DELFT UNIVERSITY OF TECHNOLOGY

MODELING AND CONTROL OF HYBRID SYSTEMS
SC42075

Hybrid System Assignment

Authors:
Group 18
Liuyi Zhu (5808383)
Qingyi Ren (5684803)

June 19, 2023

I. Hybrid system example

In this assignment, an intersection with traffic lights shown in Fig (1) is analysed as a hybrid system and a description of the system and its corresponding hybrid automaton is given.

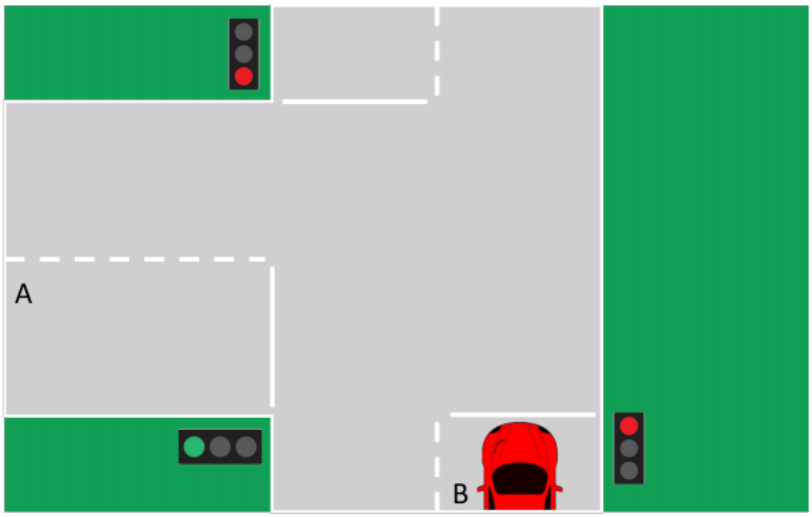

Fig. 1. An intersection with traffic lights.

A. Description of hybrid system

In this system, at the intersection with traffic lights, both lights at road A and road B are used for all directions for cars. If the light at road A is green, the car could either turn left or turn right. While the light of road B is green, cars could turn left or drive straight forward. In this case only path for cars is taken into consideration and pedestrians and bicycles are exclude. The traffic rule can be depicted as:

1) The sequential order of traffic light switching between road A and B is as follows: Initially, both roads will display a red signal, allowing other traffic to proceed. Next, road A will transition to a green or orange signal while road B remains red. After that, road A will switch back to red and then both of lights are red temporarily. Finally road A will keep red while road B will display a green or orange signal. The circle will keep going.
2) The duration of the green or orange signal on road A while road B remains red depends on the number of cars waiting on road B and the current duration. If there are more than 5 cars waiting on road B or the duration exceeds 15 seconds, the lights will shift to road A displaying a red signal and road B showing a green or orange signal.
3) The duration of the red signal on road A while road B displays a green or orange signal is determined by the number of cars waiting on road A and the current duration. If there are more than 7 cars waiting on road A or the duration surpasses 15 seconds, the lights will switch to road A showing a red signal and road B reverting to a red signal as well.

Based on that, the intersection of traffic lights can be modelled and depicted as hybrid system. There are three states of hybrid system, $x_1(t)$ and $x_2(t)$ represent the amount of cars waiting on road A and road B respectively $x_3(t)$ represents the duration of the green or orange signal for road A and road B, as

well as the duration of the dedicated time for pedestrians and bicycles to pass. For simplicity, the green and orange signals can be taken as the same which allows the cars to pass.

Then traffic lights at the intersection have three different modes $\{A, B, C\}$, where mode A represents the state when road A has a green or orange signal while road B has a red signal; mode B represents the state when road A has a red signal while road B has a green or orange signal; mode C represents the state when both road A and road B have a red signal, providing a dedicated time for pedestrians and bicycles to pass.

Mathematically, the Hybrid automaton H could be taken as a collection $H = (Q, X, f, Init, Inv, E, G, R)$ where

1) $Q = \{A, B, C\}$ is finite set of modes
   a) A: The state when road A has a green or orange signal while road B has a red signal
   b) B: The mode B represents the state when road A has a red signal while road B has a green or orange signal
   c) C: The mode C represents the state when both road A and road B have a red signal
2) $X = \{x_1(t), x_2(t), x_3(t)\}$ is set of continuous states
   a) $x_1(t) \in \{0, 1, \ldots, N_1\}$: Represents the number of cars waiting on road A where N_1 is the maximal number of cars waiting on road A
   b) $x_2(t) \in \{0, 1, \ldots, N_2\}$: Represents the number of cars waiting on road B where N_2 is the maximal number of cars waiting on road B
   c) $x_3(t) \in R^+$: Represents the duration of the green or orange signal for road A and road B, as well as the duration of the dedicated time for pedestrians and bicycles to pass
3) $f : Q \times X \to X$ is vector field
   a) In Mode A: $\dot{x} = f(x_1, x_2, x_3) = \begin{bmatrix} -8 & 6 & 1 \end{bmatrix}^T$
   b) In Mode B: $\dot{x} = f(x_1, x_2, x_3) = \begin{bmatrix} 6 & -8 & 1 \end{bmatrix}^T$
   c) In Mode C: $\dot{x} = f(x_1, x_2, x_3) = \begin{bmatrix} 12 & 6 & 1 \end{bmatrix}^T$
4) $Init \subseteq Q \times X$ is the set of initial states, in this case the initial states are set to be: $x_0 = (x_{10}, x_{20}, x_{30})^T = (5, 0, 0)^T$
5) $Inv : Q \to P(X)$ describes invariants
   a) Inv of Mode A: $(x_1, x_2, x_3) \in \{(x_1, x_2, x_3) | x_2 \leq 5, x_3 \leq 15\}$
   b) Inv of Mode B: $(x_1, x_2, x_3) \in \{(x_1, x_2, x_3) | x_1 \leq 7, x_3 \leq 15\}$
   c) Inv of Mode C: $(x_1, x_2, x_3) \in \{(x_1, x_2, x_3) | x_3 \leq 2\}$
6) $E \subseteq Q \times Q$ is the set of transitions
   a) Transition 1: From Mode A to Mode C when Guard 1 is true
   b) Transition 2: From Mode B to Mode C when Guard 2 is true
   c) Transition 3: From Mode C to Mode A when Guard 3 is true and the former mode is Mode B or not existing
   d) Transition 4: From Mode C to Mode B when Guard 3 is true and the former mode is Mode A
7) $G : E \to P(X)$ is the guard condition
   a) Guard 1: Checks if the number of cars waiting on road B is greater than 5 or the duration of road A's green or orange signal exceeds 15 seconds
   b) Guard 2: Checks if the number of cars waiting on road A is greater than 7 or the duration of road B's green or orange signal surpasses 15 seconds
   c) Guard 3: Checks if the duration of the pedestrian/bicycle crossing phase exceeds 2 seconds
8) $R : E \to P(X \times X)$ is reset map
   a) Reset of transition 1: $x_3 = 0$

b) Reset of transition 2: $x_3 = 0$
c) Reset of transition 3: $x_3 = 0$
d) Reset of transition 4: $x_3 = 0$

B. Hybrid automaton

By using the definition of Hybrid automaton H, the diagram of hybrid automaton is constructed in Fig (2).

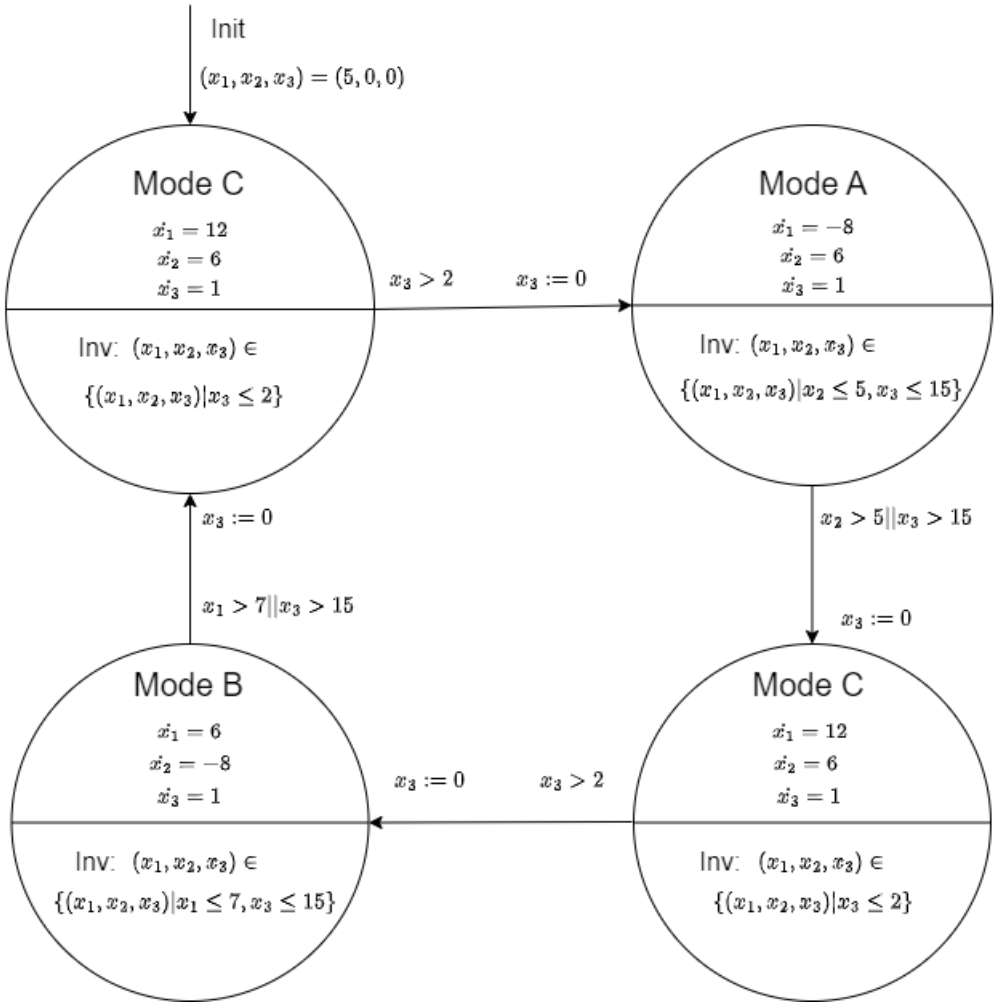

Fig. 2. Diagram of Hybrid automaton H.

In Simulink, the Hybrid automaton H is depicted in **stateflow** chart shown in Fig (3), it is worth mentioning that in order to define the next Mode from the current C Mode, the counter of previous mode is built. If the counter's value is odd, the Transition 4 is made otherwise the Transition 3 is made from Mode 3.

II. Adaptive cruise control

In this part, two cars are driving one after another are considered as the setup of adaptive cruise control(ACC) shown in Fig 4.

The goal of ACC of the two cars system is to keep the minimal distance between the two vehicles, i.e., safety distance and an adaptation to the speed which could make the differences of the speed of two

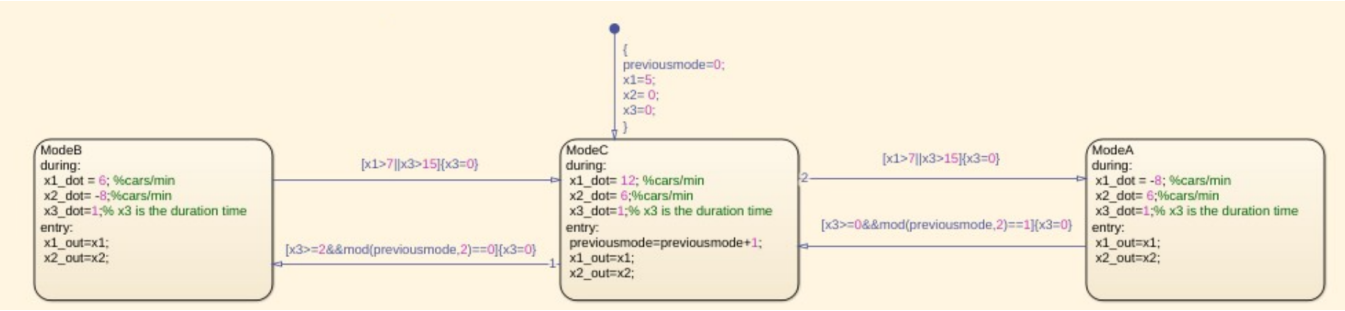

Fig. 3. The Hybrid automaton H built in Simulink.

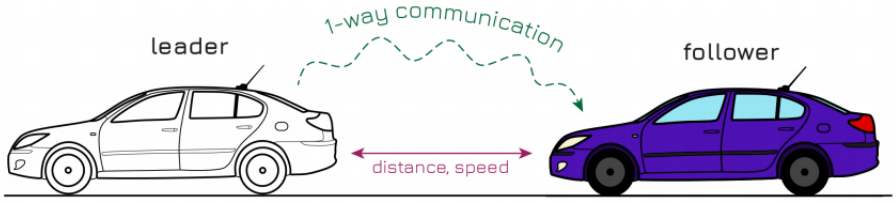

Fig. 4. ACC set-up considered in the assignment

vehicles as small as possible. To simplify, ACC of the two cars system only takes the speed adaptation control into consideration and the leading vehicle communicates its speed to the following one to make the follower track the speed of the leader.

The vehicle dynamics of the two car system could be built by Newton's second law of motion with the forces act on the vehicle (which has mass m):

$$a = \frac{F_{drive} - F_{gradient} - F_{friction}}{m} \tag{1}$$

where a is acceleration/deceleration and the forces are clarified as followed:

1) The "driving" force $F_{drive}(t)$, which is related to the throttle input $u(t)$ and the gear ratio $r(t)$ by the following equation:

$$F_{drive}(t) = \frac{b}{1 + \gamma r(t)} u(t) \tag{2}$$

where b and γ are two constants, the throttle input $u(t)$ is the control input and $r(t) \in \{1, 2\}$ indicates the gear ratio at time t. The gear ratio is taken to depend on the speed of the vehicle in the following way:

$$r(t) = \begin{cases} 1 & \text{if } v(t) < v_g \\ 2 & \text{if } v_g \leq v(t) \end{cases} \tag{3}$$

2) The road gradient resistance force $F_{gradient}(t)$ is a function of the slope of the road θ:

$$F_{gradient}(t) = mgsin(\theta)(x(t)) \tag{4}$$

3) A dynamic friction force $F_{friction}(t)$ that is proportional to the square of the speed $v(t)$ of the vehicle:

$$F_{friction}(t) = cv^2(t) \tag{5}$$

In the braking scenario, a negative throttle could be applied to the simulation. Therefore, the gear ratio influences both the acceleration and deceleration forces (acceleration and deceleration forces corresponding to positive and negative values of a separately). It is assumed that the vehicles drive in the forward direction, so the speed will always be non-negative. For the sake of passenger comfort, a maximal acceleration/deceleration is defined as:

$$|a(t)| \leq a_{comf,max} \tag{6}$$

The parameters of vehicle are shown in Appendix B.

A. *Step 2.1*

In this part, by taking the position $x(t)$ and the speed $v(t)$ as the states $[x(t), v(t)]^T$, the continuous-time state space model could be written as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x(t) \\ v(t) \end{bmatrix} = \begin{bmatrix} v(t) \\ \frac{F_{drive}-F_{gradient}-F_{friction}}{m} \end{bmatrix} = \begin{bmatrix} v(t) \\ \underbrace{\frac{b/m}{1+\gamma r(t)}u(t)}_{F_{drive}} - \underbrace{gsin\theta(x(t))}_{F_{gradient}} - \underbrace{\frac{c}{m}v^2(t)}_{F_{friction}} \end{bmatrix} \tag{7}$$

Given the $\theta = 0$, the acceleration/deceleration could be written as:

$$a = \begin{cases} \frac{b/m}{1+\gamma}u(t) - g\frac{c}{m}v^2(t) & v(t) < v_g \\ \frac{b/m}{1+2\gamma}u(t) - g\frac{c}{m}v^2(t) & v_g \leq v(t) \end{cases} \tag{8}$$

This could be taken as event-driven hybrid system. Mode 1 occurs when $v(t) < v_g$ will switch to mode 2 when $v_g \leq v(t)$. It is easily found that the velocity $v(t)$ in mode 2 is always higher than that in mode 1, thus the maximum speed v_{max} could be found in mode 2.

By given maximal throttle input $u_{max} > 0$, as stated before, the vehicle is accelerating with $a \geq 0$, which means the rate of change of velocity is positive, so when the acceleration is zero, the velocity is not changing and therefore must be at a maximum. The maximal speed v_{max} is calculated as:

$$\frac{b/m}{1+2\gamma}u_{max} - g\frac{c}{m}v^2(t) = 0 \Rightarrow v_{max} = v = \sqrt{\frac{bu_{max}}{c(1+2\gamma)}} = 50.0435\ m/s$$

The maximal acceleration $a_{acc,max}$ could be obtained by plotting the acceleration in the velocity domain $0, v_{max}]$ shown in Fig 5. By applying the function **max** in MATLAB, the maximal acceleration $a_{acc,max}$ could be obtained at $v = 0$ m/s with the value of 2.9388 m/s^2.

Similarly, when given the maximal braking input $u_{min} < 0$, the vehicle is braking with $a < 0$. The maximal deceleration $a_{dec,max}$ could be obtained by plotting the deceleration in the velocity domain $[0, v_{max}]$ shown in Fig 5. By applying the function **min** in MATLAB, the maximal deceleration $a_{dec,max}$ could be obtained at $v = v_{max}$ m/s with the value of -3.3509 m/s^2.

The MATLAB code is listed as followed:

```
% maximal speed Vmax
v_max=sqrt(b/c*umax/(1+2*gamma));

% Given minimal braking input umin, maximal acceleration and deceleration

u=umin;
v=0:0.1:v_max;
a=(b/m/(1+gamma)*u-c/m*v.^2).*(v<vg)+(b/m/(1+2*gamma)*u-c/m*v.^2).*(v>=vg);
figure();
```

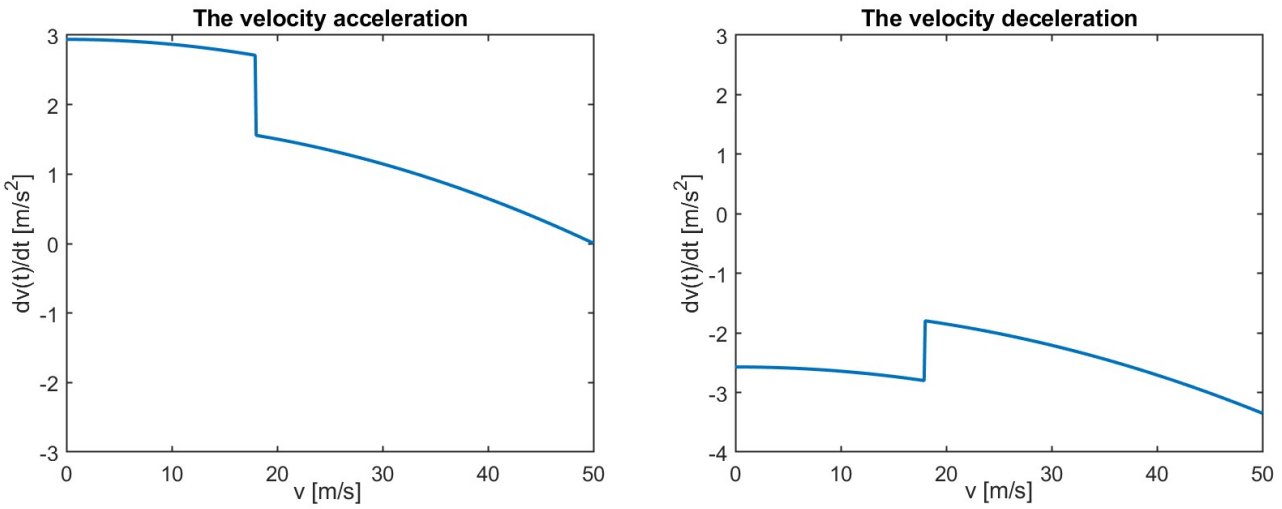

Fig. 5. Plots of acceleration given maximal throttle input u_{max} and deceleration given the maximal braking input u_{min} in the velocity domain $[0, v_{max}]$.

```
plot(v,a);
axis([0 v_max -4 3]);
title('The velocity deceleration')
xlabel('v [m/s]');
ylabel('dv(t)/dt [m/s^2]');
a_acc_min=min(a)

% Given maximal braking input umax, maximal acceleration and deceleration
u=umax;
v=0:0.1:v_max;
a=(b/m/(1+gamma)*u-c/m*v.^2).*(v<vg)+(b/m/(1+2*gamma)*u-c/m*v.^2).*(v>=vg);
figure();
plot(v,a);
axis([0 v_max -3 3]);
title('The velocity acceleration')
xlabel('v [m/s]');
ylabel('dv(t)/dt [m/s^2]');
a_acc_max=max(a)
```

B. Step 2.2

The piecewise-affine (PWA) approximation P with 2 regions of v of friction force curve

$$V : [0, v_{max}] \longrightarrow R : v \mapsto v^2$$

is applied based on perfectly matching two points $(0, 0)$ and (v_{max}, cv_{max}^2) on the friction force curve. The squared area S between the friction force curve V and piecewise-affine (PWA) approximation P is minimized:

$$S = \int_0^{v_{max}} (V(v) - P(v))^2 dv \tag{9}$$

By assuming the coordinates (α, β) is the middle edge point of the piecewise-affine (PWA) approximation P, the piecewise-affine (PWA) function could be written as

$$P(v) = \begin{cases} p_1 v = \frac{\beta}{\alpha} v & \text{if } v \leq \alpha \\ p_2 v + q_2 & \text{if } v > \alpha \end{cases}$$

where $p_2 = (cv_{max}^2 - \beta)/(v_{max} - \alpha)$ and $q_2 = \beta - p\alpha$. the squared area could be further written as two the sum of two sub squared area $S = S1 + S2$, where

$$S1 = \int_0^{\alpha} (V(v) - P(v))^2 dv = \int_0^{\alpha} (cv^2 - \frac{\beta}{\alpha} v)^2 dv$$
$$S2 = \int_{\alpha}^{v_{max}} (V(v) - P(v))^2 dv = \int_{\alpha}^{v_{max}} (cv^2 - p_2 v - q_2)^2 dv$$

Because p_2 and q_2 only depend on α and β and c, v_{max} are constants, the squared area $S = S1 + S2$ only depend on α and β as well. Therefore the squared area $S = f(\alpha, \beta)$ is a two-dimensional nonlinear function of α and β. In order to find the minimal point, the following steps are needed to be taken:

1) The partial derivatives of the function $S = f(\alpha, \beta)$ with respect to each variable. The Jacobian of $S = f(\alpha, \beta)$ could be written as

$$\left[\frac{\partial}{\partial \alpha} f(\alpha, \beta) \quad \frac{\partial}{\partial \beta} f(\alpha, \beta)\right]$$

Due to the highly non-linearity of $S = f(\alpha, \beta)$, symbolic function of S is built in MATLAB to assist in calculating the definite integrals with explicit math function. The variables $n1$, $n2$ and v represent the α, β and speed respectively. The function **int** is applied to calculate the definite integrals as $s1$ and $s2$ and **jacobian** is used to obtain the math functions of $\frac{\partial}{\partial \alpha} f(\alpha, \beta)$ and $\frac{\partial}{\partial \beta} f(\alpha, \beta)$.

```
syms n1 n2 v

S1=(c*v^2-n2/n1*v)^2;
S1=int(S1,[0 n1]);
p=(c*v_max.^2-n2)/(v_max-n1);
q=n2-p*n1;
S2=(c*v^2-p*v-q)^2;
S2=int(S2,[n1 v_max])
JAC=jacobian(S1+S2,[n1,n2]);
```

2) Set the partial derivatives equal to zero and solve for the variables α and β to find the critical points.

$$\begin{cases} \frac{\partial}{\partial \alpha} f(\alpha, \beta) = 0 \\ \frac{\partial}{\partial \beta} f(\alpha, \beta) = 0 \end{cases}$$

```
eqns = [JAC(1,1) == 0,JAC(1,2) == 0];
S = solve(eqns,[n1 n2]);
```

The obtained critical points are shown in Fig 6. According to definition of friction force curve, the domain of definition is $[0, v_{max}]$ and the middle point (α, β) of piecewise-affine (PWA) function should also need to be in the domain of $[0, v_{max}]$.
More specifically, the squared area S is constructed by dividing the domain $[0, v_{max}]$ into two smaller intervals $[0, \alpha]$ and $[\alpha, v_{max}]$ which compose the sub squared areas $S1$ and $S2$. Thus there is only critical point in the domain of $[0, v_{max}]$, that is, $\alpha = 25.0217$ and $\beta = 234.7826$. The friction force curve $V(v)$ and the approximation piecewise-affine (PWA) function are shown in Fig 6 from which it could be observed that piecewise-affine (PWA) function passes through two points $(0, 0)$ and (v_{max}, cv_{max}^2) on the friction force curve.

3) Use the second partial derivative test to determine whether each critical point corresponds to a minimum, maximum, or saddle point. The Hessian matrix is firstly obtained as:

$$H = \begin{bmatrix} \frac{\partial}{\partial \alpha^2} f(\alpha, \beta) & \frac{\partial}{\partial \alpha \partial \beta} f(\alpha, \beta) \\ \frac{\partial}{\partial \beta \partial \alpha} f(\alpha, \beta) & \frac{\partial}{\partial \beta^2} f(\alpha, \beta) \end{bmatrix}$$

In MATLAB, by using the function **hessian**, the Hessian matrix could be calculated as:

```
Hess(n1,n2)=hessian(S1+S2,[n1,n2])
H=Hess(S.n1(3),S.n2(3));
```

4) Evaluate the function at each critical point to find the minimum value. There are two conditions to be checked by using the second partial derivative test [2].
   a) Firstly, the quantity of $\frac{\partial}{\partial\alpha^2}f(\alpha,\beta)\frac{\partial}{\partial\beta^2}f(\alpha,\beta)-(\frac{\partial}{\partial\alpha\partial\beta}f(\alpha,\beta))^2$ is calculated as a positive value of 1.0888×10^5, then (α,β) is either a minimal point or a maximal point.
   b) The quantity of $\frac{\partial}{\partial\alpha^2}f(\alpha,\beta)$ is positive with the value of 2.4151×10^4, thus the (α,β) is a minimal point.

Overall, the optimal values of α and β are 25.0217 and 234.7826 respectively. The PWA function could be written as:

$$P(v)=\begin{cases}p_1v=9.3750v & \text{if } v\leq 25.0217\\ p_2v+q_2=40.6413v-781.6577 & \text{if } v>25.0217\end{cases} \tag{10}$$

where $p_1=9.3750$, $p_2=40.6413$ and $q_2=-781.6577$.

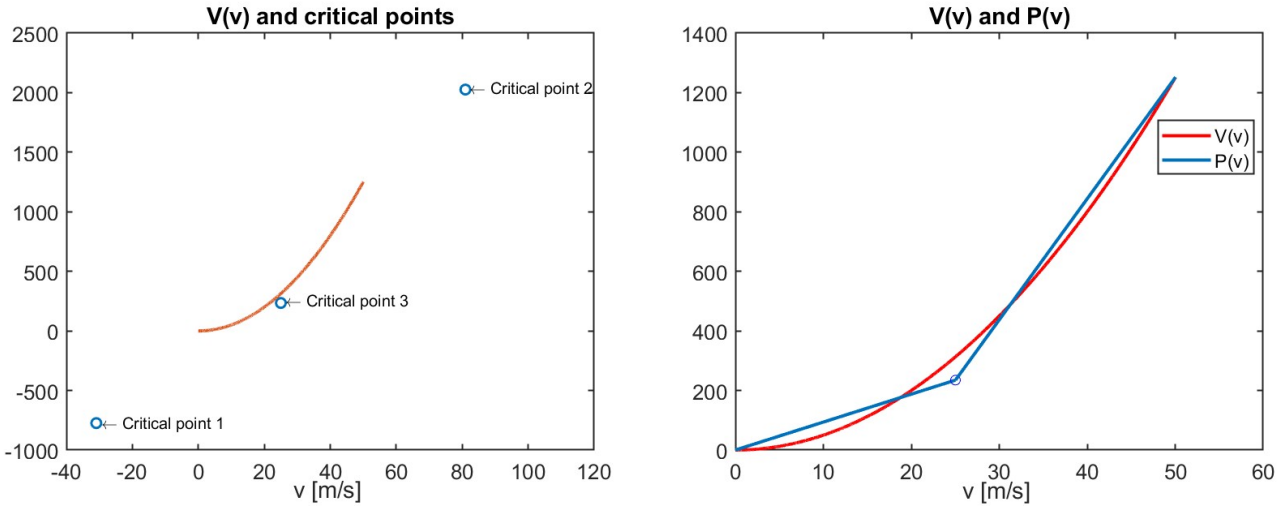

Fig. 6. Plots of acceleration given maximal throttle input u_{max} and deceleration given the maximal braking input u_{min} in the velocity domain $[0,v_{max}]$.

C. Step 2.3

In this part, the gear is assumed to be constant regardless of the speed of vehicle, $r(t)=1$ for all t. Based on the solution of Step 2.2, the friction force V is approximated by using the PWA function P. Then the continuous time state spade model is obtained as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}x(t)\\v(t)\end{bmatrix}=\begin{bmatrix}v(t)\\ \frac{F_{drive}-F_{gradient}-F_{friction}}{m}\end{bmatrix}=\begin{bmatrix}v(t)\\ \frac{b/m}{1+\gamma}u(t)-\frac{c}{m}v^2(t)\end{bmatrix} \tag{11}$$

Replacing $V(v)=cv^2(t)$ with $P(v)$ shown in equation (10), the state space model could be written in the form of Piecewise affine system (PWA),

$$\dot{z}=A_iz+B_iu+f_i,\ i\in\{1,2\}$$

for $z=\begin{bmatrix}x & v\end{bmatrix}^T\in\Omega_i$. For the two regions, Ω_1 and Ω_2, of Piecewise affine system (PWA), the sub dynamic systems of PWA are written as:

1) If $z \in \Omega_1 = \{z \in R^2 \mid 0 \leq v < \alpha\}$, $A_i = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{p_1}{m} \end{bmatrix}$, $B_i = \begin{bmatrix} 0 \\ \frac{b/m}{1+\gamma} \end{bmatrix}$, $f_i = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.

2) If $z \in \Omega_2 = \{z \in R^2 \mid \alpha \leq v \leq v_{max}\}$, $A_i = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{p_2}{m} \end{bmatrix}$, $B_i = \begin{bmatrix} 0 \\ \frac{b/m}{1+\gamma} \end{bmatrix}$, $f_i = \begin{bmatrix} 0 \\ -\frac{q_2}{m} \end{bmatrix}$.

Overall, the Piecewise affine system is written as:

$$\dot{z} = \begin{cases} \begin{bmatrix} 0 & 1 \\ 0 & -\frac{p_1}{m} \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{b/m}{1+\gamma} \end{bmatrix} \mathrm{u} & \text{if } z \in \Omega_1 \\ \begin{bmatrix} 0 & 1 \\ 0 & -\frac{p_2}{m} \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{b/m}{1+\gamma} \end{bmatrix} u + \begin{bmatrix} 0 \\ -\frac{q_2}{m} \end{bmatrix} & \text{if } z \in \Omega_2 \end{cases} \tag{12}$$

In order to analyse the different performance of continuous-time PWA model and original model for a sinusoidal throttle input, the two models are firtly built in MALTAB as:

1) Continuous-time PWA model: According to equation (12), the Continuous-time PWA model could be written in the form of differential equation as:

```
function [dydt,a] = vdp1(t,y,qq,alpha,beta,v_max,c,m)
   a1=beta/alpha;
   a2=(c*v_max^2-beta)/(v_max-alpha);
   b2=(a1-a2)*alpha;
   P=(a1*y(2)).*(y(2)<alpha)+(a2*y(2)+b2).*(y(2)>=alpha);
   u=sin(t);
   dydt = [y(2); qq*u-P/m];
end
```

2) Original model: According to equation (11), the original system could be in the form of differential equation as:

```
function [dydt,a] = vdp2(t,y,qq,m,c)
   u=sin(t);
   dydt = [y(2); qq*u-c/m*y(2).^2];
end
```

By applying a sinusoidal throttle input $u = sin(t)$ and initial state $[0, 40]$ and using the MATLAB solver **ode45**, the plots of states $x(t)$,$v(t)$ and $a(t)$ of Continuous-time PWA model and Original model are shown in Figs 7. It can be observed that the $x(t)$ of PWA system and that of original system is almost the same. However, there is dependency occurs to the plot of speed $v(t)$.

Before time $t = 18.727$ s the speed $v(t)$ of PWA system is slightly lower than that of original system. After $t = 18.727$ s, the speed of PWA system is slightly larger than that of original system. The reason for this case may be that before time $t = 18.727$, the speed is overall larger than $\alpha = 25.0217$, the approximation $P(v) = p_2 v + q_2$ is used to replace $V(v)$. In this case, the $P(v) > V(v)$ for the most simulated speed values, the acceleration a of the original continuous system is computed as $\frac{b/m}{1+\gamma}u(t)$ minus a larger value $\frac{P(v)}{m}$ in PWA model. This approach results in a higher deceleration during braking in the PWA model compared to the original system, causing the PWA model's speed to decrease more rapidly. Similarly, during acceleration, the PWA model exhibits a lower acceleration than the original system, leading to a slower increase in speed. Consequently, the speed (v) values of the PWA model are consistently lower than those of the original system.

When $t > 18.727$s, In this case, the $P(v) < V(v)$ for the most simulated speed values. This approach results in a lower deceleration during braking and higher acceleration in the speeding up process of

PWA model compared to the original system. Consequently, the speed (v) values of the PWA model are consistently higher than those of the original system. Also in plot of acceleration, the character of plot $v(t)$ could be verified with relatively lower acceleration before $t = 18.727$ s and relatively higher acceleration after $t = 18.727$ s.

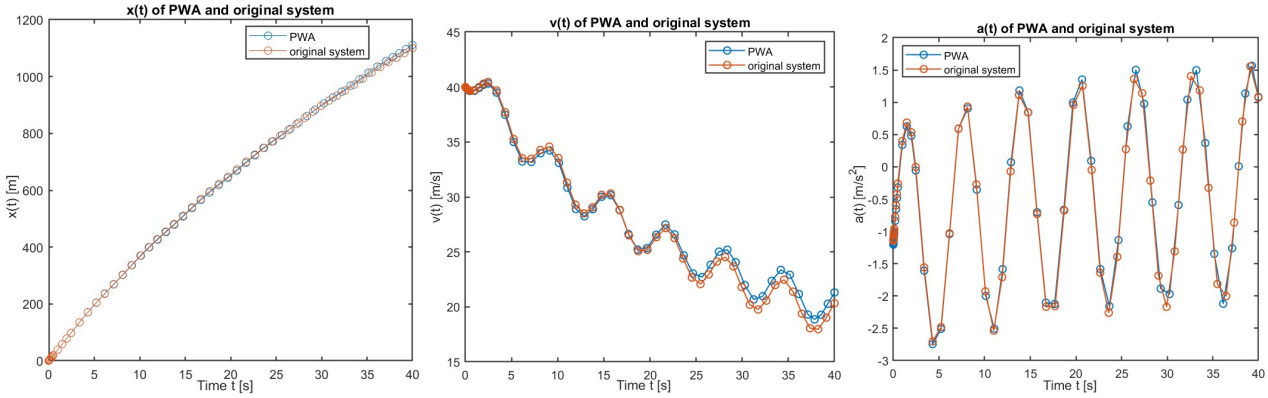

Fig. 7. Plots of states $x(t)$ and $v(t)$ and $a(t)$ of Continuous-time PWA model and Original model.

D. Step 2.4

In this step, the gear ratio $r(t)$ is changed based on the vehicle speed. Considering both the gear ratio and the approximation of friction force, the state space of model can be divided into three convex sets:

$$\Omega_1 = \left\{ z \in R^2 \mid 0 \leq z_2 < v_g \right\}$$
$$\Omega_2 = \left\{ z \in R^2 \mid v_g \leq z_2 < \alpha \right\}$$
$$\Omega_3 = \left\{ z \in R^2 \mid \alpha \leq z_2 \leq v_{max} \right\}$$

The continuous piecewise affine system can be written as:

$$\dot{z} = \begin{cases} \begin{bmatrix} 0 & 1 \\ 0 & -\frac{p_1}{m} \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{b/m}{1+\gamma} \end{bmatrix} \mathbf{u} & \text{if } z \in \Omega_1 \\ \begin{bmatrix} 0 & 1 \\ 0 & -\frac{p_1}{m} \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{b/m}{1+2\gamma} \end{bmatrix} \mathbf{u} & \text{if } z \in \Omega_2 \\ \begin{bmatrix} 0 & 1 \\ 0 & -\frac{p_2}{m} \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{b/m}{1+2\gamma} \end{bmatrix} u + \begin{bmatrix} 0 \\ -\frac{q_2}{m} \end{bmatrix} & \text{if } z \in \Omega_3 \end{cases} \tag{13}$$

Then we are going to consider the slope of the road. The height of the road can be described as a function of position x:

$$y = min \left\{ \frac{2h}{w}x, \frac{h}{w}x + h, 3h, \frac{3h}{2w}x + 9h \right\} \tag{14}$$

The position x is always non-negative. After comparing the value of four linear functions, we can simplify this function to the following form:

$$y = \begin{cases} \frac{2h}{w}x & \text{if } 0 \leq x < w \\ \frac{h}{w}x + h & \text{if } w \leq x < 2w \\ 3h & \text{if } x \geq 2w \end{cases} \tag{15}$$

The angles between these three linear functions and the horizontal line are 0.38, 0.197 and 0 respectively. Assume $sin\theta \approx \theta$, we can write the function of $sin\theta$ based on position x.

$$sin\theta = \begin{cases} \theta_1 = 0.38 & \text{if } 0 \leq x < w \\ \theta_2 = 0.197 & \text{if } w \leq x < 2w \\ 0 & \text{if } x \geq 2w \end{cases} \tag{16}$$

Considering the slope of road and the change of gear ratio and friction force based on speed, the state space of model can be divided into nine convex sets:

$$\Omega_1 = \left\{ z \in R^2 \mid 0 \leq z_1 < w, 0 \leq z_2 < v_g \right\}$$

$$\Omega_2 = \left\{ z \in R^2 \mid w \leq z_1 < 2w, 0 \leq z_2 < v_g \right\}$$

$$\Omega_3 = \left\{ z \in R^2 \mid z_1 \geq 2w, 0 \leq z_2 < v_g \right\}$$

$$\Omega_4 = \left\{ z \in R^2 \mid 0 \leq z_1 < w, v_g \leq z_2 < \alpha \right\}$$

$$\Omega_5 = \left\{ z \in R^2 \mid w \leq z_1 < 2w, v_g \leq z_2 < \alpha \right\}$$

$$\Omega_6 = \left\{ z \in R^2 \mid z_1 \geq 2w, v_g \leq z_2 < \alpha \right\}$$

$$\Omega_7 = \left\{ z \in R^2 \mid 0 \leq z_1 < w, \alpha \leq z_2 \leq v_{max} \right\}$$

$$\Omega_8 = \left\{ z \in R^2 \mid w \leq z_1 < 2w, \alpha \leq z_2 \leq v_{max} \right\}$$

$$\Omega_9 = \left\{ z \in R^2 \mid z_1 \geq 2w, \alpha \leq z_2 \leq v_{max} \right\}$$

The continuous piecewise affine system can be written as:

$$\dot{z} = \begin{cases} \begin{bmatrix} 0 & 1 \\ 0 & -\frac{p_1}{m} \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{b/m}{1+\gamma} \end{bmatrix} \mathrm{u} + \begin{bmatrix} 0 \\ -g\theta_1 \end{bmatrix} & \text{if } z \in \Omega_1 \\ \begin{bmatrix} 0 & 1 \\ 0 & -\frac{p_1}{m} \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{b/m}{1+\gamma} \end{bmatrix} \mathrm{u} + \begin{bmatrix} 0 \\ -g\theta_2 \end{bmatrix} & \text{if } z \in \Omega_2 \\ \begin{bmatrix} 0 & 1 \\ 0 & -\frac{p_1}{m} \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{b/m}{1+\gamma} \end{bmatrix} \mathrm{u} & \text{if } z \in \Omega_3 \\ \begin{bmatrix} 0 & 1 \\ 0 & -\frac{p_1}{m} \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{b/m}{1+2\gamma} \end{bmatrix} \mathrm{u} + \begin{bmatrix} 0 \\ -g\theta_1 \end{bmatrix} & \text{if } z \in \Omega_4 \\ \begin{bmatrix} 0 & 1 \\ 0 & -\frac{p_1}{m} \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{b/m}{1+2\gamma} \end{bmatrix} \mathrm{u} + \begin{bmatrix} 0 \\ -g\theta_2 \end{bmatrix} & \text{if } z \in \Omega_5 \\ \begin{bmatrix} 0 & 1 \\ 0 & -\frac{p_1}{m} \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{b/m}{1+2\gamma} \end{bmatrix} \mathrm{u} & \text{if } z \in \Omega_6 \\ \begin{bmatrix} 0 & 1 \\ 0 & -\frac{p_2}{m} \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{b/m}{1+2\gamma} \end{bmatrix} u + \begin{bmatrix} 0 \\ -\frac{q_2}{m} - g\theta_1 \end{bmatrix} & \text{if } z \in \Omega_7 \\ \begin{bmatrix} 0 & 1 \\ 0 & -\frac{p_2}{m} \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{b/m}{1+2\gamma} \end{bmatrix} u + \begin{bmatrix} 0 \\ -\frac{q_2}{m} - g\theta_2 \end{bmatrix} & \text{if } z \in \Omega_8 \\ \begin{bmatrix} 0 & 1 \\ 0 & -\frac{p_2}{m} \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{b/m}{1+2\gamma} \end{bmatrix} u + \begin{bmatrix} 0 \\ -\frac{q_2}{m} \end{bmatrix} & \text{if } z \in \Omega_9 \end{cases} \tag{17}$$

E. Step 2.5

Use forward Euler rule, we can discretize the PWA model in step 2.4. For the continuous model:

$$\dot{z} = A_i z + B_i u + f_i$$

The corresponding discrete model can be written as:

$$z_{k+1} = z_k + T(A_i z_k + B_i u_k + f_i)$$

where T is the sampling period.
We set $T = 0.1s$,then the discrete PWA model of vehicle is written as:

$$z_{k+1} = \begin{cases} \begin{bmatrix} 1 & T \\ 0 & -\frac{Tp_1}{m}+1 \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{Tb/m}{1+\gamma} \end{bmatrix} \mathrm{u} + \begin{bmatrix} 0 \\ -gT\theta_1 \end{bmatrix} & \text{if } z \in \Omega_1 \\ \begin{bmatrix} 1 & T \\ 0 & -\frac{Tp_1}{m}+1 \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{Tb/m}{1+\gamma} \end{bmatrix} \mathrm{u} + \begin{bmatrix} 0 \\ -gT\theta_2 \end{bmatrix} & \text{if } z \in \Omega_2 \\ \begin{bmatrix} 1 & T \\ 0 & -\frac{Tp_1}{m}+1 \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{Tb/m}{1+\gamma} \end{bmatrix} \mathrm{u} & \text{if } z \in \Omega_3 \\ \begin{bmatrix} 1 & T \\ 0 & -\frac{Tp_1}{m}+1 \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{Tb/m}{1+2\gamma} \end{bmatrix} \mathrm{u} + \begin{bmatrix} 0 \\ -gT\theta_1 \end{bmatrix} & \text{if } z \in \Omega_4 \\ \begin{bmatrix} 1 & T \\ 0 & -\frac{Tp_1}{m}+1 \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{Tb/m}{1+2\gamma} \end{bmatrix} \mathrm{u} + \begin{bmatrix} 0 \\ -gT\theta_2 \end{bmatrix} & \text{if } z \in \Omega_5 \\ \begin{bmatrix} 1 & T \\ 0 & -\frac{Tp_1}{m}+1 \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{Tb/m}{1+2\gamma} \end{bmatrix} \mathrm{u} & \text{if } z \in \Omega_6 \\ \begin{bmatrix} 1 & T \\ 0 & -\frac{Tp_2}{m}+1 \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{Tb/m}{1+2\gamma} \end{bmatrix} u + \begin{bmatrix} 0 \\ -\frac{Tq_2}{m} - gT\theta_1 \end{bmatrix} & \text{if } z \in \Omega_7 \\ \begin{bmatrix} 1 & T \\ 0 & -\frac{Tp_2}{m}+1 \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{Tb/m}{1+2\gamma} \end{bmatrix} u + \begin{bmatrix} 0 \\ -\frac{Tq_2}{m} - gT\theta_2 \end{bmatrix} & \text{if } z \in \Omega_8 \\ \begin{bmatrix} 1 & T \\ 0 & -\frac{Tp_2}{m}+1 \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{Tb/m}{1+2\gamma} \end{bmatrix} u + \begin{bmatrix} 0 \\ -\frac{Tq_2}{m} \end{bmatrix} & \text{if } z \in \Omega_9 \end{cases} \tag{18}$$

F. Step 2.6

According to the description in the Assignment file, we can determine the bound of states and input of the system. The first state, as well as the position of vehicle, does not have a maximum value. However, for computational convenience, we set its maximum value to a very large number. The bound of states and input is shown as below:

$$0 \leq z_1 \leq x_{max} = 10000 \tag{19}$$

$$0 \leq z_2 \leq v_{max} \tag{20}$$

$$u_{min} \leq u \leq u_{max} \tag{21}$$

By combining constraints (19) to (21), we can obtain the standard form of the constraint as follows:

$$\begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} z_1(k) \\ z_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 1 \end{bmatrix} u(k) + \mathbf{0}\delta(k) + \mathbf{0}n(k) \leq \begin{bmatrix} 0 \\ 10000 \\ 0 \\ v_{max} \\ -u_{min} \\ u_{max} \end{bmatrix} \tag{22}$$

$$E_{11}z(k) + E_{21}u(k) + E_{31}\delta(k) + E_{41}n(k) \leq g_1 \tag{23}$$

where $z(k)$ is the state of system, $u(k)$ is the input, $\delta(k)$ is the vector of boolean auxiliary variables in MLD system, $n(k)$ is the vector of real-valued auxiliary variables
Since the set of feasible states and inputs of PWA model are bounded, the discrete model in Step 2.5 can be rewritten as MLD system. There are nine state space with different system expression, so we need at least 4 binary variables to present 9 possible combinations.
We define these 4 binary variables as:

$$[\delta_1(k) = 1] \Leftrightarrow [z_1(k) \leq w] \tag{24}$$

$$[\delta_2(k) = 1] \Leftrightarrow [z_1(k) \geq 2w] \tag{25}$$

$$[\delta_3(k) = 1] \Leftrightarrow [z_2(k) \leq v_g] \tag{26}$$

$$[\delta_4(k) = 1] \Leftrightarrow [z_2(k) \geq \alpha] \tag{27}$$

It can also be presented as linear inequalities:

$$\begin{cases} (x_{max} - w)\delta_1(k) \leq x_{max} - z_1(k) \\ (w + \epsilon)\delta_1(k) \geq \epsilon + w - z_1(k) \end{cases} \tag{28}$$

$$\begin{cases} 2w\delta_2(k) \leq z_1(k) \\ (x_{max} - 2w + \epsilon)\delta_2(k) \geq z_1(k) - 2w + \epsilon \end{cases} \tag{29}$$

$$\begin{cases} (v_{max} - v_g)\delta_3(k) \leq v_{max} - z_2(k) \\ (v_g + \epsilon)\delta_3(k) \geq \epsilon + v_g - z_2(k) \end{cases} \tag{30}$$

$$\begin{cases} \alpha\delta_4(k) \leq z_2(k) \\ (v_{max} - \alpha + \epsilon)\delta_4(k) \geq z_2(k) - \alpha + \epsilon \end{cases} \tag{31}$$

where ϵ is machine precision.
By combining constraints (28) to (31), we can obtain the standard form of the constraint as follows:

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ -1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & -1 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} z_1(k) \\ z_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} x_{max} - w & 0 & 0 & 0 & 0 & 0 & 0 \\ -w - \epsilon & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2w & 0 & 0 & 0 & 0 & 0 \\ 0 & -x_{max} + 2w - \epsilon & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & v_{max} - v_g & 0 & 0 & 0 & 0 \\ 0 & 0 & -v_g - \epsilon & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & -v_{max} + \alpha - \epsilon & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_1(k) \\ \delta_2(k) \\ \delta_3(k) \\ \delta_4(k) \\ \delta_5(k) \\ \delta_6(k) \\ \delta_7(k) \end{bmatrix}$$

$$+\mathbf{0}n(k) \leq \begin{bmatrix} x_{max} \\ -\epsilon - w \\ 0 \\ 2w - \epsilon \\ v_{max} \\ -\epsilon - v_g \\ 0 \\ \alpha - \epsilon \end{bmatrix} \tag{32}$$

$$E_{12}z(k) + E_{22}u(k) + E_{32}\delta(k) + E_{42}n(k) \leq g_2 \tag{33}$$

Additionally, it is impossible for both δ_1 and δ_2 to be 1 simultaneously. The same applies to δ_3 and δ_4. Therefore, we set other two binary variables $\delta_5 = \delta_1\delta_2$ and $\delta_6 = \delta_3\delta_4$. The value of δ_5 and δ_6 is always be 0. It is equivalent to the following linear inequalities:

$$\begin{cases} \delta_5 = 0 \\ -\delta_1 + \delta_5 \leq 0 \\ -\delta_2 + \delta_5 \leq 0 \\ \delta_1 + \delta_2 - \delta_5 \leq 1 \end{cases} \tag{34}$$

$$\begin{cases} \delta_6 = 0 \\ -\delta_3 + \delta_6 \leq 0 \\ -\delta_4 + \delta_6 \leq 0 \\ \delta_3 + \delta_4 - \delta_6 \leq 1 \end{cases} \tag{35}$$

In the state function of the system, constant terms may arise. To simplify calculations, we introduce a extra binary variable δ_7 that multiplies with the constant term, and δ_7 is always equal to 1:

$$\delta_7 = 1 \tag{36}$$

By combining constraints from (34) to (36), we can obtain the standard form of the constraint as follows:

$$\mathbf{0}\begin{bmatrix} z_1(k) \\ z_2(k) \end{bmatrix} + \mathbf{0}u(k) + \begin{bmatrix} 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} \delta_1(k) \\ \delta_2(k) \\ \delta_3(k) \\ \delta_4(k) \\ \delta_5(k) \\ \delta_6(k) \\ \delta_7(k) \end{bmatrix} + \mathbf{0}n(k) \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ -1 \end{bmatrix} \tag{37}$$

$$E_{13}z(k) + E_{23}u(k) + E_{33}\delta(k) + E_{43}n(k) \leq g_3 \tag{38}$$

The system can be presented as:

$$\begin{aligned} z(k+1) = &\begin{bmatrix} 1 & T \\ 0 & 1-\frac{T}{m}[\delta_3 p_1 + \delta_4 p_2 + (1-\delta_3-\delta_4)p_1] \end{bmatrix} z(k) \\ &+ \begin{bmatrix} 0 \\ \frac{Tb}{m}[\delta_3 \frac{1}{1+\gamma} + \delta_4 \frac{1}{1+2\gamma} + (1-\delta_3-\delta_4)\frac{1}{1+2\gamma}] \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ \delta_4(-\frac{Tq_2}{m}) \end{bmatrix} \\ &+ \begin{bmatrix} 0 \\ -gT[\delta_1\theta_1 + (1-\delta_1-\delta_2)\theta_2] \end{bmatrix} \end{aligned}$$

After simplification, the expression of the system can be represented as follows:

$$\begin{aligned} z(k+1) = &\begin{bmatrix} 1 & T \\ 0 & 1-\frac{T}{m}p_1 \end{bmatrix} z(k) + \begin{bmatrix} 0 \\ \frac{Tb}{m(1+2\gamma)} \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ gT(\theta_2-\theta_1) \end{bmatrix} \delta_1(k) \\ &+ \begin{bmatrix} 0 \\ gT\theta_2 \end{bmatrix} \delta_2(k) + \begin{bmatrix} 0 \\ \frac{T}{m}(p_1-p_2) \end{bmatrix} \delta_4(k) z_2(k) + \begin{bmatrix} 0 \\ \frac{Tb}{m}(\frac{1}{1+\gamma} - \frac{1}{1+2\gamma}) \end{bmatrix} \delta_3(k) u(k) \\ &+ \begin{bmatrix} 0 \\ -\frac{Tq_2}{m} \end{bmatrix} \delta_4(k) + \begin{bmatrix} 0 \\ -gT\theta_2 \end{bmatrix} \end{aligned}$$

We replace nonlinear part in the system by several auxiliary real variables: $n_1(k) = \delta_4(k) z_2(k)$, $n_2(k) = \delta_3(k) u(k)$. It is equivalent to the following linear inequalities:

$$\begin{cases} n_1(k) \le v_{max}\delta_4(k) \\ n_1(k) \ge 0 \\ n_1(k) \le z_2(k) \\ n_1(k) \ge z_2(k) - v_{max}(1-\delta_4(k)) \end{cases} \tag{39}$$

$$\begin{cases} n_2(k) \le u_{max}\delta_3(k) \\ n_2(k) \ge u_{min}\delta_3(k) \\ n_2(k) \le u(k) - u_{min}(1-\delta_3(k)) \\ n_2(k) \ge u(k) - u_{max}(1-\delta_3(k)) \end{cases} \tag{40}$$

By combining constraints (39) to (40), we can obtain the standard form of the constraint as follows:

$$\begin{aligned} &\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & -1 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} z_1(k) \\ z_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 1 \end{bmatrix} u(k) + \begin{bmatrix} 0 & 0 & 0 & -v_{max} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & v_{max} & 0 & 0 & 0 \\ 0 & 0 & -u_{max} & 0 & 0 & 0 & 0 \\ 0 & 0 & u_{min} & 0 & 0 & 0 & 0 \\ 0 & 0 & -u_{min} & 0 & 0 & 0 & 0 \\ 0 & 0 & u_{max} & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_1(k) \\ \delta_2(k) \\ \delta_3(k) \\ \delta_4(k) \\ \delta_5(k) \\ \delta_6(k) \\ \delta_7(k) \end{bmatrix} \\ &+ \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} n_1(k) \\ n_2(k) \end{bmatrix} \le \begin{bmatrix} 0 \\ 0 \\ 0 \\ v_{max} \\ 0 \\ 0 \\ -u_{min} \\ u_{max} \end{bmatrix} \end{aligned} \tag{41}$$

$$E_{14}z(k) + E_{24}u(k) + E_{34}\delta(k) + E_{44}n(k) \leq g_4 \tag{42}$$

The standard form of the MLD (Mixed Integer Linear Programming) system can be represented as follows:

$$\begin{aligned}
z(k+1) &= \begin{bmatrix} 1 & T \\ 0 & 1-\frac{T}{m}p_1 \end{bmatrix} z(k) + \begin{bmatrix} 0 \\ \frac{Tb}{m(1+2\gamma)} \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ gT(\theta_2-\theta_1) \end{bmatrix} \delta_1(k) + \begin{bmatrix} 0 \\ gT\theta_2 \end{bmatrix} \delta_2(k) \\
&+ \begin{bmatrix} 0 \\ \frac{T}{m}(p_1-p_2) \end{bmatrix} n_1(k) + \begin{bmatrix} 0 \\ \frac{Tb}{m}(\frac{1}{1+\gamma} - \frac{1}{1+2\gamma}) \end{bmatrix} n_2(k) + \begin{bmatrix} 0 \\ -\frac{Tq_2}{m} \end{bmatrix} \delta_4(k) + \begin{bmatrix} 0 \\ -gT\theta_2 \end{bmatrix} \delta_7(k) \\
&= \begin{bmatrix} 1 & T \\ 0 & 1-\frac{T}{m}p_1 \end{bmatrix} z(k) + \begin{bmatrix} 0 \\ \frac{Tb}{m(1+2\gamma)} \end{bmatrix} u(k) + \begin{bmatrix} 0 & 0 \\ \frac{T}{m}(p_1-p_2) & \frac{Tb}{m}(\frac{1}{1+\gamma} - \frac{1}{1+2\gamma}) \end{bmatrix} \begin{bmatrix} n_1(k) \\ n_2(k) \end{bmatrix} \\
&+ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ gT(\theta_2-\theta_1) & gT\theta_2 & 0 & -\frac{Tq_2}{m} & 0 & 0 & -gT\theta_2 \end{bmatrix} \begin{bmatrix} \delta_1(k) \\ \delta_2(k) \\ \delta_3(k) \\ \delta_4(k) \\ \delta_5(k) \\ \delta_6(k) \\ \delta_7(k) \end{bmatrix} \\
&= Az(k) + B_1u(k) + B_2\delta(k) + B_3n(k)
\end{aligned} \tag{43}$$

$$\underbrace{\begin{bmatrix} E_{11} \\ E_{12} \\ E_{13} \\ E_{14} \end{bmatrix}}_{E_1} z(k) + \underbrace{\begin{bmatrix} E_{21} \\ E_{22} \\ E_{23} \\ E_{24} \end{bmatrix}}_{E_2} u(k) + \underbrace{\begin{bmatrix} E_{31} \\ E_{32} \\ E_{33} \\ E_{34} \end{bmatrix}}_{E_3} \delta(k) + \underbrace{\begin{bmatrix} E_{41} \\ E_{42} \\ E_{43} \\ E_{44} \end{bmatrix}}_{E_4} n(k) \leq \underbrace{\begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix}}_{g} \tag{44}$$

G. Step 2.7

1) MLD system to MILP problem: To determine the optimal Model predictive control (MPC) input sequence for a given sample step k, the MLD model needs to be firtly transformed into a mixed-integer linear programming problem (MILP). The first step is to collect the values of $\delta(k)$, $n(k)$ and $u(k)$ up to the time step $k + N_p - 1$, where N_p is prediction horizon which represents the length of time into the future for which the system's future behavior is predicted. These values could be written in the form of vector:

$$\hat{\delta}(k) = \begin{bmatrix} \hat{\delta}(k|k) \\ \hat{\delta}(k+1|k) \\ \vdots \\ \hat{\delta}(k+N_p-1|k) \end{bmatrix} \quad \hat{n}(k) = \begin{bmatrix} \hat{n}(k|k) \\ \hat{n}(k+1|k) \\ \vdots \\ \hat{n}(k+N_p-1|k) \end{bmatrix} \quad \hat{u}(k) = \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+N_p-1) \end{bmatrix}$$

However, by employing the control algorithm to steer a desired control sequence, we could only have the length of sequences of $\delta(k)$, $n(k)$ and $u(k)$ equal to the length of time into the future for which the control algorithm considers the effects of its actions. This length is taken as control horizon N_c. For the values from $k+N_c$ until $k+N_p-1$ will remain the same as the values at $k+N_c-1$ (usually $N_c \leq N_P$). These values with the length of N_c could be written in the form of vector as:

$$\hat{\delta}(k) = \begin{bmatrix} \tilde{\delta}(k|k) \\ \hat{\delta}(k+1|k) \\ \vdots \\ \hat{\delta}(k+N_c-1|k) \end{bmatrix} \quad \tilde{n}(k) = \begin{bmatrix} \hat{n}(k|k) \\ \hat{n}(k|k) \\ \vdots \\ \hat{n}(k+N_c-1|k) \end{bmatrix} \quad \tilde{u}(k) = \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+N_c-1) \end{bmatrix}$$

The transformation from $\hat{u}(k)$ to $\tilde{u}(k)$ could be further represented as:

$$\hat{u}(k) = \begin{bmatrix} u(k) \\ \vdots \\ u(k+N_c-1) \\ u(k+N_c) \\ \vdots \\ u(k+N_p-1) \end{bmatrix} = \begin{bmatrix} I_{N_u} & 0 & \dots & 0 \\ 0 & I_{N_u} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & I_{N_u} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & I_{N_u} \end{bmatrix} \begin{bmatrix} u(k) \\ \vdots \\ u(k+N_c-1) \end{bmatrix} = K_u \tilde{u}(k) \tag{45}$$

where I_{N_u} represents identity matrix with the length of $u(k)$. In our case $u(k)$ is one dimensional and the $I_{N_u} = 1$. Similarly, the transformation from $\hat{\delta}(k)$ to $\tilde{\delta}(k)$ and $\hat{n}(k)$ to $\tilde{n}(k)$ could be written in $\hat{\delta}(k) = K_\delta \tilde{\delta}(k)$ and $\hat{n}(k) = K_n \tilde{n}(k)$. For simplicity, the $\tilde{u}(k)$, $\tilde{\delta}(k)$ and $\tilde{n}(k)$ could be combined as

$$\tilde{V}(k) = \begin{bmatrix} \tilde{u}(k) \\ \tilde{\delta}(k) \\ \tilde{n}(k) \end{bmatrix}$$

In order to convert MLD into MILP problem, the MLD model could be further written in the successive substitution:

$$\begin{aligned}
&z(k+1) = Az(k) + B_1u(k) + B_2\delta(k) + B_3n(k) \\
&z(k+2) = A(Az(k) + B_1u(k) + B_2\delta(k) + B_3n(k)) + B_1u(k+1) + B_2\delta(k+1) + B_3n(k+1) \\
&= A^2z(k) + AB_1u(k) + AB_2\delta(k) + AB_3n(k) + B_1u(k+1) + B_2\delta(k+1) + B_3n(k+1) \\
&z(k+3) = Az(k+2) + B_1u(k+2) + B_2\delta(k+2) + B_3n(k+2) = A^3z(k) + A^2B_1u(k) + A^2B_2\delta(k) \\
&+ A^2B_3n(k) + AB_1u(k+1) + AB_2\delta(k+1) + AB_3n(k+1) + B_1u(k+2) + B_2\delta(k+2) + B_3n(k+2) \\
&\vdots \\
&z(k+j) = A^jx(k) + \sum_{i=0}^{j-1} A^{j-i-1}(B_1u(k+i) + B_2\delta(k+i) + B_3n(k+i))
\end{aligned} \tag{46}$$

The equations (46) could be written in the form of matrix representation:

$$\begin{bmatrix} z(k+1) \\ z(k+2) \\ \vdots \\ z(k+N_p) \end{bmatrix} = \underbrace{\begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{N_p} \end{bmatrix}}_{M_2} z(k) + \underbrace{\begin{bmatrix} B_1 & 0 & \dots & 0 \\ AB_1 & B_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N_p-1}B_1 & A^{N_p-2}B_1 & \dots & B_1 \end{bmatrix}}_{T_1} \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+N_p-1) \end{bmatrix}$$

$$+ \underbrace{\begin{bmatrix} B_2 & 0 & \dots & 0 \\ AB_2 & B_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N_p-1}B_2 & A^{N_p-2}B_2 & \dots & B_2 \end{bmatrix}}_{T_2} \begin{bmatrix} \delta(k) \\ \delta(k+1) \\ \vdots \\ \delta(k+N_p-1) \end{bmatrix} + \underbrace{\begin{bmatrix} B_3 & 0 & \dots & 0 \\ AB_3 & B_3 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N_p-1}B_3 & A^{N_p-2}B_3 & \dots & B_3 \end{bmatrix}}_{T_3} \begin{bmatrix} n(k) \\ n(k+1) \\ \vdots \\ n(k+N_p-1) \end{bmatrix}$$

In the prediction horizon N_p, the values of $\hat{z}(k+1|k)$ from $k+1$ until $k+N_p$ could be compacted and transformed into data vector $\hat{z}(k)$ as $\hat{z}(k) = \begin{bmatrix}\hat{z}(k+1|k) & \hat{z}(k+2|k) & \dots & \hat{z}(k+N_p|k)\end{bmatrix}^T$. By using function $z(k)$ and $\hat{u}(k) = [u(k), u(k+1), \dots, u(k+N_p-1)]$, the $\tilde{z}(k)$, $\tilde{\delta}(k)$ and $\tilde{n}(k)$ could be represented in a more compactly form as:

$$\begin{aligned}\hat{z}(k) &= M_2 z(k) + T_1\hat{u}(k) + T_2\hat{\delta}(k) + T_3\hat{n}(k)\\ &= M_2 z(k) + T_1 K_u\tilde{u}(k) + T_2 K_\delta\tilde{\delta}(k) + T_3 K_n\tilde{n}(k)\\ &= M_1\tilde{V}(k) + M_2 z(k)\end{aligned} \tag{47}$$

where $M_1 = \begin{bmatrix}T_1K_u & T_2K_\delta & T_3K_n\end{bmatrix}$.

The next step is to rewrite the constraint (44) into the inequality by using $x(k)$ and $\tilde{V}(k)$. In the prediction horizon, we have N_p constrains in total:

$$E_1 z(k) + E_2 u(k) + E_3\hat{\delta}(k|k) + E_4\hat{n}(k|k) \le g$$
$$E_1\hat{z}(k+1|k) + E_2 u(k+1) + E_3\hat{\delta}(k+1|k) + E_4\hat{n}(k+1|k) \le g$$
$$\vdots$$
$$E_1\hat{z}(k+N_p-1|k) + E_2 u(k+N_p-1) + E_3\hat{\delta}(k+N_p-1|k) + E_4\hat{n}(k+N_p-1|k) \le g$$

$\hat{z}(k+1|k)$,...,$\hat{z}(k+N_p-1|k)$ in these constraints can be represented by $z(k)$ and $\tilde{V}(k)$ according to (46). Then the constraints can be written as:
for $l = 0, 1, ..., N_p - 1$:

$$\begin{aligned}&E_1A^l x(k) + E_1A^{l-1}B_1u(k) + E_1A^{l-2}B_1u(k+1) + ... + E_1B_1u(k+l-1) + E_2u(k+l)\\ &+ E_1A^{l-1}B_2\hat{\delta}(k|k) + E_1A^{l-2}B_2\hat{\delta}(k+1|k) + ... + E_1B_2\hat{\delta}(k+l-1|k) + E_3\hat{\delta}(k+l|k)\\ &+ E_1A^{l-1}B_3\hat{n}(k|k) + E_1A^{l-2}B_3\hat{n}(k+1|k) + ... + E_1B_3\hat{n}(k+l-1|k) + E_4\hat{n}(k+l|k) \le g\end{aligned} \tag{48}$$

All constraints can be written in the form of matrix representation:

$$\underbrace{\begin{bmatrix}E_1\\E_1A\\E_1A^2\\\vdots\\E_1A^{N_p-1}\end{bmatrix}}_{F_{31}} z(k) + \underbrace{\begin{bmatrix}E_2 & 0 & 0 & \dots & 0\\E_1B_1 & E_2 & 0 & \dots & 0\\E_1AB_1 & E_1B_1 & E_2 & \dots & 0\\\vdots & \vdots & \vdots & \ddots & \vdots\\E_1A^{N_p-2}B_1 & E_1A^{N_p-3}B_1 & E_1A^{N_p-4}B_1 & \dots & E_2\end{bmatrix}}_{Q_1}\begin{bmatrix}u(k)\\u(k+1)\\u(k+2)\\\vdots\\u(k+N_p-1)\end{bmatrix}$$
$$+\underbrace{\begin{bmatrix}E_3 & 0 & 0 & \dots & 0\\E_1B_2 & E_3 & 0 & \dots & 0\\E_1AB_2 & E_1B_2 & E_3 & \dots & 0\\\vdots & \vdots & \vdots & \ddots & \vdots\\E_1A^{N_p-2}B_2 & E_1A^{N_p-3}B_2 & E_1A^{N_p-4}B_2 & \dots & E_3\end{bmatrix}}_{Q_2}\begin{bmatrix}\hat{\delta}(k|k)\\\hat{\delta}(k+1|k)\\\hat{\delta}(k+2|k)\\\vdots\\\hat{\delta}(k+N_p-1|k)\end{bmatrix}$$
$$+\underbrace{\begin{bmatrix}E_4 & 0 & 0 & \dots & 0\\E_1B_3 & E_4 & 0 & \dots & 0\\E_1AB_3 & E_1B_3 & E_4 & \dots & 0\\\vdots & \vdots & \vdots & \ddots & \vdots\\E_1A^{N_p-2}B_3 & E_1A^{N_p-3}B_3 & E_1A^{N_p-4}B_3 & \dots & E_4\end{bmatrix}}_{Q_3}\begin{bmatrix}\hat{n}(k|k)\\\hat{n}(k+1|k)\\\hat{n}(k+2|k)\\\vdots\\\hat{n}(k+N_p-1|k)\end{bmatrix} \le \underbrace{\begin{bmatrix}g\\g\\g\\\vdots\\g\end{bmatrix}}_{F_{21}}$$

$$
\begin{aligned}
&F_{31}z(k) + Q_1\hat{u}(k) + Q_2\hat{\delta}(k) + Q_3\hat{n}(k) \leq F_{21} \\
&F_{31}z(k) + Q_1K_u\tilde{u}(k) + Q_2K_\delta\tilde{\delta}(k) + Q_3K_n\tilde{n}(k) \leq F_{21} \\
&F_{11}\tilde{V}(k) + F_{31}z(k) \leq F_{21}
\end{aligned} \tag{49}
$$

where $F_{11} = \begin{bmatrix} Q_1K_u & Q_2K_\delta & Q_3K_n \end{bmatrix}$.
Additionally, we have to consider the comfort constraint:$-a_{comf,max} \leq a(k) \leq a_{comf,max}$ and speed ($z_2(k)$) of system should always be positive. In constraint (20), we have already set the constraint of speed. Therefore, we only need to add constraints regarding acceleration.
In the discrete-time system, the comfort constraint can be represented as:

$$
-a_{comf,max}T \leq z_2(k+1) - z_2(k) = \begin{bmatrix} 0 & 1 \end{bmatrix} (z(k+1) - z(k)) \leq a_{comf,max}T \tag{50}
$$

where T is the sampling time.
In the prediction horizon, the acceleration of all k should satisfy the comfort constraint:

$$
\begin{gathered}
-a_{comf,max}T \leq \begin{bmatrix} 0 & 1 \end{bmatrix} (\hat{z}(k+1|k) - z(k)) \leq a_{comf,max}T \\
-a_{comf,max}T \leq \begin{bmatrix} 0 & 1 \end{bmatrix} (\hat{z}(k+2|k) - \hat{z}(k+1|k)) \leq a_{comf,max}T \\
\vdots \\
-a_{comf,max}T \leq \begin{bmatrix} 0 & 1 \end{bmatrix} (\hat{z}(k+N_p|k) - \hat{z}(k+N_p-1|k)) \leq a_{comf,max}T
\end{gathered}
$$

$\hat{z}(k+1|k) - z(k)$,...,$\hat{z}(k+N_p|k) - \hat{z}(k+N_p-1|k)$ in these constraints can be represented by $z(k)$ and $\tilde{V}(k)$ according to (46):
for $l = 0, 1, ..., N_p - 1$:

$$
\begin{aligned}
&\hat{z}(k+l+1|k) - \hat{z}(k+l|k) = (A^{l+1} - A^l)z(k) \\
&+ (A^lB_1 - A^{l-1}B_1)u(k) + ... + (AB_1 - B_1)u(k+l-1) + B_1u(k+l) \\
&+ (A^lB_2 - A^{l-1}B_2)\hat{\delta}(k|k) + ... + (AB_2 - B_2)\hat{\delta}(k+l-1|k) + B_2\hat{\delta}(k+l|k) \\
&+ (A^lB_3 - A^{l-1}B_3)\hat{n}(k|k) + ... + (AB_3 - B_3)\hat{n}(k+l-1|k) + B_3\hat{n}(k+l|k)
\end{aligned} \tag{51}
$$

All comfort constraints can be written in the form of matrix representation:

$$
R_1 = \begin{bmatrix}
0 & 1 & 0 & 0 & \dots & 0 & 0 \\
0 & 0 & 0 & 1 & \dots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & 0 & \dots & 0 & 1
\end{bmatrix}_{(2N_p)*(4N_p)} \tag{52}
$$

$$R_1 \underbrace{\begin{bmatrix} A-I \\ A^2-A \\ \vdots \\ A^{N_p}-A^{N_p-1} \\ I-A \\ A-A^2 \\ \vdots \\ A^{N_p-1}-A^{N_p} \end{bmatrix}}_{P_1} z(k) + R_1 \underbrace{\begin{bmatrix} B_1 & 0 & \dots & 0 \\ AB_1-B_1 & B_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N_p-1}B_1-A^{N_p-2}B_1 & A^{N_p-2}B_1-A^{N_p-3}B_1 & \dots & B_1 \\ -B_1 & 0 & \dots & 0 \\ B_1-AB_1 & -B_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N_p-2}B_1-A^{N_p-1}B_1 & A^{N_p-3}B_1-A^{N_p-2}B_1 & \dots & -B_1 \end{bmatrix}}_{P_2} \begin{bmatrix} u(k) \\ u(k+1) \\ u(k+2) \\ \vdots \\ u(k+N_p-1) \end{bmatrix}$$

$$+ R_1 \underbrace{\begin{bmatrix} B_2 & 0 & \dots & 0 \\ AB_2-B_2 & B_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N_p-1}B_2-A^{N_p-2}B_2 & A^{N_p-2}B_2-A^{N_p-3}B_2 & \dots & B_2 \\ -B_2 & 0 & \dots & 0 \\ B_2-AB_2 & -B_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N_p-2}B_2-A^{N_p-1}B_2 & A^{N_p-3}B_2-A^{N_p-2}B_2 & \dots & -B_2 \end{bmatrix}}_{P_3} \begin{bmatrix} \hat{\delta}(k|k) \\ \hat{\delta}(k+1|k) \\ \hat{\delta}(k+2|k) \\ \vdots \\ \hat{\delta}(k+N_p-1|k) \end{bmatrix}$$

$$+ R_1 \underbrace{\begin{bmatrix} B_3 & 0 & \dots & 0 \\ AB_3-B_3 & B_3 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N_p-1}B_3-A^{N_p-2}B_3 & A^{N_p-2}B_3-A^{N_p-3}B_3 & \dots & B_3 \\ -B_3 & 0 & \dots & 0 \\ B_3-AB_3 & -B_3 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N_p-2}B_3-A^{N_p-1}B_3 & A^{N_p-3}B_3-A^{N_p-2}B_3 & \dots & -B_3 \end{bmatrix}}_{P_4} \begin{bmatrix} \hat{n}(k|k) \\ \hat{n}(k+1|k) \\ \hat{n}(k+2|k) \\ \vdots \\ \hat{n}(k+N_p-1|k) \end{bmatrix} \leq \underbrace{\begin{bmatrix} a_{comf,max}T \\ a_{comf,max}T \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ a_{comf,max}T \end{bmatrix}}_{F_{22}}$$

$$\begin{aligned} &R_1P_1z(k) + R_1P_2\hat{u}(k) + R_1P_3\hat{\delta}(k) + R_1P_4\hat{n}(k) \leq F_{22} \\ &R_1P_1z(k) + R_1P_2K_u\tilde{u}(k) + R_1P_3K_\delta\tilde{\delta}(k) + R_1P_4K_n\tilde{n}(k) \leq F_{22} \\ &F_{12}\tilde{V}(k) + F_{32}z(k) \leq F_{22} \end{aligned} \tag{53}$$

where $F_{12} = \begin{bmatrix} R_1P_2K_u & R_1P_3K_\delta & R_1P_4K_n \end{bmatrix}$, $F_{32} = R_1P_1$.
Combine (49) and (53), we can get the standard form of constraints in MILP system:

$$\underbrace{\begin{bmatrix} F_{11} \\ F_{12} \end{bmatrix}}_{F_1} \tilde{V}(k) \leq \underbrace{\begin{bmatrix} F_{21} \\ F_{22} \end{bmatrix}}_{F_2} + \underbrace{\begin{bmatrix} -F_{31} \\ -F_{32} \end{bmatrix}}_{F_3} z(k) \tag{54}$$

2) transform objective function to MILP problem: We are in group 18. According to the Table 2 in assignment file, the performance J of MPC problem is:

$$\begin{aligned} J(k) &= J_{track}(k) + \lambda J_{input}(k) \\ &= \|\hat{v}(k) - \hat{v}_{ref}(k)\|_1 + \lambda\|\Delta^2\hat{u}(k)\|_\infty \end{aligned} \tag{55}$$

where

$$\hat{v}(k) = [v(k+1)...v(k+N_p)]^T = [z_2(k+1)...z_2(k+N_p)]^T$$
$$\hat{v}_{ref}(k) = [v_{ref}(k+1)...v_{ref}(k+N_p)]^T$$
$$\hat{u}(k) = [u(k)...u(k+N_p-1)]^T$$
$$\Delta^2\hat{u}(k) = \Delta\hat{u}(k) - \Delta\hat{u}(k-1) = \hat{u}(k) - 2\hat{u}(k-1) + \hat{u}(k-2)$$

To transform the 1-Norm and ∞-Norm in the objective function into a linear programming problem, we introduce (N_p+1) dummy variables to the optimization problem: $\rho_1, \rho_2, ...\rho_{N_p}, \tau$.
Where $\rho_1, \rho_2, ...\rho_{N_p}$ can represent the absolute difference between the velocity and the velocity reference within the prediction horizon. The constraints for these N_p dummy variables are:

$$-\rho_i \leq \hat{z}_2(k+i) - \hat{v}_{ref}(k+i) \leq \rho_i, \ \ for \ i = 1, 2..., N_p \tag{56}$$

And τ represents the maximum value of all $|\Delta^2\hat{u}(k)|$ in the prediction horizon. The constraint for τ is:

$$-\tau \leq \hat{u}(l) - 2\hat{u}(l-1) + \hat{u}(l-2) \leq \tau, \ \ for \ l = k, ..., k+N_p-1 \tag{57}$$

To get a well-defined objective function, we assume that $\hat{u}(k-1) = \hat{u}(k-2) = 0$.
After introducing the dummy variables, the objective function and constraints of MPC problem can be written as:

$$min \ (\rho_1 + \rho_2 + ... + \rho_{N_p} + \lambda\tau) \tag{58}$$

$$\begin{aligned} s.t. \quad & -\rho_i \leq \hat{z}_2(k+i) - \hat{v}_{ref}(k+i) \leq \rho_i, \ \ for \ i = 1, 2..., N_p \\ & -\tau \leq \hat{u}(l) - 2\hat{u}(l-1) + \hat{u}(l-2) \leq \tau, \ \ for \ l = k, ..., k+N_p-1 \\ & F_1\tilde{V}(k) \leq F_2 + F_3 z(k) \end{aligned} \tag{59}$$

To get the standard form of MILP problem, we extend the variable vector $\tilde{V}(k)$ to $\tilde{V}_e(k)$:

$$\tilde{V}_e(k) = \begin{bmatrix} \tilde{u}^T(k) & \tilde{\delta}^T(k) & \tilde{n}^T(k) & \rho_1 & \dots & \rho_{N_p} & \tau \end{bmatrix}^T \tag{60}$$

Thus, the objective function to be minimized can be formulated as:

$$\begin{aligned} & \rho_1 + \rho_2 + ... + \rho_{N_p} + \lambda\tau \\ & = [\underbrace{0 \ \dots \ 0}_{N_c(N_u+N_\delta+N_n)} \ \ \underbrace{1 \ \dots \ 1}_{N_p} \ \ \lambda]\tilde{V}_e(k) \\ & = S\tilde{V}_e(k) \end{aligned} \tag{61}$$

where N_u, N_δ, N_n are the length of sequence $u(k), \delta(k)$ and $n(k)$ respectively.
The next step is to transform the constraints (59) into the standard form of MILP problem.

For the first constraint in (59), we can perform the following transformation:
Assume $\rho = \begin{bmatrix} \rho_1 & \dots & \rho_{N_p} \end{bmatrix}^T$, then the constraint can be represented as:

$$-\rho \leq \hat{v}(k) - \hat{v}_{ref}(k) \leq \rho$$

$$-\rho + \hat{v}_{ref}(k) \leq \hat{v}(k) \leq \rho + \hat{v}_{ref}(k)$$

Assume

$$R_2 = \begin{bmatrix} 0 & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix}_{(N_p)*(2N_p)}$$

$$\hat{v}(k) = [z_2(k+1)...z_2(k+N_p)]^T = R_2\hat{z}(k)$$

Combine (47), the constraint can be written as:

$$-\rho + \hat{v}_{ref}(k) \leq R_2(M_1\tilde{V}(k) + M_2 z(k)) \leq \rho + \hat{v}_{ref}(k)$$

For the right-hand side of the inequality, we can simplify it as follows:

$$R_2M_1\tilde{V}(k) + R_2M_2z(k) \leq \rho + \hat{v}_{ref}(k)$$

$$R_2M_1\tilde{V}(k) - \rho \leq -R_2M_2z(k) + \hat{v}_{ref}(k)$$

$$\begin{bmatrix} R_2M_1 & -I_{N_p} & 0 \end{bmatrix} \tilde{V}_e(k) \leq \hat{v}_{ref}(k) - R_2M_2z(k) \tag{62}$$

For the left-hand side of the inequality, we can simplify it as follows:

$$-R_2M_1\tilde{V}(k) - R_2M_2z(k) \leq \rho - \hat{v}_{ref}(k)$$

$$-R_2M_1\tilde{V}(k) - \rho \leq R_2M_2z(k) - \hat{v}_{ref}(k)$$

$$\begin{bmatrix} -R_2M_1 & -I_{N_p} & 0 \end{bmatrix} \tilde{V}_e(k) \leq -\hat{v}_{ref}(k) + R_2M_2z(k) \tag{63}$$

Combine (62) and (63), we can determine the standard form of the first constraint:

$$\underbrace{\begin{bmatrix} R_2M_1 & -I_{N_p} & 0 \\ -R_2M_1 & -I_{N_p} & 0 \end{bmatrix}}_{F_{e11}} \tilde{V}_e(k) \leq \underbrace{\begin{bmatrix} \hat{v}_{ref}(k) \\ -\hat{v}_{ref}(k) \end{bmatrix}}_{F_{e21}} + \underbrace{\begin{bmatrix} -R_2M_2 \\ R_2M_2 \end{bmatrix}}_{F_{e31}} z(k) \tag{64}$$

For the second constraint in (59), we can perform the following transformation:

$$\begin{bmatrix} -\tau \\ -\tau \\ -\tau \\ \vdots \\ -\tau \end{bmatrix} \leq \begin{bmatrix} u(k) \\ u(k+1) - 2u(k) \\ u(k+2) - 2u(k+1) + u(k) \\ \vdots \\ u(k+N_p-1) - 2u(k+N_p-2) + u(k+N_p-3) \end{bmatrix} \leq \begin{bmatrix} \tau \\ \tau \\ \tau \\ \vdots \\ \tau \end{bmatrix}$$

Assume

$$H_1 = \begin{bmatrix} 1 & & & & & & \\ -2 & 1 & & & & & \\ 1 & -2 & 1 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ 0 & 0 & 0 & \dots & 1 & -2 & 1 \end{bmatrix}_{N_p*N_p}$$

Then

$$\begin{bmatrix} u(k) \\ u(k+1)-2u(k) \\ u(k+2)-2u(k+1)+u(k) \\ \vdots \\ u(k+N_p-1)-2u(k+N_p-2)+u(k+N_p-3) \end{bmatrix} = H_1\hat{u}(k) = H_1K_u\tilde{u}(k)$$

For the right-hand side of the inequality, we can simplify it as follows:

$$H_1K_u\tilde{u}(k) \leq \begin{bmatrix} \tau \\ \tau \\ \vdots \\ \tau \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & \dots & 0 & 1 \\ 0 & \dots & 0 & 1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & 1 \end{bmatrix}}_{H_2} \tilde{V}_e(k)$$

where the size of H_2 is $N_p * (N_c(N_u + N_\delta + N_n) + N_p + 1)$.

$$\underbrace{\begin{bmatrix} H_1K_u & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{H_3} \begin{bmatrix} \tilde{u}(k) \\ \tilde{\delta}(k) \\ \tilde{n}(k) \\ \rho \\ \tau \end{bmatrix} \leq H_2\tilde{V}_e(k)$$

$$(H_3 - H_2)\tilde{V}_e(k) \leq \mathbf{0} \tag{65}$$

For the left-hand side of the inequality, we can simplify it as follows:

$$-H_1K_u\tilde{u}(k) = -H_3\tilde{V}_e(k) \leq \begin{bmatrix} \tau \\ \tau \\ \vdots \\ \tau \end{bmatrix} = H_2\tilde{V}_e(k)$$

$$(-H_3 - H_2)\tilde{V}_e(k) \leq \mathbf{0} \tag{66}$$

Combine (65) and (66), we can get the standard form of the second constraint:

$$\underbrace{\begin{bmatrix} H_3 - H_2 \\ -H_3 - H_2 \end{bmatrix}}_{F_{e12}} \tilde{V}_e(k) \leq \mathbf{0} \tag{67}$$

For the third constraint in (59), we can perform the following transformation:

$$\begin{aligned} F_1\tilde{V}(k) &\leq F_2 + F_3z(k) \\ \underbrace{\begin{bmatrix} F_1 & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix}}_{F_{e13}} \tilde{V}_e(k) &\leq F_2 + F_3z(k) \end{aligned} \tag{68}$$

Take all constraints (64), (67) and (68) into account, we can determine the standard form of (59):

$$\underbrace{\begin{bmatrix} F_{e11} \\ F_{e12} \\ F_{e13} \end{bmatrix}}_{F_{e1}} \tilde{V}_e(k) \leq \underbrace{\begin{bmatrix} F_{e21} \\ \mathbf{0} \\ F_2 \end{bmatrix}}_{F_{e2}} + \underbrace{\begin{bmatrix} F_{e31} \\ \mathbf{0} \\ F_3 \end{bmatrix}}_{F_{e3}} z(k) \tag{69}$$

Combine (61) and (69), we can finally convert the MPC problem to a standard MILP problem:

$$\begin{aligned} &\min_{\tilde{V}_e(k)} \ S\tilde{V}_e(k) \\ &s.t. \ \ F_{e1}\tilde{V}_e(k) \leq F_{e2} + F_{e3}z(k) \end{aligned} \tag{70}$$

This has to be translated and optimised on MATLAB. The glpk function of the MPT toolbox version 3 is used for this purpose. The code is attached in Appendix A of this report. In this case, the sampling horizon is 250 which means the number of simulation time steps is 250 (according the receding horizon principle the MILP problem is solved 250 times), and the prediction horizon and control horizon are set to be $N_p = 5$ and $N_c = 4$ respectively. Inside a loop that runs from the start to the end of the simulation time steps, the MILP model is computed at each time step with its corresponding constraints. Then the optimisation matrices c, A and b at time k are defined as required by the glpk function as

$$c = S, \quad A = F_{e1}, \qquad b = F_{e2} + F_{e3}z(k)$$

and the optimal matrix $\tilde{V}_e(k)$ is computed for this time step. Also the $u(k)$ is computed as the first optimal MPC control input coming from $\tilde{V}_e(k)$ and the $z(k+1)$ is updated by MILP system dynamic $z(k+1) = Az(k) + B_1u(k) + B_2\delta(k) + B_3n(k)$ (43), where $u(k)$, $\delta(k)$ and $n(k)$ are also from the obtained optimal matrix $\tilde{V}_e(k)$. The initial state $z(0)$ is set to be $[0, 10]^T$. What is more, the reference of speed v is set to be $v_{ref}(k) = 20$ m/s.

H. Step 2.8

In this part, the closed-loop simulation is made and applied to the original continuous-time ACC model as simulation model (7). In the previous section, the MILP problem with objective function and constraints is built based at time step k (only at one simulation time step). Here, the loop is closed, which means in each simulation time step the receding horizon principle is used and only the first optimal MPC control input is recomputed and applied to the system.

For comparison, the differential equations of continuous-time model of ACC are used to update the states of system over time. As The state variables of a system represent the dynamic quantities that change over time, the numerical integration methods are used to update states that describes the behavior of the system. In MATLAB, the ode45 solver is a powerful to integrate the system of differential equations within the time span with initial conditions of states.

In Figs (8), the plots of position x and speed v of MLD model and original continuous model respectively are shown and compared. It is observed that the plots of two states position x and speed v of MID model is slightly below those of original continuous model, which means the MLD model underestimate the true values of real states of original system.

This discrepancy can be attributed to the approximation of $V(v)$ as $P(v)$ during the construction of the MLD model. In equation (10) and Figs (6), it is evident that for $v < 20$, the value of $V(v)$ is smaller than that of $P(v)$. Additionally, the simulated speed values mostly fall within the range $v \in (0, 20]$. Analyzing equation (11), which describes the acceleration a of the original continuous system, it is clear that the MLD model computes the acceleration as $\frac{b/m}{1+\gamma}u(t)$ minus a larger value $\frac{P(v)}{m}$. This approach results in a higher deceleration during braking in the MLD model compared to the original system, causing the MLD model's speed to decrease more rapidly. Similarly, during acceleration, the MLD model exhibits a lower acceleration than the original system, leading to a slower increase in speed. Consequently, the speed (v) values of the MLD model are consistently lower than those of the original system. As a result, the speed (v) values of the MLD model consistently remain lower than those of the original system. Notably, at time $t = 19.8$s, the speed v of the MLD model reaches its reference of 20 m/s. In the region around

a speed of 20 m/s, the car is in the acceleration process, leading to a higher final speed in the original system compared to the MLD model.

Furthermore, as the position (x) is the time integral of speed (v), and since the speed values are positive and lower in the MLD model compared to the original system, the position of the MLD model is also lower than that of the original system.

Despite the slight underestimation of the states x and v by the MLD model, the MLD model still serves as a suitable approximation of the original system. The observed discrepancy between the models is not significant, and the MLD model is capable of capturing the dynamics of the states. This implies that the trends of the states, such as where changes occur and the values of turning points, remain consistent between the MID model and the original system.

In a noteworthy achievement, the speed (v) of the MLD model reaches the reference speed v_{ref} of 20 m/s precisely at time $t = 19.8$ s. This remarkable outcome serves as a compelling evidence of the effectiveness and success of the MILP problem.

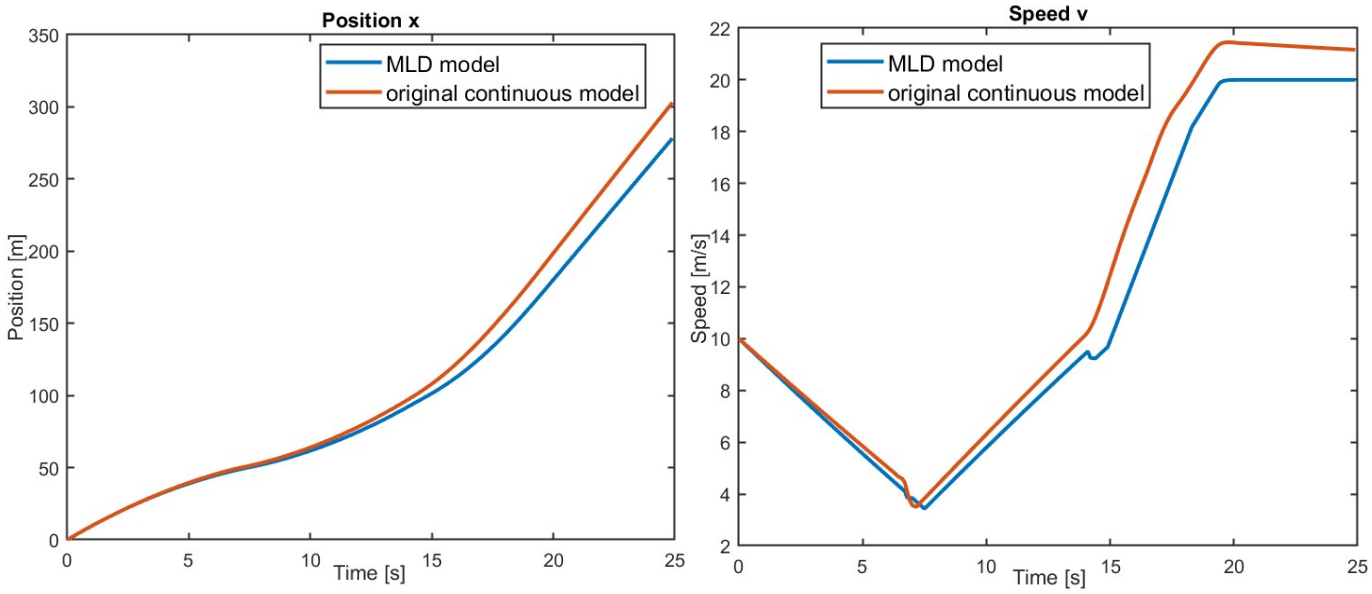

Fig. 8. Plots of position x and speed v of MLD model and original continuous model respectively .

I. Step 2.9

In this part, the speed reference signal is defined as:

$$v_{ref}(t) = \begin{cases} 0.85\alpha & 0 \leq t \leq 3 \\ 1.2\alpha & 3 < t \leq 9 \\ 1.2\alpha - \frac{1}{12}\alpha(t-9) & 9 < t \leq 15 \\ 0.7\alpha & 15 < t \leq 18 \\ 0.7\alpha + \frac{4}{15}\alpha(t-18) & 18 < t \leq 21 \\ 0.9\alpha & 21 < t \leq 30 \end{cases} \tag{71}$$

Instead of using the constant $v_{ref} = 20$ m/s like step 2.8 does, the time-varying speed reference signal is used. According to equation (59), the constraints for MPC problem which set limit for speed v is written as

$$-\rho_i \leq \hat{v}(k+i) - \hat{v}_{ref}(k+i) = \hat{z}_2(k+i) - \hat{v}_{ref}(k+i) \leq \rho_i, \ \ for\ i = 1, 2..., N_p$$

where $\hat{v}_{ref}(k+i) = v_{ref}(k+i)$ according equation (71). Assume $\rho = \begin{bmatrix}\rho_1 & \dots & \rho_{N_p}\end{bmatrix}^T$, $\hat{v}(k) = \begin{bmatrix}\hat{v}(k+1) & \dots & \hat{v}(k+N_p)\end{bmatrix}^T$ and $\hat{v}_{ref}(k) = \begin{bmatrix}\hat{v}_{ref}(k+1) & \dots & \hat{v}_{ref}(k+N_p)\end{bmatrix}^T$, then the constraint can be represented as:

$$-\rho \leq \hat{v}(k) - \hat{v}_{ref}(k) \leq \rho$$

$$-\rho + \hat{v}_{ref}(k) \leq \hat{v}(k) \leq \rho + \hat{v}_{ref}(k)$$

Then assume

$$R_2 = \begin{bmatrix} 0 & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix}_{(N_p)*(2N_p)}$$

$$\hat{v}(k) = [z_2(k+1)...z_2(k+N_p)]^T = R_2\hat{z}(k)$$

$$-\rho + \hat{v}_{ref}(k) \leq R_2\hat{z}(k) \leq \rho + \hat{v}_{ref}(k) \tag{72}$$

The equation (72) could be further transformed into constraint $F_{e11}\tilde{V}_e(k) \leq F_{e21} + F_{e31}$ with $F_{e21} = \begin{bmatrix}\hat{v}_{ref}(k) \\ -\hat{v}_{ref}(k)\end{bmatrix}$ which is the sub-matrix of constraint matrix F_{e2} in MILP problem . That means, the MILP problem is not constant model but with sub-matrix F_{e21} in matrix F_{e2} changing with in different simulation time steps. By changing matrix F_{e21} each time step in simulation horizon according to speed reference signal (71). In this case the parameters of (N_p, N_c) are chosen as the two pairs $(N_p = 5, N_c = 4)$ and $(N_p = 7, N_c = 5)$ respectively.

The plots of the profile of road, the position of x, and acceleration are shown in Figs (9). The road profile provides information about the slope of the road surface. The surface profile can be divided into three distinct regions:

$$y = \begin{cases} \frac{2h}{w}x & \text{if } 0 \leq x < w \\ \frac{h}{w}x + h & \text{if } w \leq x < 2w \\ 3h & \text{if } x \geq 2w \end{cases}$$

To facilitate a more comprehensive analysis of the MILP problem's performance, we denote the time instances when the car enters different regions in all the plots representing acceleration, reference velocity (v_{ref}), actual velocity (v), velocity deviation ($v_{v_{ref}}$), optimal input (u), and input variation (Δu). When $t = 2.4$s, the car drives from road region 1 to road region 2. When $t = 5.2$s, the car drives from road region 2 to road region 3. When the car is in road region 1, it undergoes a braking process characterized by a negative time derivative of speed. In road region 2, the car experiences nearly zero acceleration, but the plot with $(N_p = 5, N_c = 4)$ shows larger fluctuations around zero compared to the plot with $(N_p = 7, N_c = 5)$. When the car enters region 3, the acceleration changes at time points 9, 15, 18 and 21 s, which are the time turning points of speed reference signal.

The plots of v_{ref}, v and $v - v_{ref}$ are shown in Figs (10). It could be found that the speed could not track the reference with negative time derivative of speed in road region 1. The speed decreased from 23.1451 to 18 m/s. Through analysis of the control input u, it is observed that during this period, the control input reaches the maximum throttle input u_{max}. This indicates that the car has already reached its maximum capability for acceleration. However, in road region 1, the car's ability to speed up is hindered by the large slope of road, causing the acceleration due to gravity to be stronger than the car's acceleration capacity. In the majority of road region 2, the car maintains a speed of 18 m/s. However, the plot with $(N_p = 5, N_c = 4)$ exhibits larger fluctuations around 18 m/s compared to the plot with $(N_p = 7, N_c = 5)$. Additionally, in region 2, the car struggles to track the reference speed despite employing a large throttle

input that is close to u_{max}. In this scenario, the car's ability to accelerate is balanced by its ability to decelerate due to the large slope of the road. This equilibrium results in an acceleration due to gravity, which serves as evidence that the slope in region 1 is steeper than that in region 2. In region 3, the car reaches the region 3 which is flat with zero slope. In this case, the car is not affected by the slope of roads. However, due to the constraint of maximal acceleration/deceleration, where the absolute value of the acceleration at any time, denoted as $|a(t)|$, must not exceed $a_{comf,max}$, the car's speed can only increase with limited acceleration. As a result, the speed may not be able to perfectly match the speed reference in regions where the car's speed deviates significantly from the reference speed or when the rate of change of the speed reference exceeds the upper acceleration limit.

The plots of optimal input u and Δu are shown in Figs (11). The impulses of plot Δu occur at time points where there are significant changes in the difference between the current speed (v) and the reference speed (v_{ref}), and when comparing the plots of acceleration, optimal input, and Δu for the parameter pairs $(N_p = 5, N_c = 4)$ and $(N_p = 7, N_c = 5)$, fluctuations can be observed, indicating that the stability of the MILP system with $(N_p = 5, N_c = 4)$ is inferior to that of the MILP system with $(N_p = 7, N_c = 5)$.

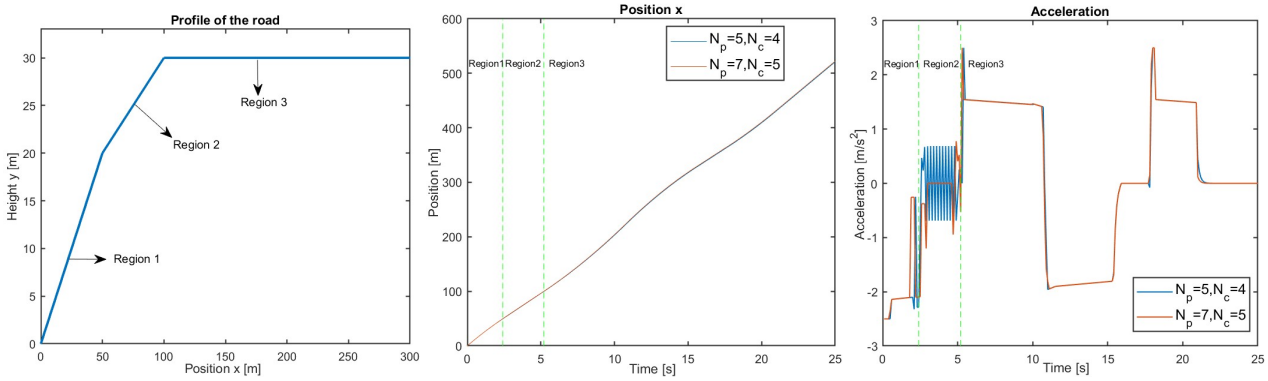

Fig. 9. Plots of the profile of road, the position of x, and acceleration .

J. Step 2.10

When applying MPC for controlling hybrid systems, using an MILP solver to compute the optimal input at each step can consume a significant amount of computation time and resources. Particularly, when dealing with a large prediction horizon, the solver's analysis time becomes even longer. If the analysis time for each step exceeds the sampling time, the computed optimal solution will no longer be applicable to the real-time system.

In this section, we employ an explicit MPC approach to solve this problem. Firstly, we construct a lookup table for the optimal inputs for each combination of prediction horizon and control horizon. Then, we use interpolation techniques to find the best input based on current voltage and voltage reference and then compared it with the result obtained by implicit MPC approach previously.

1) generate lookup table of optimal inputs $u^*(k)$: In the previous section, we utilized two combinations of horizons, namely (N_p = 5, N_c = 4) and (N_p = 7, N_c = 5). For each of these horizon sets, we will create a separate lookup table to compare the results.

Once the prediction horizon and control horizon are determined, the magnitude of the optimal input depends solely on the current velocity, the reference velocity, and the current road gradient. In this assignment, there are three sections of the road with different gradient. Therefore, in a single lookup table, we have categorized three different road conditions and computed the corresponding $u^*(k)$ for driving on each section of the road.

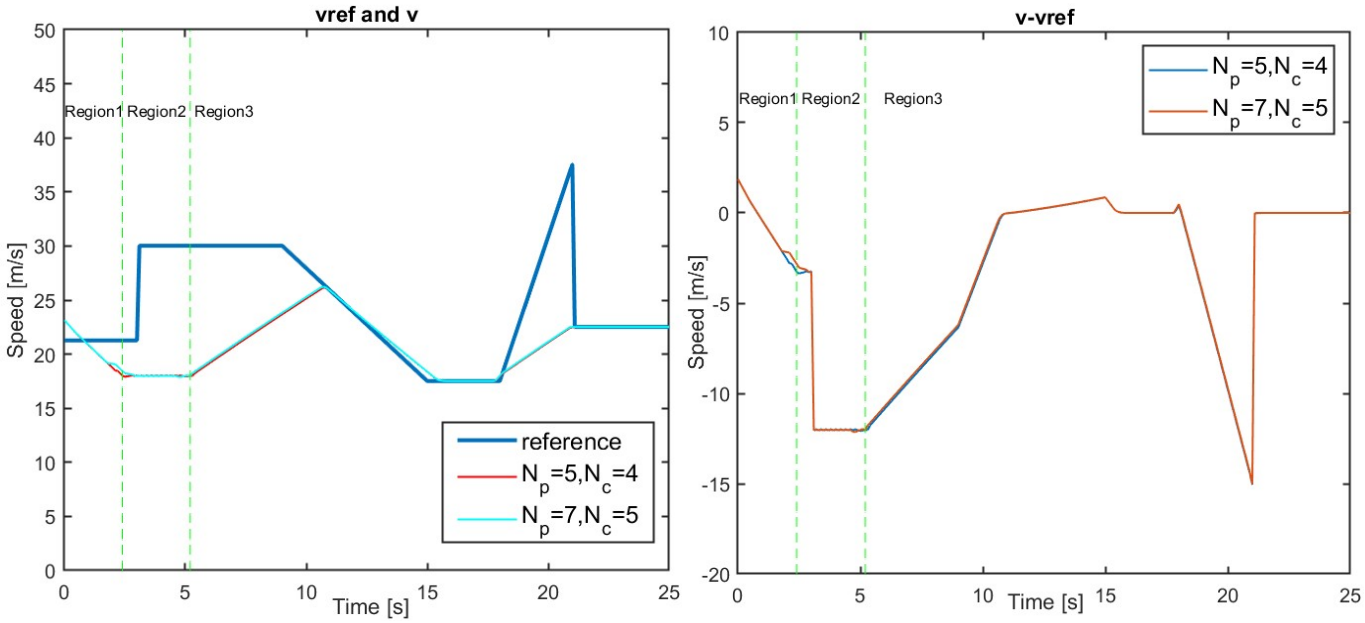

Fig. 10. Plots of v_{ref}, v and $v - v_{ref}$.

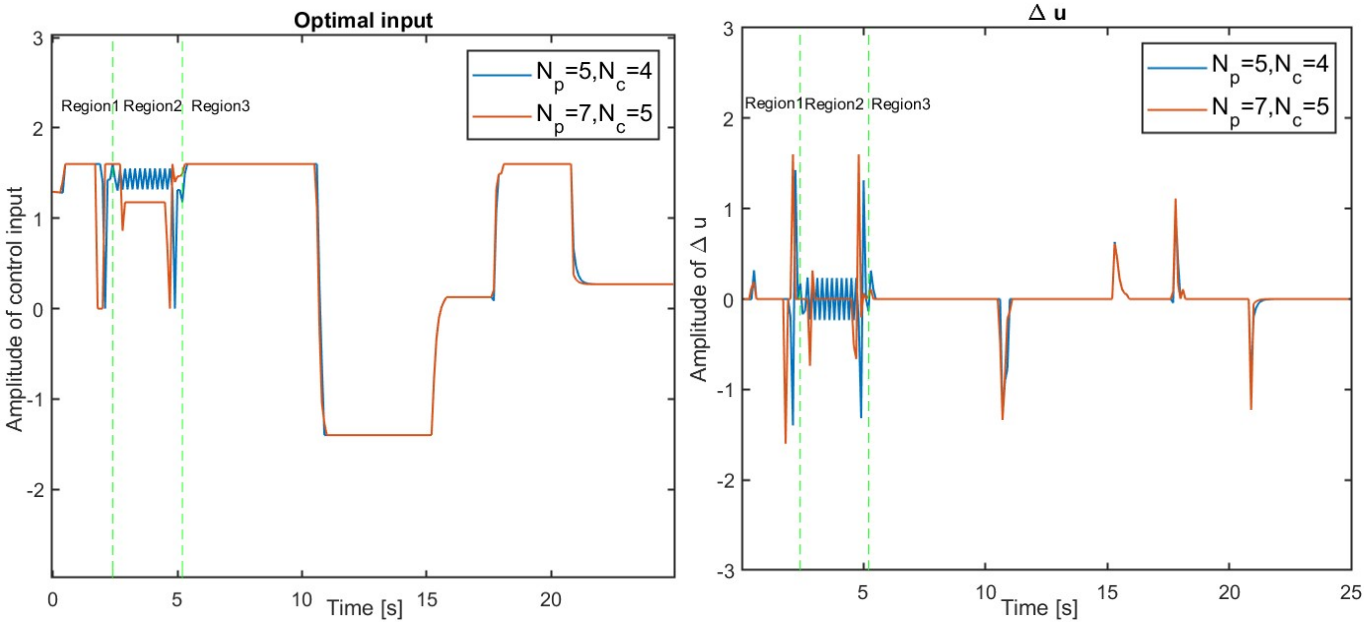

Fig. 11. The plots of optimal input u and Δu respectively .

Firstly, we will determine the range for the velocity and reference velocity to create the lookup table. The reference velocity range can be set from 0 to v_{max}. In principle, the velocity range should also be from 0 to v_{max}. However, based on the simulation results from the previous section, the maximum velocity observed was around $26m/s$. Therefore, to save computation time, we will limit the velocity range in the lookup table to 0-30.

Next, we set $v = 0 : 1 : 30$ and $v_{ref} = 0 : 1 : 50$ and use MILP solver to calculate $31 * 51 = 1581$ optimal inputs for each road section and then store all the results in a look-up table.

Once we generate look-up table, it can be used to apply the explicit MPC approach to control the system. At each step, we start by using **if** statement to assess the current section of the road where the vehicle is located. Subsequently, we employ **interp2** function to interpolate based on the current velocity and reference velocity, using the relevant data from the lookup table corresponding to the particular road section.

2) repeat step 2.7 and 2.8: In step 2.7, we set initial state $z_0 = [0; 10]$ and reference velocity as a constant value $v_{ref} = 20m/s$. The comparison of results between the explicit MPC method and the implicit MPC method is depicted in Figure 12.

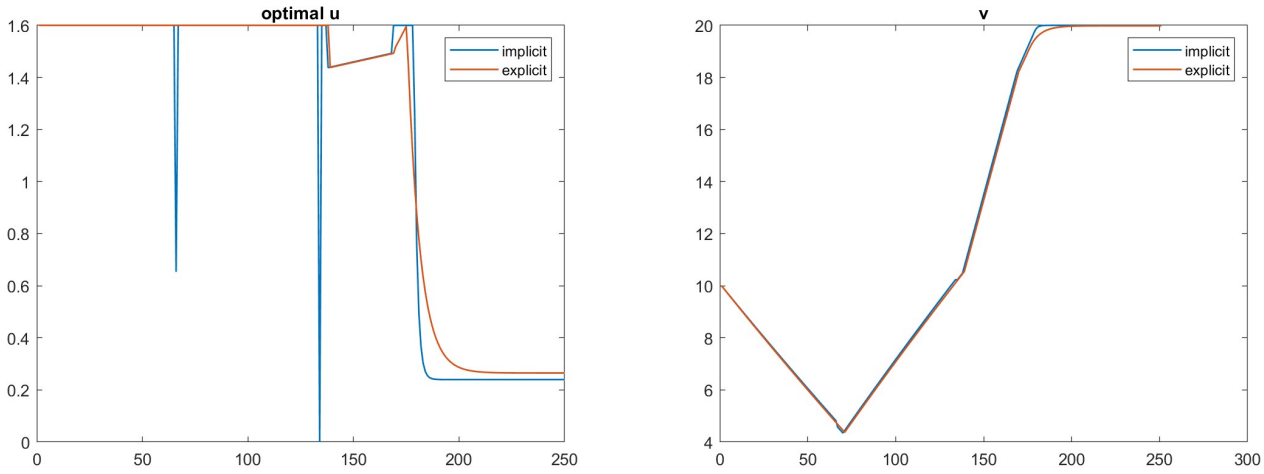

Fig. 12. results comparison between explicit and implicit MPC method .

The graph reveals that when the reference velocity is constant, the results obtained from both methods are largely similar. Additionally, the explicit method produces a smoother input sequence without any spikes. When there are abrupt changes in the input, the explicit method exhibits a relatively slower rate of change.

The reason for the similarity in results between the two methods is that we set the reference velocity as a constant. When there is a significant difference between the current velocity and the reference velocity, a wide range of reference velocities will yield the same optimal input. For instance, when the current velocity is 1 and the reference velocity ranges from 5 to 50, the optimal input obtained remains the same. In such cases, the interpolated input obtained through the interpolation method aligns with the actual optimal input.

The reason behind the smoother input curve obtained from the explicit method is that, in the online method, it can compute the optimal input based on real-time calculations for future scenarios when there are sudden changes in road conditions or reference velocity. In this process, the optimal input sequence may contain spikes or abrupt changes to help the actual velocity converge quickly towards the reference value. On the other hand, during interpolation, we only calculate the optimal inputs for a few specific points, potentially overlooking any spikes or abrupt changes between those points. As a result, we obtain a smoother input sequence. This is also why the implicit method can reach the reference velocity faster than the explicit method because the interpolated inputs do not represent the true optimal inputs.

Then we repeat 2.8 to apply the input sequence to the original continuous-time model. The comparison of result in step 2.8 between result obtained by explicit MPC method is shown in Figure 13.

From the figure, we can observe that the results of both methods are almost the same, with only minor differences appearing towards the end. This is because the control inputs obtained from the two methods

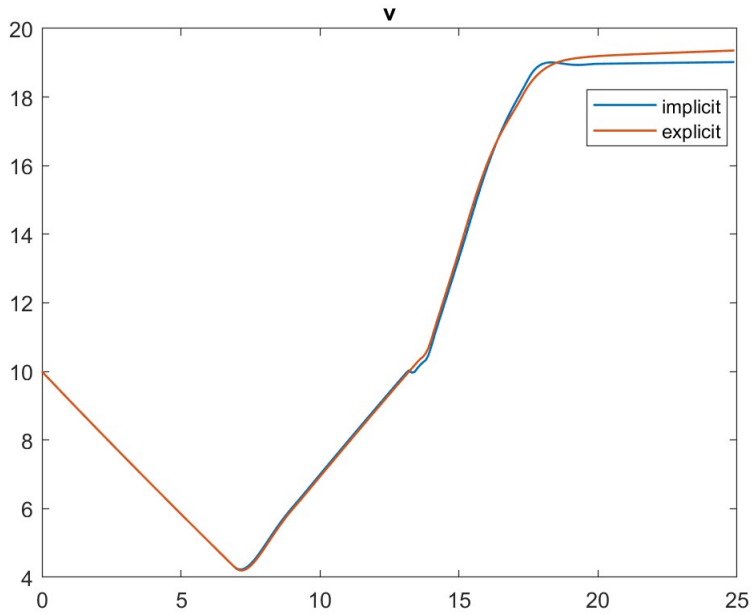

Fig. 13. 2.8 results comparison between explicit and implicit MPC method

exhibit slight variations during the final segment, the implicit method reaching the reference velocity faster. As a result, these subtle differences manifest in the performance of original continuous-time system during the last section.

3) repeat step 2.9: We repeat step 2.9 and compare with results we obtained before. When $N_p = 5, N_c = 4$, the result comparison is shown in Figure 14.

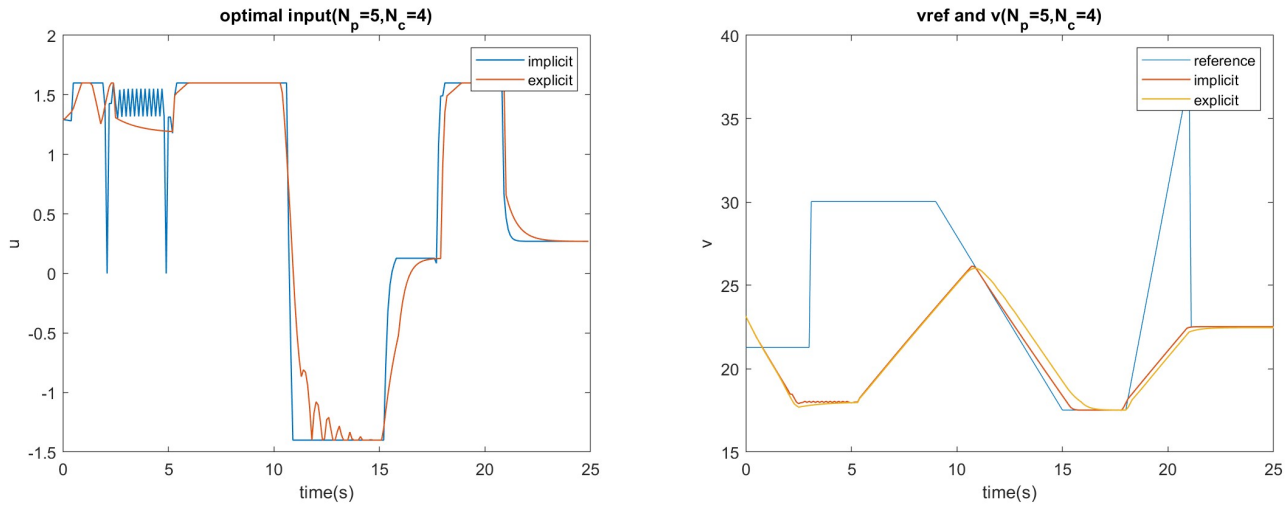

Fig. 14. results comparison between explicit and implicit MPC method($N_p = 5, N_c = 4$) .

From the graph, it is clear that the explicit method generates a smoother input sequence. For instance, between 2 and 5 seconds, the input obtained by the implicit method fluctuates up and down, whereas the input resulted from explicit method forms a smooth curve. We have already discussed the underlying reasons in the previous section.

Furthermore, it is worth noting that there are notable differences between the results obtained by the two methods between 11 and 17 seconds. This is attributed to the constant reference velocity used when creating the lookup table, which does not account for the time-varying nature of the actual reference velocity. For instance, during the mentioned time interval in the graph, the reference velocity decreases over time. The explicit method's interpolated inputs assume that the future reference velocity will remain unchanged, causing the actual velocity to decline at a slower rate. Consequently, the explicit method fails to accurately predict the relationship between the actual velocity and the future reference values.

When $N_p = 7, N_c = 5$, the result comparison is shown in Figure 15.

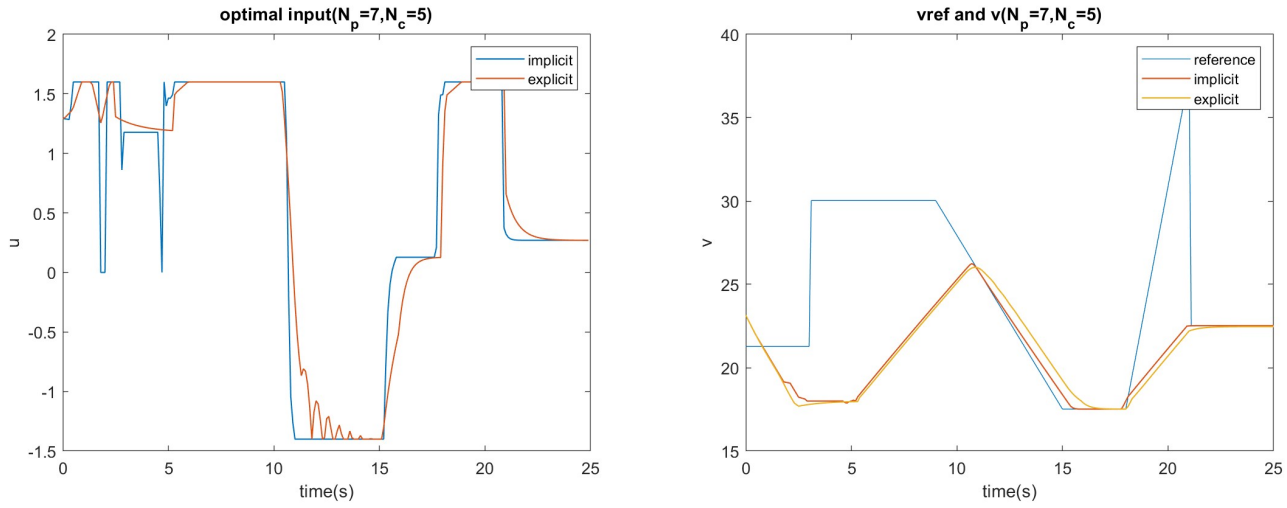

Fig. 15. results comparison between explicit and implicit MPC method($N_p = 7, N_c = 5$) .

Similar to the comparison results of the previous set of horizon combinations, overall, the explicit method produces a smoother input sequence, especially when there are sudden changes in the reference signal. The two methods exhibit differences in the results between 11 and 17 seconds, while they have similar results during other time intervals.

4) compare computation times: In this section, we used the same conditions as in step 2.9, including the reference velocity, initial state, and simulation time. Then we apply the implicit method and explicit method to three different sets of horizons: $N_p = N_c \in \{2, 3, 4\}$ and compare its computation times. The computation time of different horizons and methods are shown in table 1.

TABLE I
COMPUTATION TIME

$N_p = N_c$	off-line	on-line
2	0.0597s	1.6620s
3	0.0472s	1.6993s
4	0.0522s	1.7730s

It is evident from the table that the online computation time is generally much longer than the offline computation time.

For the explicit method, the computation time for different horizons is consistently around 0.05 seconds. This similarity can be attributed to the same size of each lookup table, resulting in similar interpolation times for obtaining the results.

For the implicit method, it is clear that the computation time increases as the horizon size grows larger. This is because we convert the MPC problem into an MILP problem and then solve it using an MILP solver. During the conversion process, as the horizon size increases, certain matrices have more rows, resulting in longer computation times for the MILP solver to provide the solution.

III. Conclusion

In the first question, we described a hybrid system that is commonly encountered in our daily life. We provided a detailed description of various elements within the traffic hybrid system, including different modes, guard conditions, transitions, reset maps, and so on. By exploring this question, we have successfully established a connection between abstract concepts and the various elements present in real-world systems. As a result, we have gained a more profound comprehension of the abstract concepts through their practical application.

In step 2.1, we established the mathematical representation of the system and then derive the state function of system based on the real-world motion equations for vehicles. The original system is a nonlinear system, to simplify the question, we construct a piece wise-affine approximation to transfer the system to a linear system in step 2.2.

From step 2.3 to step 2.5, we built the PWA model step by step and finally we got the PWA model with nine regions in total. In step 2.6, we transformed the PWA model into an MLD model with binary and real variables only. The system's state is divided into three segments within different regions, it is quite challenging to represent these segments using binary variables. Later on, we managed to overcome this challenge. To represent three different cases, we needed at least two binary variables. However, since these two binary variables corresponded to the same state, they were not independent. To impose additional constraints on their relationship, we introduced an additional binary variable equal to the product of these two variables. Furthermore, during the computation of the state function, a constant term emerged. However, the standard form of the MLD model does not include a constant term. To address this, we introduced a binary variable that is always equal to 1 and multiplied it by the constant term. As a result, the constant term became incorporated into the coefficient matrix of the binary variables. Indeed, there are many tricks to transform various complex hybrid systems into a standard form of the MLD model, making subsequent computations more convenient. The process of exploring and utilizing these tricks can be highly engaging and fascinating.

In step 2.7 we are going to design an MPC controller to control the MLD model. In this assignment, the MPC cost function consisted of both the 1-norm and the infinity-norm. To convert the MPC problem into an MILP problem, we incorporated additional dummy variables into the formulation. The transformation process involved complex matrix calculations, and once we obtained the final standard form, we implemented the computed matrices in MATLAB. We selected the MILP solver **glpk** to solve the transformed MILP problem. Initially, we encountered difficulties in obtaining a feasible solution due to minor errors in the previous matrix computations. Fortunately, our calculation process was divided into many independent small steps, and all the computations are kept symbolic. As a result, once an error was identified, we only needed to make small modifications to rectify the issue, allowing for overall progress to be achieved.

In step 2.8, we applied the inputs obtained in step 2.7 to the original continuous nonlinear system and observed that the results were roughly similar to those obtained in step 2.7. This indicates that the MLD model we constructed earlier was correct.

In step 2.9, we conducted a detailed analysis of the impact of different horizons on MPC control by comparing reference velocity with real velocity, acceleration, optimal inputs, and other data. By examining the graphs depicting the reference velocity versus real velocity and the variations in optimal inputs, we

discovered that the control inputs in different regions of the system were reasonable and aligned with expectations. This confirms the correctness of the MILP formulation we computed in step 2.7.

In step 2.10, we addressed the issue of long computation times associated with the implicit MPC method by exploring the use of the explicit MPC method. In the first half of step 2.10, we compared the control performance of both methods and observed that, in most cases, the control results obtained from both methods were highly similar. In the second half, we evaluated the computation times of the two methods and found that, even with a small horizon (N_p=2), the explicit MPC method exhibited significantly shorter computation times compared to the implicit MPC method. Considering these results, we can conclude that the explicit MPC method is preferable due to its shorter computation times while still achieving control performance comparable to the implicit MPC method.

Overall, this assignment was challenging for us. We invested a significant amount of time to complete it, but we also learned a lot about hybrid system control concepts and techniques.

APPENDIX

A. *Code*

The overall code in MATLAB is listed as:

```
clear;
close all;

%% parameters definition

m=700;
g=9.8;
c=0.5;
b=3600;
umax=1.6;
umin=-1.4;
a_comf_max=2.5;
gamma=1.8;
vg=18;
h=10;
w=50;

%% Question 2.1

% maximal speed Vmax
v_max=sqrt(b/c*umax/(1+2*gamma));

% Given minimal braking input umin, maximal acceleration and deceleration

u=umin;
v=0:0.1:v_max;
a=(b/m/(1+gamma)*u-c/m*v.^2).*(v<vg)+(b/m/(1+2*gamma)*u-c/m*v.^2).*(v>=vg);
figure();
plot(v,a);
axis([0 v_max -4 3]);
title('The velocity deceleration')
xlabel('v [m/s]');
ylabel('dv(t)/dt [m/s^2]');
a_acc_min=min(a)
```

```
% Given maximal braking input umax, maximal acceleration and deceleration
u=umax;
v=0:0.1:v_max;
a=(b/m/(1+gamma)*u-c/m*v.^2).*(v<vg)+(b/m/(1+2*gamma)*u-c/m*v.^2).*(v>=vg);
figure();
plot(v,a);
axis([0 v_max -3 3]);
title('The velocity acceleration')
xlabel('v [m/s]');
ylabel('dv(t)/dt [m/s^2]');
a_acc_max=max(a)

%% Question 2.2
syms n1 n2 v

S1=(c*v^2-n2/n1*v)^2;
S1=int(S1,[0 n1]);
p=(c*v_max.^2-n2)/(v_max-n1);
q=n2-p*n1;
S2=(c*v^2-p*v-q)^2;
S2=int(S2,[n1 v_max])
JAC=jacobian(S1+S2,[n1,n2]);


eqns = [JAC(1,1) == 0,JAC(1,2) == 0];
SS = solve(eqns,[n1 n2])
% solution1 = solve(JAC(1,1) == 0, [n1,n2])
% solution2 = solve(JAC(1,2) == 0, [n1,n2])

xo=[SS.n1(1); SS.n1(2);SS.n1(3)] ;
yo=[SS.n2(1); SS.n2(2);SS.n2(3)];
uo=['\leftarrow Critical point 1'; '\leftarrow Critical point 2'; '\leftarrow
    Critical point 3'];

figure()
plot(xo,yo,'o')
for i=1:length(xo)
text(xo(i),yo(i),num2str(uo(i,:)))
end
vv=0:0.01:v_max;
hold on
plot(vv,c*vv.^2)
title('V(v) and critical points')
xlabel('v [m/s]');


figure()
plot(vv,c*vv.^2,'r')
hold on
line([0,SS.n1(3)],[0,SS.n2(3)]);
hold on;
scatter(SS.n1(3),SS.n2(3),'b')
hold on
line([SS.n1(3),v_max],[SS.n2(3),c*v_max^2]);
title('V(v) and P(v)')
xlabel('v [m/s]');
```

```
legend('V(v)','P(v)');

% Test the second partial derivative

Hess(n1,n2)=hessian(S1+S2,[n1,n2])
H=Hess(SS.n1(3),SS.n2(3));

%tes1 H(1,1)*H(2,2)-H(1,2)^2>0,local minimal or maximal
HH=H(1,1)*H(2,2)-H(1,2)^2
%tes2 H(1,1)>0, local minimal
H(1,1)

%% Question 2.3

alpha=25.0217;
beta=234.7826;
a1=beta/alpha;
a2=(c*v_max^2-beta)/(v_max-alpha);
b2=(a1-a2)*alpha;


qq=b/m/(1+gamma);
t=0:0.1:20;
tspan=[0 200];
x0=[0;20];
% x0=[0;10];
% x0=[0;38];
[t,y]=ode45(@(t,y)vdp1(t,y,qq,alpha,beta,v_max,c,m),tspan,x0);
[t1,y1]=ode45(@(t,y)vdp2(t,y,qq,m,c),tspan,x0);
figure()
plot(t,y(:,1),'-o',t1,y1(:,1),'-o')
title('x(t) of PWA and original system');
xlabel('Time t [s]');
ylabel('x(t) [m]');
legend('PWA','original system')
figure()
plot(t,y(:,2),'-o',t1,y1(:,2),'-o')
title('v(t) of PWA and original system');
xlabel('Time t [s]');
ylabel('v(t) [m/s]');
legend('PWA','original system')
P=(a1*y(:,2)).*(y(:,2)<alpha)+(a2*y(:,2)+b2).*(y(:,2)>=alpha);
a1=qq*sin(t)-P/m;
a2=qq*sin(t1)-c/m*y1(:,2).^2;
figure()
plot(t,a1,'-o',t1,a2,'-o')
title('a(t) of PWA and original system');
xlabel('Time t [s]');
ylabel('a(t) [m/s^2]')
legend('PWA','original system')

%% Question 2.4

% Relationship of y to x

x=0:0.01:150;
```

```
[y,selection]=min([2*h/w.*x;h/w.*x+h.*ones(size(x));3*h.*ones(size(x));3*h/2/w.*x+9*h
    .*ones(size(x))]);
figure()
plot(x,y)
xlabel('m [m/s]');
ylabel('y [m/s]');
title('Height y');

% Approximate sin(theta) to theta

for i=1:(150/0.01+1)
    if selection(i)==1
        theta(i)=atan(2*h/w)*pi/180;
    elseif selection(i)==2
        theta(i)=atan(h/w)*pi/180;
    elseif selection(i)==3
        theta(i)=0;
    else
        theta(i)=atan(3*h/2/w)*pi/180;
    end
end

figure()
plot(x,theta)
xlabel('theta');
ylabel('y [m/s]');
title('Height y');

%% Question 2.7

T=0.1;%sample time
theta1=0.38;
theta2=0.197;
p1 = 9.3750;
p2 = 40.6413;
q2 = -781.657;

%unknown
epsilon=0;
xmax=10000;

%define variable dimension
dim.nu=1;
dim.ndel=7;
dim.nn=2;
dim.nz=2;

%define constraint dimension
constraint.con1=6;
constraint.con2=8;
constraint.con3=12;
constraint.con4=8;

%define system matrix A,B1,B2,B3
A=[ 1                   T;
    0               1-T*p1/m];
```

```
B1=[0;T*b/(m*(1+2*gamma))];
B2=[0                          0     0     0   0 0       0;
    g*T*(theta2-theta1) g*T*theta2 0 -T*q2/m 0 0 -g*T*theta2];
B3=[     0                      0                      ;
    T*(p1-p2)/m     T*b/m*(1/(1+gamma)-1/(1+2*gamma));];

%define MPC parameters
Np=5;
Nc=4;

% define constraint matrices and vectors
E11=[-1  0;
      1  0;
      0 -1;
      0  1;
      0  0;
      0  0;];
E21=[ 0;
      0;
      0;
      0;
     -1;
      1];
E31=zeros(constraint.con1,dim.ndel);
E41=zeros(constraint.con1,dim.nn);
g1=[0;xmax;0;v_max;-umin;umax];

E12=[1  0;
     -1 0;
     -1 0;
     1  0;
     0  1;
     0 -1;
     0 -1;
     0  1];
E22=zeros(constraint.con2,dim.nu);
E32=[xmax-w     0               0                  0            0 0 0;
    -w-epsilon  0               0                  0            0 0 0;
    0          2*w              0                  0            0 0 0;
    0    -xmax+2*w-epsilon      0                  0            0 0 0;
    0           0            v_max-vg              0            0 0 0;
    0           0           -vg-epsilon            0            0 0 0;
    0           0               0               alpha           0 0 0;
    0           0               0        -v_max+alpha-epsilon   0 0 0;];
E42=zeros(constraint.con2,dim.nn);
g2=[xmax;-epsilon-w;0;2*w-epsilon;v_max;-epsilon-vg;0;alpha-epsilon];

E13=zeros(constraint.con3,dim.nz);
E23=zeros(constraint.con3,dim.nu);
E33=[ 0  0  0  0 -1  0  0;
      0  0  0  0  1  0  0;
     -1  0  0  0  1  0  0;
      0 -1  0  0  1  0  0;
      1  1  0  0 -1  0  0;
      0  0  0  0  0 -1  0;
      0  0  0  0  0  1  0;
```

```
      0   0 -1   0   0   1   0;
      0   0   0 -1   0   1   0;
      0   0   1   1   0 -1   0;
      0   0   0   0   0   0   1;
      0   0   0   0   0   0 -1;];
E43=zeros(constraint.con3,dim.nn);
g3=[0 0 0 0 1 0 0 0 0 1 1 -1]';

E14=[0 0;
     0 0;
     0 -1;
     0 1;
     0 0;
     0 0;
     0 0;
     0 0;];
E24=[zeros(constraint.con4-2,dim.nu);-1;1];
E34=[  0       0       0       -v_max    0 0 0;
       0       0       0         0       0 0 0;
       0       0       0         0       0 0 0;
       0       0       0        v_max    0 0 0;
       0       0     -umax       0       0 0 0;
       0       0      umin       0       0 0 0;
       0       0     -umin       0       0 0 0;
       0       0      umax       0       0 0 0;];
E44=[1  0;
    -1  0;
     1  0;
    -1  0;
     0  1;
     0 -1;
     0  1;
     0 -1;];
g4=[0 0 0 v_max 0 0 -umin umax]';

E1=[E11;E12;E13;E14];
E2=[E21;E22;E23;E24];
E3=[E31;E32;E33;E34];
E4=[E41;E42;E43;E44];
vecg=[g1;g2;g3;g4];


I_nu=eye(dim.nu);
I_ndel=eye(dim.ndel);
I_nn=eye(dim.nn);

Cell_I_ndel1 = repmat({I_ndel}, 1, Nc);
Cell_I_ndel2 = [zeros((Np-Nc)*dim.ndel,dim.ndel*(Nc-1)),repmat(I_ndel,(Np-Nc),1)];
Cell_I_ndel = blkdiag(Cell_I_ndel1{:});

Cell_I_nu1 = repmat({I_nu}, 1, Nc);
Cell_I_nu2 = [zeros((Np-Nc)*dim.nu,dim.nu*(Nc-1)),repmat(I_nu,(Np-Nc),1)];
Cell_I_nu = blkdiag(Cell_I_nu1{:});

Cell_I_nn1 = repmat({I_nn}, 1, Nc);
Cell_I_nn2 = [zeros((Np-Nc)*dim.nn,dim.nn*(Nc-1)),repmat(I_nn,(Np-Nc),1)];
```

```
Cell_I_nn = blkdiag(Cell_I_nn1{:});

K_u=[Cell_I_nu;Cell_I_nu2];
K_delta=[Cell_I_ndel;Cell_I_ndel2];
K_n=[Cell_I_nn;Cell_I_nn2];

% define M2,T1,T2,T3
dim.p=size(A,1);
M2=A;
for i=2:Np
    M2=[M2;A^i];
end

for i=1:Np
    for j=1:i
        T1((i-1)*dim.p+1:i*dim.p,(j-1)*dim.nu+1:j*dim.nu)=A^(i-j)*B1;
    end
end

for i=1:Np
    for j=1:i
        T2((i-1)*dim.p+1:i*dim.p,(j-1)*dim.ndel+1:j*dim.ndel)=A^(i-j)*B2;
    end
end

for i=1:Np
    for j=1:i
        T3((i-1)*dim.p+1:i*dim.p,(j-1)*dim.nn+1:j*dim.nn)=A^(i-j)*B3;
    end
end

% define M1
M1 = [T1*K_u T2*K_delta T3*K_n];

% define F31, Q1,Q2,Q3
F31=E1;
for i=2:Np
    F31=[F31;E1*A^(i-1)];
end

dim.e=size(E1,1);
for i=1:Np
    Q1((i-1)*dim.e+1:i*dim.e,(i-1)*dim.nu+1:i*dim.nu)=E2;
end
for i=2:Np
    for j=1:i-1
        Q1((i-1)*dim.e+1:i*dim.e,(j-1)*dim.nu+1:j*dim.nu)=E1*A^(i-j-1)*B1;
    end
end

for i=1:Np
    Q2((i-1)*dim.e+1:i*dim.e,(i-1)*dim.ndel+1:i*dim.ndel)=E3;
end
for i=2:Np
    for j=1:i-1
        Q2((i-1)*dim.e+1:i*dim.e,(j-1)*dim.ndel+1:j*dim.ndel)=E1*A^(i-j-1)*B2;
```

```
    end
end

for i=1:Np
    Q3((i-1)*dim.e+1:i*dim.e,(i-1)*dim.nn+1:i*dim.nn)=E4;
end
for i=2:Np
    for j=1:i-1
        Q3((i-1)*dim.e+1:i*dim.e,(j-1)*dim.nn+1:j*dim.nn)=E1*A^(i-j-1)*B3;
    end
end


% define F11
F11 = [Q1*K_u Q2*K_delta Q3*K_n];

% define R1
R1=[0 1];
R1=repmat({R1}, 1, 2*Np);
R1=blkdiag(R1{:});

% define P1 P2 P3 P4
I=eye(dim.nz);
P1=A-I;
for i=2:Np
    P1=[P1;(A-I)*A^(i-1)];
end
P1 =[P1;-P1];


for i=1:Np
    P2((i-1)*dim.p+1:i*dim.p,(i-1)*dim.nu+1:i*dim.nu)=B1;
end
for i=2:Np
    for j=1:i-1
        P2((i-1)*dim.p+1:i*dim.p,(j-1)*dim.nu+1:j*dim.nu)=A^(i-j)*B1-A^(i-j-1)*B1;
    end
end
P2 =[P2;-P2];

for i=1:Np
    P3((i-1)*dim.p+1:i*dim.p,(i-1)*dim.ndel+1:i*dim.ndel)=B2;
end
for i=2:Np
    for j=1:i-1
        P3((i-1)*dim.p+1:i*dim.p,(j-1)*dim.ndel+1:j*dim.ndel)=A^(i-j)*B2-A^(i-j-1)*B2
   ;
    end
end
P3 =[P3;-P3];

for i=1:Np
    P4((i-1)*dim.p+1:i*dim.p,(i-1)*dim.nn+1:i*dim.nn)=B3;
end
for i=2:Np
    for j=1:i-1
```

```
        P4((i-1)*dim.p+1:i*dim.p,(j-1)*dim.nn+1:j*dim.nn)=A^(i-j)*B3-A^(i-j-1)*B3;
    end
end
P4 =[P4;-P4];

% define F12, F32, F21, F22
F12 = [R1*P2*K_u R1*P3*K_delta R1*P4*K_n];
F32 = R1*P1;
F21 = repmat({vecg}, Np, 1);
F21 = cell2mat(F21);
F22 = a_comf_max*T*ones(2*Np,1);

% define F1,F2,F3
F1=[F11;F12];
F2=[F21;F22];
F3=[-F31;-F32];


% define obejctive function

lambda=0.1;% tune this value
vref=20*ones(Np,1);% set reference


H1=getH1(Np);
H2=zeros(Np,(Nc*(dim.nu+dim.nn+dim.ndel)+Np+1));
H2(:,end)=ones(Np,1);
H3=[H1*K_u zeros(Np,Nc*(dim.ndel+dim.nn)+Np+1)];
S=[zeros(1,Nc*(dim.nu+dim.nn+dim.ndel)) ones(1,Np) lambda];
R2=[0 1];
R2=repmat({R2}, 1, Np);
R2=blkdiag(R2{:});

Fe11=[R2*M1 -eye(Np) zeros(Np,1);
     -R2*M1 -eye(Np) zeros(Np,1)];
Fe21=[vref;-vref];
Fe31=[-R2*M2;R2*M2];
Fe12=[H3-H2;-H3-H2];
Fe13=[F1 zeros(size(F1,1),Np+1)];
Fe1=[Fe11;Fe12;Fe13];
Fe2=[Fe21;zeros(size(Fe12,1),1);F2];
Fe3=[Fe31;zeros(size(Fe12,1),2);F3];

%% MPC
N=250;          %simulation horizon
z0=[0;10];      %initial state
zopt=zeros(2,N+1);
zopt(:,1)=z0;
uopt=zeros(1,N);

% glpk

for i=1:N
    c=S;
    a=Fe1;
    bineq=Fe2+Fe3*zopt(:,i);
```

```
    lb=[];
    ub=[];
    ctype=repmat('U', 1, size(Fe1,1));
    vartype = [repmat('C', 1, Nc*dim.nu), repmat('B', 1, Nc*dim.ndel),repmat('C',1,Nc
   *dim.nn+Np+1)];
    sense=1;
    [xmin, fmin, status, extra] = glpk (c, a, bineq, lb, ub, ctype, vartype, sense);
    uopt(i)=xmin(1);
    zopt(:,i+1)=A*zopt(:,i)+B1*xmin(1:dim.nu)+...
        B2*xmin(Nc*dim.nu+1:Nc*dim.nu+dim.ndel)+...
        B3*xmin(Nc*(dim.nu+dim.ndel)+1:Nc*(dim.nu+dim.ndel)+dim.nn);
end

%% Question 2.8
tspan = 0:T:N*T-T;
[t_28, zcon] = ode45(@(t, y) conmodel(t, y, interp1(tspan, uopt, t)), tspan, z0);

figure()
plot(t_28,zopt(1,1:end-1));
hold on;
plot(t_28,zcon(:,1));
title('x')
legend('MLD model','original continuous model')

figure()
plot(t_28,zopt(2,1:end-1));
hold on;
plot(t_28,zcon(:,2));
title('v')
legend('MLD model','original continuous model')

%% Question 2.9
vref1=[0.85*alpha*ones(31,1);1.2*alpha*ones(60,1);...
    1.2*alpha*ones(60,1)-1/12*alpha*(0.1:0.1:6)';0.7*alpha*ones(30,1);...
    0.7*alpha*ones(30,1)+4/15*alpha*(0.1:0.1:3)';0.9*alpha*ones(90,1)];


z01=[0;alpha*0.925];     %initial state
zopt2=zeros(2,N+1);
zopt2(:,1)=z01;
uopt2=zeros(1,N);

% glpk

for i=1:N
    vref_9=vref1(i+1:i+Np);
    Fe21_9=[vref_9;-vref_9];
    Fe2_9=[Fe21_9;zeros(size(Fe12,1),1);F2];
    c=S;
    a=Fe1;
    bineq=Fe2_9+Fe3*zopt2(:,i);
    lb=[];
    ub=[];
    ctype=repmat('U', 1, size(Fe1,1));
    vartype = [repmat('C', 1, Nc*dim.nu), repmat('B', 1, Nc*dim.ndel),repmat('C',1,Nc
   *dim.nn+Np+1)];
```

```
    sense=1;
    [xmin, fmin, status, extra] = glpk (c, a, bineq, lb, ub, ctype, vartype, sense);
    uopt2(i)=xmin(1);
    zopt2(:,i+1)=A*zopt2(:,i)+B1*xmin(1:dim.nu)+...
        B2*xmin(Nc*dim.nu+1:Nc*dim.nu+dim.ndel)+...
        B3*xmin(Nc*(dim.nu+dim.ndel)+1:Nc*(dim.nu+dim.ndel)+dim.nn);
end


%% 2.9 comparison
sim_1=importdata('sim_1.mat');
sim_2=importdata('sim_2.mat');
vref1=importdata('vref1.mat');
uopt1=sim_1.uopt;
uopt2=sim_2.uopt;
zopt1=sim_1.zopt;
zopt2=sim_2.zopt;


%vref and v
figure()
plot(0:0.1:25,vref1(1:251))
hold on
plot(0:0.1:25,zopt1(2,:))
hold on
plot(0:0.1:25,zopt2(2,:))
title('vref and v')
legend('reference','N_p=5,N_c=4','N_p=9,N_c=6')

%x
figure()
plot(0:0.1:25,zopt1(1,:))
hold on
plot(0:0.1:25,zopt2(1,:))
title('x')
legend('N_p=5,N_c=4','N_p=9,N_c=6')

%v-vref
figure()
plot(0:0.1:25,zopt1(2,:)-vref1(1:251)')
hold on
plot(0:0.1:25,zopt2(2,:)-vref1(1:251)')
title('v-vref')
legend('N_p=5,N_c=4','N_p=9,N_c=6')

%optimal input(u)
figure()
plot(0:0.1:24.9,uopt1)
hold on
plot(0:0.1:24.9,uopt2)
title('optimal input')
legend('N_p=5,N_c=4','N_p=9,N_c=6')

%delta_u
figure()
plot(0.1:0.1:24.9,uopt1(2:end)-uopt1(1:end-1))
```

```
hold on
plot(0.1:0.1:24.9,uopt2(2:end)-uopt2(1:end-1))
title('\Delta u')
legend('N_p=5,N_c=4','N_p=9,N_c=6')

%acceleration
figure()
plot(0.1:0.1:25,(zopt1(2,2:end)-zopt1(2,1:end-1))/T)
hold on
plot(0.1:0.1:25,(zopt2(2,2:end)-zopt2(2,1:end-1))/T)
title('acceleration')
legend('N_p=5,N_c=4','N_p=9,N_c=6')

%% Question 2.10
%make lookup table
[Vm,Vrefm]=meshgrid(1:30,1:50);
Uoptm=zeros(50,30);
for i=1:50
    for j=1:30
            vref_off=i*ones(Np,1);
            Fe21_off=[vref_off;-vref_off];
            Fe2_off=[Fe21_off;zeros(size(Fe12,1),1);F2];
            c=S;
            a=Fe1;
            bineq=Fe2_off+Fe3*[120;j];
            lb=[];
            ub=[];
            ctype=repmat('U', 1, size(Fe1,1));
            vartype = [repmat('C', 1, Nc*dim.nu), repmat('B', 1, Nc*dim.ndel),repmat(
   'C',1,Nc*dim.nn+Np+1)];
            sense=1;
            [xmin, fmin, status, extra] = glpk (c, a, bineq, lb, ub, ctype, vartype,
   sense);
            Uoptm(i,j) = xmin(1);
    end
end

%repeat 2.7
zopt_7ex=zeros(2,N+1);
zopt_7ex(:,1)=z0;
uopt_7ex=zeros(1,N);

for i=1:N
    if zopt_7ex(1,i)<w
    uopt_7ex(i)= interp2(Vm,Vrefm,ulookup_54.u1,zopt_7ex(2,i),vref(1),'linear');
    elseif zopt_7ex(1,i)<2*w
     uopt_7ex(i)= interp2(Vm,Vrefm,ulookup_54.u2,zopt_7ex(2,i),vref(1),'linear');
    else
        uopt_7ex(i)= interp2(Vm,Vrefm,ulookup_54.u3,zopt_7ex(2,i),vref(1),'linear');
    end
    zopt_7ex(:,i+1)= [1,T;...
        0,...
        1-T/m*((p1*(zopt_7ex(2,i)<alpha))+p2*(zopt_7ex(2,i)>=alpha))]*zopt_7ex(:,i)
   +...
        [0;T*b/(m*(1+gamma*((zopt_7ex(2,i)<vg)+(zopt_7ex(2,i)>=vg)*2)))]*uopt_7ex(i)
   +...
```

```
        [0;-T*q2/m*(zopt_7ex(2,i)>alpha)]+...
        [0;-g*T*((theta1*(zopt_7ex(1,i)<w)+theta2*(zopt_7ex(1,i)>=w&&zopt_7ex(1,i)<2*
    w)))];
end

%repeat 2.8
[t_28, zcon_8ex] = ode45(@(t, y) conmodel(t, y, interp1(tspan, uopt_7ex, t)), tspan,
    z0);

%repeat 2.9
zopt2_9ex=zeros(2,N+1);
zopt2_9ex(:,1)=z01;
uopt2_9ex=zeros(1,N);
for i=1:N
    vref_9ex=vref1(i+1:i+Np);
    if zopt2_9ex(1,i)<w
    uopt2_9ex(i)= interp2(Vm,Vrefm,ulookup_75.u1,zopt2_9ex(2,i),vref_9ex(1),'linear')
    ;
    elseif zopt2_9ex(1,i)<2*w
     uopt2_9ex(i)= interp2(Vm,Vrefm,ulookup_75.u2,zopt2_9ex(2,i),vref_9ex(1),'linear'
    );
    else
        uopt2_9ex(i)= interp2(Vm,Vrefm,ulookup_75.u3,zopt2_9ex(2,i),vref_9ex(1),'
    linear');
    end
    zopt2_9ex(:,i+1)= [1,T;...
        0,...
        1-T/m*((p1*(zopt2_9ex(2,i)<alpha))+p2*(zopt2_9ex(2,i)>=alpha))]*zopt2_9ex(:,i
    )+...
        [0;T*b/(m*(1+gamma*((zopt2_9ex(2,i)<vg)+(zopt2_9ex(2,i)>=vg)*2)))]*uopt2_9ex(
    i)+...
        [0;-T*q2/m*(zopt2_9ex(2,i)>alpha)]+...
        [0;-g*T*((theta1*(zopt2_9ex(1,i)<w)+theta2*(zopt2_9ex(1,i)>=w&&zopt2_9ex(1,i)
    <2*w)))];
end

%% compare computation time
%% 2.2 off-line
zopt22=zeros(2,N+1);
zopt22(:,1)=z01;
uopt22=zeros(1,N);
tic
for i=1:N
    vref_9ex=vref1(i+1:i+Np);
    if zopt22(1,i)<w
    uopt22(i)= interp2(Vm,Vrefm,lookup4.u1,zopt22(2,i),vref_9ex(1),'linear');
    elseif zopt22(1,i)<2*w
     uopt22(i)= interp2(Vm,Vrefm,lookup4.u2,zopt22(2,i),vref_9ex(1),'linear');
    else
        uopt22(i)= interp2(Vm,Vrefm,lookup4.u3,zopt22(2,i),vref_9ex(1),'linear');
    end
    zopt22(:,i+1)= [1,T;...
        0,...
        1-T/m*((p1*(zopt22(2,i)<alpha))+p2*(zopt22(2,i)>=alpha))]*zopt22(:,i)+...
        [0;T*b/(m*(1+gamma*((zopt22(2,i)<vg)+(zopt22(2,i)>=vg)*2)))]*uopt22(i)+...
        [0;-T*q2/m*(zopt22(2,i)>alpha)]+...
```

```
        [0;-g*T*((theta1*(zopt22(1,i)<w)+theta2*(zopt22(1,i)>=w&&zopt22(1,i)<2*w)))];
end
time_2off=toc;

%% 2.2 on-line
zopt222=zeros(2,N+1);
zopt222(:,1)=z01;
uopt222=zeros(1,N);
tic
for i=1:N
    vref_9=vref1(i+1:i+Np);
    Fe21_9=[vref_9;-vref_9];
    Fe2_9=[Fe21_9;zeros(size(Fe12,1),1);F2];
    c=S;
    a=Fe1;
    bineq=Fe2_9+Fe3*zopt222(:,i);
    lb=[];
    ub=[];
    ctype=repmat('U', 1, size(Fe1,1));
    vartype = [repmat('C', 1, Nc*dim.nu), repmat('B', 1, Nc*dim.ndel),repmat('C',1,Nc
   *dim.nn+Np+1)];
    sense=1;
    [xmin, fmin, status, extra] = glpk (c, a, bineq, lb, ub, ctype, vartype, sense);
    uopt222(i)=xmin(1);
    zopt222(:,i+1)=A*zopt222(:,i)+B1*xmin(1:dim.nu)+...
        B2*xmin(Nc*dim.nu+1:Nc*dim.nu+dim.ndel)+...
        B3*xmin(Nc*(dim.nu+dim.ndel)+1:Nc*(dim.nu+dim.ndel)+dim.nn);
end
time_2on=toc;

%% Functions

function [dydt,a] = vdp1(t,y,qq,alpha,beta,v_max,c,m)
   a1=beta/alpha;
   a2=(c*v_max^2-beta)/(v_max-alpha);
   b2=(a1-a2)*alpha;
   P=(a1*y(2)).*(y(2)<alpha)+(a2*y(2)+b2).*(y(2)>=alpha);
   u=sin(t);
   dydt = [y(2); qq*u-P/m];
end

function [dydt,a] = vdp2(t,y,qq,m,c)
   u=sin(t);
   dydt = [y(2); qq*u-c/m*y(2).^2];
end


function y=Mat1(n,E1,E2,A,B)
if n==1
    y=E2;
else
    y=E1*(A^(n-1))*B;
end
end

function y=Mat2(n,A,B,I)
```

```
if n==1
    y=B;
else
    y=A^(n-2)*(A-I)*B;
end
end

function y=getH1(Np)
y=zeros(Np);
for i=1:Np
    for j=1:Np
        if i==j
            y(i,j)=1;
        elseif i-j==1
            y(i,j)=-2;
        elseif i-j==2
            y(i,j)=1;
        end
    end
end
end

function dydt = conmodel(t, y, u)
    if y(2)<18 && y(1)<50
    dydt = [y(2);3600/(700*(1+1.8))*u-9.8*0.38-0.5/700*y(2)^2];
    elseif y(2)>=18 && y(1)<50
    dydt = [y(2);3600/(700*(1+3.6))*u-9.8*0.38-0.5/700*y(2)^2];
    elseif y(2)<18 && y(1)>=50 && y(1)<100
         dydt = [y(2);3600/(700*(1+1.8))*u-9.8*0.197-0.5/700*y(2)^2];
    elseif y(2)>=18 && y(1)>=50 && y(1)<100
         dydt = [y(2);3600/(700*(1+3.6))*u-9.8*0.197-0.5/700*y(2)^2];
    elseif y(2)<18 && y(1)>=100
        dydt = [y(2);3600/(700*(1+1.8))*u-0.5/700*y(2)^2];
    else
        dydt = [y(2);3600/(700*(1+3.6))*u-0.5/700*y(2)^2];
    end
end
```

B. The parameters of vehicle

Parameters	Values	Units
m	700	kg
g	9.8	m/s^2
c	0.5	kg/m
b	3600	N
u_{max}	1.6	-
u_{min}	-1.4	-
$a_{comf,max}$	2.5	m/s^2
γ	1.8	-
v_g	18	m/s
h	10	m
w	50	m

TABLE II
VEHICLE PARAMETERS.

REFERENCES

[1] Optimization for Systems and Control (SC42056). (2020, October 17). Retrieved from https://www.dcsc.tudelft.nl/ bdeschutter/osc

[2] Second partial derivative test. Khan Academy. (2023, May 12). Retrieved from https://www.khanacademy.org/math/multivariable-calculus/applications-of-multivariable-derivatives/optimizing-multivariable-functions/a/second-partial-derivative-test.

[3] LinearInequalityConstraintExample. (2022, October 18). Retrieved from https://www.mathworks.com/help/optim/ug/fmincon.html

[4] Choosing the Algorithm - MATLAB Simulink. (2022, October 18). Retrieved from https://www.mathworks.com/help/optim/ug/choosing-the-algorithm.htmlbsbwxm7