MATLAB/Simulink model for Cruise & Adaptive Cruise Control

Prepared by,

Kanwar Talha Bin Liaqat

Date: 05/06/2018

Table of Contents

Table of Figures

1. Introduction

The purpose of using Models i.e. physical, mathematical, or other logical representation of a system, is to generate a basis for simulations to develop data as a basis for decision making. Simulation, furthermore, can facilitate in understanding a system's behavior without testing the system in the real world.

In this project a scenario was developed in Simulink where a vehicle with Adaptive Cruise Control follows a Vehicle with simple cruise-controlled functionality. The follower vehicle must adjust speed according to the environment and obstacles come in its way.

The project was divided into two separate parts. 1st part was to develop a vehicle with a simplified vehicle model and cruise control functionality. In the 2nd part, another vehicle with adaptive cruise control and enhanced vehicle model was developed. These two models were then simulated together to test the working of both models.

2. Cruise Control

2.1 System Explanation

A simplified Cruise Control system of an automatic transmission vehicle is to be modelled from Newton's second law of motion as follow:

$$m\frac{dv}{dt} + bv(t) = u(t) \tag{2.1}$$

where,

- **m** indicates the mass of a car in kg
- **v** is the velocity of the car in m/s
- **b** is the friction obtained by the car in kg/s
- **u** is the force from the engine in Nm

The cruise-controlled vehicle model must fulfil the following requirements:

- **R1:** The vehicle speed should be kept at the reference specified by the driver under road conditions that do not exceed the physical limit of the system
- **R2:** A main reference speed shall synthetize the accelerator and brake pedal's actions of the driver

- **R3:** The driver can change the reference speed by means of two push buttons: “up button” and “down button”
- **R4:** Rise by 5 km/h the actual reference speed and keep it at the new reference when “up button” has been once activated
- **R5:** Decrease by 5 km/h the actual reference speed and keep it at the new reference when “down button” has been once activated
- **R6:** A change in the main reference speed shall overrule the buttons’ action

2.2 Definition of Input(s) and Output(s)

From the mathematical model and the functional requirements, the inputs and outputs were defined as follows:

Inputs:

- ***Input Speed,*** a reference speed that synthetize the accelerator and brake pedal’s actions.
- ***Up button,*** indicates an increase of 5 km/h in speed
- ***Down button,*** indicates a decrease of 5 km/h in speed

Outputs:

- ***vehicle speed,*** vehicle actual speed

Inputs and outputs of vehicle plant are as follows:

Inputs:

- ***u(t),*** the force from the engine which is synthesized by PI controller

Outputs:

- ***v(t),*** vehicle actual speed

Constants:

- ***m = 1000 kg***, vehicle mass
- ***b = 50 kg/s***, friction obtained by the car

2.3 Implementation in Simulink

The above described system was implemented in Simulink, as shown in **Fig 2.1,** with 3 significant blocks.

- A Signal Builder
- A State flow chart (Cruise Control)

- A subsystem (Vehicle Model) that uses PI controller and **equation 2.1** to produce speed.

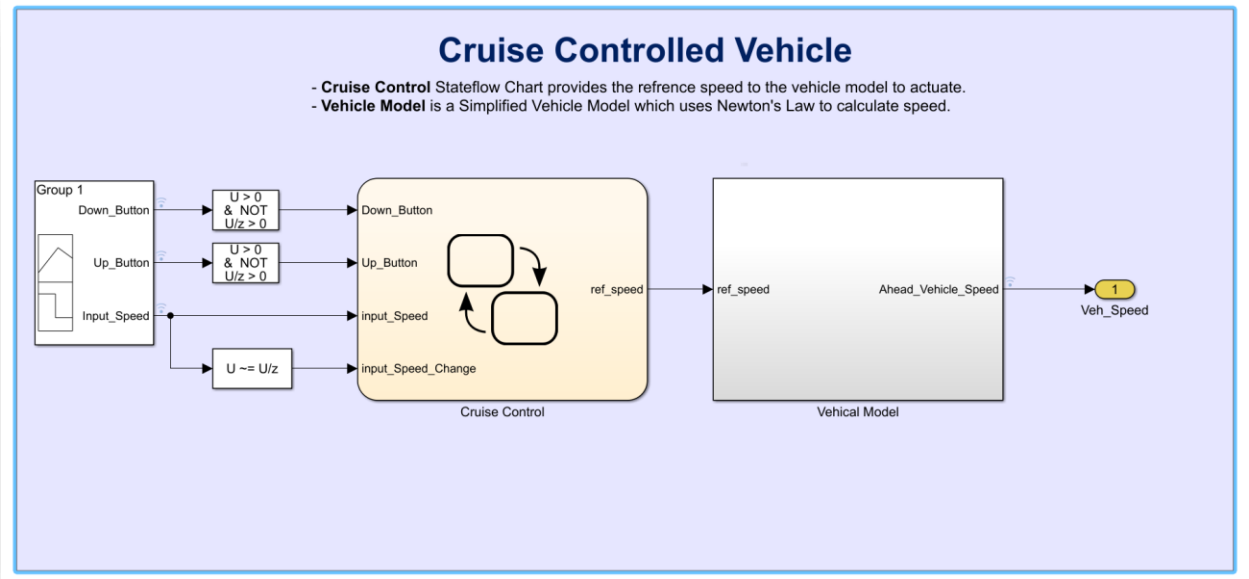

Fig. 2-1: Cruise Controlled Vehicle Model

As seen in **Fig. 2-1** there are three inputs to the system, **up_button**, **down_button** and the **input_speed**. Based on these inputs the logic of cruise control was implemented in a state flow chart. A reference speed calculated by '**Cruise Control**' state flow is given to the '**Vehicle Model**' block which gives the vehicle speed as the output.

2.3.1 Signal Builder

Simulink's signal builder block was used to provide inputs to the system as per the requirements **R2** and **R3** in **section 2.1**. Two rising edge detectors were used to show change in the button values. The signals created in signal builder are shown in **Fig. 2-2**.

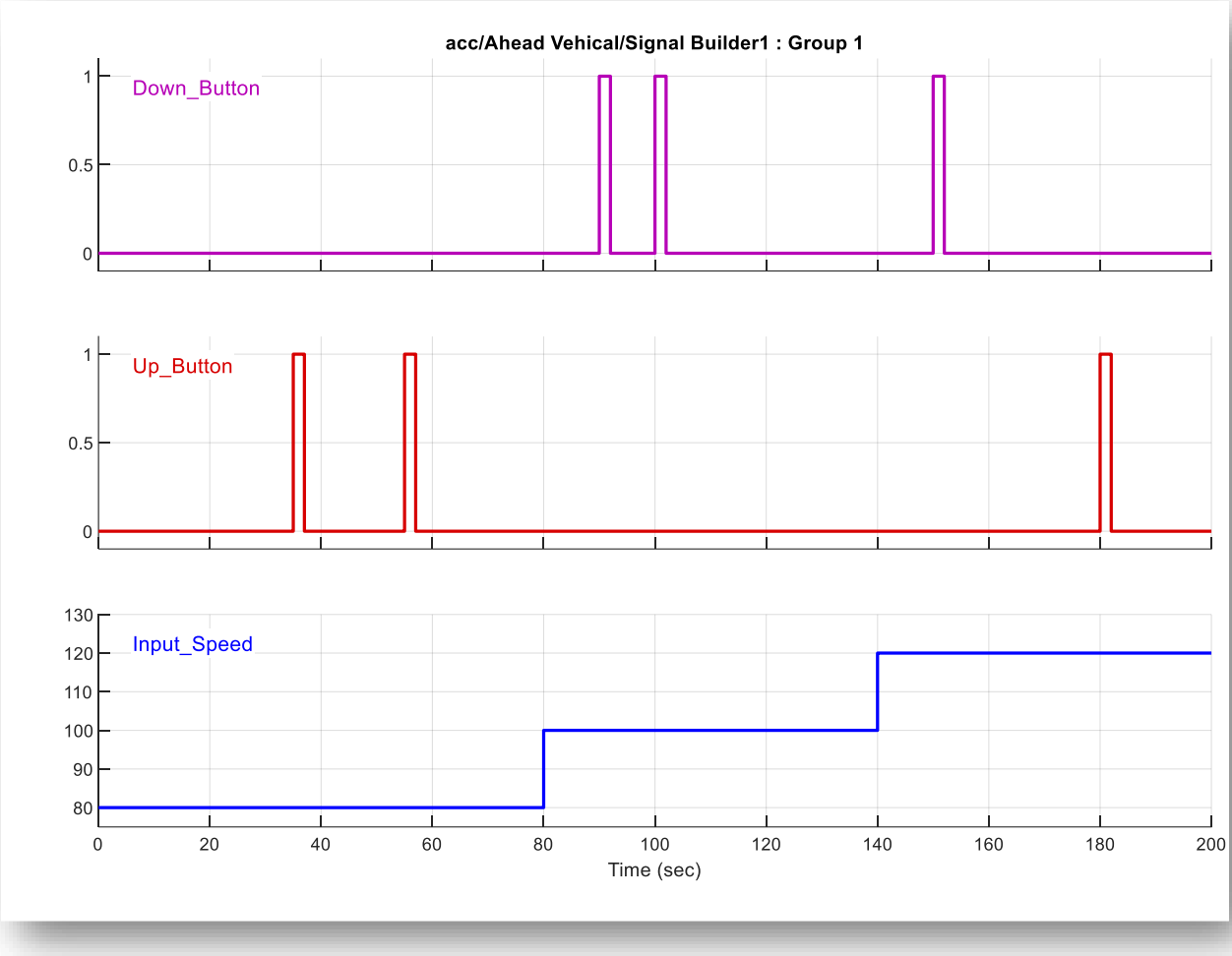

Fig. 2-2: Signals created in signal builder

As seen in Fig. 2-2, the input speed attains 3 different values and Up and Down buttons both are pressed 3 times at different times.

2.3.2 Cruise Control (State machine)

A Simulink's state flow chart was used to implement the logic of cruise control as per the requirements **R4**, **R5**, and **R6,** mentioned in section 2.1.

Four states were used to fulfil the requirements, as shown in **Fig. 2-3**.

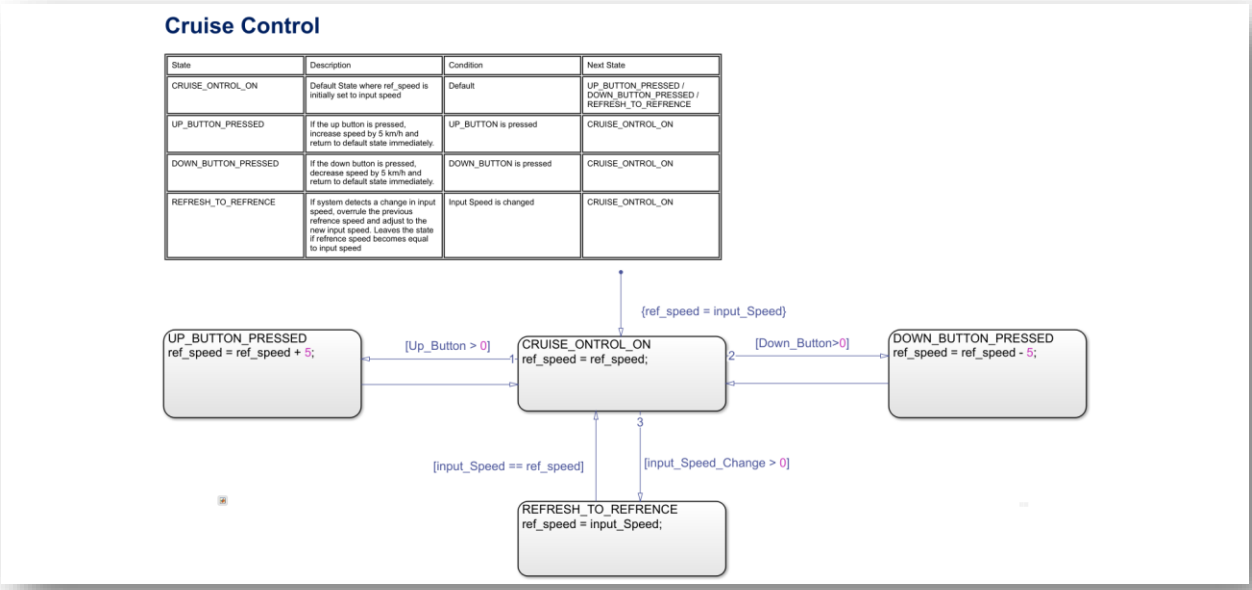

Fig. 2-3: Cruise Control State Flow Chart

The state machine controls the reference speed based on the decisions made according to its inputs. **Table 2-1** explains the state flow.

Table 2-1: State Machine Logic

State	Description	Condition of Entry	Next State
CRUISE_ONTROL_ON	Default State where ref_speed[1] is initially set to input speed	Default	UP_BUTTON_PRESSED / DOWN_BUTTON_PRESSED / REFRESH_TO_REFRENCE
UP_BUTTON_PRESSED	If the up button is pressed, increase speed by 5 km/h and return to default state immediately.	UP_BUTTON is pressed	CRUISE_ONTROL_ON
DOWN_BUTTON_PRESSED	If the down button is pressed, decrease speed by 5 km/h and return to default state immediately.	DOWN_BUTTON is pressed	CRUISE_ONTROL_ON
REFRESH_TO_REFRENCE	If system detects a change in input speed, overrule the previous reference speed and adjust to the new input speed. Leaves the state if reference speed becomes equal to input speed	Input Speed is changed	CRUISE_ONTROL_ON

[1] Output of state machine which is applied to the vehicle model to follow.

2.3.3 Vehicle Model

According to the requirement **R1** in **section 2.1**, the vehicle model was designed using a PI controller block and a transfer function block. **Vehicle Model equation 2.1** was used as transfer function. **Fig. 2-4** shows the Vehicle Model block.

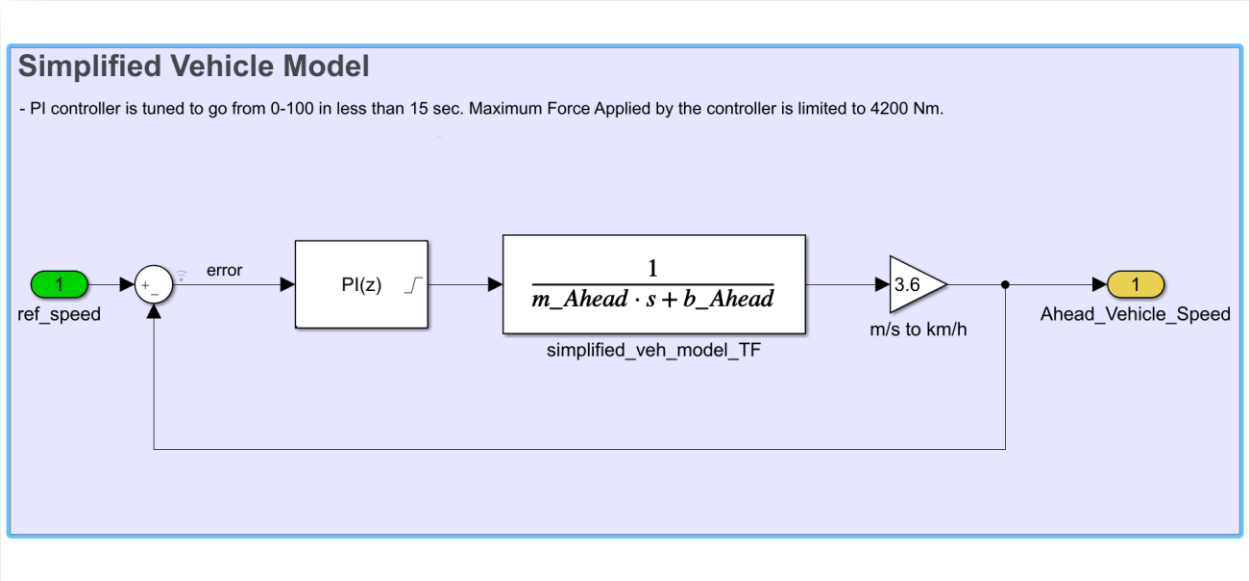

Fig. 2-4: Simplified Vehicle Model

A **discrete PI controller** was used in a closed loop with **1 ms** as sampling time. The maximum force applied by the controller was limited to **4200 Nm** (force at the wheels) by setting 4200 as the upper saturation limit of the PI controller. PI controller was tuned using $K_p = 125$ and $K_i = 7.2$ to obtain a reaction time less than 15 sec going from 0 to 100 Km/h.

As the vehicle model gives speed in m/s, as described in section 2.1, a gain block of 3.6 is used to convert **m/s to km/h**.

2.4 Test and Results

To simulate the above created system, all the values for the constants were provided in a .m file as shown in **Fig. 2-5**.

```
clc;    % Clear the command window.
clear;  % Erase all
```

Vehicle Properties

```
m_Ahead = 1000;      % mass [kg]
b_Ahead = 50;        % friction in kg/s
```

Fig. 2-5: Constant values provided in a m-file

The test scenario is created using the signal builder as shown in **Fig. 2-2**. **Table 2-2** and **Table 2-3** shows the test scenario.

Table 2-2: Vehicle Input Speed

T1 (sec)	T2 (sec)	Speed (km/h)
0	80	80
80	140	100
140	200	120

Table 2-3: Buttons Pressed at times

Button	Pressed at (sec)		
Up Button	35	55	180
Down Button	90	100	150

After using the above described test scenario, the following results were obtained:

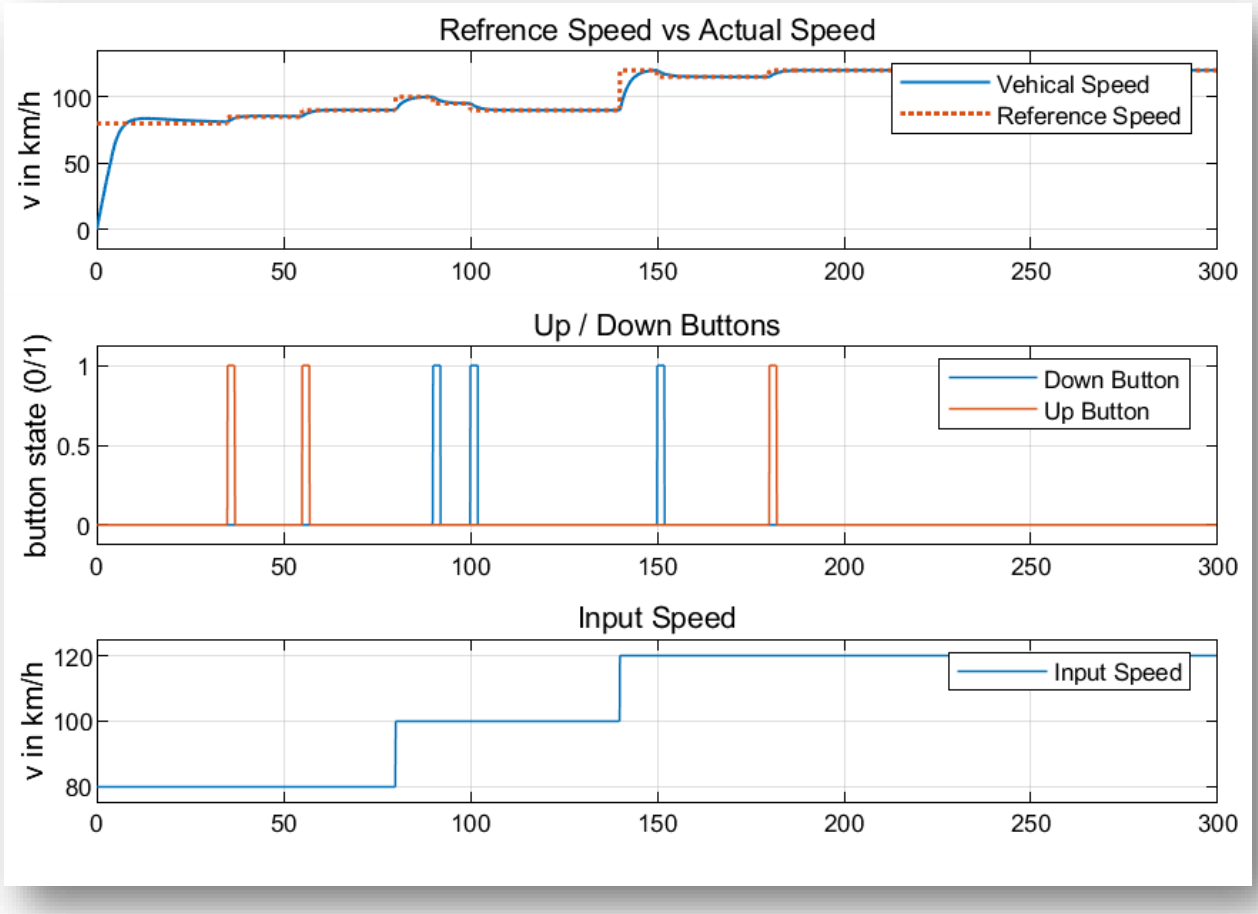

Fig. 2-6: Simulation Results

In **Fig. 2-6**, the behavior of the system can be observed. As per the requirements, the following results were obtained:

- Vehicle speed is following the reference speed as required.
- From **0 – 35 sec**, the reference speed is equal to the input speed i.e. **80 km/h**.
- At **35 sec**, up button is pressed and the reference speed goes up to **85 km/h**.
- At **35 sec**, up button is pressed again and the reference speed goes up to **90 km/h**.
- Input Speed changes at **80 sec** and goes to **100 km/h**. Reference speed follows the change and adapt the new speed.
- The down button is pressed twice, at **90 sec** & **100 sec**. The reference speed comes down to **95** and the **90 km/h**.
- At **140 sec**, the input speed changes again and reference speed follows the change.

3. Adaptive Cruise Control

The objective of adaptive cruise control (ACC) function is to sustain a steady distance to an ahead driving vehicle under varying road conditions, thus allowing the vehicle operator to relax from constant foot throttle manipulation. The ACC system include radar sensors to measure the rate of closure to other vehicles and adjust the speed to maintain a constant distance.

For the testing of the function, two vehicles are required to be implemented. The first one is the vehicle modeled in **section 2** which is considered to be the vehicle ahead. The second one is the enhanced vehicle model which additional functionality.

3.1 System Explanation

The vehicle model designed in **section 2** has to be enhanced by splitting the tractive from the braking forces. The vehicle comprises a completely nonlinear system which can be summarized into three nonlinear equations describing the engine, brake system and dynamics of the vehicle.
The vehicle speed shall be modelled based on the first order dynamic equation:

$$\dot{v}_h = \varphi_t u_t + \varphi_b u_b - \varphi_c \tag{3.1}$$

Where,

- ut **[0,100]** indicates throttle opening
- ub **[−100,0]** indicates the brake pedal position
- φt indicates the tractive force produced by wheels of the vehicle
- φb indicates the braking force
- φc indicates the external forces effecting vehicle dynamics (air drag, rolling resistance and influence of road gradient)

3.1.1 Traction Force

The traction force φt is given by the equation:

$$\varphi_t = \frac{0.01}{m \cdot r} R_{tr} R_f C_{tr} T_{max} \tag{3.2}$$

Where,

- **m** indicates the mass of the vehicle = **1500 kg**
- **r** indicates the wheel radius = **0.326 m**
- $\mathbf{R_{tr}}$ indicates the gear ratio = **1**
- $\mathbf{R_f}$ indicates the final drive ratio = **3.28**

- C_{tr} indicates the torque ratio of the torque converter = **1.6**
- T_{max} indicates is the maximum torque the engine can provide at a certain speed.

$$T_{max} = 528.7 + 0.152N_e - 0.0000217N_e^2 \quad (3.3)$$

Where,

- N_e indicates the actual engine speed in rpm. For this project, the engine speed was set constantly to **4000 rpm**.

3.1.2 Breaking Force

The breaking force φb is given by the equation:

$$\varphi_b = \frac{1.5}{m \cdot r} K_b K_c \quad (3.4)$$

where,

- K_b is the break pressure gain = **0.005**
- K_c is the lumped gain of the entire brake system = **1**

3.1.3 External Forces

External forces, like air drag, rolling resistance and influence of the gradient of the road, on the vehicle are calculated as follows:

$$\varphi_c = \underbrace{\frac{1}{2m}\rho A C_d v_h^2}_{F_{aerodynamics}} + \underbrace{C_r g cos(\theta)}_{F_{rolling\ resistance}} + \underbrace{g sin(\theta)}_{F_{gravitational}} \quad (3.5)$$

Where,

- ρ, the air density = **1.225 kg/m²**
- C_d, the drag coefficient depending on the body shape (Cw-value)
- **A,** the vehicle cross area
- C_r, the rolling resistance coefficient = **0.015**
- g, the gravitational acceleration = **9.8 kg/m²**
- θ **[°]**, the inclination of the road = **0**
- $\rho A C_d$ was taken as **0.98** for this project

3.1.4 ACC Control

The control of the distance is assured by comparing the two velocities of the follower and ahead vehicle providing the distance between those, as follows:

$$d = \int (v_h - v)dt \tag{3.5}$$

3.1.5 Functional Requirements

In addition to the cruise-controlled vehicle requirements mentioned in section 2.1, The adaptive cruise control model must fulfil the following requirements:

- **R1:** The CC function shall be enabled if the vehicle speed is above 30km/h
- **R2:** The speed shall be fixed if the set button is pressed afterwards the accelerator (input speed) shall be possible to release
- **R3:** The CC function shall be stopped if the brake pedal is pressed
- **R4:** The distance between the vehicles shall be selectable by the driver
- **R5:** In case a vehicle ahead is below the set distance, the ACC must overrule the CC function by keeping the desired distance => Distance tracking mode
- **R6:** In case the vehicle ahead accelerates, the vehicle shall accelerate and retake the set vehicle speed and leaf ACC mode if distance is bigger than maximum. => Speed tracking mode

3.2 Definition of Input(s) and Output(s)

Considering the above mentioned functional requirements, the inputs and outputs were defined as follows:

Inputs:

- **input_Speed,** a reference speed that will be used to calculate the driver throttle power.
- **Up_button,** indicates an increase of 5 km/h in speed
- **Down_button,** indicates a decrease of 5 km/h in speed
- **Speed_Set_Button,** a button press signal after which the speed shall be fixed
- **Driver_break,** indicates driver's breaking paddle position.
- **Ahead_veh_speed,** speed of the vehicle ahead
- **Desired_Distance,** min distance required from the ahead vehicle, set by the driver.

Outputs:

- **vehicle speed,** vehicle actual speed

- **MODE,** indicates the mode of tracking of the vehicle.

3.3 Implementation in Simulink

The above described system was implemented in Simulink, as shown in **Fig 3-1,** with 2 significant blocks.

- Follower Vehicle
- Obstacle and Environment

The **Obstacle and Environment** block has a simplified vehicle model with cruise control which was developed in **section 2**.

Follower vehicle has the adaptive cruise control functionality, which means it can deal with the obstacles in the way and adapt its speed accordingly.

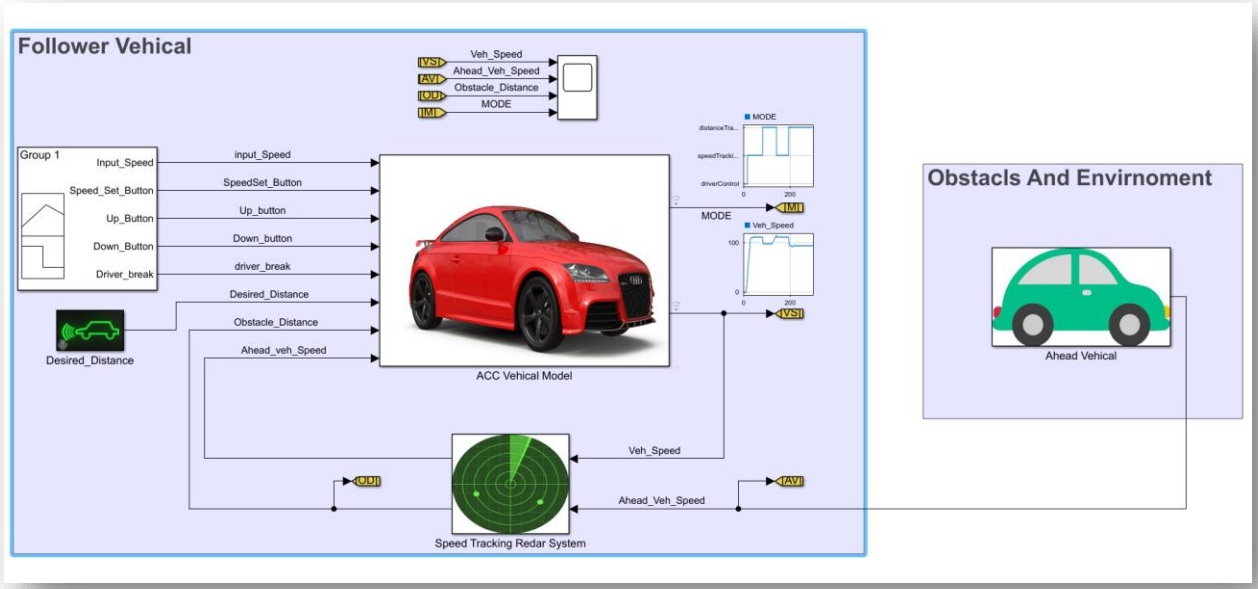

Fig. 3-1: Adaptive Cruise Control Model in Action

The Follower vehicle model was divided into the following parts:

- Signal Builder
- Speed Tracking (Radar System)
- ACC Vehicle Model

The description of these model is as follows:

3.3.1 Signal Builder

Simulink's signal builder block was used to provide inputs to the system based on our test scenario. Following signals are generated from the signal builder block:

- **Input_Speed**: Vehicle has to follow this speed if the speed is not set by the driver
- **Set_Speed_Button**: If this button is pressed, the speed shall be kept constant unless the break paddle is not pressed.
- **Up_Button**: If pressed, the vehicle speed should go up by 5 km/h (If in a cruise control mode).
- **Down_Button**: If pressed, the vehicle speed should go down by 5 km/h
- **Driver_Break:** Indicates the breaking paddle position

The definition of these signals will be discussed in **section 3.4**

3.3.2 Desired Speed

To fulfil the requirement **R4** in section 3.1.5 a separate block was built and masked to provide ease to select the minimum distance desired to keep between the vehicles (in meters).

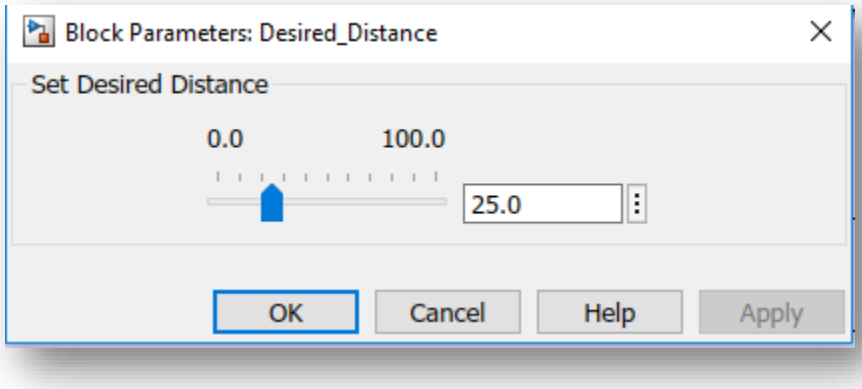

Fig. 3-2: Desired Distance selected by the driver

3.3.3 Speed Tracking (Radar System)

To model the function of a radar system and to calculate the distance between the two vehicles, the block takes two inputs; **Ahead vehicle speed** and **Follower vehicle speed**.

The system uses differential **equation 3.5** to calculate the distance as an output. As the distance is taken in meters, the conversion from km/h to m/s was done as shown in **Fig 3-3.**

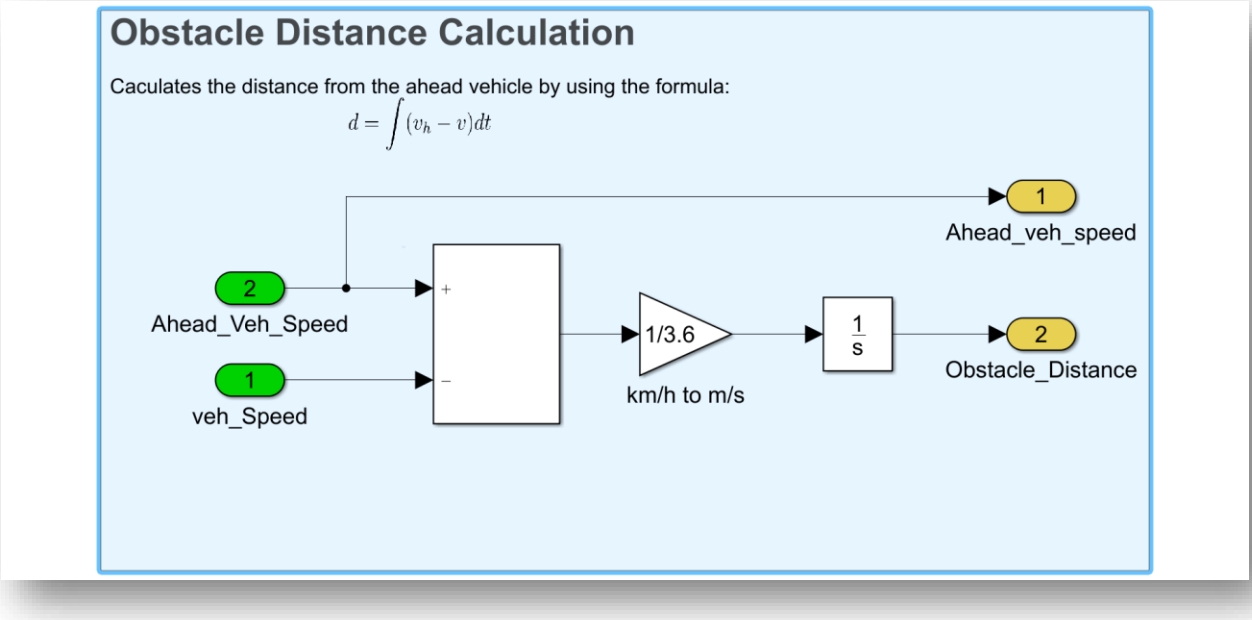

Fig. 3-3: Obstacle Distance Calculation Block

3.3.4 ACC Vehicle Model

According to the requirement in **section 3.1.5**, the Adaptive Cruise Control with enhanced vehicle model was designed as shown in **Fig. 3-4**.

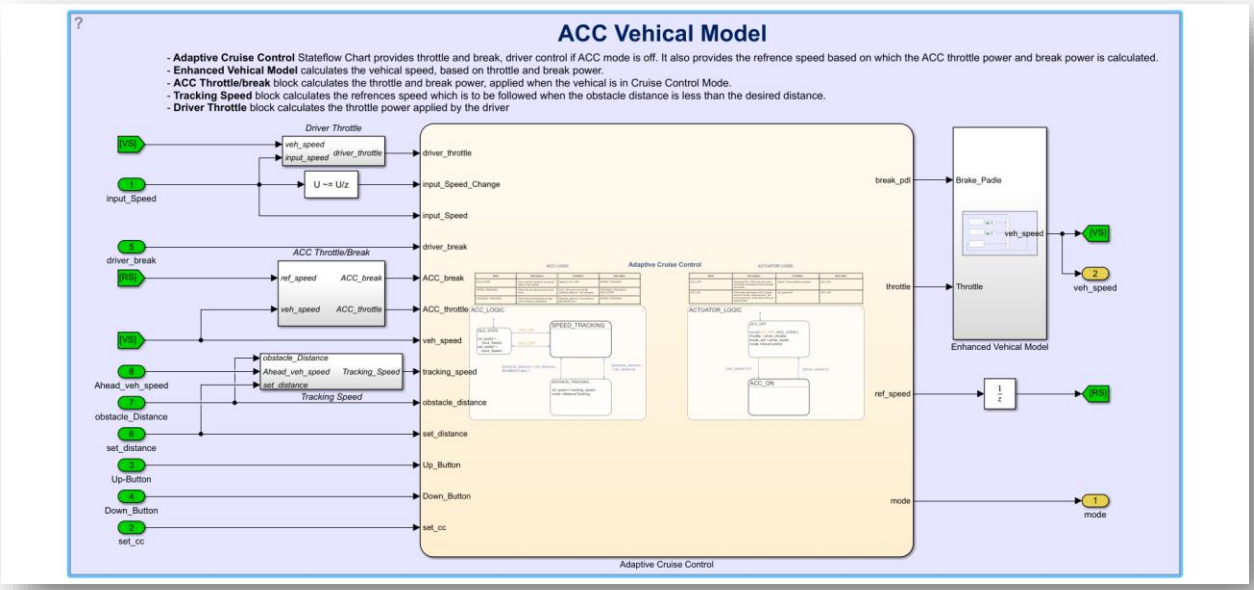

Fig. 3-4: ACC Enhanced Vehicle model

The enhanced vehicle model works with traction and breaking forces separately, hence the logic was built to separate throttle and breaking forces for both driver control and Cruise control Modes.
This block was further divided into following blocks:

3.3.4.1 Driver Throttle

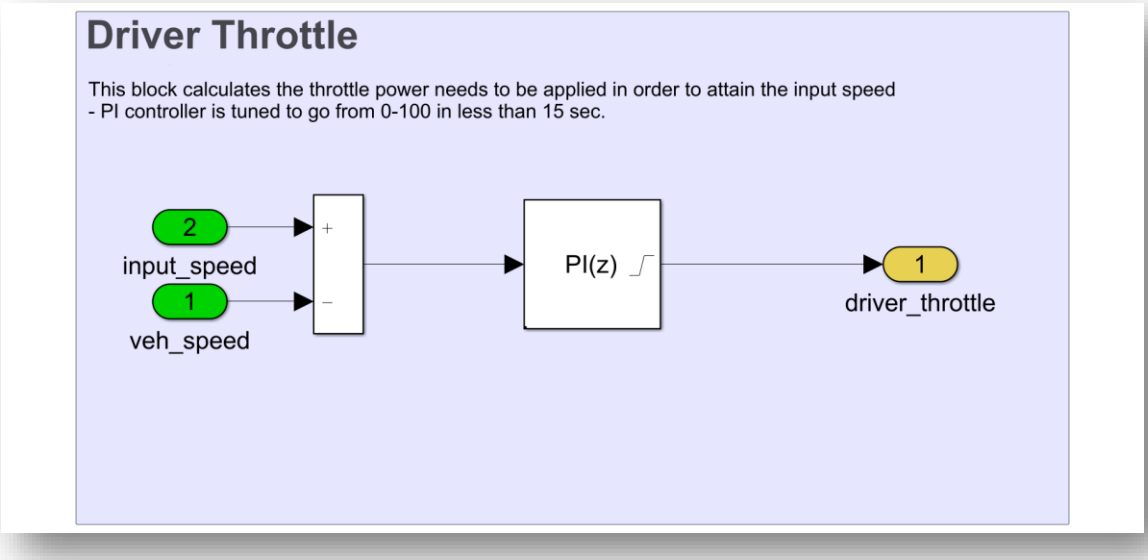

Fig. 3-5: Driver Throttle Block

This block calculates the driver-controlled throttle power that is needed to keep the vehicle at the **input_speed**. This throttle opening is taken as input by the Enhanced vehicle model as described in section 3.1 . A PI controller was used to set the response time to **0 -100** in less than **15 sec** by setting $\mathbf{K_p}$ = 5 and $\mathbf{K_i}$ = 0.5. The upper limit is set to 100 as per the requirement of **equation 3.1**

3.3.4.2 ACC Throttle/Break

When the vehicle is in cruise control mode, the vehicle has to follow the reference speed specified by the cruise control logic. The ACC Throttle/Break separation block calculates the throttle and break openings, which are used by the enhanced vehicle model if the vehicle is in Cruise Control Mode.

This block takes **ref_speed**, calculated by ACC Logic (explained in the next sections), and the **veh_speed** as input and apply throttle and break to regulate vehicle speed to the reference speed. **Fig. 3-6** shown the model implementation.

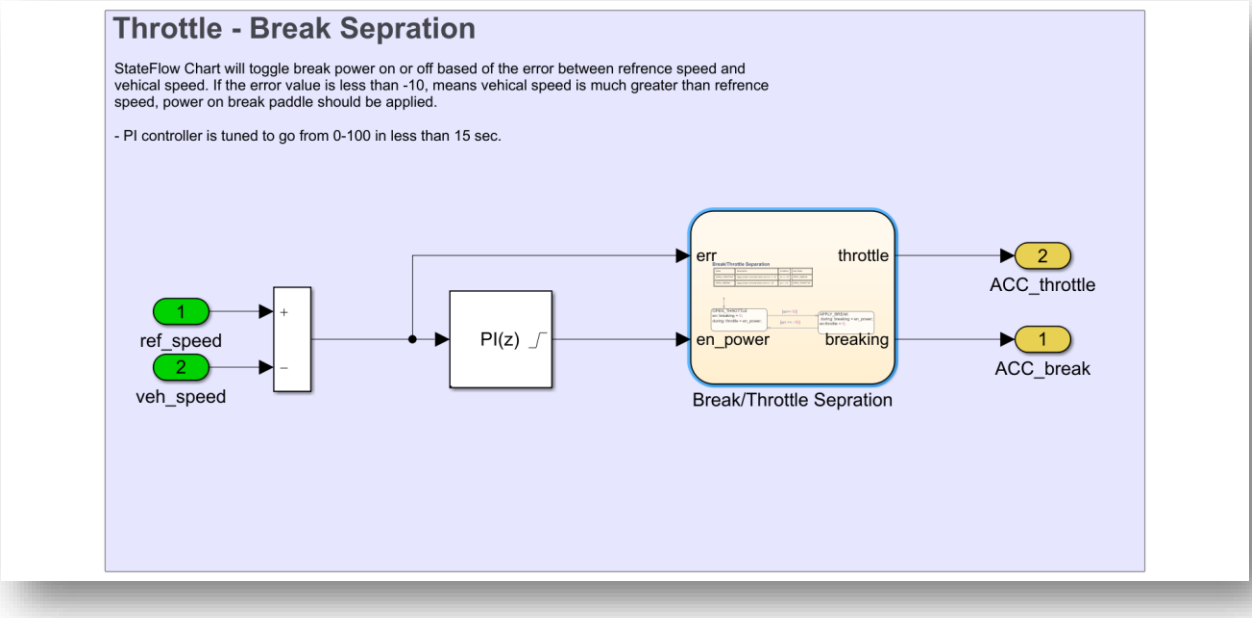

Fig. 3-6: Throttle/Break Separation Block

A PI controller was used to set the response time to **0 -100** in less than **15 sec** by setting $\mathbf{K_p}$ = 5 and $\mathbf{K_i}$ = 0.5. The upper limit was set to **100** and lower limit was set to **-100.**

A state flow chart was introduced in the system to separate the throttle and breaking power. **Fig 3.7** shows the state flow chart.

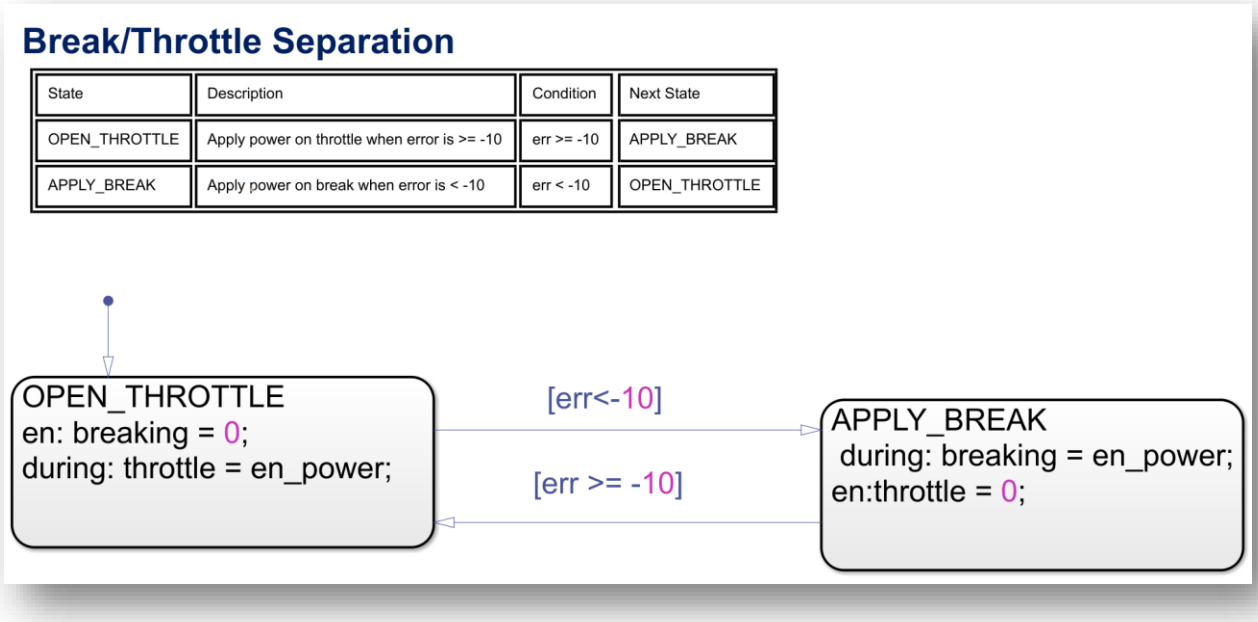

Fig. 3-7: ACC Throttle Break Separation Logic

The state logic is described in **Table 3-1.**

Table 3-1: Throttle / Break Separation Logic

State	Description	Condition of Entry	Next State
OPEN_THROTTLE	If the vehicle speed is close to reference speed i.e. err >= -10 Send power to throttle	Default / Err >= -10	APPLY_BREAK
APPLY_BREAK	If the vehicle speed is much greater than reference speed i.e. err < -10. Send power to break	Err < - 10	OPEN_THROTTLE

3.3.4.3 Tracking Speed

According to the functional requirement **R5,** in case a vehicle ahead is below the set distance, the ACC must overrule the CC function by keeping the desired distance and goes into the Distance tracking mode.

Tracking speed block calculates the tracking speed required to maintain the distance between the two vehicles. **Fig. 3-8** shows the Tracking Speed block.

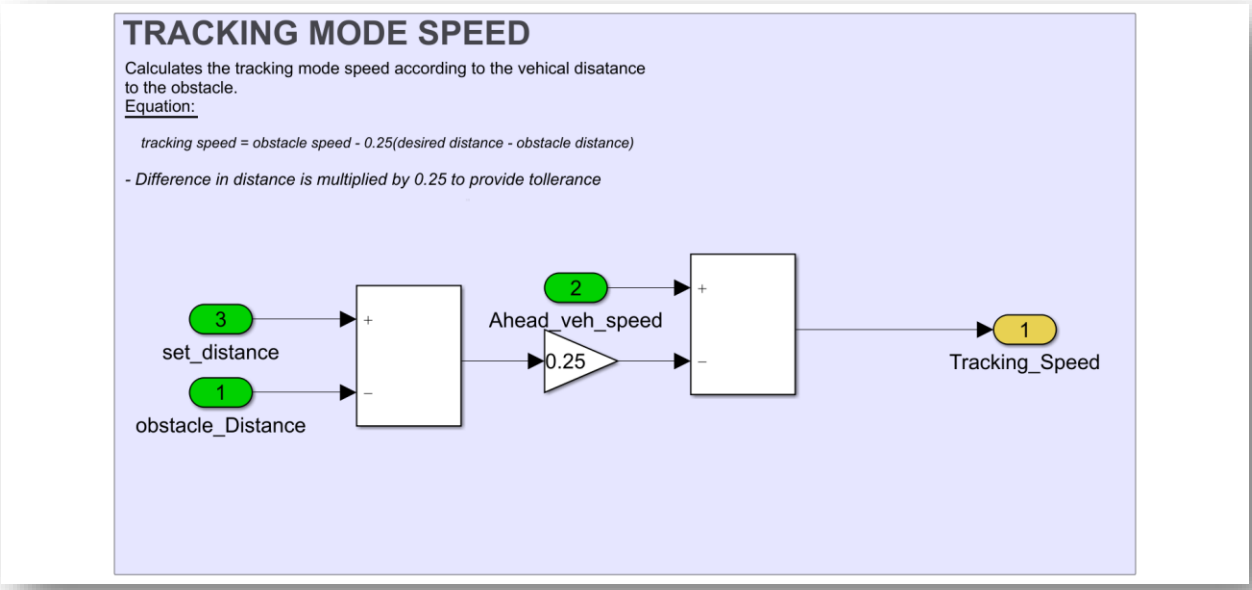

Fig. 3-8: Tracking Speed Block

The tracking speed is calculated as follows:

$$v_h = v - 0.25(d_{set} - d_{actual}) \tag{3.6}$$

Where,

- v_h, is follower vehicle speed
- **v,** is ahead vehicle speed
- d_{set}, is desired distance
- d_{actual}, is actual distance from the vehicle

3.3.4.4 Adaptive Cruise Control

The Adaptive Cruise Control State machine was used to actuate the functional requirements **R1**, **R2**, **R3**, **R5**, and **R6** mentioned is **section 3.1.5.**

Two parallel states **ACTUATOR_LOGIC** and **ACC_LOGIC**, in the respective execution order, were implemented. The ACTUATOR_LOGIC decides if the vehicle is in driver control mode or if it is in Cruise Control mode. The ACC_LOGIC goes from idle to speed tracking state when ACC_ON event occurs. **Fig. 3-9** shows the Adaptive Cruise Control State Machine.

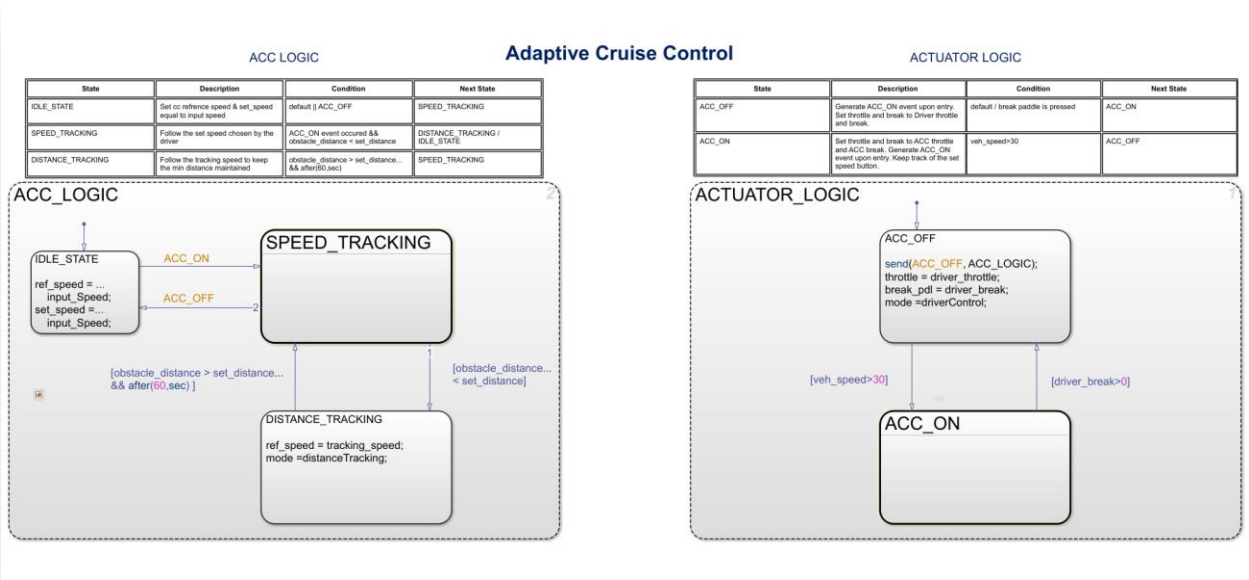

Fig. 3-9: Adaptive Cruise Control State Machine

For better representation of the control mode of the vehicle, a custom data type, **control_mode_type**, was created in Simulink that has three members; **driverControl**, **speedTracking**, and **distanceTracking** as shown in **Fig. 3-10**.

```
classdef (Enumeration) control_mode_type < Simulink.IntEnumType
    enumeration
        driverControl(0)
        speedTracking(1)
        distanceTracking(2)

    end
end
```

Fig. 3-10: Custom Data Type for Vehicle Control Modes

3.3.4.4.1 ACTUATOR_LOGIC

This state decides the control mode of the vehicle, from driverControl to ACC (adaptive cruise control) mode. And based on the mode it decides whether to output driver throttle and break,

or to use ACC throttle and break (described in section 3.3.4.1 & 3.3.4.2) . **Table 3-2** shows the state flow of ACTUATOR_LOGIC state.

Table 3-1: ACTUATOR_LOGIC State

State	Description	Condition of Entry	Next State
ACC_OFF	Generate ACC_OFF event upon entry and broad cast it to ACC_LOGIC state. Set throttle and break to Driver throttle and break.	Default / driver_break > 0	ACC_ON
ACC_ON	Activated when vehicle speed exceeds 30 km/h then Set throttle and break to ACC throttle and ACC break. Generate ACC_ON event upon entry. Keep track of the set speed button.	Veh_speed > 30	ACC_OFF

The **ACC_ON** state has 2 further grouped states, **SS_BUTTON_UNPRESSED** and **SS_BUTTON_PRESSED**. These states keep track of the set speed button. If the button is pressed, ss_button is set to be one and set_speed is set as the input speed at that instance. These values are then used in parallel ACC_LOGIC state. The ACC_ON grouped states are shown in **Fig. 3-11**.

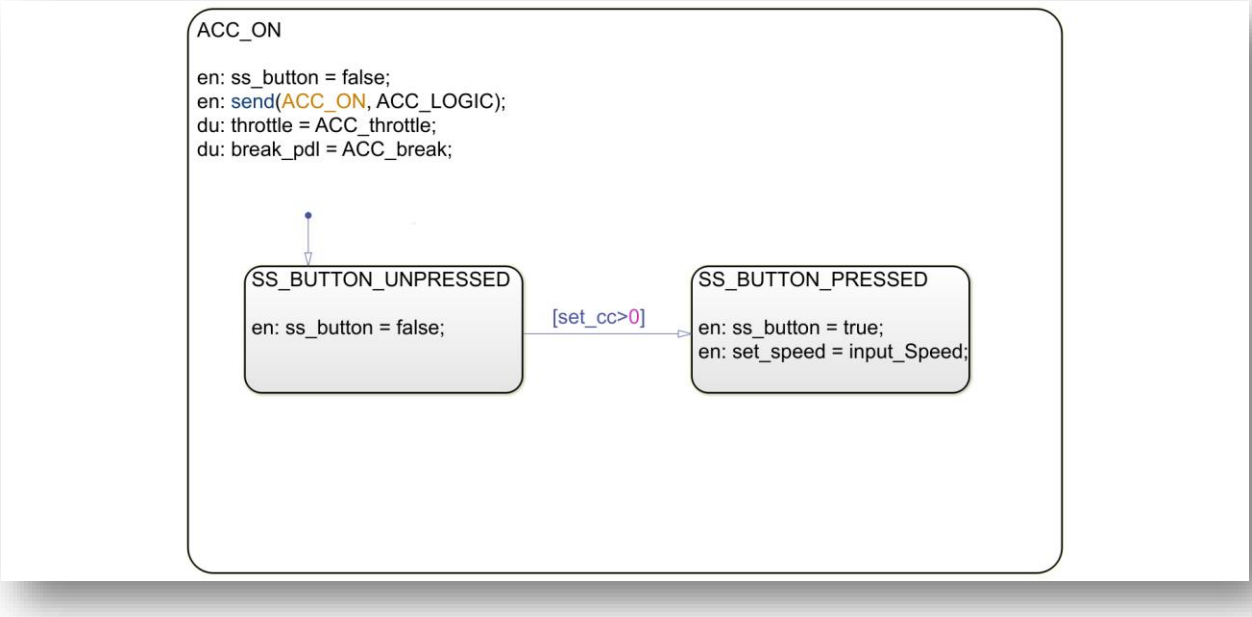

Fig. 3-11: ACC_ON Grouped States

3.3.4.4.2 ACC_LOGIC

When the vehicle is in **driverControl** mode, the **ACC_LOGIC** State remains in idle state and follows the input speed. Upon receiving the **ACC_ON** event, the vehicle goes to **speedTracking** mode. **SPEED_TRACKING** state is the same as the Simplified **CRUISE_CONTROL** state machine developed in **section 2.3.2** but with the implementation of a few more requirements.

- The speed shall be fixed if the set button is pressed afterwards the accelerator shall be possible to release. i.e. any change in input speed won't affect the vehicle speed.

Fig. 3-12 shows the upgraded Cruise Control logic.

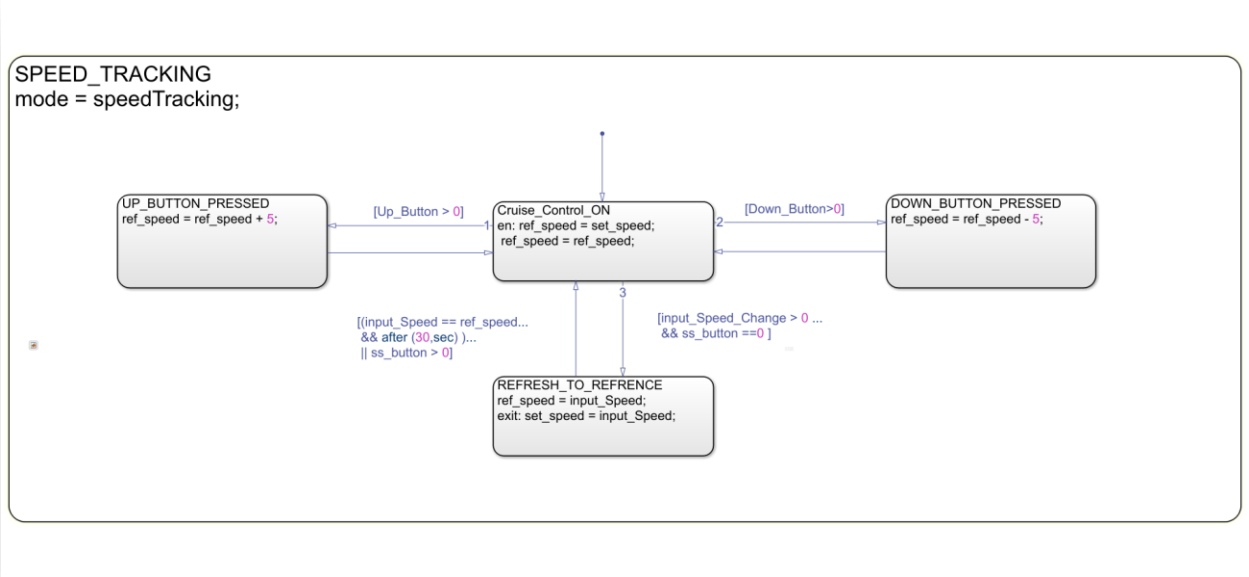

Fig. 3-12: SPEED_TRACKING State

Table 3-3 shows the state description of ACC_LOGIC state.

Table 3-2: ACC_LOGIC State

State	Description	Condition of Entry	Next State
IDLE_STATE	Set cc reference speed & set_speed equal to input speed	default || ACC_OFF	SPEED_TRACKING
SPEED_TRACKING	Follow the input speed unless speed set button is pressed. When button is pressed, follow the set speed.	ACC_ON event occured && obstacle_distance < set_distance	DISTANCE_TRACKING / IDLE_STATE
DISTANCE_TRACKING	Follow the tracking speed to keep the min distance maintained	obstacle_distance > set_distance... && after(60,sec)	SPEED_TRACKING

3.3.4.5 Enhanced Vehicle Model

An Enhanced vehicle model was developed using the **equation 3.1.** Throttle and Break_Paddle are the inputs to the system. The system generates the speed in m/s which is being converted to km/h. Fig. 3-13 shows the model of the system.

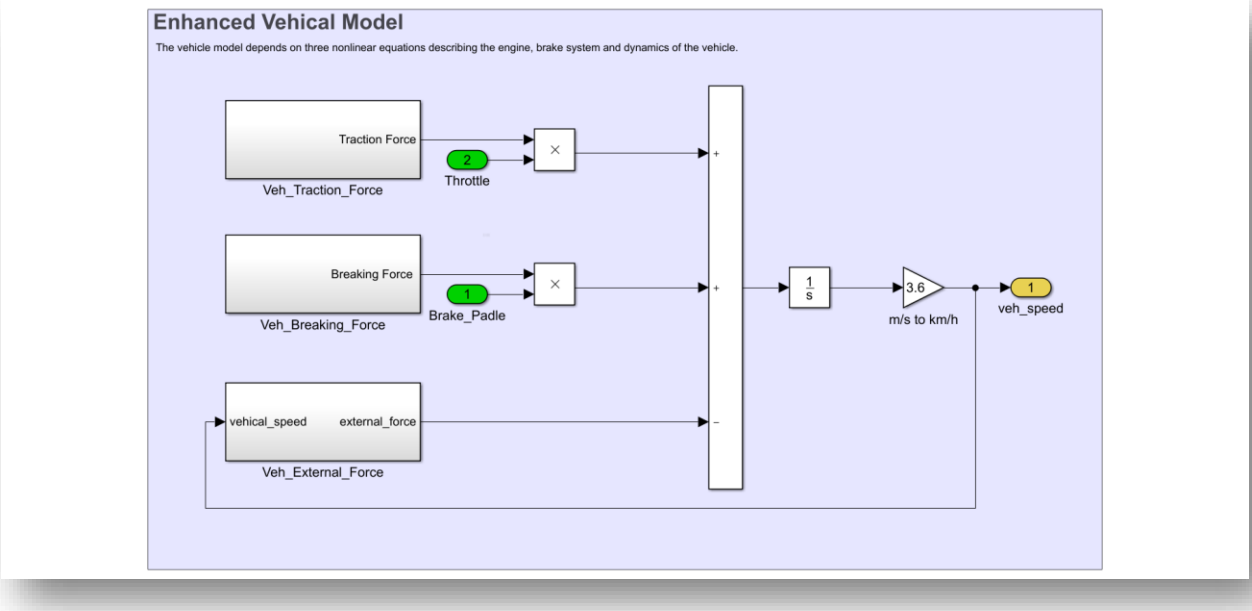

Fig. 3-13: Enhanced Vehicle Model

3.3.4.5.1 Vehicle Traction Force

To implement vehicle traction force, **equation 3.2** from **section 3.1.1** was used. **Fig. 3-14** shows the vehicle traction force model. Values of these constants were declared in a .m file shown in **Fig. 3-17**

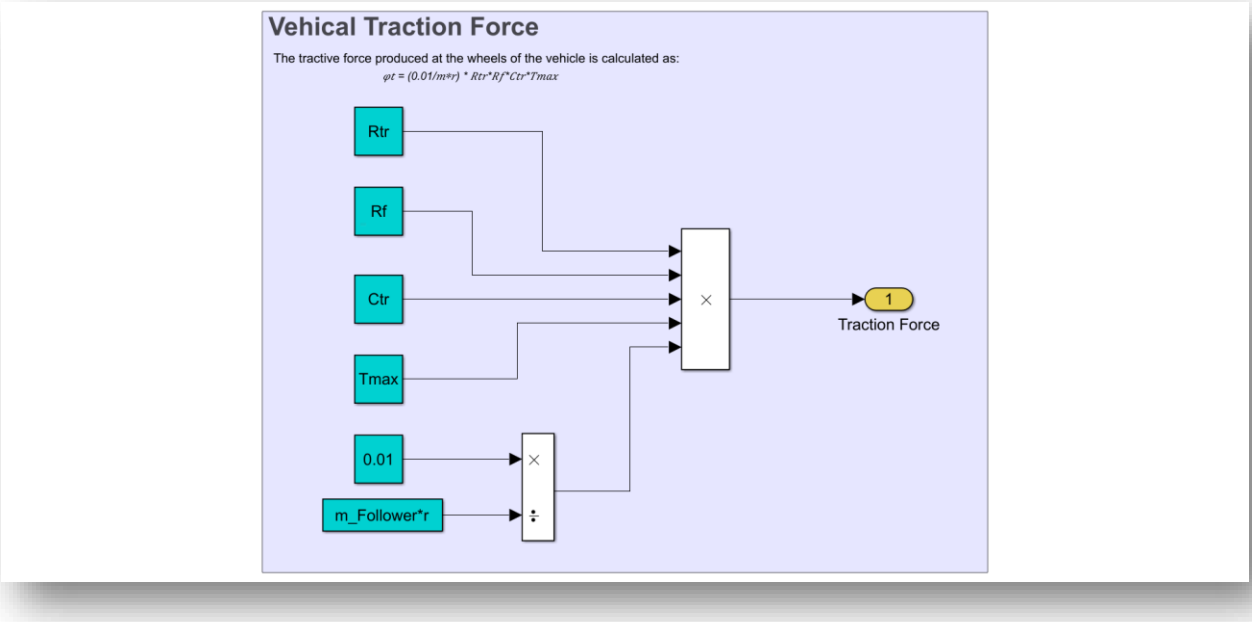

Fig. 3-14: Vehicle Traction Force

3.3.4.5.2 Vehicle Breaking Force

To implement vehicle breaking force, **equation 3.4** from **section 3.1.2** was used. **Fig. 3-15** shows the vehicle breaking force model. Values of these constants were declared in a .m file shown in **Fig. 3-17**

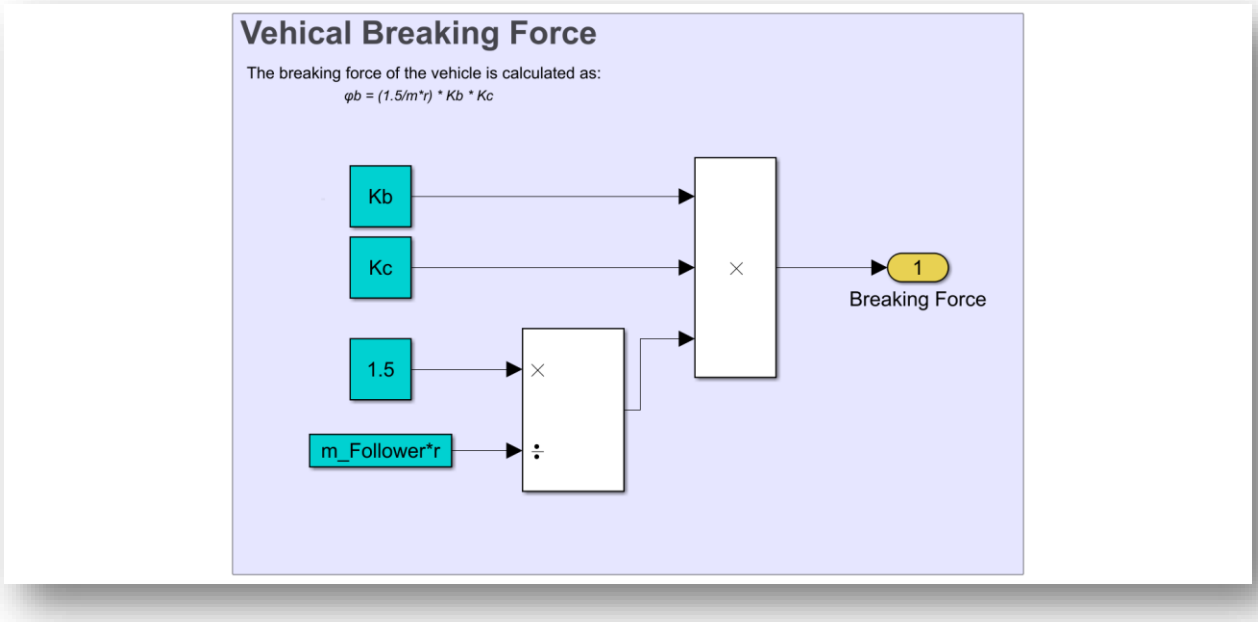

Fig. 3-15: Vehicle Breaking Force

3.3.4.5.3 Vehicle External Forces

To implement vehicle external forces, **equation 3.5** from **section 3.1.3** was used. **Fig. 3-16** shows the vehicle external forces model. Values of these constants were declared in a .m file as shown in **Fig. 3-17**

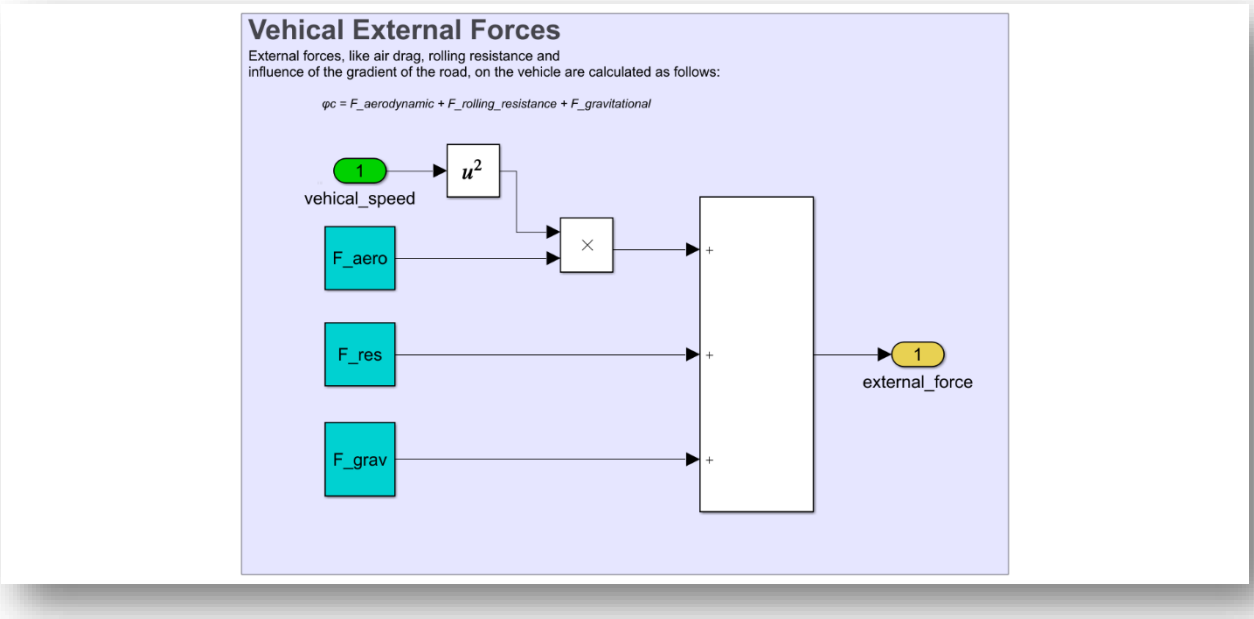

Fig. 3-16: Vehicle External Forces

3.4 Test Scenario

To simulate the above created system, all the values for the constants were provided in a .m file as shown in **Fig. 3-17**.

Ahead Vehicle

```
m_Ahead = 1000;      % mass [kg]
b_Ahead = 50;        % friction in kg/s
```

Follower Vehicle

```
m_Follower = 1500;   % mass [kg]
r  = 0.326;          % wheel radius [m]
Rf = 3.28;           % Final Drive Ratio
Rtr = 1;             % Gear Ratio
Ctr = 1.6;           % Torque ratio of torque converter
Ne = 4000;           % Engine Speed [rpm]
Kc = 1;              % Brake Pressure Gain
Kb = 0.005;          % Lumped Gain of entire braking system
Ro_A_Cd = 0.98;      % Aerodynamic factor
Cr = 0.015;          % Rolling Resistance coefficient
g = 9.81;            % Gravitational Acceleration [m/s^2]
theta = 0;           % Inclination of the road [degree]
F_aero = 1/(2*m_Follower) * Ro_A_Cd;
F_res  = Cr * g * cos(theta);
F_grav = g * sin(theta);
Tmax = 528.7 + 0.152*Ne - 0.0000217*Ne^2;
```

Fig. 3-17: Declaration of constants in m-file

A test scenario was created to test the behavior of the model. All the input signals were provided using the signal builder. **Table 3-4** and **Table 3-5** shows the input speeds of Ahead and Follower vehicles in km/h.

Table 3-3: Follower Vehicle Input Speed

T1 (sec)	T2 (sec)	Mode	Speed (km/h)
12	30	Accelerate	112
30	42	Keep	112
42	43	Reduce	0
43	300	Keep	0

Table 3-4: Ahead Vehicle Input Speed

T1 (sec)	T2 (sec)	Mode	Speed (km/h)
5	25	Accelerate	100
25	38	Keep	100
38	80	Reduce	98
80	120	Keep	98
120	140	Accelerate	115
140	175	Keep	115
175	195	Reduce	94

3.4.1 Signal Builder

Two signal Builders were used to drive two vehicles. **Fig. 3-18** & **Fig. 3-19** shows the signals created as inputs to the system.

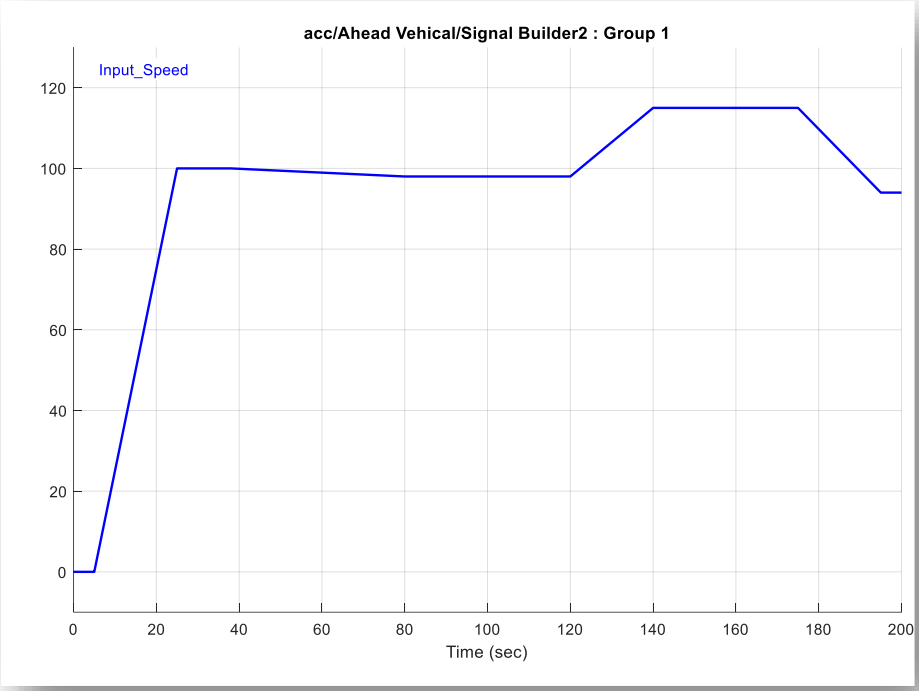

Fig. 3-18: Ahead Vehicle Input Speed

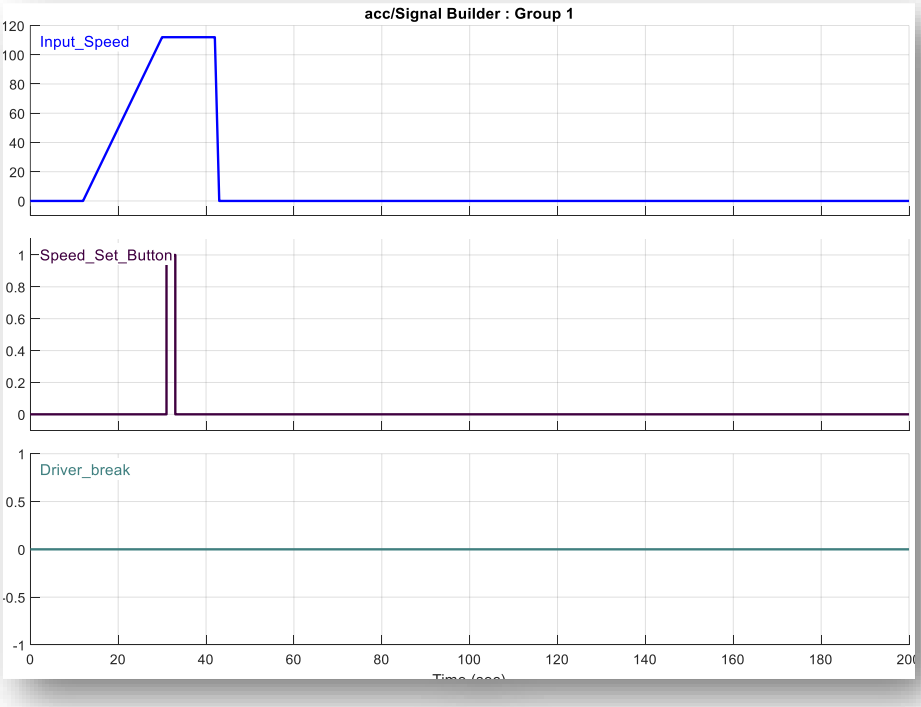

Fig. 3-19: Follower Vehicle Input Signals

3.5 Test Results

After realizing the above developed scenario, the following results were obtained:

3.5.1 Speed set point vs. Actual Speed (Vehicle Ahead)

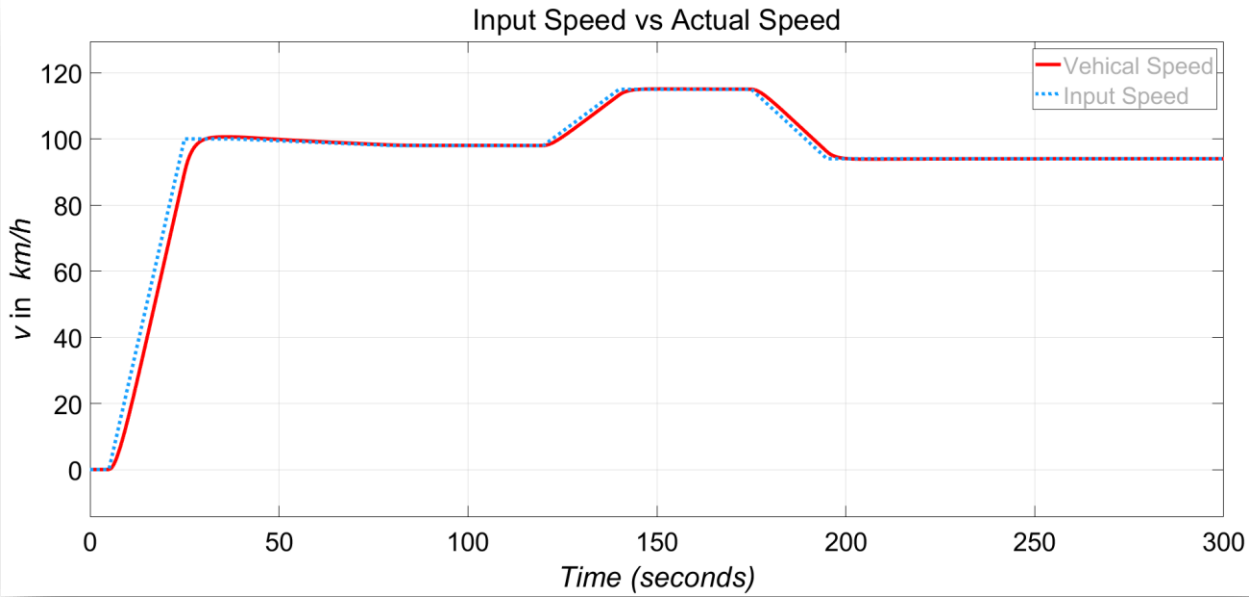

Fig. 3-20: Ahead Vehicle Input Speed vs. Actual Speed

Fig. 3-20 shows that Ahead Vehicle, which was developed in **section 2** using a simplified vehicle model, follows the input speed with a very less reaction time.

3.5.2 Speed set point vs. Actual Speed (Follower Vehicle)

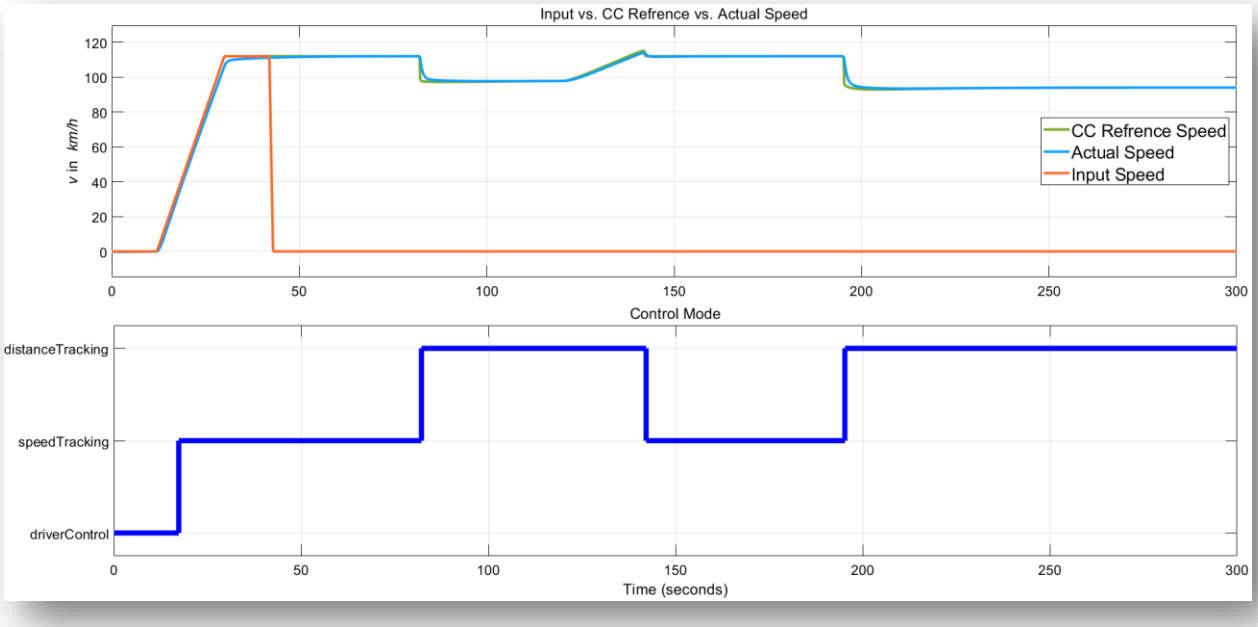

Fig. 3-21: Follower Vehicle Speed Set Point vs. Actual Speed

Figure 3-21 shows the follower **vehicle actual speed**, **cc reference speed**, **input speed**, and the **mode of control**. As seen in the **sub-graph 1**, the actual speed follows the set reference speed with a very high response time. When the speed set button is pressed at **31 sec**, the car follows the cc reference speed instead of the input speed.

The Cruise Control mode was turned on as the vehicle crosses **30 km/h**.

3.5.3 Reference Distance vs. Actual Distance

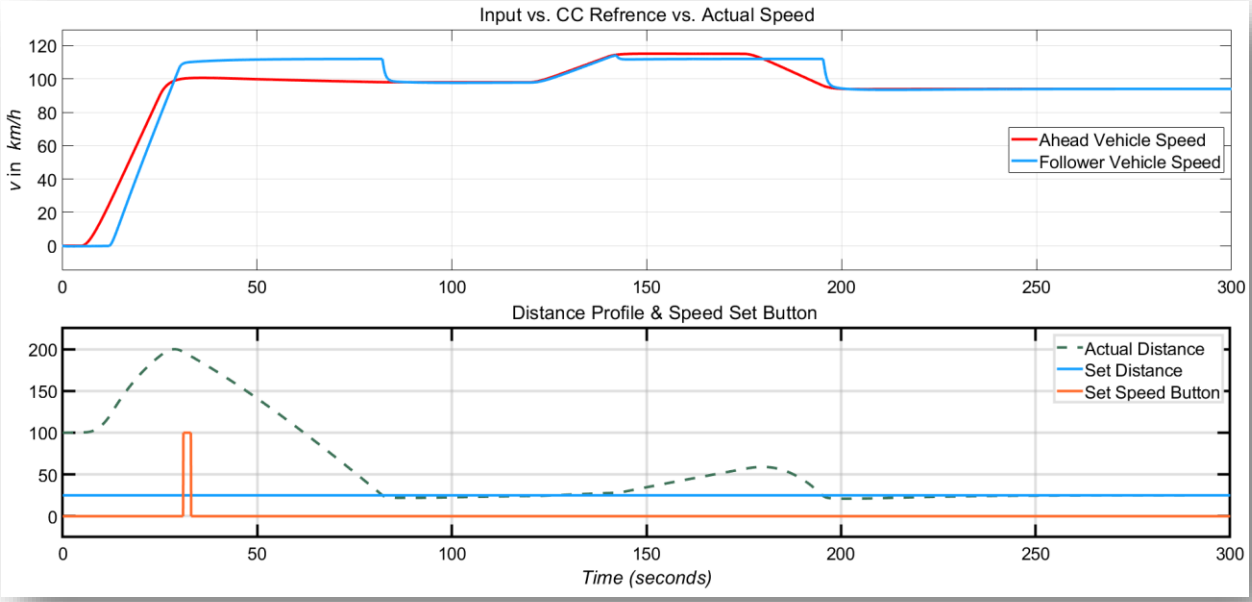

Fig. 3-22: Actual Distance vs. Set Distance

Fig. 3-22 shows the Follower vehicle speed and ahead vehicle speed in sub-graph 1 and their difference in distance in sub-graph 2. It can be seen than initially the distance was set to **100 m**. As the follower vehicle accelerates, from **5 – 30 sec**, follower vehicle speed is less than the ahead vehicle speed, so the distance is increasing. At **30 sec**, follower vehicle speed becomes greater than ahead vehicle speed and the distance start decreasing. At **31 sec**, the **set speed button** is pressed so the follower vehicle speed is then fixed.

When the distance becomes less than the set distance, which is set to **25 m**, the follower vehicle enters the **distance tracking mode** and follows the ahead vehicle speed to keep the minimum distance maintained. At **127 sec**, when the distance becomes greater then set distance, the follower vehicle goes back to the **speed tracking** mode and attain the set speed. At **195 sec** the distance becomes less than the required distance, so the follower vehicle enters the **distance tracking** mode again and follows the ahead vehicle speed for the rest of the simulation.

3.5.4 Actual Distance vs. ACC Tracking speed Profile

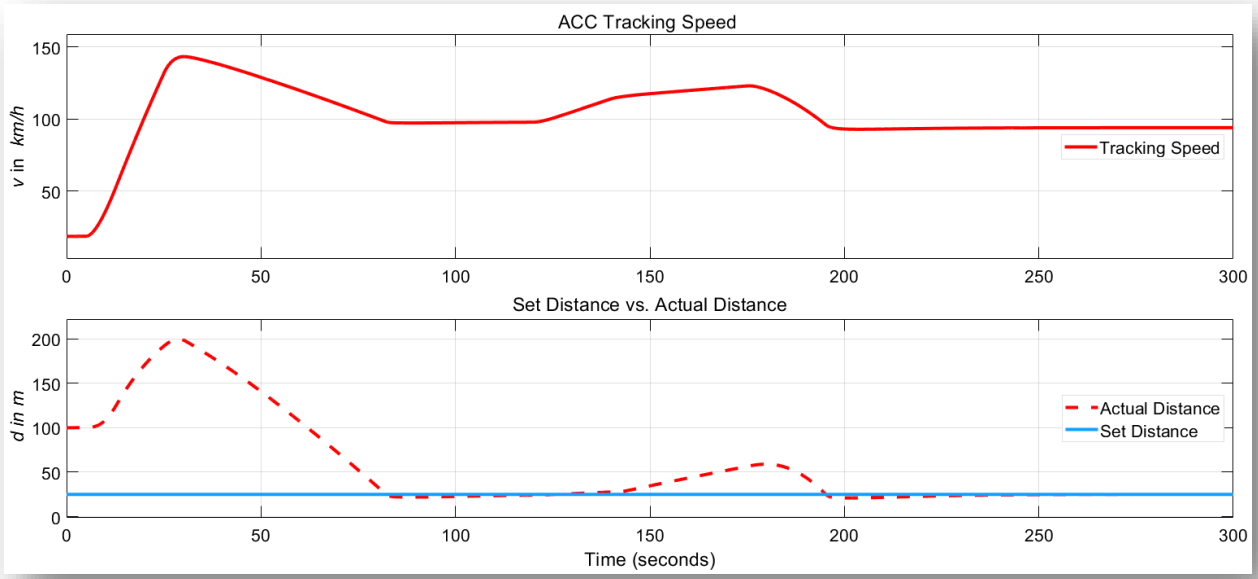

Fig. 3-23: Actual Distance vs ACC Tracking Speed

When the vehicle is distance tracking mode it must follow the calculated tracking speed to maintain the desired distance between the vehicles.

Fig. 3-23 shows the tracking speed generated based on the difference in distance. It can be seen that the tracking speed is **proportional** to the distance. As the distance decreases, tracking speed also decreases to attain the set distance. This speed will only be followed by the vehicle when it is in **distance tracking mode** i.e. distance is less than the desired distance.

3.5.5 Follower Vehicle Modes of Control

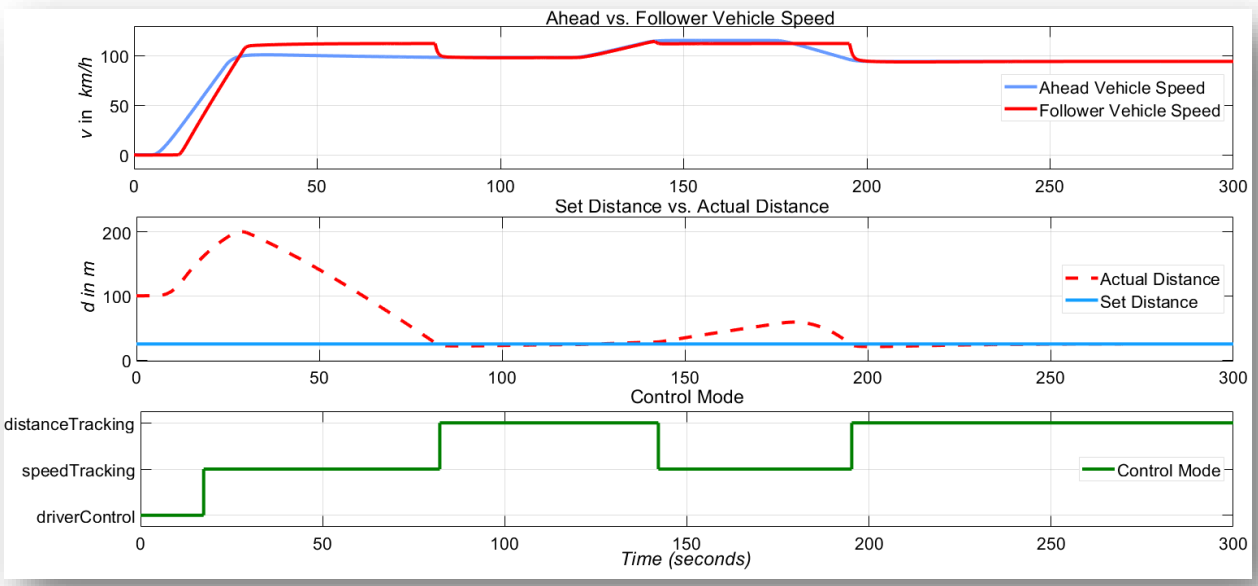

Fig. 3-24: Control Modes of Follower Vehicle

Fig. 3-24 shows the different modes of the ACC vehicle. The vehicle can attain three modes i.e. Driver Control, Speed Control. Vehicle must stay in driver control mode if the vehicle speed is less than **30 km/h** and goes into speed control when speed increases from **30 km/h**. Distance tracking mode is activated when the distance to the ahead vehicle is less than the desired distance. **Table 3-6** shows the control mode profile of the vehicle.

Table 3-5: Follower Vehicle Control Mode Profile

T1 (s)	T2(s)	Control Mode	Control Mode Condition
0	17.3	Driver Control	Speed is less than 30 km/h
17.3	82	Speed Tracking	Speed is greater than 30 km/h and distance is greater than desired distance
82	142	Distance Tracking	Distance becomes less than desired distance
142	195	Speed Tracking	Distance is greater than desired distance
195	300	Distance Tracking	Distance becomes less than desired distance.

3.5.6 Speed Set Button Behavior

Adaptive cruise control provides the additional functionality where if the set speed button is pressed, the speed should be fixed, and the driver can release the throttle. This speed shall be followed until the break paddle is not pressed. Fig. 3-25 shows the behavior of vehicle when the set speed button is pressed.

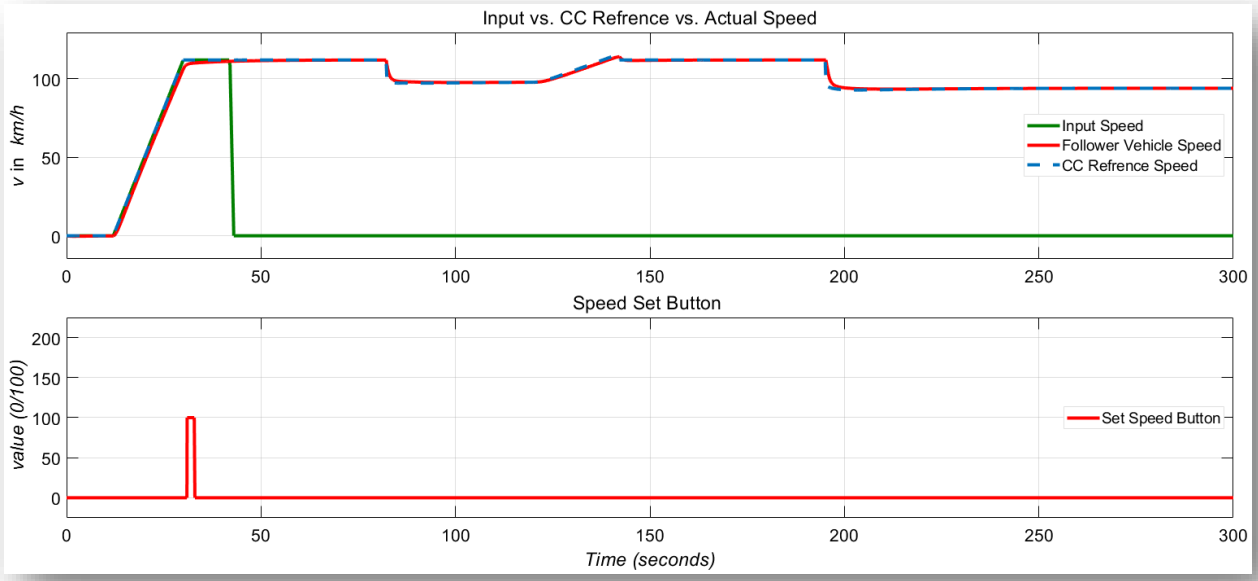

Fig. 3-25: Set Speed Button Pressed

It can be observed that when the button is pressed, the speed is set to a constant value and the vehicle continues to follow the set speed (when in speed tracking mode), when the driver has released the throttle (input speed is zero).

Fig. 3-26 shows the behavior of the vehicle when set speed button is not pressed. The vehicle follows the input speed and gradually comes to zero as the input speed is zero. The negative speed value occurs due to the separation of break and throttle provided by the **ACC Break/Throttle Separation** block. But the settle at 0 for the rest of the simulation.

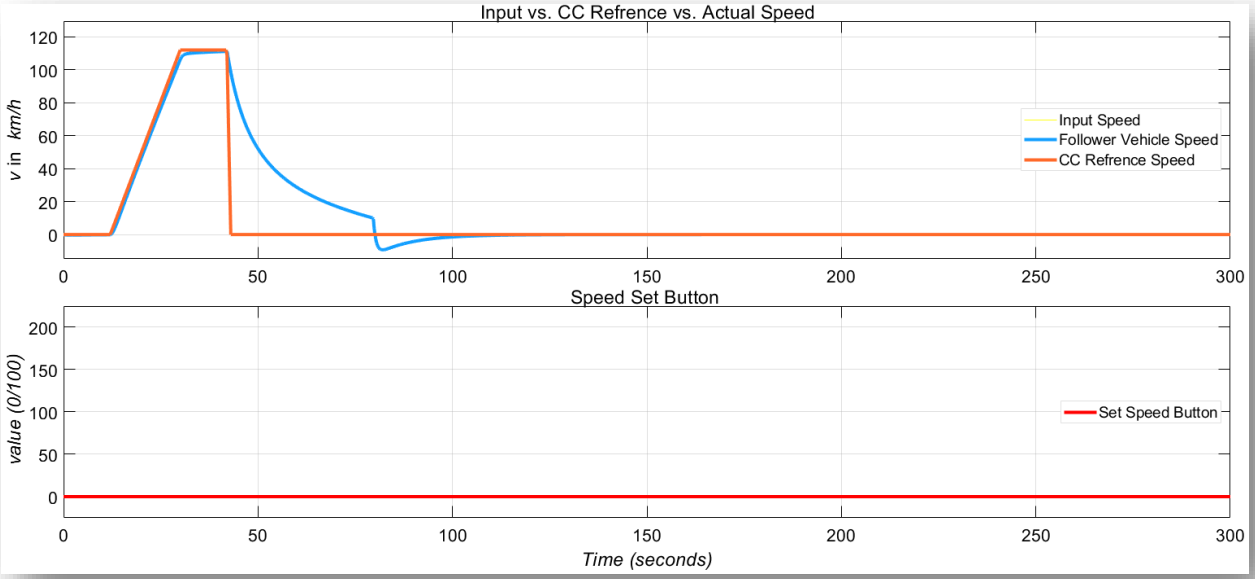

Fig. 3-26: Speed Set Button Not Pressed

3.5.7 Conclusion

After performing the above-mentioned tests, it was established that the Adaptive cruise control model was working according to the requirements mentioned in **section 3.1.5.**