep., Feb. 2021.

[17] Engineering ToolBox, "Drag Coefficient," https://www.engineeringtoolbox.com/drag-coefficient-d_627.html, 2004, accessed: 2022-Oct-04.

[18] ——, "Rolling Resistance," https://www.engineeringtoolbox.com/rolling-friction-resistance-d_1303.html, 2008, accessed: 2022-Oct-04.

[19] Wikipedia, "Automobile drag coefficient," https://en.wikipedia.org/wiki/Automobile_drag_coefficient#:~:text=Typical%20drag%20coefficients,-Learn%20more&text=The%20average%20modern%20automobile%20achieves,of%20body%20of%20the%20vehicle., Sep. 2022, accessed: 2022-Oct-05.

[20] ——, "Gravitational acceleration," https://en.wikipedia.org/wiki/Gravitational_acceleration, May 2022, accessed: 2022-Oct-05.

[21] Henry C., "Solving Continuous Control environment using Deep Deterministic Policy Gradient (DDPG) agent," https://medium.com/@kinwo/solving-continuous-control-environment-using-deep-deterministic-policy-gradient-ddpg-agent-
Oct. 2018, accessed: 2022-Oct-07.

[22] MathWorks, "Deep Deterministic Policy Gradient (DDPG) Agents," https://www.mathworks.com/help/reinforcement-learning/ug/ddpg-agents.html, accessed: 2022-Oct-07.

[23] Sunny Guha, "Deep Deterministic Policy Gradient (DDPG): Theory and Implementation," https://towardsdatascience.com/deep-deterministic-policy-gradient-ddpg-theory-and-implementation-747a3010e82f#:~:text=DDPG%20being%20an%20actor%2Dcritic,a%20probability%20distribution%20over%20actions., May 2020, accessed: 2022-Oct-09.

[24] MathWorks, "RL Agent," https://www.mathworks.com/help/reinforcement-learning/ref/rlagent.html, accessed: 2022-Oct-07.