Automotive Control Systems (MEEM/EE5812)

Intelligent Cruise Control

Instructor

Dr. Bo Chen

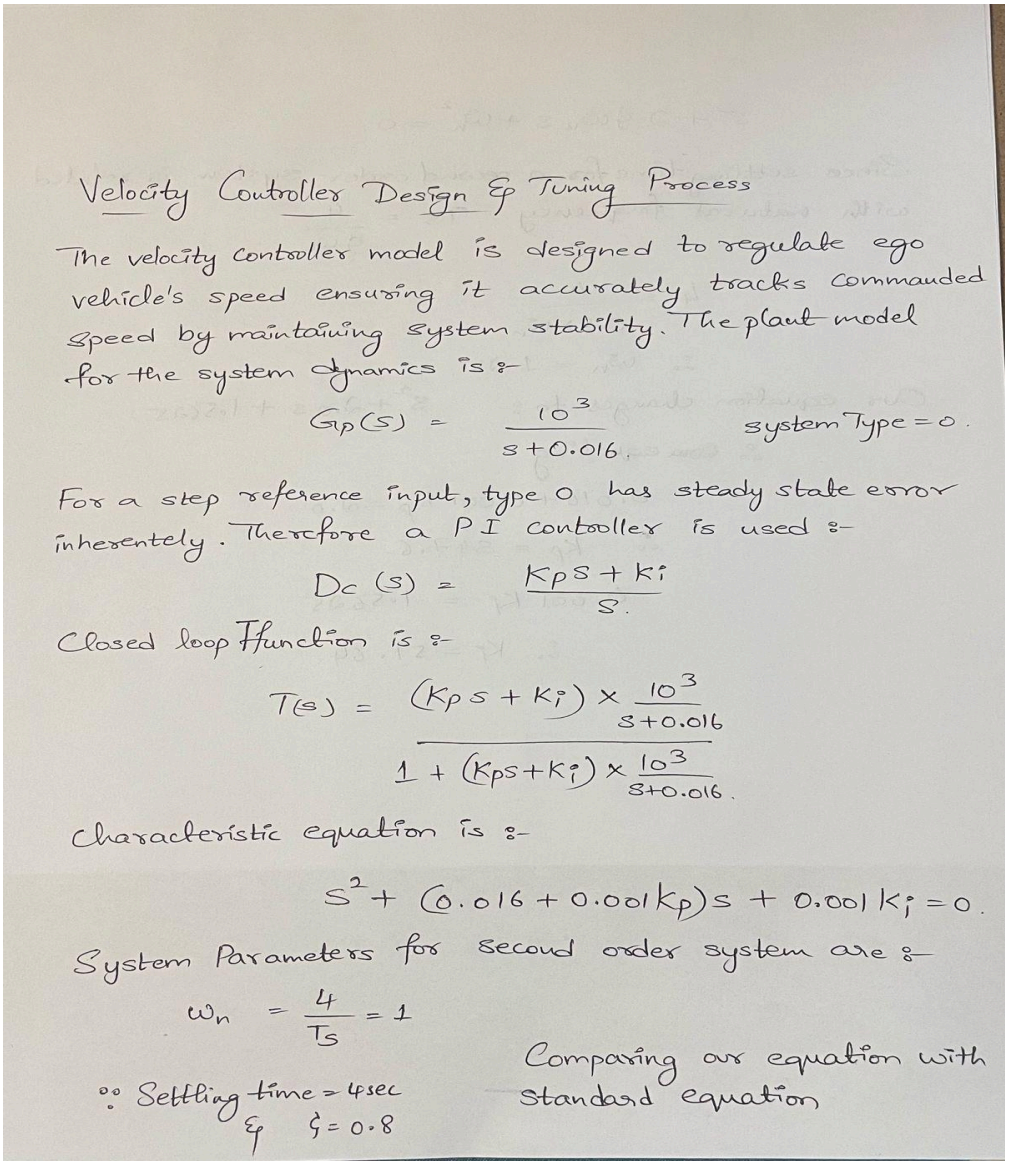
Submitted By

Rahul Srikanth Kandi

VELOCITY CONTROLLER:

The velocity controller was designed to regulate the ego vehicle's speed to match the desired speed while ignoring headway measurements. The PI (Proportional-Integral) control approach was implemented to achieve the required performance criteria.

Design Process & Controller Gains:



Velocity Controller Design & Tuning Process

The velocity controller model is designed to regulate ego vehicle's speed ensuring it accurately tracks commanded speed by maintaining system stability. The plant model for the system dynamics is :-

$$G_p(s) = \frac{10^3}{s+0.016}, \qquad \text{system Type} = 0.$$

For a step reference input, type 0 has steady state error inherently. Therefore a PI controller is used :-

$$D_c(s) = \frac{K_p s + K_i}{s}$$

Closed loop Tfunction is :-

$$T(s) = \frac{(K_p s + K_i) \times \frac{10^3}{s+0.016}}{1 + (K_p s + K_i) \times \frac{10^3}{s+0.016}}$$

Characteristic equation is :-

$$s^2 + (0.016 + 0.001 K_p)s + 0.001 K_i = 0.$$

System Parameters for second order system are :-

$$\omega_n = \frac{4}{T_s} = 1$$

$\because$ Settling time = 4sec

$\therefore \zeta = 0.8$

Comparing our equation with standard equation

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0.$$

Since settling time for a second order system is related with natural frequency $T_s = \frac{4}{\zeta\omega_n}$

$$4 = \frac{4}{0.8 \times \omega_n}$$

$$\therefore \omega_n = 1.25.$$

Our equation changes to :- $s^2 + 2.0s + 1.5625$.

∴ equating

$$0.016 + 0.001 K_p = 2.0$$

$$\therefore K_p = 347.6.$$

$$0.001 K_I = 1.5625$$

$$\therefore K_I = 51.64$$

PI controller for a Velocity cruise controller.

$$u_{c\_cruise} = -K_p V_c(s) + \frac{K_I}{s} e_{cruise}(s)$$

Initial value of Integrator in velocity controller

$$u_{c\,cruise}(t_0) = -K_p V_c(t_0) + K_I I_{10} = u_{cheadway}.$$

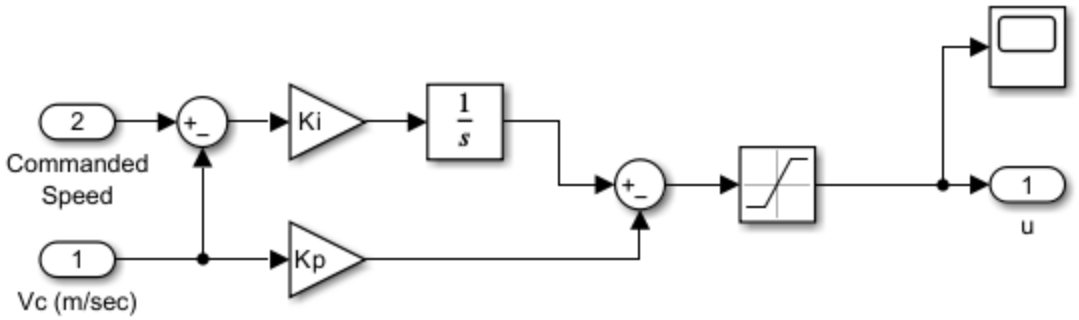
$$I_{10} = \frac{u_{cheadway} + K_p V_c(t_0)}{K_I}$$

1. The control input (traction force) was determined based on the error between commanded and actual velocity.

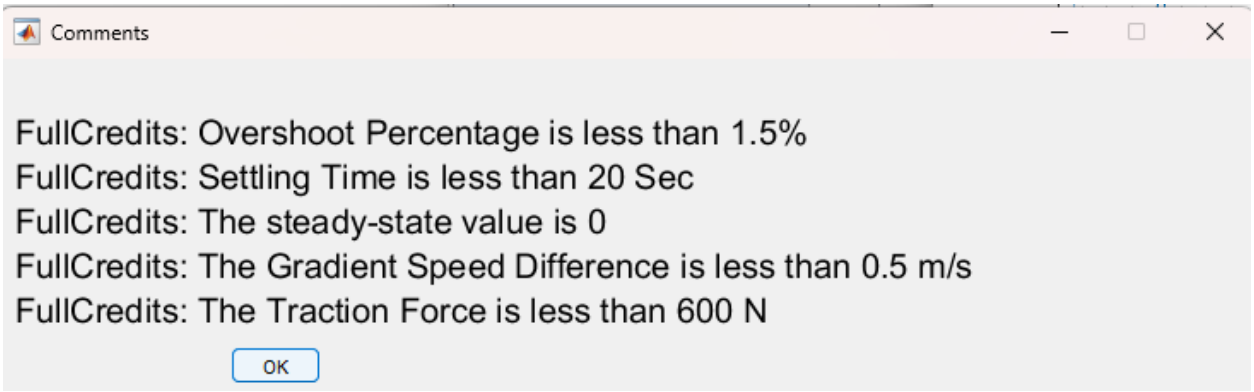
2. The gains for the PI controller were tuned to satisfy the given performance criteria:

Kp = 370;

Ki=47;



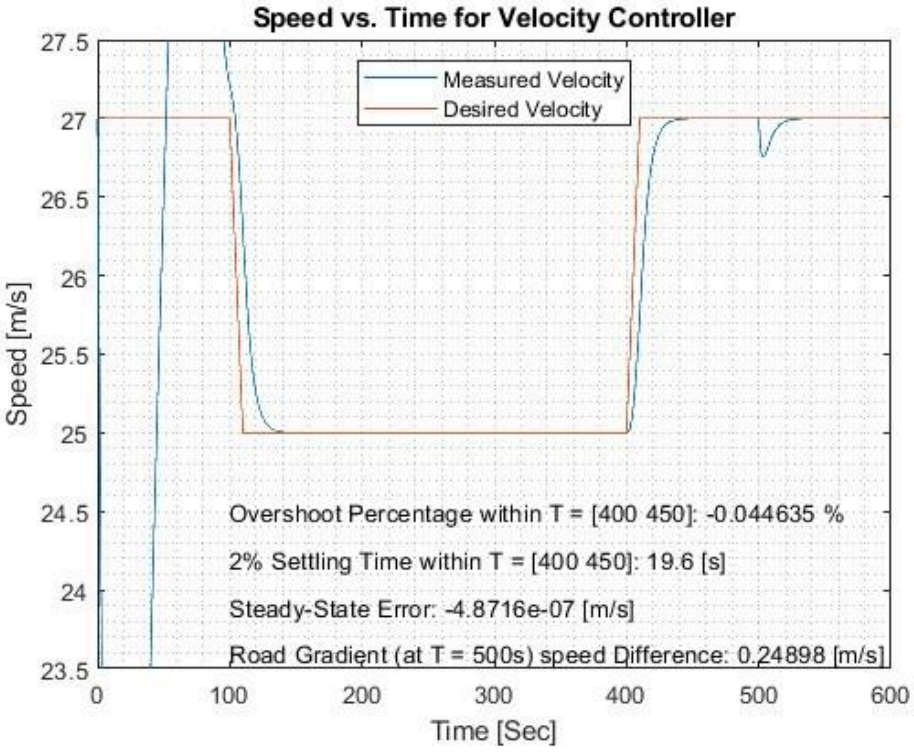
Above is the model for the velocity controller.
The controller successfully met all the specifications and earned full credits in all performance metrics.



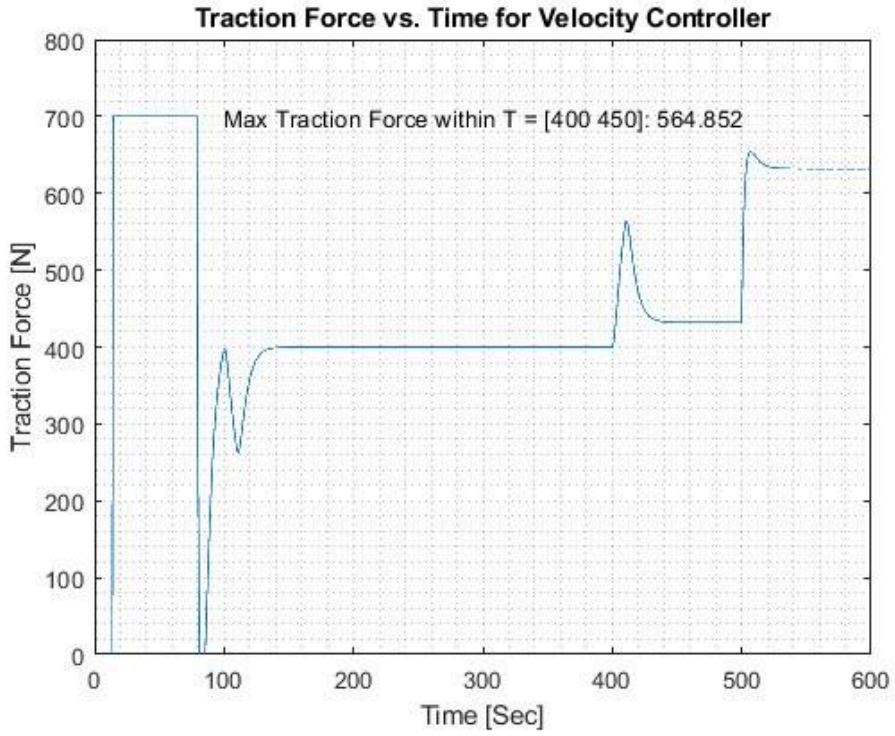
Performance Validation & Simulation Results:

The simulation was run in Velocity Control Mode, and the following requirements were verified:

1. Overshoot in velocity between 400s and 450s was less than 1.5%.
2. Settling time (2%) was less than 20 seconds.
3. Steady-state error was approximately zero ($\leq 10^{-5}$ m/s).
4. Speed difference due to gradient change at 500s was less than 0.5 m/s.
5. Tractive force remained below 600N between 400s and 450s.



Velocity vs Time Plot



Traction_force vs Time Plot

Observations from Simulation:

- The measured velocity closely tracked the commanded velocity.
- The traction force adjusted dynamically to maintain speed, staying within the required limits.
- The controller responded effectively to disturbances such as road gradients and changes in commanded speed.

HEADWAY CONTROLLER:

The headway control system was formulated using a state-space representation, ensuring that the ego vehicle dynamically adjusted its speed based on the headway error. The process involved the following key steps:

State-Space Model Formulation:

1. The system dynamics were derived considering headway error, vehicle velocity, and external disturbances.
2. A state-space model was established to define the relationships between state variables and control input.

Integration for Disturbance Rejection:

1. To ensure a steady-state error of zero in response to a step disturbance input, an additional integrator state was incorporated into the model.
2. This transformation allowed the system to handle sudden changes in lead vehicle velocity while maintaining headway.

LQR Design and Gain Calculation:

1. The Q and R matrices were selected to minimize deviations in headway while avoiding excessive control effort.
2. The A and B matrices were defined based on the system dynamics, ensuring stability and robust tracking performance.

Tuning the Gains:

1. Initial gains were determined based on system response characteristics.
2. The final LQR gains were adjusted to eliminate steady-state error while ensuring smooth transition between velocity and headway controllers.

The system was implemented according to plant function $G_p(s) = \frac{10^{-3}}{s+0.016}$.

From first order system $= \frac{K_c}{\tau_c s + 1}$

Therefore; $K_c = 0.063$ & $\tau_c = 63$. [bounded]

Adding two integrators to make a Type 2 system & steady state error = 0.

$\therefore u_c\_headway(s) = -K_1 R(s) - K_2 V_c(s) - K_3 X_3(s) - K_4 X_4(s)$

From the states, we define :-

$X_1(t) = d(t) = X_l - X_c$

$X_2(t) = V_c(t)$

$X_3(t) = \int (r(t) - d(t))\, dt$

$X_4(s) = \frac{X_3(s)}{s}$

$w_1(t) = V_l(t)$

$\dot{x}_1(t) = \dot{d}(t) = \dot{x}_l - \dot{x}_c = V_l(t) - V_c(t) = -x_2(t) + w_1(t)$

$\dot{x}_2(t) = \frac{-1}{\tau_c} x_2(t) + \frac{K_c}{\tau_c} u_c(t)$

$\dot{x}_3(t) = r(t) - x_1(t)$

Adding additional integral state to eliminate steady state errors with ramp disturbances.

$\dot{x}_4(t) = x_3$

State matrix is as follows :-

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \\ \dot{x}_4(t) \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & \frac{-1}{\tau_c} & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{A} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{K_c}{\tau_c} \\ 0 \\ 0 \end{bmatrix} u_c(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} r(t) + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} w_1$$

$e_{headway}(s) = R_c(s) - R(s)$

$R_c(s)$: 3 second distance threshold.

$R(s)$: Headway.

Initial value of Integrators in headway Controller.

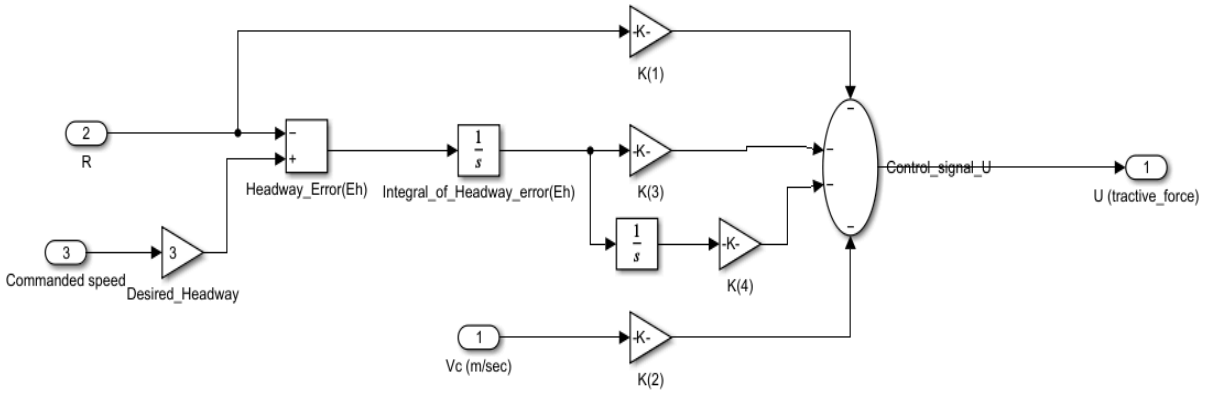
$u_{c\,headway}(t_0) = -K_1 R(t_0) - K_2 V_c(t_0) - K_3 I_{20} - K_4 I_{30} = u_{c\,cruise}$

Set $I_{20} = 0$.

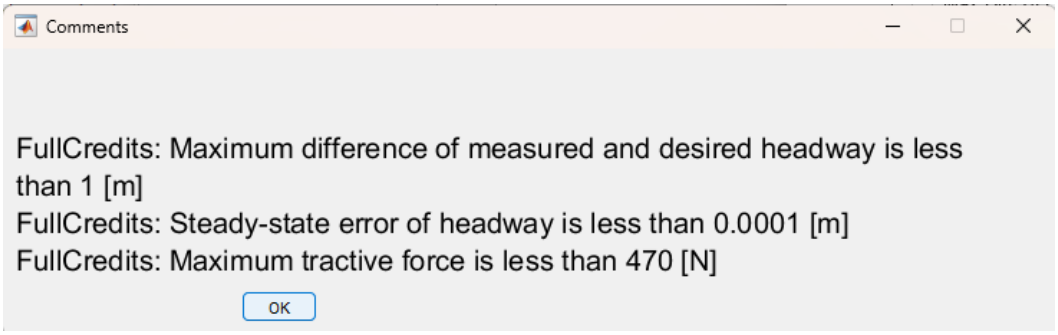
$$I_{30} = \frac{-u_{c\,cruise} - K_1 R(t_0) - K_2 V_c(t_0)}{K_4}$$

```
%% Velocity Controller
Kp= 370;
Ki= 47;

% Step size
Step_size = 0.1;
%% Headway Controller
tauc = 63;
Kc= 0.063;
A = [0 -1 0 0;
     0 -1/tauc 0 0;
     -1 0 0 0;
     0 0 1 0];

B = [0;Kc/tauc;0;0];

Q = [8 0  0  0;
     0  0  0  0;
     0  0  4  0;
     0  0  0  35];

R = 0.62;
```

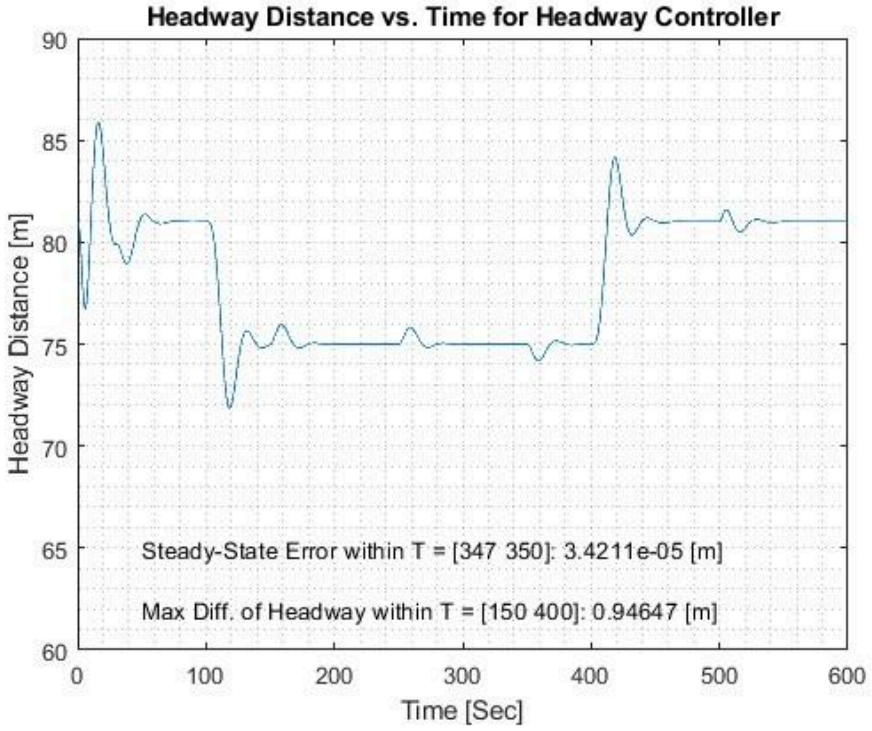
This code is used for the Headway controller after tuning some of the Q and R values.



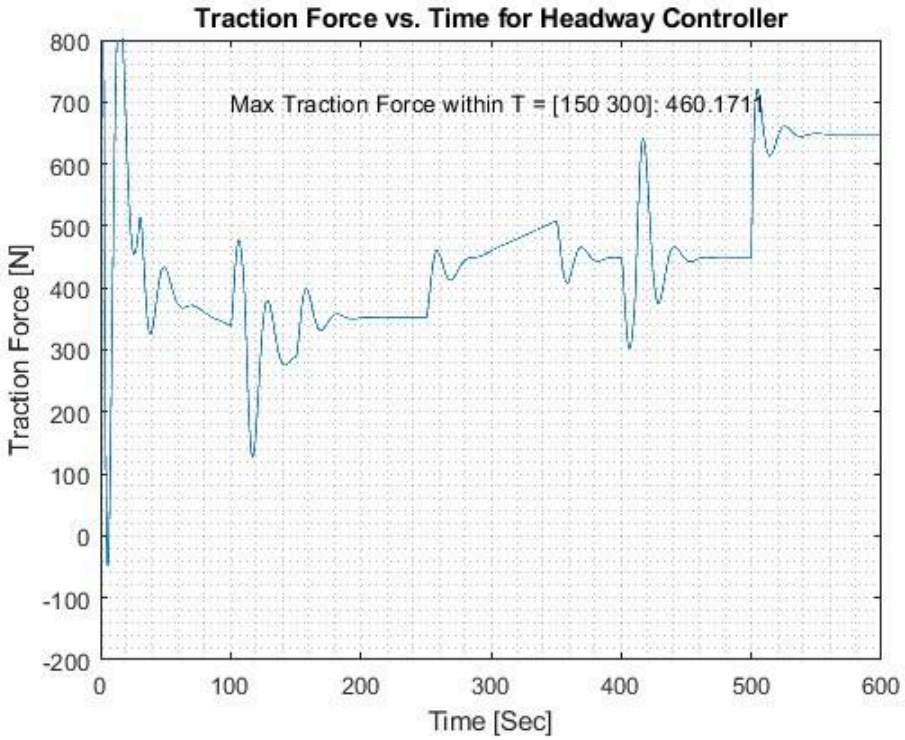
Above is the model for the headway controller.



The headway controller met all performance criteria and earned full credits in all parameters.



Headway_distance vs Time Plot



Traction_force vs Time Plot

Performance Validation & Simulation Results:

The simulation was run in Headway Control Mode, and the following requirements were verified:

1. Maximum absolute headway error between 150s and 400s was less than 1 meter.
2. Steady-state error in headway between 347s and 350s was zero ($\leq 10^{-4}$ m).
3. Maximum tractive force between 150s and 300s was less than 470N

Observations from Simulation:

- The ego vehicle consistently maintained a safe following distance from the lead vehicle.
- The tractive force stayed within acceptable limits, remaining below 470N for smooth operation.
- The controller successfully transitioned from velocity control to headway control when the lead vehicle approached.
- Both the velocity and headway controllers were designed, implemented, and tested effectively.
- The results validated that all performance criteria were met, ensuring the intelligent cruise control system operated safely and efficiently.

The next step is integrating these controllers into the Intelligent Cruise Controller, ensuring seamless switching between the two modes.

INTELLIGENT CRUISE CONTROLLER:

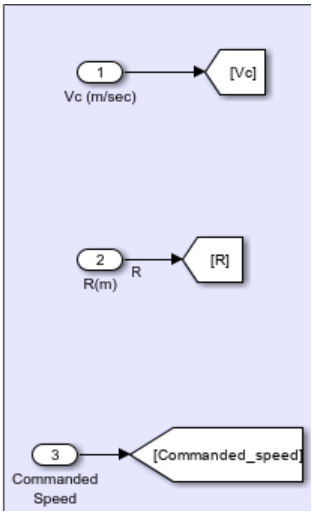
The Intelligent Cruise Controller (ICC) integrates both velocity control and headway control, allowing the ego vehicle to autonomously adjust its speed based on road conditions, traffic, and the lead vehicle's movement. The system seamlessly switches between the two modes, prioritizing velocity control when there is no obstruction ahead and headway control when a lead vehicle is detected.

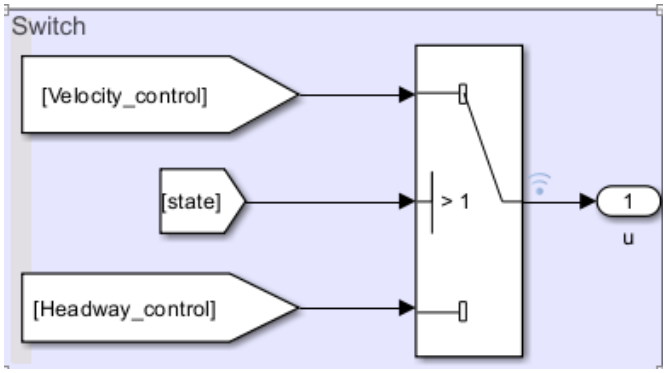
Design Process & Controller Structure:
The ICC consists of:

1. Velocity Controller – Keeps the vehicle moving at a steady speed when there are no obstacles ahead.
2. Headway Controller – Maintains a safe distance from the lead vehicle by adjusting speed accordingly.
3. Mode Switching Logic – Determines the right moment to switch between velocity and headway control based on traffic conditions.

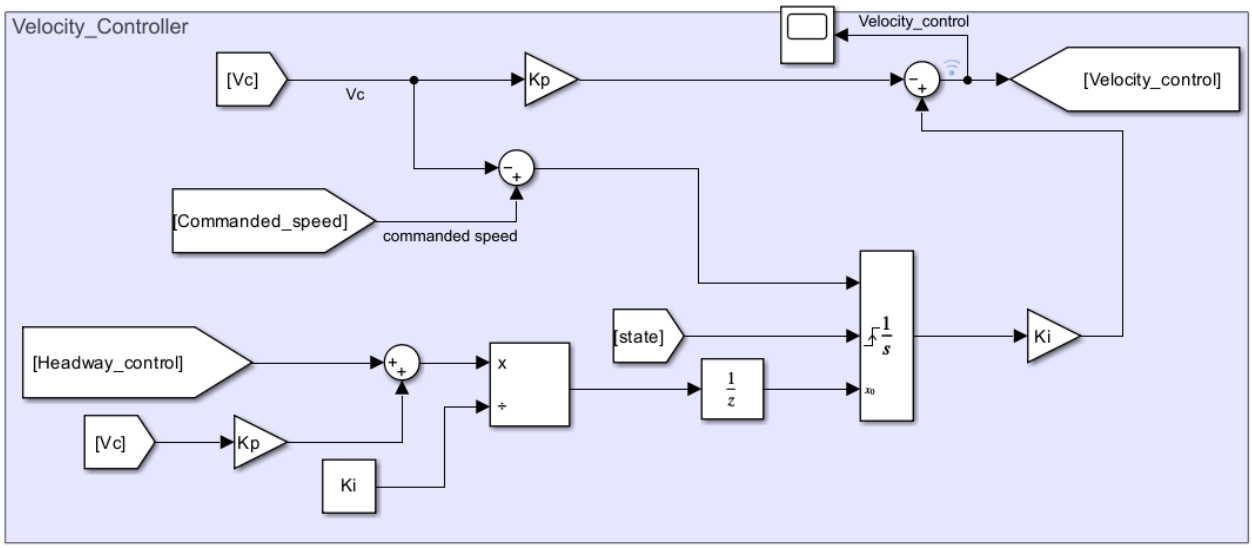
The switching mechanism monitors the relative distance (headway) and lead vehicle speed to decide:

- If the lead vehicle is far → Use Velocity Control (maintains desired speed).
- If the lead vehicle is close → Use Headway Control (adjusts speed to maintain a safe gap).

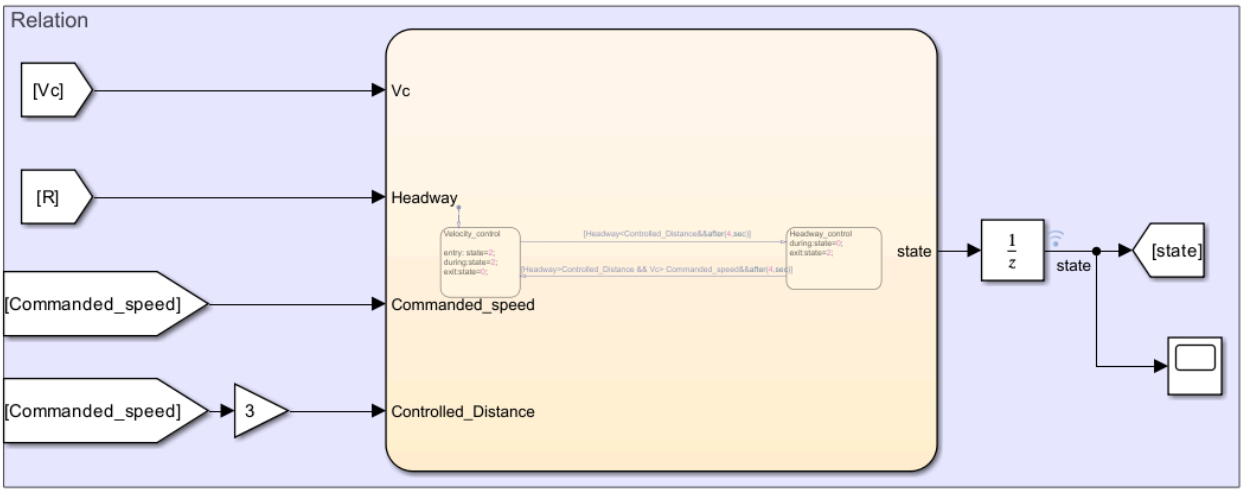




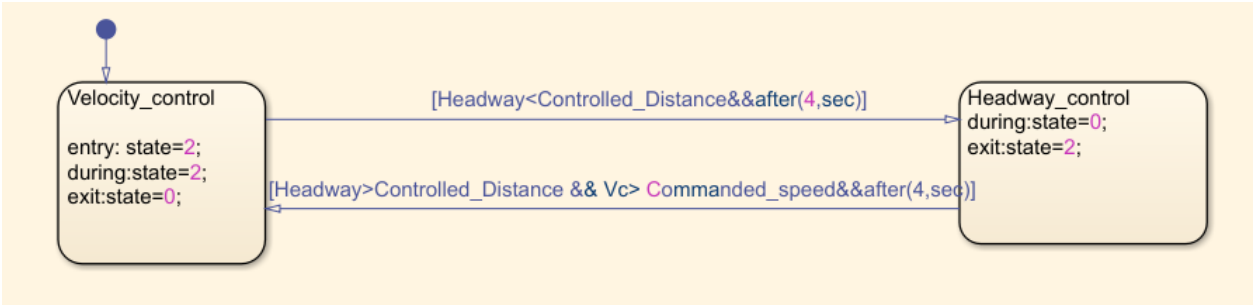
Above are the initial goto blocks and switch block.



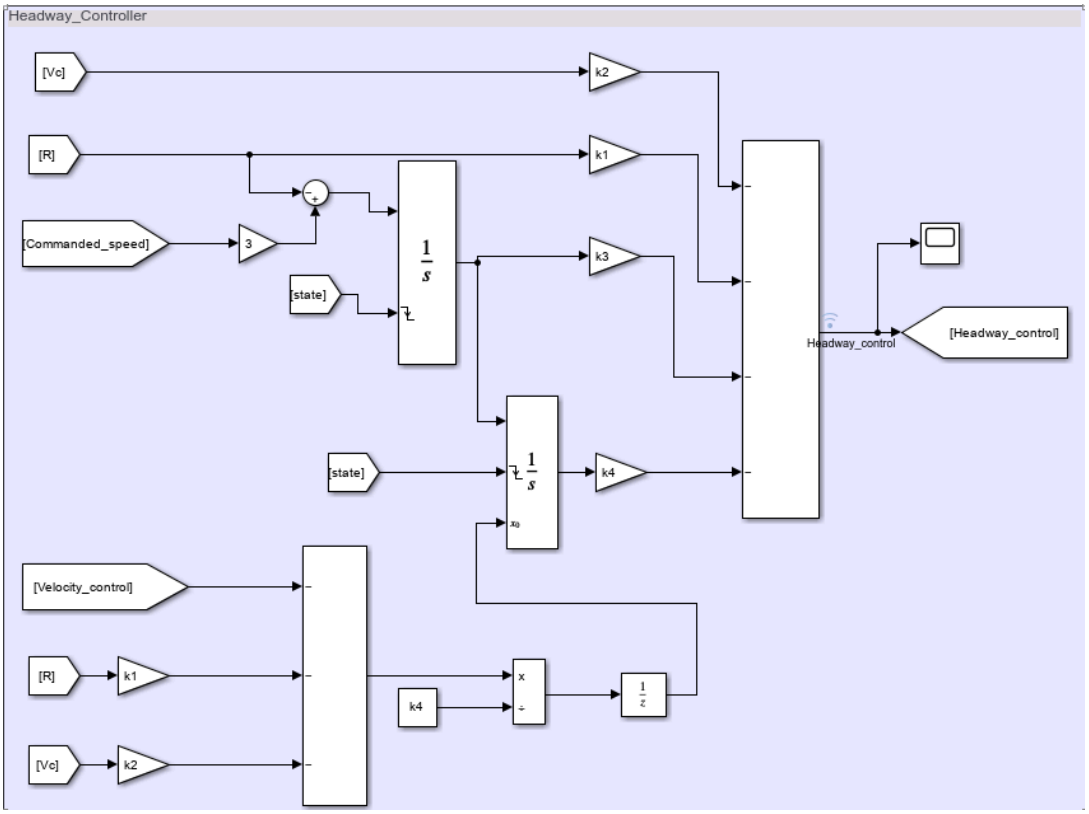
Above is the velocity controller model along with an integrator taking Input, state and initial condition. In this controller the integrator is set with the Rising and External option. Whereas in the Headway Controller they are set to Falling and Internal as well as Falling and External for two integrators.



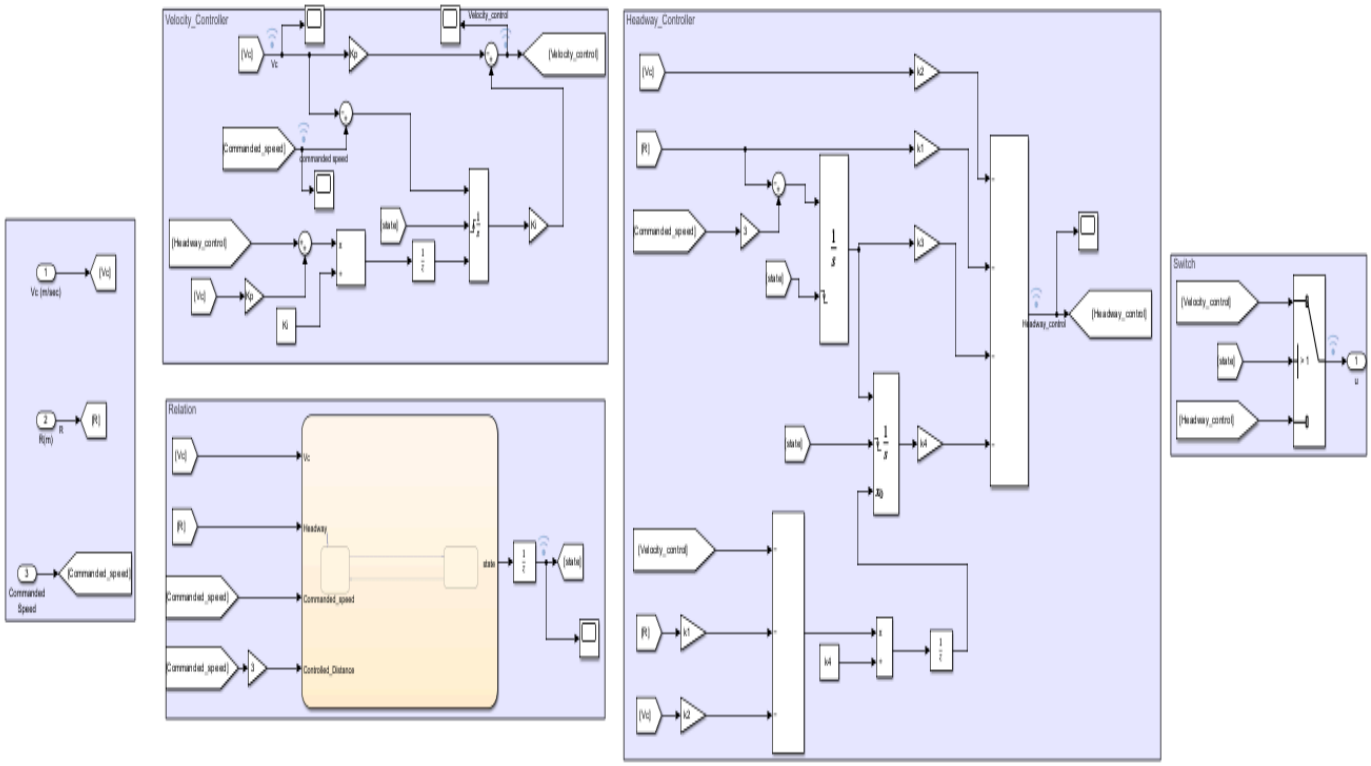
This is the block for relation in stateflow.



Detailed stateflow logic for velocity and headway controllers for switching and working simultaneously when needed.



Above is the headway controller with two integrators.

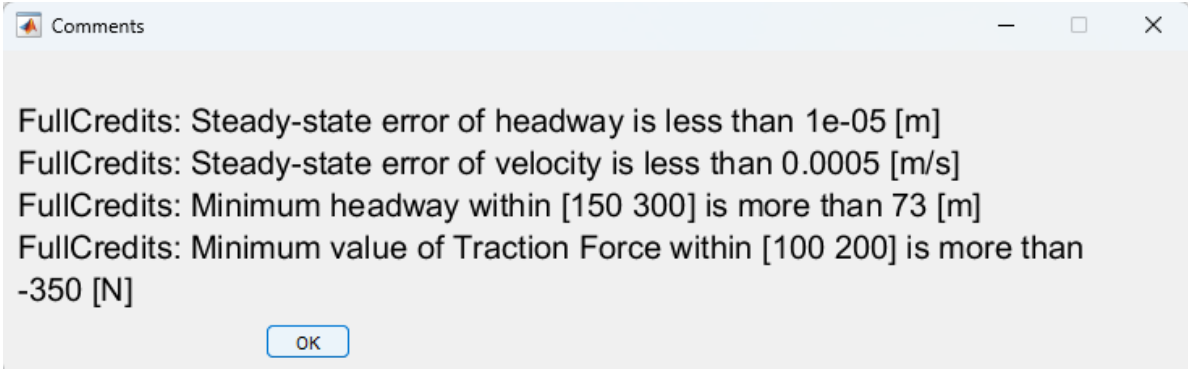


This is the full model representation of the Intelligent cruise controller.

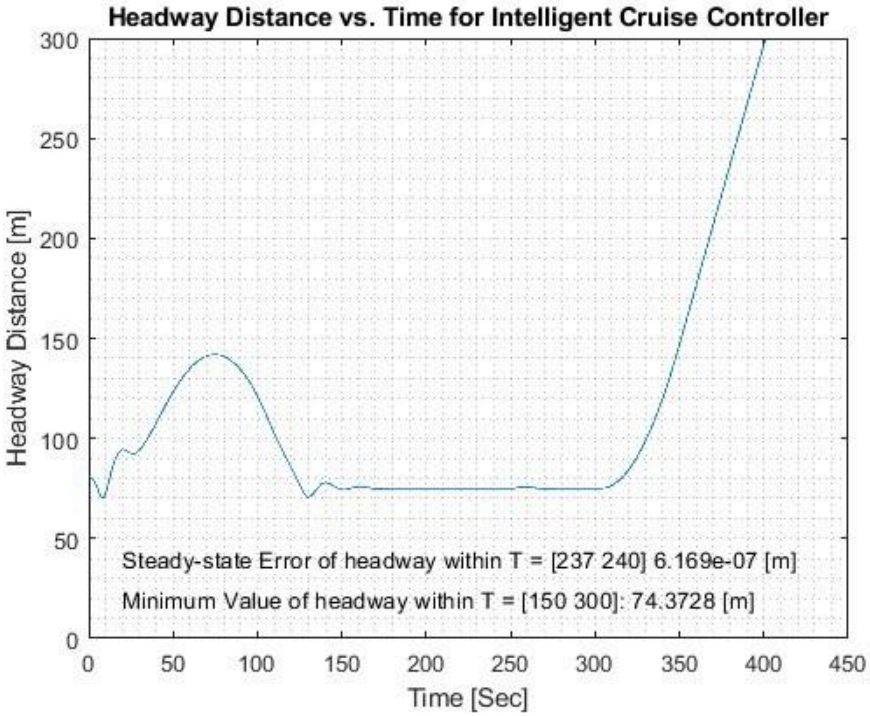
Performance Validation & Simulation Results:

The simulation was run in Intelligent Cruise Control Mode, and the following requirements were validated:

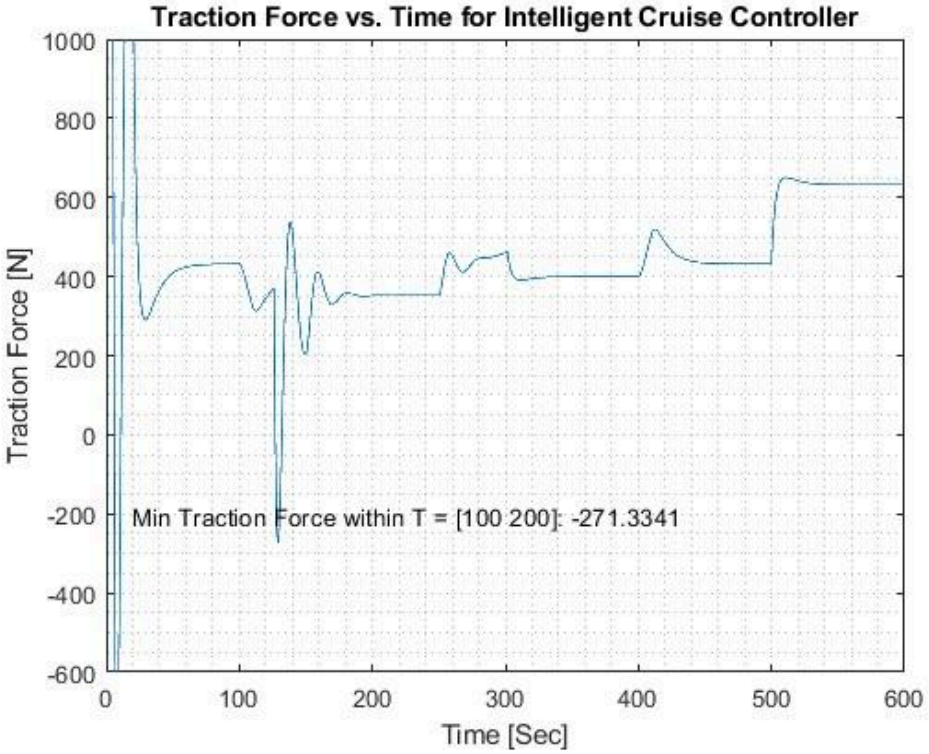
1. Steady-state error of headway between 237s and 240s was ≤ 10^{-5} m.
2. Steady-state error of velocity between 450s and 490s was ≤ 0.5×10^{-3} m/s.
3. Minimum headway between 150s and 300s was more than 73m.
4. Minimum tractive force between 100s and 200s was greater than -350N.



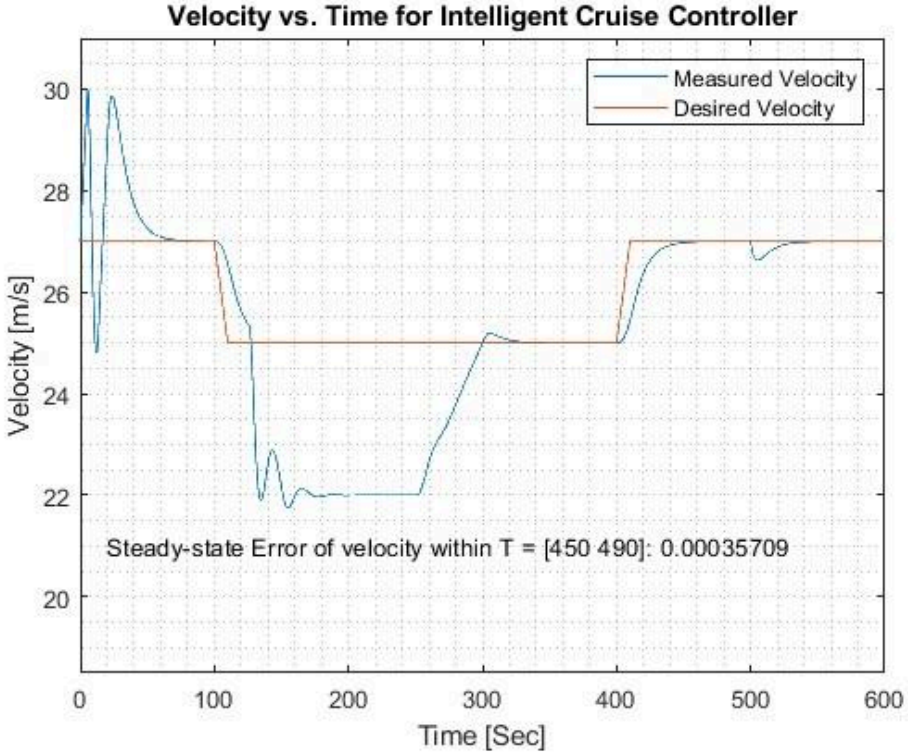
The ICC successfully met all performance requirements and received full credits in all parameters.



Headway_distance vs Time Plot



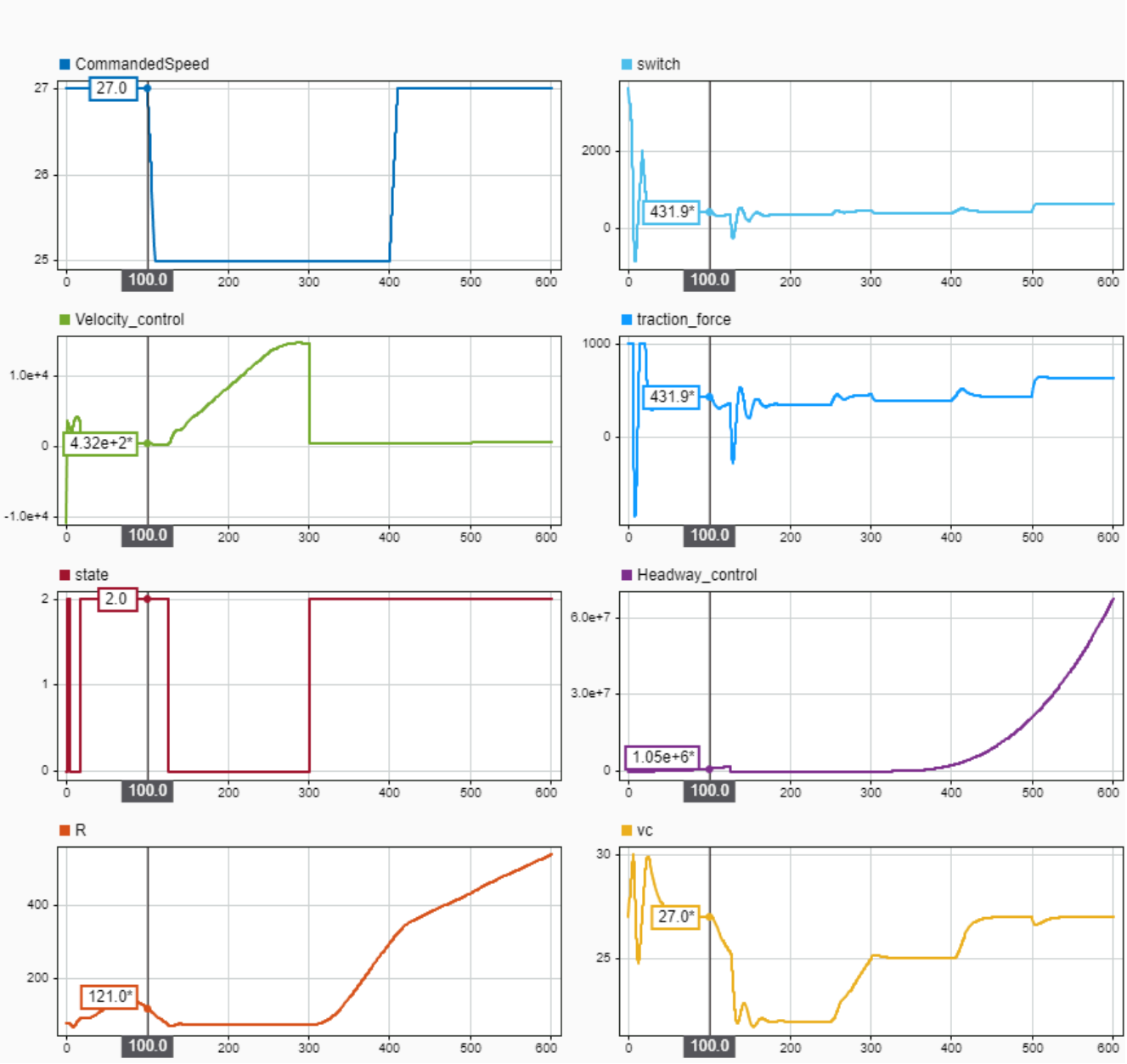
Traction_force vs Time Plot



Velocity vs Time Plot

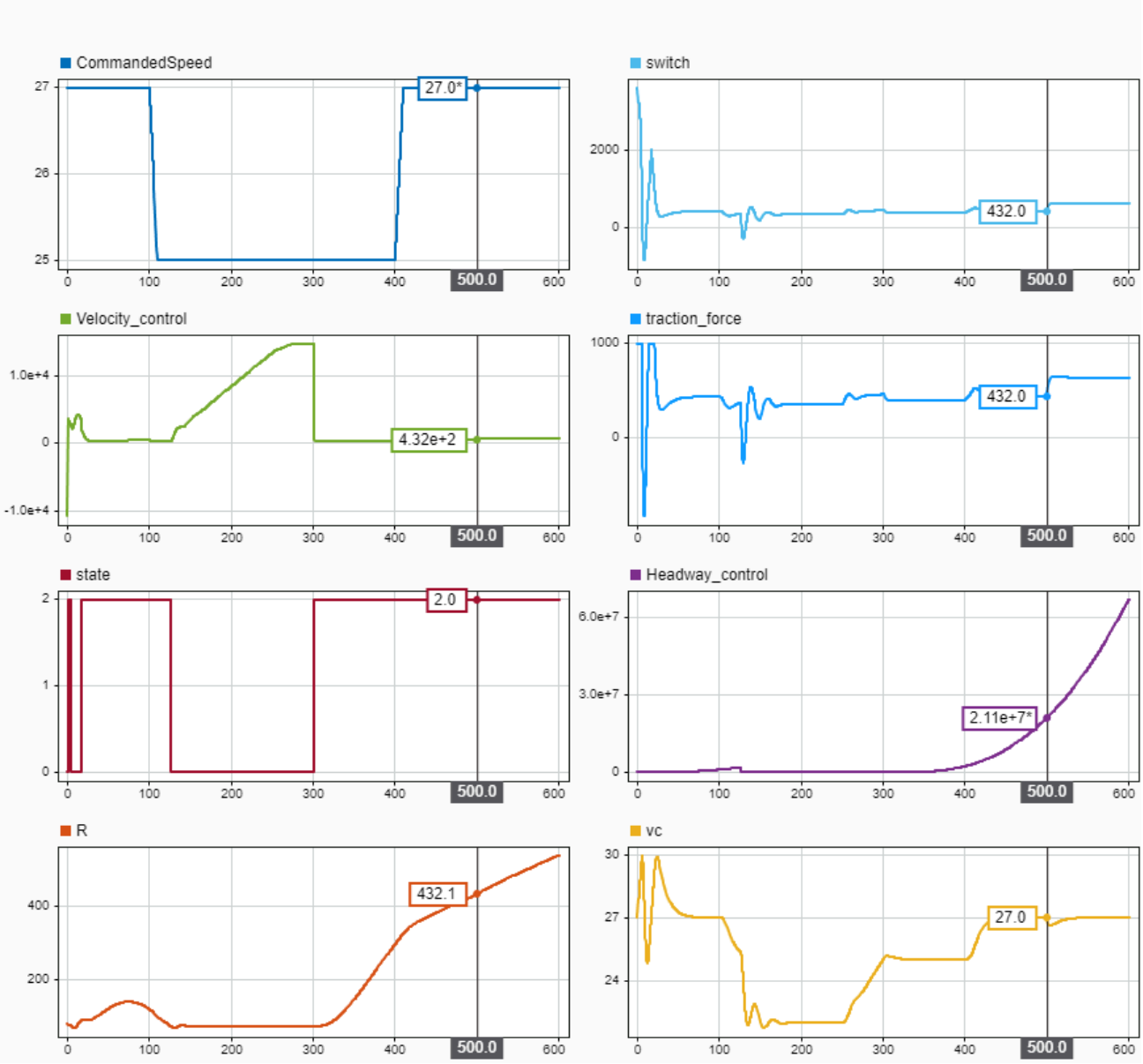
1. Velocity Command Ramp Down at 100 Seconds

- The commanded speed decreases, causing the measured velocity to follow the reduction.
- The tractive force decreases significantly, indicating that less force is required to maintain the reduced speed.



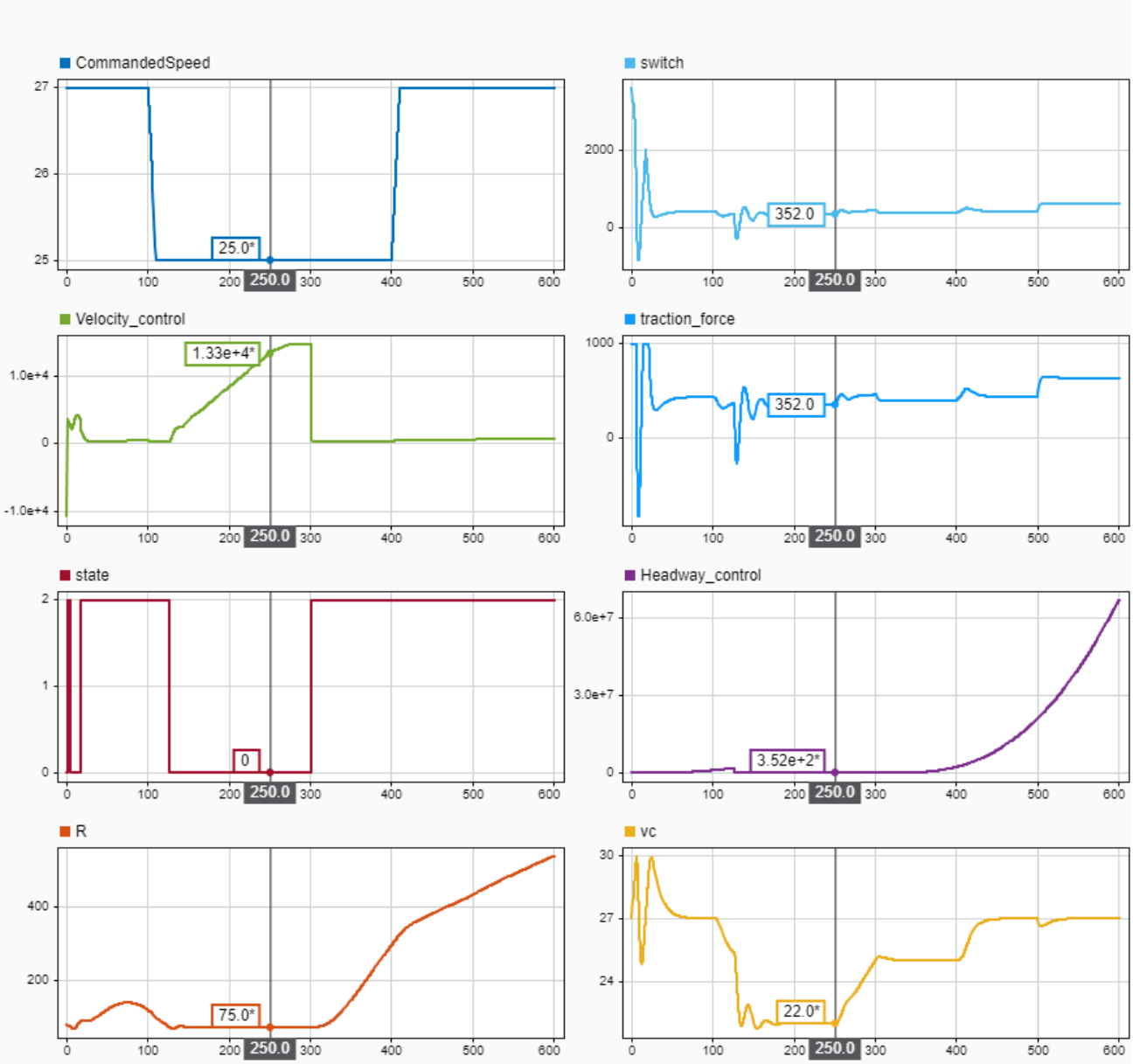
2. Switching from Velocity Control to Headway Control at 150 Seconds

- The state switches from 2 (Velocity Control) to 0 (Headway Control), and the ego vehicle starts adjusting based on the lead vehicle.
- A sharp decrease in tractive force occurs as the system prioritizes maintaining a safe following distance.



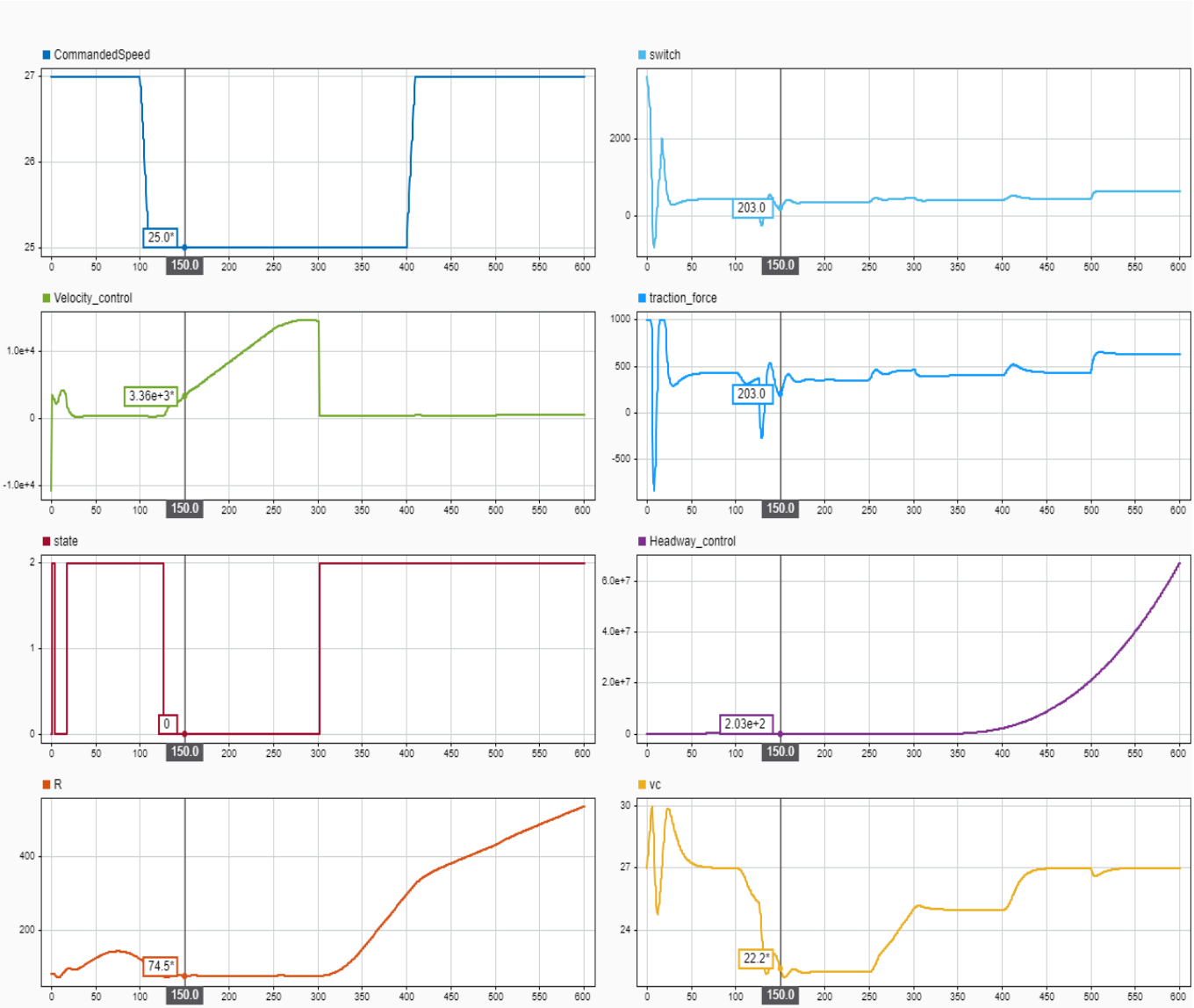
3. Lead Vehicle Accelerating at 250 Seconds

- The headway distance increases, showing that the lead vehicle is moving away.
- The tractive force increases, and the ego vehicle's velocity starts rising to match the lead vehicle’s acceleration.



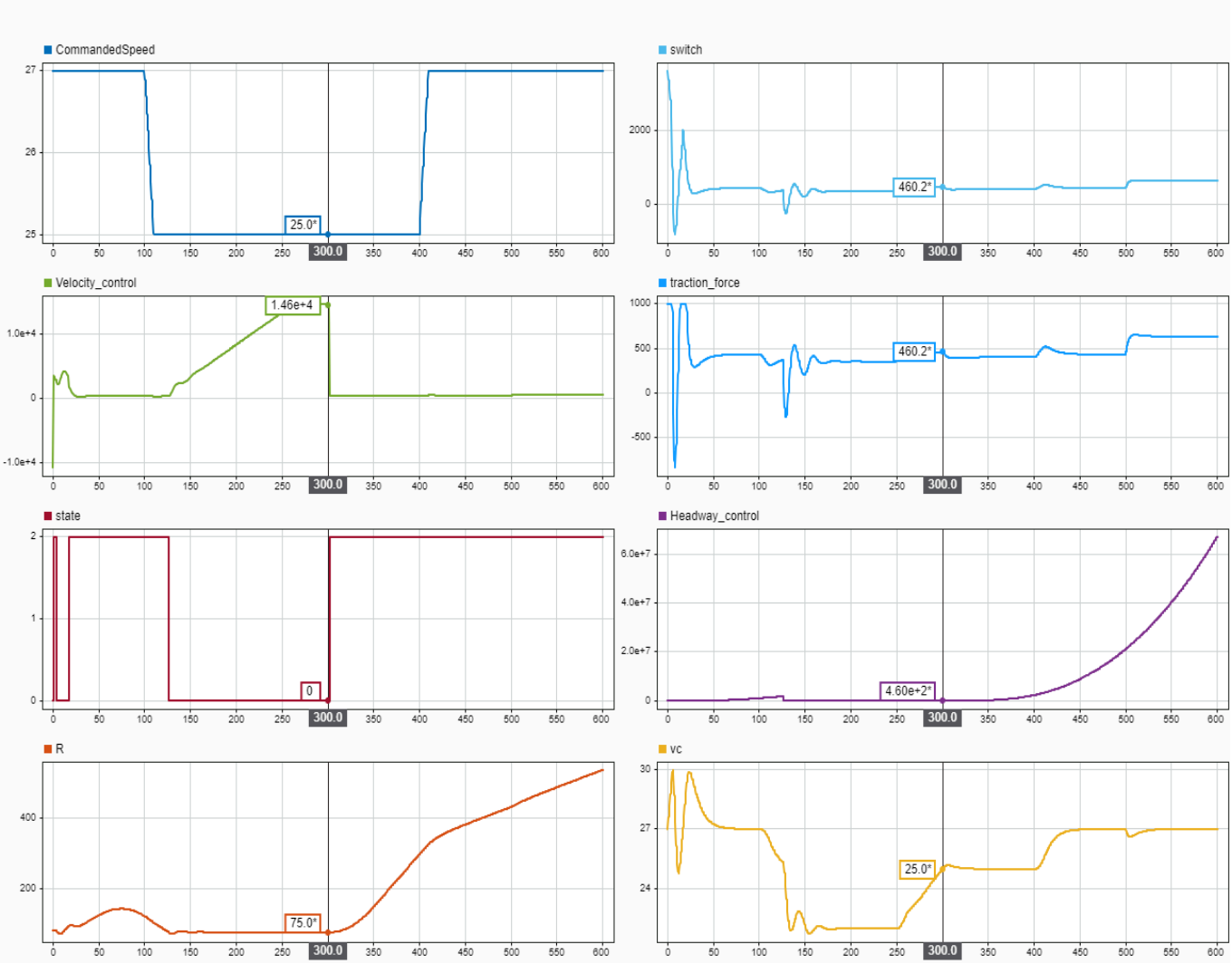
4. Switching from Headway Control to Velocity Control at 300 Seconds

- The state transitions back to 2 (Velocity Control), prioritizing maintaining the desired speed rather than following the lead vehicle.
- The tractive force stabilizes, and the velocity remains constant at the commanded speed.



5. Vehicle Going Up a Hill at 500 Seconds

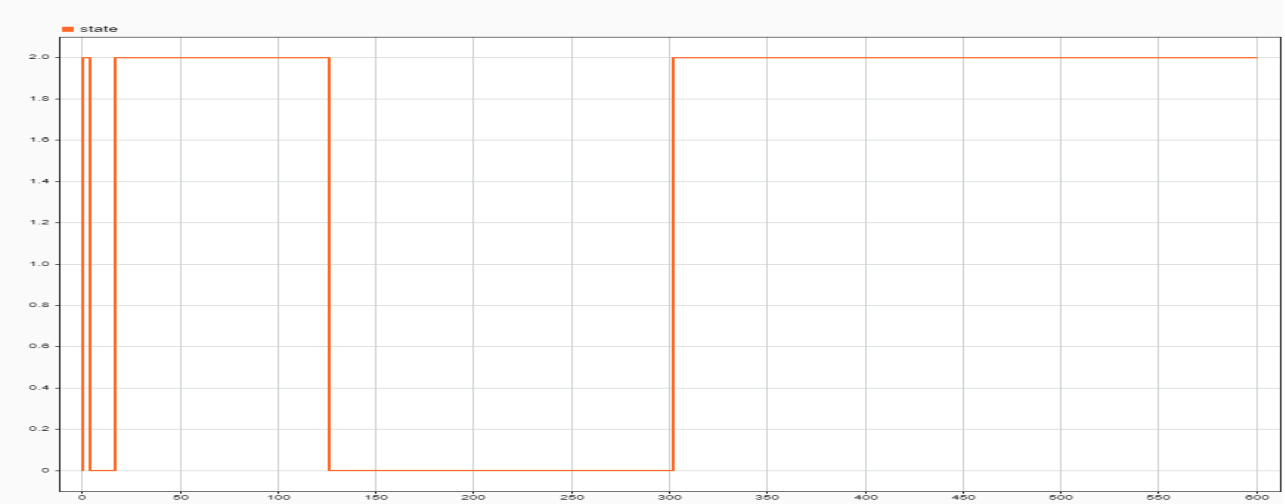
- As the vehicle encounters an incline, the tractive force increases noticeably to counteract the added resistance.
- Despite this change, the velocity remains steady, demonstrating that the controller effectively adjusts to maintain the desired speed.



```
%% Velocity Controller
Kp= 400;
Ki= 33;

% Step size
Step_size = 0.1;
%% Headway Controller
tauc = 63;
Kc= 0.063;
A = [0 -1 0 0;
     0 -1/tauc 0 0;
     -1 0 0 0;
     0 0 1 0];

B = [0;Kc/tauc;0;0];

Q = [8 0  0  0;
     0  0  0  0;
     0  0  4  0;
     0  0  0  35];

R = 0.62;
```

```
[K,~,p] = lqr(A, B, Q, R);
disp('LQR Gain:');
disp(K);
k1=K(1);
k2=K(2);
k3=K(3);
k4=K(4);
```

This is the final code used for Intelligent cruise controller model, tuned more compared to velocity cruise controller alone.



Above is the representation of the state changing from Velocity to Headway and vice-versa.

Conclusion :

The Intelligent Cruise Controller was successfully implemented and tested. The results confirmed that the controller efficiently handled speed regulation and safe following distance maintenance while dynamically switching between control modes as required.

This implementation ensures better driving safety, smoother ride comfort, and improved energy efficiency, making it a vital component for advanced driver-assistance systems (ADAS) in autonomous vehicles.