Autonomous Vehicles under Adverse Conditions

Tristan Carter

Ms. Mabrooka Chaudhry

Atholton High School

Abstract

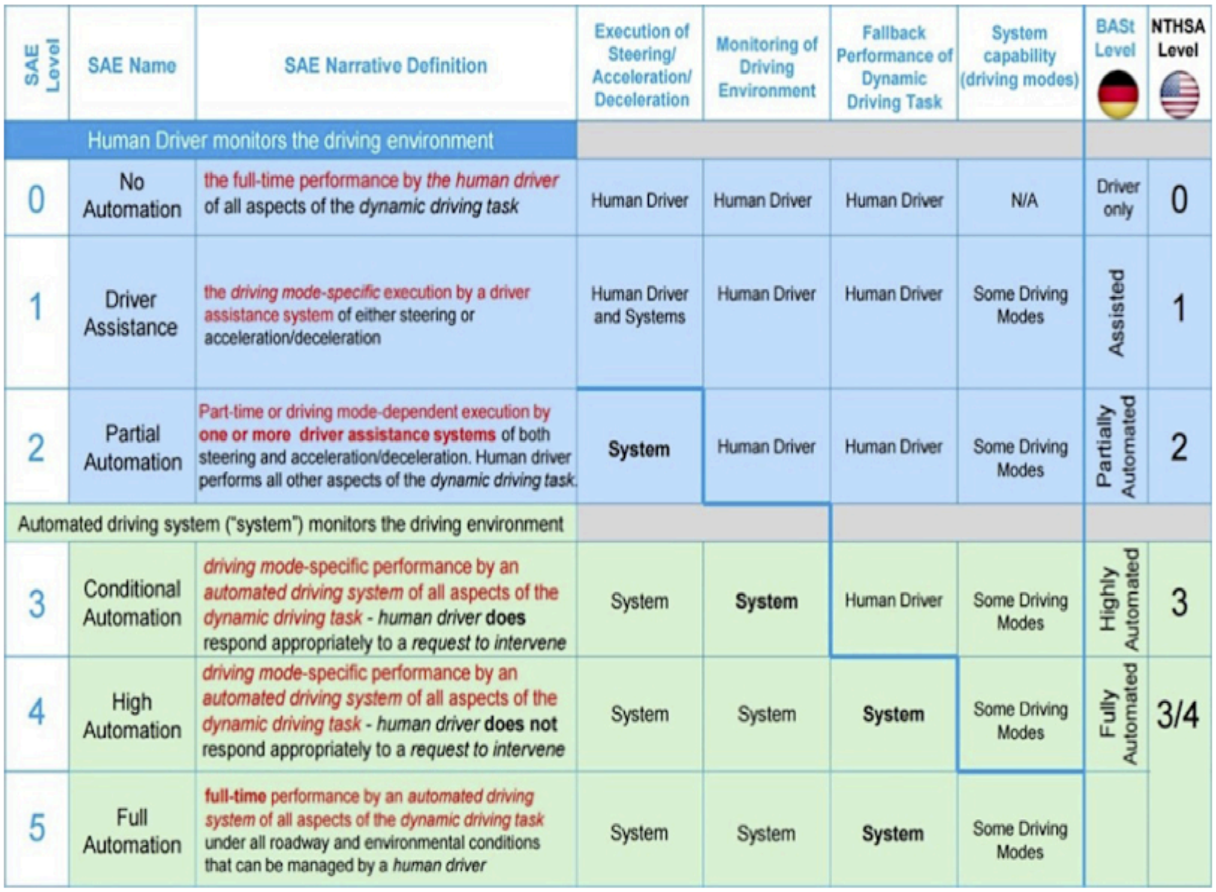
This review evaluates the adaptive cruise control performance of a simulated autonomous vehicle using radar measurements with Gaussian noise. The autonomous vehicle, using the parameters of a 2024 Mazda CX-90, uses dynamical state equations that are simulated using independently-designed Sliding Mode control laws to reach the commanded car velocity and then slow down to maintain a safe defined distance behind a slower forward vehicle. Unique low-pass digital filters are applied to improve the cruise control performance using radar measurements with Gaussian noise or to maintain the desired speed/relative distance when there is a delay since the last radar measurement. The experimental methodology applied unique control laws and digital filters with optimized parameters to demonstrate adaptive cruise control using nominal and worst-case sensor measurements. The simulated vehicle's resistance against these degraded sensor measurements shows the effectiveness of using measurement filtering algorithms.

Keywords: Adaptive Cruise Control, Gaussian Noise, Sliding Mode Control Law, Low-Pass Digital Filter

Introduction

In 2014, SAE (Society of Automotive Engineers) initially launched the SAE Levels of Driving Automation, which has since become the industry's most-cited source for driving automation. SAE has defined six levels of driving automation, that range from Level 0 (no driving automation) to Level 5 (full driving automation)(SAE International, 2021).

Autonomous vehicles implement the autonomous functions safely using a variety of sensors. Common autonomous vehicle sensor types include RADAR (Radio Detection and Ranging), LiDAR (Light Detection and Ranging), ultrasonic, forward/rear-view cameras, and a global navigation satellite receiver. The most common function adaptive vehicles have adopted is adaptive cruise control. This control system utilizes such sensors to measure the distance to preceding vehicles and controls the throttle and brakes to maintain a desired spacing. This provides enhanced comfort for drivers while also potentially improving safety (Rudolph, Veolzke, 2017). There are two main modes of operation for an adaptive cruise control system: speed control and vehicle following. The vehicle must control when it switches from speed control to vehicle following mode because it must anticipate safely how it may interact with the car directly in front of it. In vehicle following mode, the system maintains the desired spacing from the preceding vehicle.



SAE Level	SAE Name	SAE Narrative Definition	Execution of Steering/ Acceleration/ Deceleration	Monitoring of Driving Environment	Fallback Performance of Dynamic Driving Task	System capability (driving modes)	BASt Level	NTHSA Level
	Human Driver monitors the driving environment							
0	No Automation	the full-time performance by *the human driver* of all aspects of the *dynamic driving task*	Human Driver	Human Driver	Human Driver	N/A	Driver only	0
1	Driver Assistance	the *driving mode-specific* execution by a driver assistance system of either steering or acceleration/deceleration	Human Driver and Systems	Human Driver	Human Driver	Some Driving Modes	Assisted	1
2	Partial Automation	Part-time or driving mode-dependent execution by **one or more driver assistance systems** of both steering and acceleration/deceleration. Human driver performs all other aspects of the *dynamic driving task*.	**System**	Human Driver	Human Driver	Some Driving Modes	Partially Automated	2
Automated driving system ("system") monitors the driving environment								
3	Conditional Automation	*driving mode-specific* performance by an *automated driving system* of all aspects of the *dynamic driving task* - *human driver* **does** respond appropriately to a *request to intervene*	System	**System**	Human Driver	Some Driving Modes	Highly Automated	3
4	High Automation	*driving mode-specific* performance by an *automated driving system* of all aspects of the *dynamic driving task* - *human driver* **does not** respond appropriately to a *request to intervene*	System	System	**System**	Some Driving Modes	Fully Automated	3/4
5	Full Automation	**full-time** performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	**System**	Some Driving Modes		

Table 1. SAE Levels of Driving Automation

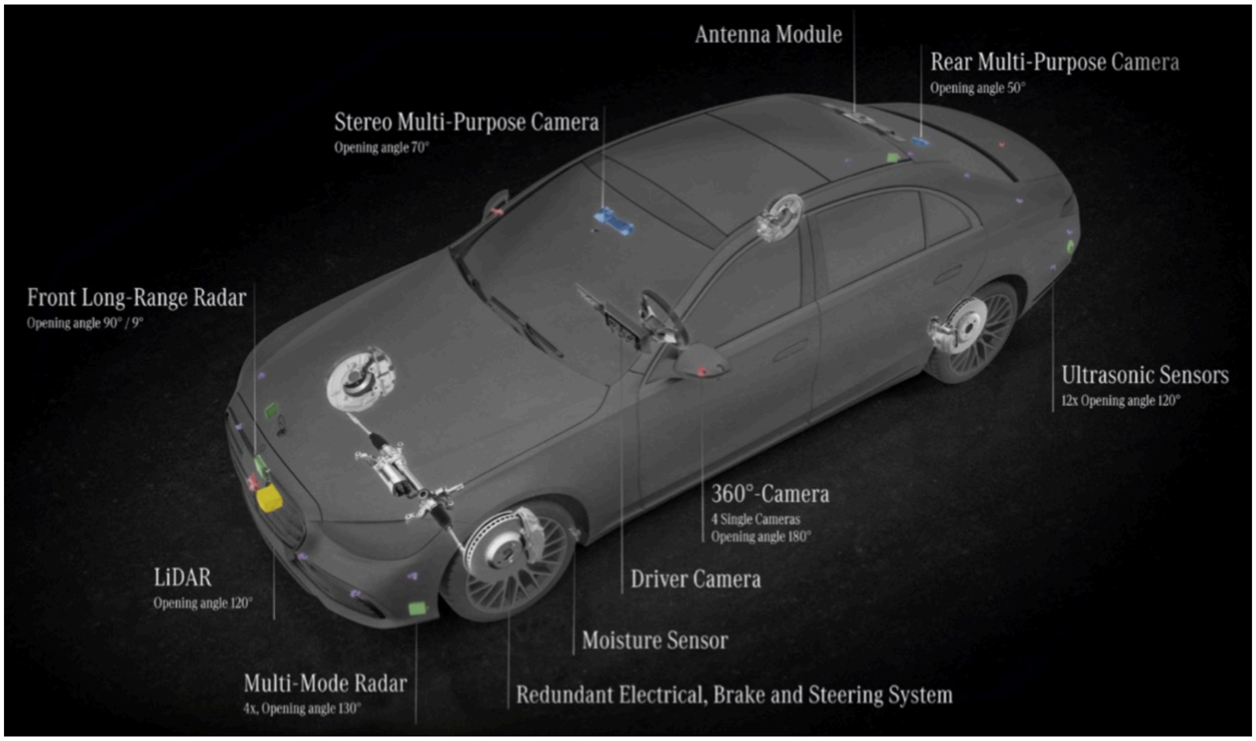


Figure 1. Sensors for the Mercedes-Benz Drive Pilot

These sensors collect measurements about the vehicle and its surrounding environment, but weather conditions like rain, fog, and snow significantly degrade sensor data through noise and missing information. For example, nearly 400 semi-autonomous car crashes were reported to National Highway Traffic Safety Administration over 11 months from July 2021 to May 2022, with a majority of the crashes being reported from Tesla vehicles (The Associated Press, 2022). Approximately 46% of weather-related accidents are caused by rain, and approximately ~17% are caused by snow (Vargas, Alsweiss, Toker, Razdan, Santos, 2021). The weather can affect both the car sensor measurements and road conditions that degrade the autonomous function performance causing these accidents and leading to passenger safety risks in rare cases.

To counteract these degradations, autonomous vehicles can utilize sensor measurement filtering algorithms. These will improve the performance of autonomous vehicle functions, based on measurement stochastic noise or missing measurement data during typical driving scenarios. Filtering measurements have been used in many control system problems to remove noise and extrapolate from past measurements for data gaps; the application of these techniques can be applied to the autonomous vehicle control modes in failure scenarios (Kachroo, Tomizuka, 1994).

Overview of Autonomous Vehicles Functions and Systems

According to the SAE international, an Autonomous Vehicle is defined as “the use of mechanics, electronics, artificial intelligence, and multi-agent computing systems to assist in the operation of a vehicle, allowing the vehicle to sense and interact with its environment with various degrees of human involvement” (SAE International, 2021).

The SAE organization has defined the six levels from no automation to full automation respectively: No driving Automation, Driver Assistance, Partial Driving Automation, Conditional Driving Automation, High Driving Automation, and Full Driving Automation. At level 0, functionalities include automated emergency braking, blind spot warning, and lane departure warning. At level 1, functionalities include lane centering or adaptive cruise control. At level 2, new functionalities include lane centering and cruise control. At level 3, new functionalities include traffic jam chauffeur. At level 4, the functionality is as a local driverless taxi. At level 5, the functionality is as a driverless chauffeur for passenger vehicles. So far as of 2024, the most common driver support systems are at level 2, and some state-of-the-art are at level 3. Examples of level 2 driver support systems include: Tesla Autopilot with “Full Self-Driving", BMW Extended Traffic Jam Assistant, Audi Traffic Jam Assist, GM Super Cruise, Ford Blue Cruise, and Mazda Traffic Jam Assist. Examples of certified level 3 driver support systems so far are: Mercedes Benz L3 Drive Pilot which has been certified in Germany (May 2022), Nevada (January 2023), and California (June 2023), and the BMW Personal Pilot L3, which has been certified only in Germany (November 2023) (MBUSA Newsroom, 2023).

Although there are various types of sensors as mentioned in the introduction, the most useful in terms of tracking the vehicle ahead is RADAR with the most accurate data on its speed and distance. Out of all the sensors, it has the best range of around 250 meters, detects speed and distance the best, and is small. In addition, it can operate “in varied weather conditions due to a wide spectrum wavelength and absence of mechanical parts, as opposed to passive visual sensors due to their lack of scope for climate range” (Vargas, Alsweiss, Toker, Razdan, Santos, 2021). However, this is not to claim that it can operate perfectly under weather conditions and other types of interference.

There exist two types of RADAR sensors: Short Range RADAR (SRR), Medium Range RADAR (MRR), and Long Range RADAR (LRR). SRRs can detect and determine the range of objects using radio waves (at 24 GHz frequency). It can effectively be used for blind spot detection, lane change assistant, park assistance, and cross-traffic monitoring. MRRs and LRRs are used to detect and determine the range of objects using radio waves (at 77 GHz frequency). These can effectively be used for brake assistance, emergency braking, and automatic distance control (Vargas, Alsweiss, Toker, Razdan, Santos, 2021).

In order to properly design a control system for an advanced vehicle system, a vehicle’s linear and angular control should be realized, and be able to be designed to fit certain requirements like ride quality or passenger comfort. An important factor of vehicle mobility is traction force. Traction force is the amount of force a vehicle exerts to travel on a tangential surface. Traction force is dependent on the friction coefficient between the road and the tire. Surfaces range from being completely wet to completely dry, with wet surfaces having a large amount of wheel slip and a low coefficient of friction and dry surfaces having low amount of wheel slip and a large coefficient of friction. It is possible to influence traction force by varying the wheel slip. As angular wheel velocity is directly measured, only vehicle velocity is needed to estimate wheel slip. By calculating the variety of variables (such as engine torque, braking torque, and adhesion coefficient), we can model the system, and design a good controller. By controlling the wheel slip, we control the tractive force to obtain the desired output, namely wheel and vehicle velocities, from the system. Thus, it is a good idea that to control tractive force effectively we should have our vehicle dynamic equations in terms of wheel slip.

To model a proper cruise control scenario, there are two important specifications for the vehicle following control system: individual vehicle stability and string stability. Individual vehicle stability means the spacing error of the adaptive cruise control vehicle converges to zero when the preceding vehicle is at a constant speed, and is non-zero when the preceding vehicle is decelerating or accelerating. String stability means spacing errors do not amplify as they propagate through a string of adaptive cruise control vehicles. This ensures large spacing errors do not develop further back in the vehicle string (Rajamani 2006).

A plausible explanation for the sensor's degradation would be weather conditions. In-depth exploration can be done to explore each focus of weather conditions. However, all weather conditions here will be generalized to being treated as Gaussian noise. This type of signal disturbance causes a randomized increment/decrement that is independently identically distributed from some normal or some Gaussian distribution.

In an interview with Dr. Michael Carter, a control systems specialist and an engineer from the Naval Research Laboratory, the discussion of this issue brought insight to some of these details. To handle the missing measurements in autonomous vehicles due to this issue, a generalized solution would be to extrapolate missing data using past slopes. He suggested that a digital signal processing approach would be a very strong way in countering bad data, and that with a good filter design, the attenuation of the frequencies above the cutoff frequency can be achieved. So, if the environment is adverse due to road conditions or weather, the Gaussian noise will become larger but a good filter will still be able to remove the gaussian noise.

Algorithms and Filters to Resolve Degradation of Measurement Data

Using a Sliding Mode Controller (SMC), a Chebyshev Type II filter, and the Runge-Kutta Order Four (RK4) process together provides a comprehensive and robust approach to controlling the dynamics and filtering the data of an autonomous vehicle.

Using the RK4 method offers several advantages for accurately and efficiently solving the differential equations that describe the autonomous vehicle's linear dynamics. RK4 is a fourth-order numerical integration method. This high accuracy makes RK4 particularly suitable for scenarios where precise control is crucial, such as in autonomous vehicles. By providing an accurate approximation of the vehicle's state over time, RK4 helps in maintaining precise control over the vehicle's trajectory and dynamics.

In control systems, especially those involving real-time applications like autonomous vehicles, stability and reliability are paramount. RK4 is known for its stability properties when dealing with linear dynamics. This stability ensures that the control system remains robust against small deviations and modeling inaccuracies, providing consistent performance over time.

Autonomous vehicles often operate in environments where the required computational effort can vary. RK4 is robust to changes in step size, allowing for adaptive step size control. This adaptability can help in managing computational resources more effectively, ensuring real-time performance without compromising accuracy (Cheever 2019).

The Chebyshev Type II filter is a very popular type of low-pass filter in digital signal processing. A low-pass filter is a filter that passes signals with a frequency lower than a selected cutoff frequency and attenuates signals with frequencies higher than the cutoff frequency. The Chebyshev Type II filter is especially characterized by its equiripple behavior in the stopband, which means it provides a very sharp cutoff outside the passband. In a equiripple filter, the ripple behavior is equalized in both the passband and stopband, and the error between the desired and actual frequency response is minimized. This sharp cutoff is particularly useful in filtering out high-frequency noise and other unwanted components that often contaminate RADAR data. By effectively attenuating these undesirable frequencies, the filter ensures that the data used for decision-making processes in the vehicle's control system is clean and accurate (Digital Signal Processing: A Practical Guide for Engineers and Scientists, 2003)

Additionally, Chebyshev Type II filters are designed to have no ripples in the passband, ensuring a smooth and flat response within the frequency range of interest. This property is crucial for maintaining the integrity of the RADAR signal’s essential components, which contain critical information about the vehicle’s surroundings. A clean and undistorted signal helps in accurately identifying and tracking objects, enhancing the reliability of the vehicle’s perception system. In the interview with Dr. Michael Carter, he claimed that this filter was a “very good simple low-pass filter”. It allows for almost a perfect pass-band in the low frequencies below the cutoff frequency that preserves the ideal measurement without noise. Even if the attenuation above the cutoff frequency is high enough, he claims that it is still a good filter because if the filter provides enough attenuation, the ripple does not matter. Low-pass filters are good about having a small phase-lag/time-delay in the output.

For a SMC system, the slip dynamic equation for deceleration is presented, and a switching surface is defined based on the sliding variable, with the objective of achieving a desired slip. As a nonlinear function, Dr. Michael Carter supports the idea of the usage of the SMC due to its guarantee that the system “converges to the desired state regardless of initial conditions”. The nonlinear function is estimated, and the control gain is bounded. The controller design involves a finite time to reach the switching surface, ensuring stability with exponential convergence. Practical challenges such as chattering, caused by non-ideal switching, are addressed by introducing a region around the switching surface. A boundary layer is defined to avoid chattering, and the control input is modified accordingly.

Sliding mode control is a robust control strategy well-suited for systems facing uncertainties, disturbances, and nonlinearities. Its strength lies in its ability to handle parametric uncertainties without destabilizing the control performance, making it less sensitive to modeling inaccuracies. This control system is particularly effective in rejecting disturbances, offering adaptability to changing system conditions, and facilitating real-time implementation. While chattering can be a concern, strategies like introducing a boundary layer help mitigate this issue. The robustness of this control method allows it to adapt to varying road surfaces, weather conditions, and uncertainties in the autonomous vehicle's dynamics. Additionally, this control method's capability to reject disturbances is crucial for maintaining control integrity when faced with unexpected events on the road. This control approach effectively handles anti-lock braking and traction control that will help the autonomous vehicle simulation perform as best as possible in how it interacts with the environment around it (Unsal, Kachroo, 1999)

Together, these three components create a synergistic system where the SMC ensures robust and stable control, the Chebyshev Type II filter delivers clean and reliable sensor data, and the RK4 process ensures precise numerical integration, resulting in optimal performance and safety of the autonomous vehicle.

Autonomous Vehicle Cruise Control Scenario through Experimentation

Vehicle Equations of Motion

The vehicle equations of motion are derived from the wheel angular momentum time derivative and the vehicle linear momentum time derivative.

■ Wheel Angular Momentum Time Derivative

The wheel angular momentum time derivative is the product of the wheel inertia, I_w, and the wheel angular acceleration, $\dot{\omega}$, and is equal to the sum of all the wheel torques, *T*:

$$I_w\dot{\omega} = \sum T \tag{1}$$

Figure 2 shows that the wheel torques include the:

- Engine torque at each wheel, T_e
- Braking torque at each wheel, T_b
- Torque from Tire Traction Force, $r_w \bullet F_{tr}$
- Torque from Wheel Rolling Resistance Frictional Force, $r_w \bullet F_{rr}$

where r_w is the wheel radius. These wheel torques can then be substituted into equation (1):

$$I_w\dot{\omega} = T_e - T_b - r_w \bullet \left(F_{tr} + F_{rr}\right) \tag{2}$$

The control torque, T_{con}, for the autonomous vehicle simulations in this paper is the sum of the engine torque and braking torque:

$$T_{con} = T_e - T_b \tag{3}$$

The tire rolling resistance frictional force, F_{rr}, is the product of the rolling resistance coefficient, μ_{rr}, and the normal force at each wheel, N_w:

$$F_{rr} = \mu_{rr} N_w = \mu_{rr} \frac{m_v}{n_w} g \tag{4}$$

The wheel normal force is the product of the vehicle mass, m_v, and gravitational acceleration, g, divided by the number of wheels, n_w. The rolling resistance coefficient is a function of tire pressure, p, in units of pounds per square inch (psi), and vehicle velocity, v, in units of miles per hour (mph):

$$\mu_{rr} = 0.005 + \frac{1}{p}\left(0.145 + 3.567e - 5\, v^2\right) \tag{5}$$

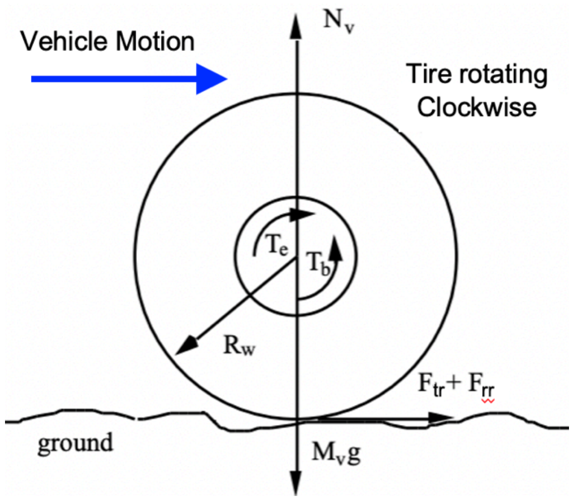


Figure 2. External Torque on each Wheel of the Vehicle

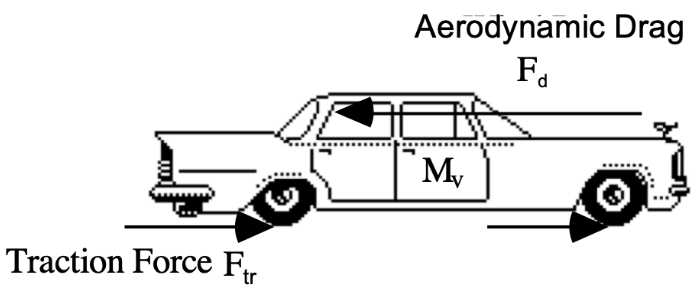


Figure 3. External Forces on the Vehicle

■ Vehicle Linear Momentum Time Derivative

The vehicle linear momentum time derivative is the product of the vehicle mass, m_v, and the vehicle acceleration, a, and is equal to the sum of all the external forces, F:

$$m_v a = \sum F \tag{6}$$

Figure 3 shows that the vehicle external forces include the:

- Traction Force for Acceleration/Braking at each wheel, T_e
- Aerodynamic Drag Force, F_d

where n_w is the number of wheels on the vehicle. These vehicle forces can then be substituted into equation (6):

$$m_v a = n_w F_{tr} - F_d \tag{7}$$

The traction force at each wheel is the product of the adhesion coefficient, μ_{ad}, and the normal force at each wheel, N_w:

$$F_{tr} = \mu_{ad} N_w = \mu_{ad} \frac{m_v}{n_w} g \tag{8}$$

The wheel normal force is the product of the vehicle mass, m_v, and gravitational acceleration, g, divided by the number of wheels, n_w. The adhesion coefficient is a function of the peak frictional coefficient based on road conditions, μ_{pk}, the wheel slip, λ, and the peak wheel slip, λ_{pk}:

$$\mu_{ad} = \mu_{pk} \frac{2\frac{\lambda}{\lambda_{pk}}}{1+\left(\frac{\lambda}{\lambda_{pk}}\right)^2} = \mu_{pk}\alpha_\lambda \quad (9)$$

α_λ is the peak frictional coefficient attenuation factor with a value between 0 and 1. The wheel slip is the normalized difference between the wheel angular velocity, ω_w, and the no-slip wheel angular velocity (the vehicle velocity, v, divided by the wheel radius, r_w):

$$\lambda = \frac{\omega_w - \frac{v}{r_w}}{max\left(\omega_w, \frac{v}{r_w}\right)} \quad (10)$$

This can also be written as:

$$\lambda = [1 - \frac{v}{r_w \omega_w},\ if\ \ v \leq r_w \omega_w\ \ \frac{r_w \omega_w}{v} - 1,\ if\ \ v > r_w \omega_w$$

(11)

Table 2. Peak Frictional Coefficients for Dry and Wet Road Surfaces

	Peak Frictional Coefficient	
Road Surface	**Dry**	**Wet**
Concrete	0.8-0.9	0.8
Asphalt	0.8-0.9	0.5-0.6
Earth road	0.68	0.55
Gravel	0.60	
Snow (hard-packed)		0.2
Ice		0.1

Table 2 provides the peak frictional coefficient, μ_{pk}, based on the road conditions (dry and wet

peak frictional coefficients of different road surfaces.

For the autonomous vehicle simulations in this paper, the peak wheel slip and peak frictional coefficients captured in Table 3 were used for the different dry and wet road surfaces. These peak wheel slips and peak frictional coefficient were used to plot the peak frictional coefficient attenuation factor, α_{λ}, and the adhesion coefficient, μ_{ad}, as a function of wheel slip in Figure 4 and Figure 5 respectively.

The aerodynamic drag force is the product of air density, ρ_{air}, vehicle velocity, v, the drag coefficient, C_d, and the vehicle cross-sectional area, A_v:

$$F_d = \frac{1}{2}\rho_{air} v^2 C_d A_v \tag{12}$$

■ Wheel Angular Acceleration and Vehicle Acceleration

The wheel angular acceleration can be written as a function of the control torque and adhesion and rolling resistance coefficients by substituting equations (3), (4) and (8) into equation (2):

$$\dot{\omega} = \frac{1}{I_w} T_{con} - \frac{r_w}{I_w} \frac{m_v g}{n_w} \left(\mu_{ad} + \mu_{rr} \right)$$

(13)

Substituting equations (5) and (9) for the above coefficients, the wheel angular acceleration can be expanded to:

$$\dot{\omega} = \frac{1}{I_w} T_{con} - \frac{r_w}{I_w} \frac{m_v g}{n_w} \left(\mu_{pk} \frac{2\frac{\lambda}{\lambda_{pk}}}{1+\left(\frac{\lambda}{\lambda_{pk}}\right)^2} + 0.005 + \frac{1}{p}\left(0.145 + 3.567e - 5\, v^2\right) \right) \quad (14)$$

The vehicle acceleration can be written as a function of the time-varying adhesion coefficient and vehicle velocity by substituting

$$a = \mu_{ad}\, g - \frac{1}{2} \frac{\rho_{air}}{m_v} C_d A_v v^2 \quad (15)$$

Substituting equation (9) for the adhesion coefficient, the wheel angular acceleration can be expanded to:

$$a = \frac{2\frac{\lambda}{\lambda_{pk}}}{1+\left(\frac{\lambda}{\lambda_{pk}}\right)^2} \mu_{pk}\, g - \frac{1}{2} \frac{\rho_{air}}{m_v} C_d A_v v^2 \quad (16)$$

Table 3. Peak Wheel Slip and Peak Frictional Coefficients for Dry and Wet Road Surfaces used for Autonomous Vehicle Simulations

	lambda_pk	mu_pk
Dry Pavement	0.17	0.92
Wet Asphalt	0.25	0.82
Unpacked Snow	0.11	0.30
Ice	0.24	0.165

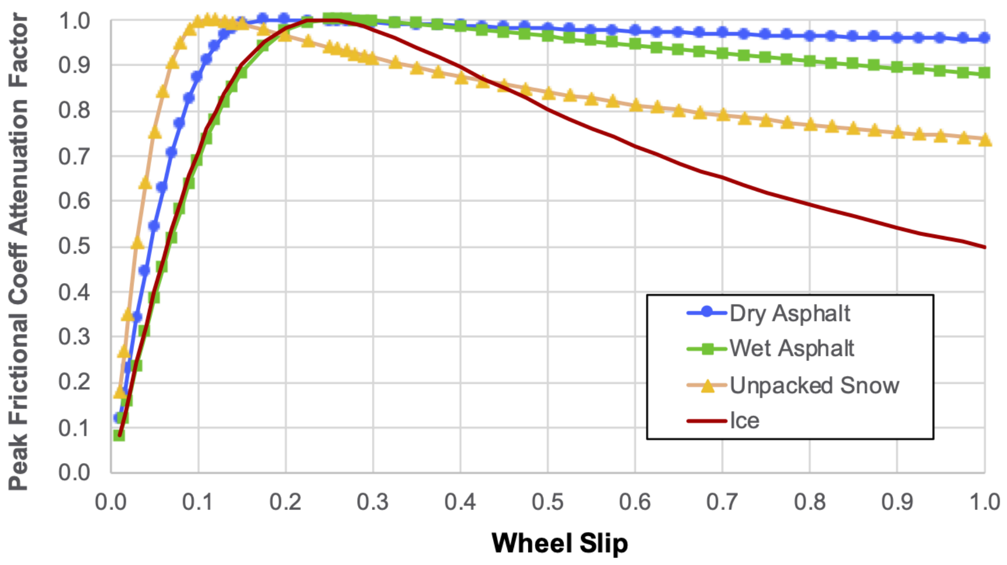


Figure 4. Peak Frictional Coefficient Attenuation Factor as a function of Wheel Slip

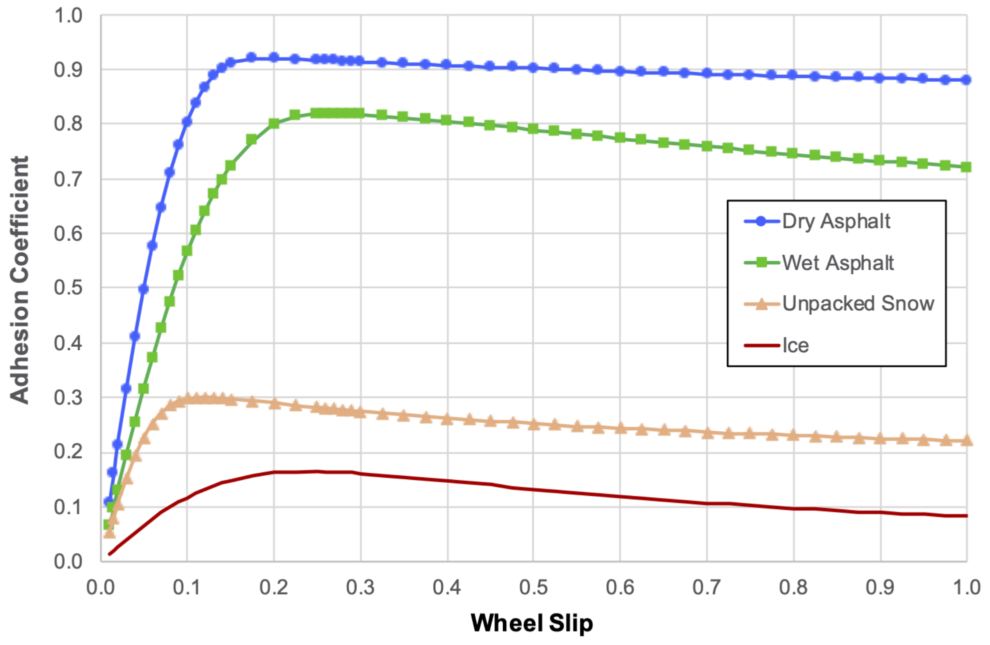


Figure 5. Adhesion Coefficient as a function of Wheel Slip

■ State Equations of Motion

The vehicle dynamical equations are rewritten as first-order differential equations of time-varying states in the form required for numerical algorithms (such as Runga-Kutta methods) to integrate over time. The states for vehicle motion include the vehicle position, the vehicle velocity and the wheel angular velocity respectively:

$$x_1 = x \, x_2 = \dot{x} = v \, x_3 = \omega \tag{17}$$

The state time derivatives for vehicle motion include the vehicle velocity, the vehicle acceleration from equation (16) and the wheel angular acceleration from equation (14):

$$\dot{x}_1 = v = x_2 \, \dot{x}_2 = a = A_{21}\alpha_\lambda - A_{22}v^2 \, \dot{x}_3 = \dot{\omega} =- A_{31}\alpha_\lambda - A_{32}v^2 - d_3 + B\,T_{con} \tag{18}$$

where the vehicle acceleration state parameters are given by:

$$A_{21} = \mu_{pk}g \tag{19}$$

$$A_{22} = \frac{1}{2}\frac{\rho_{air}}{m_v}C_d A_v \tag{20}$$

The wheel angular acceleration state parameters are given by:

$$A_{31} = \frac{r_w}{I_w}\frac{m_v}{n_w}\mu_{pk}g \tag{21}$$

$$A_{32} = \frac{3.567e-5}{p}\frac{r_w}{I_w}\frac{m_v}{n_w}g \tag{22}$$

and the wheel angular acceleration disturbance parameter is given by:

$$d_3 = \left(0.005 + \frac{0.145}{p}\right)\frac{r_w}{I_w}\frac{m_v}{n_w}g \tag{23}$$

while the wheel angular acceleration control parameter is given by:

$$B = \frac{1}{I_w} \tag{24}$$

Adaptive Cruise Control of an Autonomous Vehicle

Cruise control for a vehicle will accelerate or brake to reach and then maintain the commanded vehicle velocity. The adaptive cruise control for an autonomous vehicle will use the radar measurements to determine the distance to a slower vehicle in front of the autonomous vehicle and then match the speed of the slower vehicle at a commanded safe distance.

The control torque for cruise control was derived from the state equations of motion using a sliding mode controller:

$$T_{con_v} = \frac{1}{B}\left\{(A_{31} - k_4 A_{21})\alpha_\lambda + (A_{32} + k_4 A_{22}){v_1}^2 + d_3\right\} + \frac{1}{B}\left\{-k_3 \cdot k_4\left(v_1 - v_{1,cmd}\right) + k_4\left(\omega_w - \omega_{w,cmd1}\right)\right\}$$

(25)

where $v_{1,cmd}$ is the commanded vehicle velocity and $\omega_{w,cmd1}$ is the commanded wheel angular velocity that can be derived from the commanded vehicle velocity.

The control torque to maintain a safe driving distance behind a slow car was also derived from the state equations of motion using a sliding mode controller:

$$T_{con\_d} = \frac{1}{B}\left\{(A_{31} - k_4 A_{21})\alpha_\lambda + (A_{32} + k_4 A_{22}){v_1}^2 + d_3\right\} + \frac{1}{B}\left\{-k_5 \cdot k_6\left(r_1 - r_{1,cmd}\right) + k_6\left(\omega_w - \omega_{w,cmd2}\right)\right\}$$

(26)

where $r_{1,cmd}$ is the commanded vehicle safe relative distance behind the car in front and $\omega_{w,cmd2}$ is the commanded wheel angular velocity that can be derived from the estimated forward vehicle velocity.

The control torque for autonomous cruise control is a weighted average of the control torques to maintain the commanded vehicle velocity in equation (25) and to maintain the vehicle safe relative distance in equation (26) respectively:

$$T_{con} = \begin{bmatrix} T_{con\_v}\text{ , if outside Radar control range, } r_{ctrl}\text{, of 15 m (49.2 ft)} \\ (1-\alpha)T_{con_d+} + \alpha\, T_{con\_v}\text{, if within Radar control range} \end{bmatrix}$$

(27)

The control torque for autonomous cruise control reverts to the traditional cruise control torque, if there is no vehicle directly in front of the autonomous vehicle in the radar control range (of 15 m, or 49.2 ft, in the following simulation runs). If the relative distance to the forward vehicle is within the radar control range, the autonomous control torque will be a linear interpolation between the commanded vehicle velocity control torque and the safe relative distance to the forward vehicle control torque, based on the interpolation parameter:

$$\alpha = \frac{r_1 - r_{1,cmd}}{r_{ctrl}},\ \ \alpha \in (\frac{-r_{1,cmd}}{r_{ctrl}}, +\ 1] \qquad (28)$$

where r_1 is the radar relative distance measurement to the forward vehicle, $r_{1,cmd}$ is the commanded safe relative distance, and r_{ctrl} is the defined radar control range.

The state time derivatives for vehicle motion include the vehicle velocity, the vehicle acceleration from equation (16) and the wheel angular acceleration from equation (14).

Adaptive Cruise Control of an Autonomous Vehicle

The adaptive cruise control for a 2024 Mazda CX-90 was simulated for 30 sec by integrating the state time derivatives in equation (18) using the Runge Kutta (4,4) algorithm with a time step of 0.125 sec. The state equation parameters for this simulation are given by:

$$A_{21} = 29.6\ \frac{ft}{s}$$

$$A_{22} = 5.59e - 5\ \frac{1}{ft}$$

$$A_{31} = 1459.327\ \frac{1}{s^2}$$

$$A_{32} = 0.0189\ \frac{1}{s^2}$$

$$d_3 = 14.3201\ \frac{1}{lbm-ft^2}$$

$$B = 0.0225\frac{1}{s^2}$$

The adaptive cruise control for a 2024 Mazda CX-90 is activated when the car is stopped, and commanded to a target speed of 50 mph (73.3 ft/s). A second vehicle is 200 ft ahead of the Mazda CX-90 at the initial time, and is traveling at a constant speed of 40 mph (58.7 mph).

The adaptive cruise control will brake and adjust the Mazda car velocity to maintain a safe distance of 6 m (19.7 ft) behind the slower forward vehicle. The radar sensors for this simulation will acquire ranging distance measurements when the forward vehicle is within a 50 m (164.0 ft) range, and will adjust the control torque to maintain a safe distance when the forward vehicle is within 15 m (49.2 ft).

Based on the state equation parameters, the initial conditions, and the commanded final state, the following non-dimensional control gains were selected to optimize the autonomous cruise control performance:

$$k_3 = 1.05$$

$$k_4 = 0.6$$

$$k_5 = 0.38$$

$$k_6 = 1.04$$

The safe driving error remains ≤ 1.5 ft at 28.0 sec

Figure 6 shows how the Mazda CX-9 reaches the target 50 mph speed using the designed Sliding Mode adaptive cruise controller operating at 8 Hz in 5.5 sec. The adaptive cruise controller starts to transition to a Sliding Mode safe distance controller to remain 6 m behind the forward vehicle at 16.5 sec, when the safe distance error is less than 49.2 ft.

Figure 7 shows the Mazda CX-9 control torque is maintained at 64,000 to 68,800 foot-pounds (ft-lb) for the first 2.25 sec to accelerate from rest to > 42 mph. The control torque decreases from 18,000 to 6000 ft-lb during the 2.5 to 6.0 sec period. The control torque is fairly stable (5797 to 6000 ft-lb) from 6.0 to 16.5 sec, until the slow forward vehicle is within the Mazda CX-9 radar control distance range of 15 m. The Mazda CX-9 then brakes (negative control torque) from 17.125 to 20.0 sec. The Mazda CX-9 will slowly increase the control torque to a stable range of 3525 to 3915 ft-lb from 25 to 30 sec to maintain the 6 m relative distance with the forward car traveling at 40 mph.

Figure 8 shows the Mazda CX-9 commanded velocity error transitions smoothly to zero and maintains the commanded cruise control velocity from 5.5 to 16.5 sec, until the slow forward vehicle is within the Mazda CX-9 radar control distance range of 15 m. Similarly, the commanded wheel angular velocity error is maintained at zero from 5.5 to 16.5 sec, as shown in Figure 9.

Figure 10 shows the phase plot with the wheel angular velocity plotted with the vehicle commanded velocity error over 30 sec. The adaptive cruise control transitions from the initial error position (-73.3 ft/sec Vehicle velocity error, -57.25 rad/sec Wheel angular velocity error) to the origin of this two-dimension plot and remains there for the first 16.5 sec. However, due to the slower moving forward vehicle, the final error position on this phase plot is -14.6 ft/sec Vehicle velocity error and -11.4 rad/sec Wheel angular velocity error.

Figure 11 shows the Mazda CX-9 commanded safe relative distance error transitions smoothly to zero and maintains the commanded safe distance of 6 meters from 25 to 30 sec. Similarly, the commanded wheel angular velocity error is maintained at zero from 25.0 to 30.0 sec, as shown in Figure 12. Both of these plots only occur measurements after the slow forward vehicle is within the Mazda CX-9 radar control distance range of 15 m.

Figure 13 shows the phase plot with the wheel angular velocity plotted with the safe relative distance error from 16.5 to 30 sec. The adaptive cruise control transition from the initial error position (48.3 ft Safe relative distance error, 11.47 rad/sec Wheel Angular velocity error) to the origin of this two-dimension plot in this 13.5 sec period. The final error position on this phase plot is 1.3 ft Safe relative distance error and 0.09 rad/sec Wheel angular velocity error.

Figure 15 shows a Python 3 program made to complement this experiment. The program uses the “numpy” and “scipy” libraries to assist with making the vectors and designing the filters. The same processes described above are used in this program. This detailed simulation is vital for testing and validating the control strategies, ensuring the vehicle maintains desired performance details such as safe distance, velocity control, and response to dynamic changes. Conditions can be fine-tuned efficiently to achieve desired safety and performance. Inputs were taken from a file with the name “Independent_File.in.txt”. The inputs include the different gains for the control systems, the initial conditions, the command conditions, and the experiment conditions (e.g. the number of time steps).

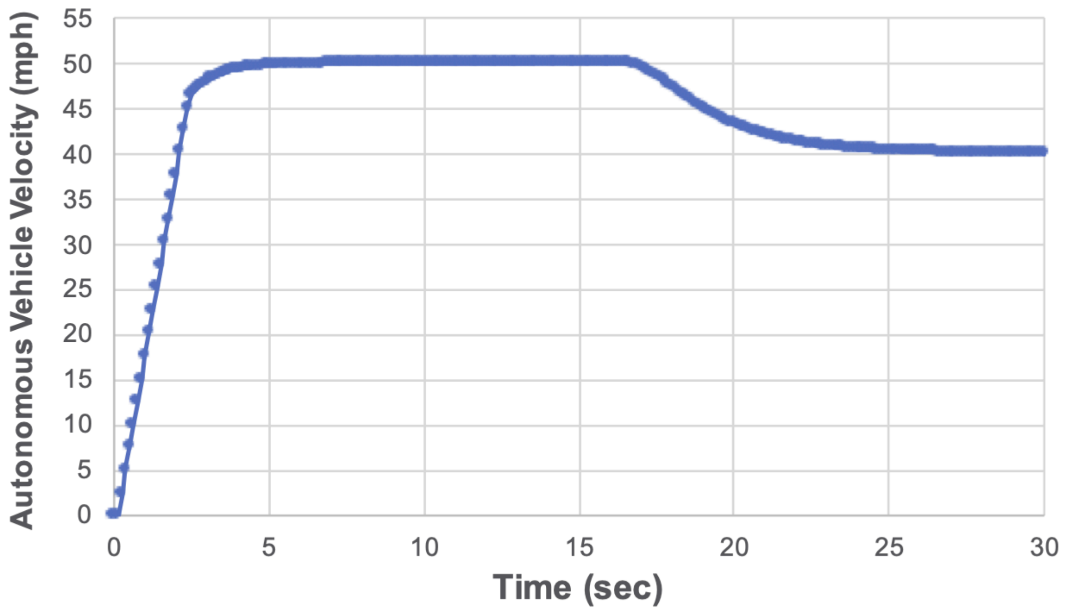


Figure 6. Mazda CX-9 Velocity over 30 sec using Adaptive Cruise Control

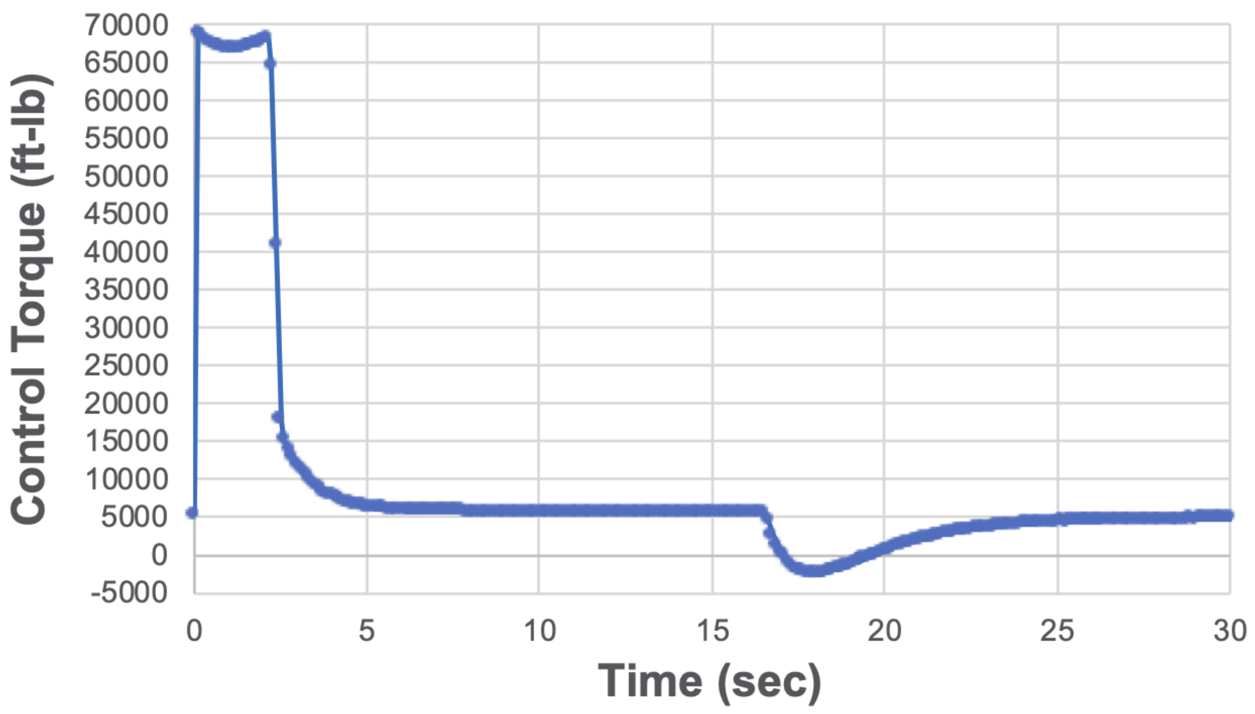


Figure 7. Mazda CX-9 Control Torque over 30 sec using Adaptive Cruise Control

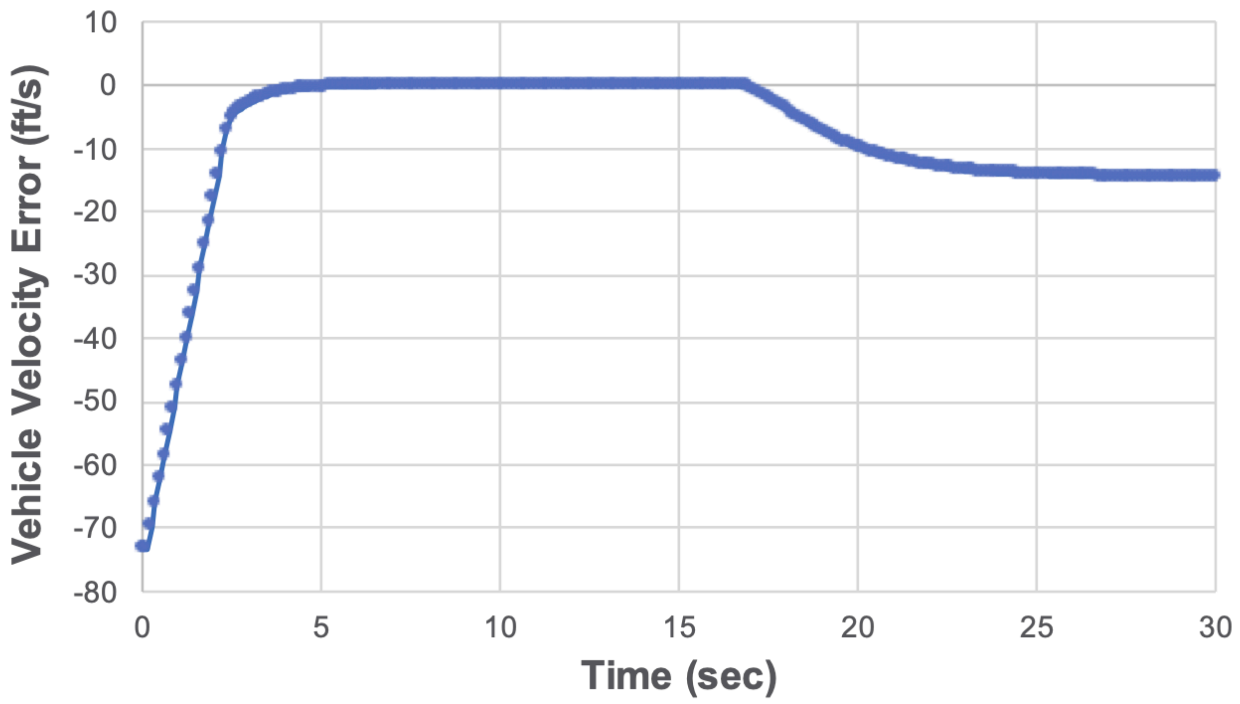


Figure 8. Mazda CX-9 Commanded Velocity Error over 30 sec using Adaptive Cruise Control

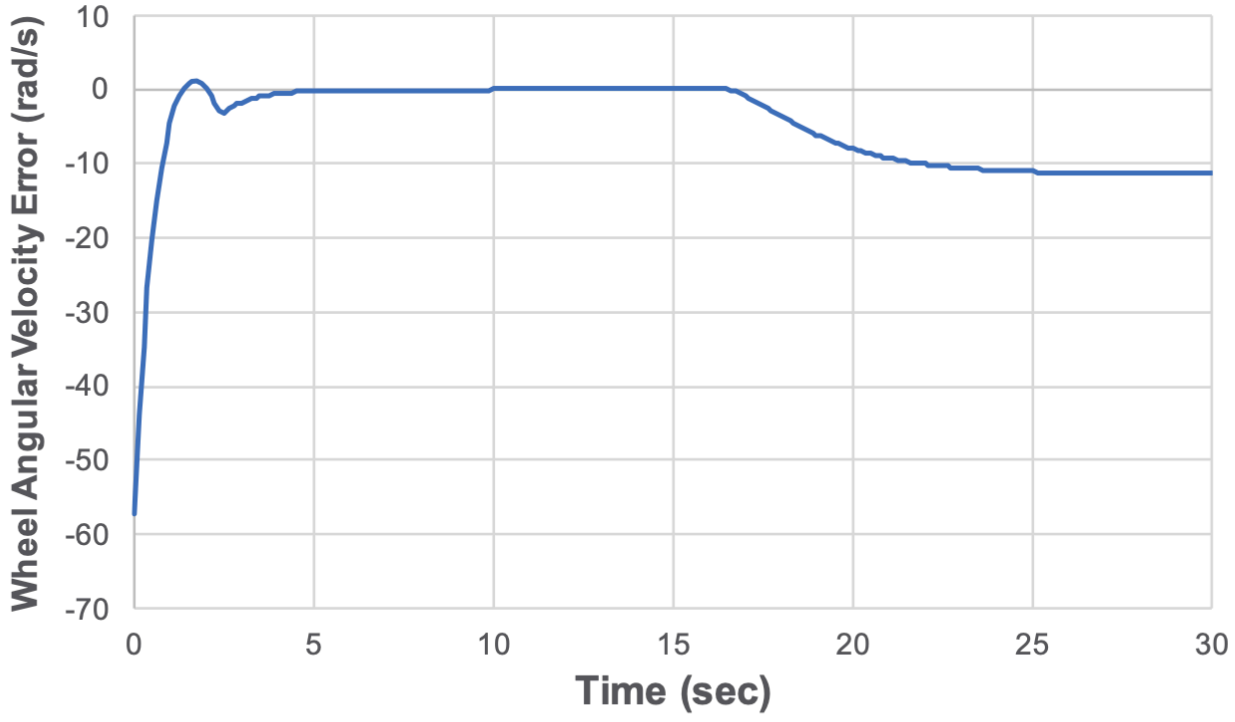


Figure 9. Mazda CX-9 Wheel Angular Velocity Error corresponding to Commanded Velocity over 30 sec using Adaptive Cruise Control

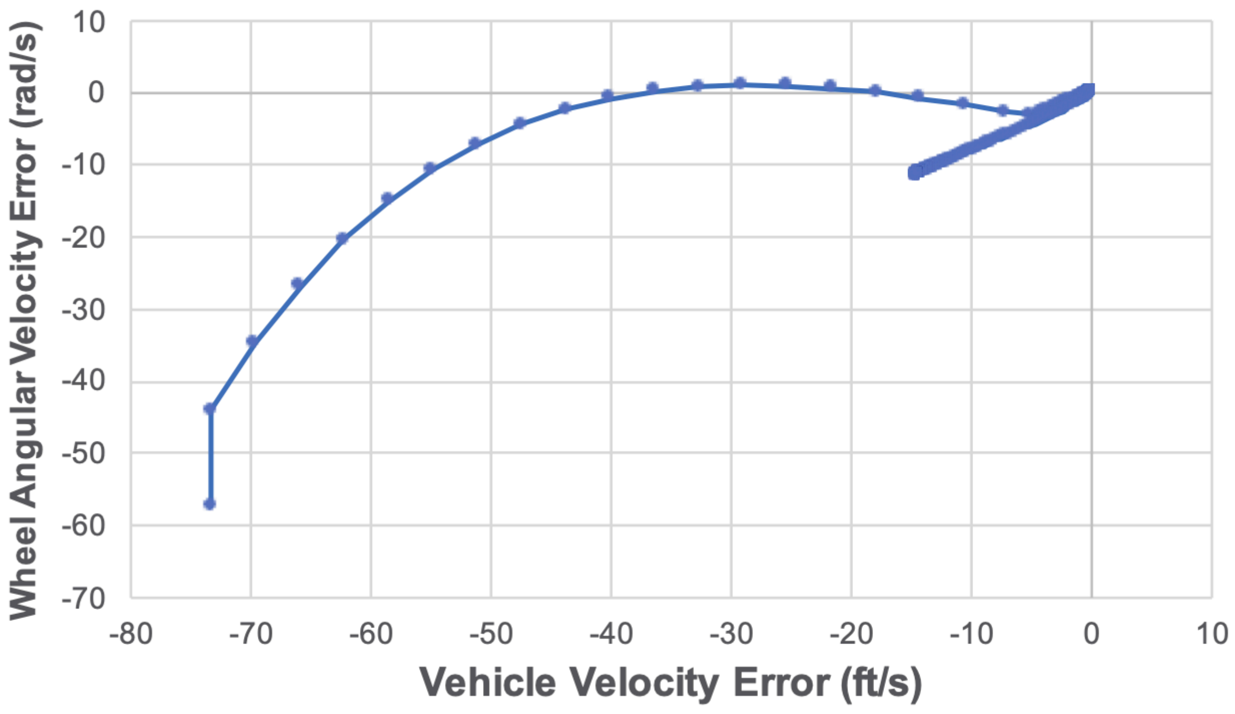


Figure 10. Mazda CX-9 Wheel Angular Velocity Error to Vehicle Commanded Velocity Error Phase Plot over 30 sec using Adaptive Cruise Control

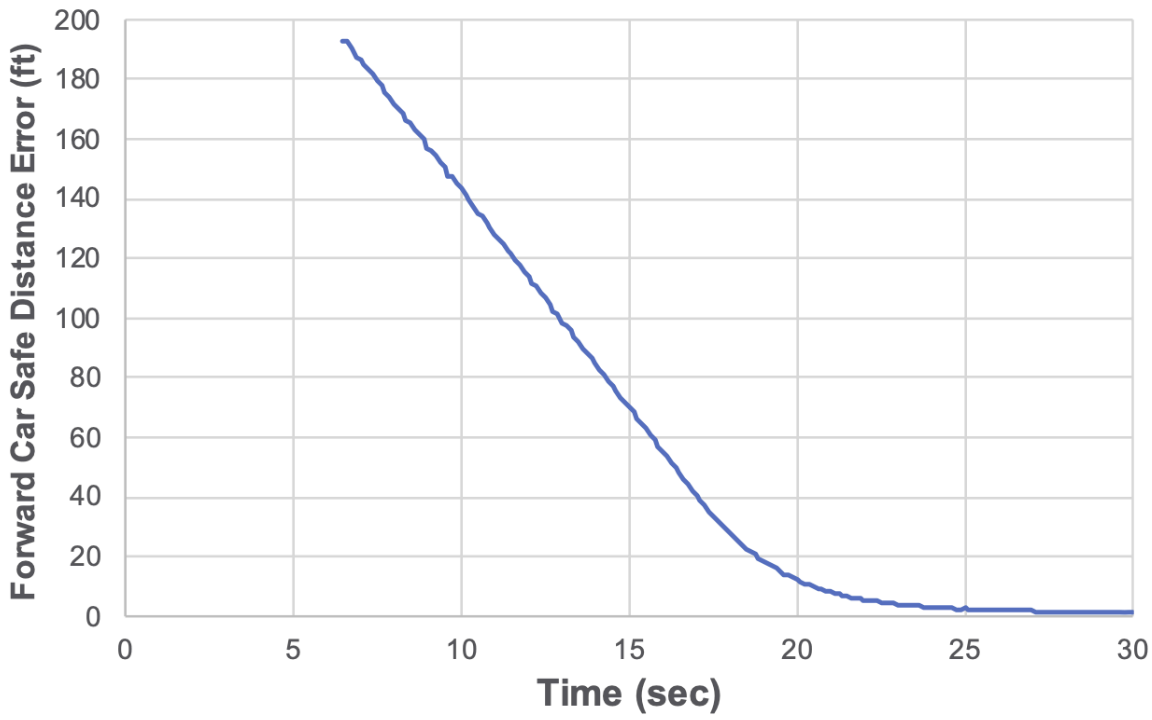


Figure 11. Mazda CX-9 Safe Distance Error over 30 sec using Adaptive Cruise Control

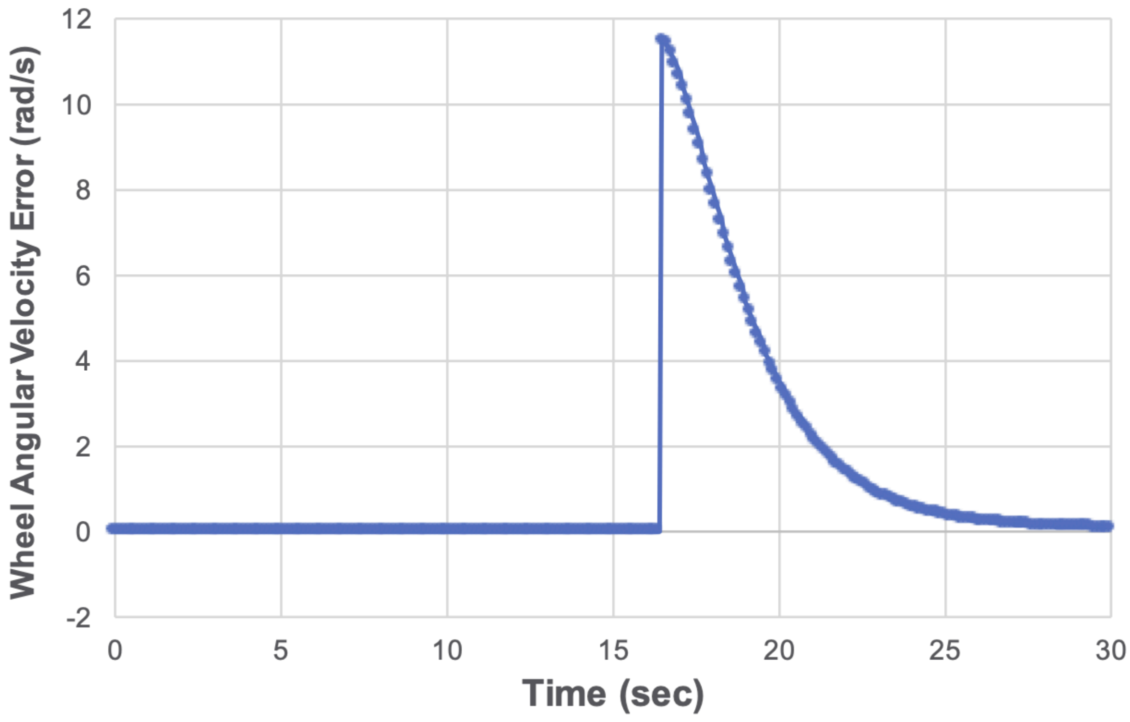


Figure 12. Mazda CX-9 Wheel Angular Velocity Error corresponding to Safe Relative Distance Error over 30 sec using Adaptive Cruise Control

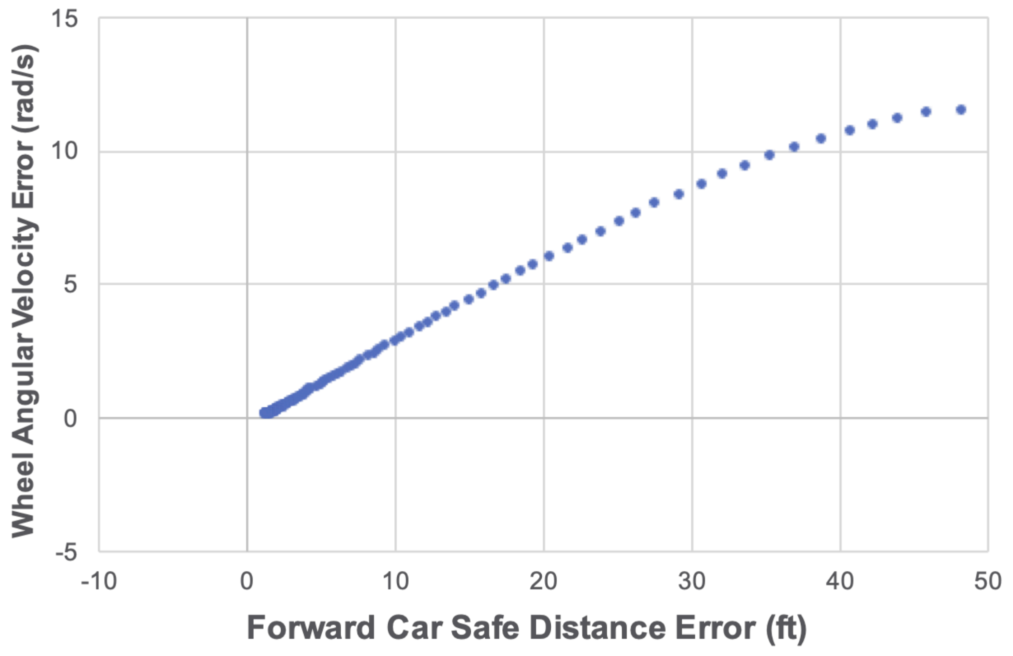
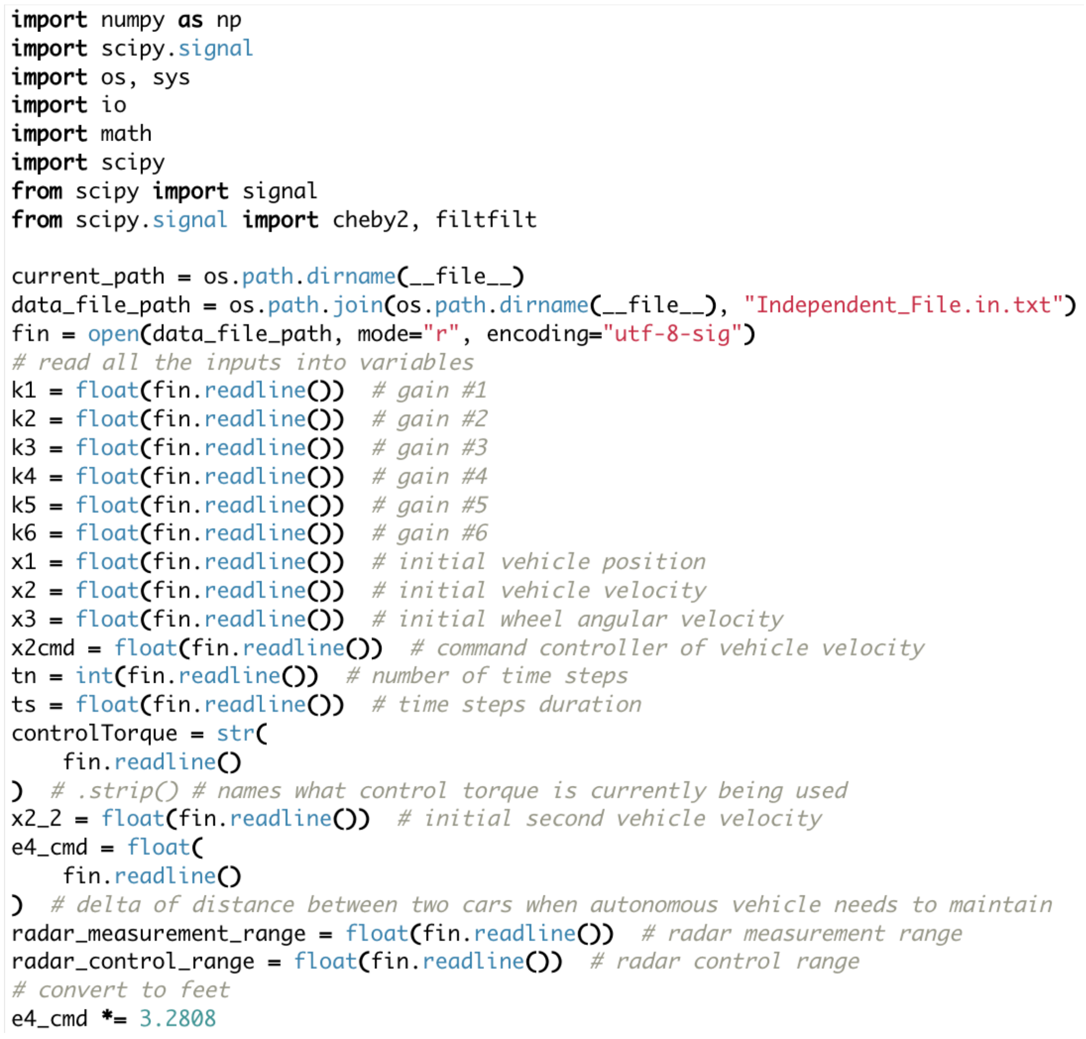
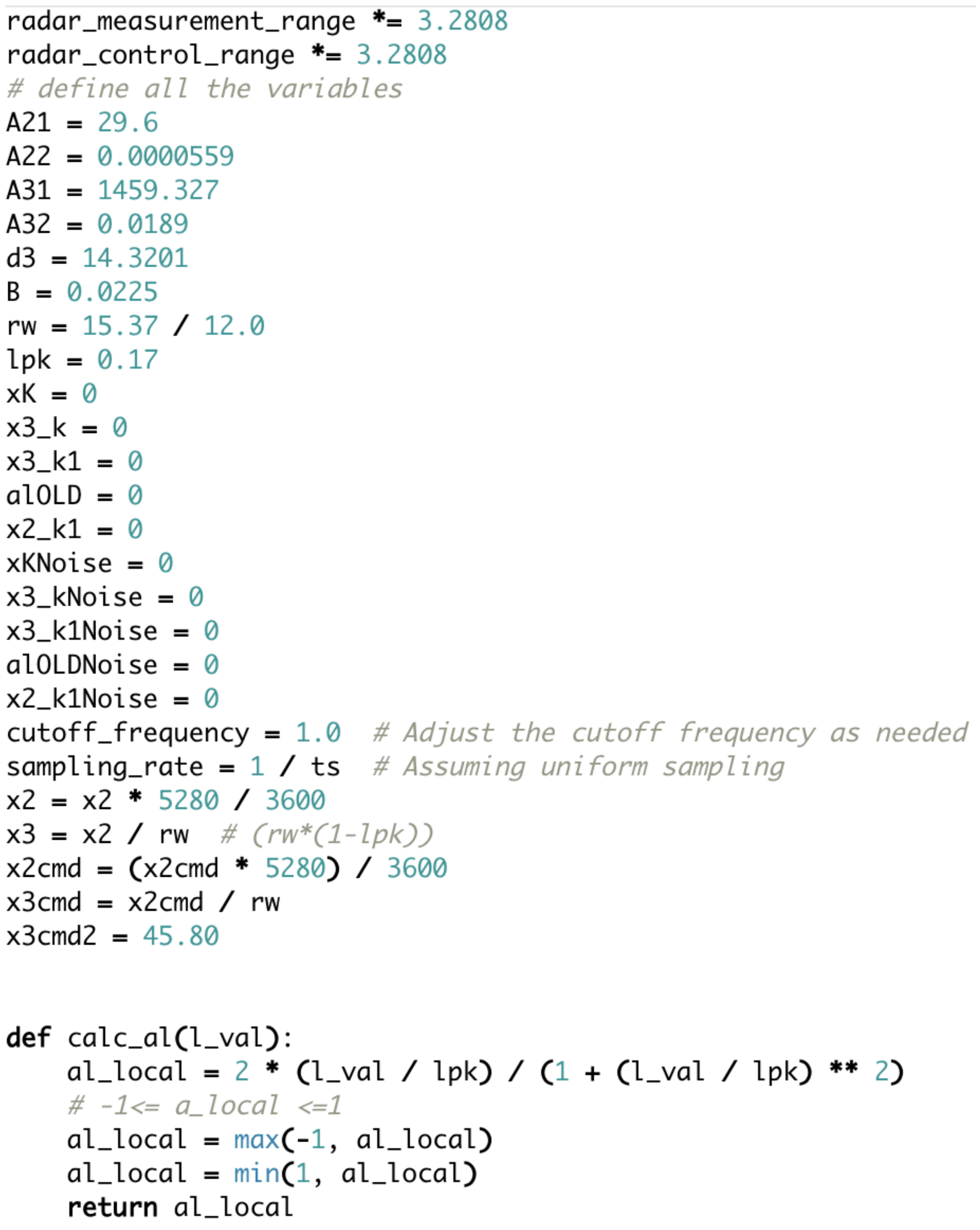


Figure 13. Mazda CX-9 Wheel Angular Velocity Error to Safe Relative Distance Error Phase Plot over 30 sec using Adaptive Cruise Control

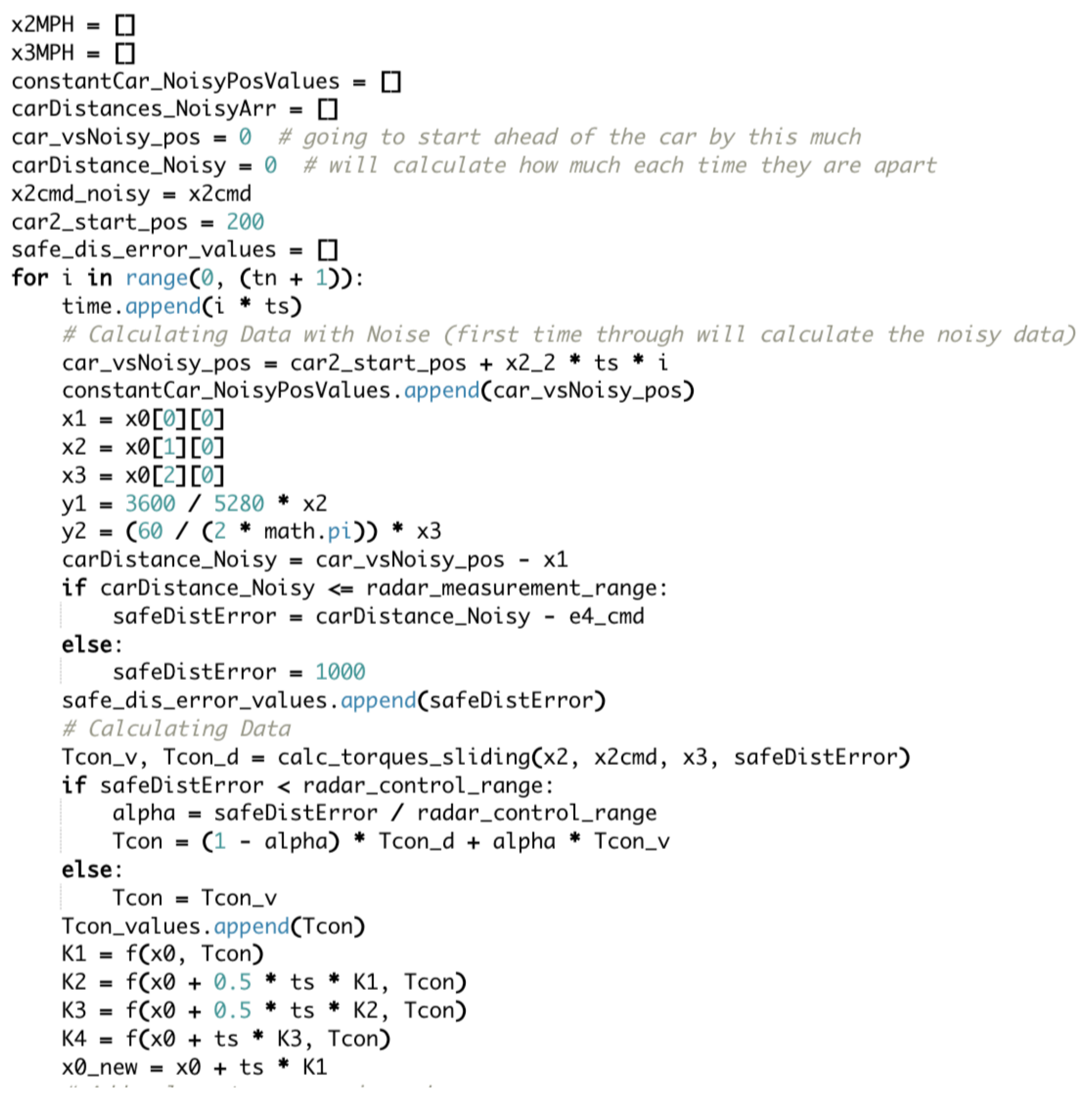


```
import numpy as np
import scipy.signal
import os, sys
import io
import math
import scipy
from scipy import signal
from scipy.signal import cheby2, filtfilt

current_path = os.path.dirname(__file__)
data_file_path = os.path.join(os.path.dirname(__file__), "Independent_File.in.txt")
fin = open(data_file_path, mode="r", encoding="utf-8-sig")
# read all the inputs into variables
k1 = float(fin.readline())  # gain #1
k2 = float(fin.readline())  # gain #2
k3 = float(fin.readline())  # gain #3
k4 = float(fin.readline())  # gain #4
k5 = float(fin.readline())  # gain #5
k6 = float(fin.readline())  # gain #6
x1 = float(fin.readline())  # initial vehicle position
x2 = float(fin.readline())  # initial vehicle velocity
x3 = float(fin.readline())  # initial wheel angular velocity
x2cmd = float(fin.readline())  # command controller of vehicle velocity
tn = int(fin.readline())  # number of time steps
ts = float(fin.readline())  # time steps duration
controlTorque = str(
    fin.readline()
)  # .strip() # names what control torque is currently being used
x2_2 = float(fin.readline())  # initial second vehicle velocity
e4_cmd = float(
    fin.readline()
)  # delta of distance between two cars when autonomous vehicle needs to maintain
radar_measurement_range = float(fin.readline())  # radar measurement range
radar_control_range = float(fin.readline())  # radar control range
# convert to feet
e4_cmd *= 3.2808
```

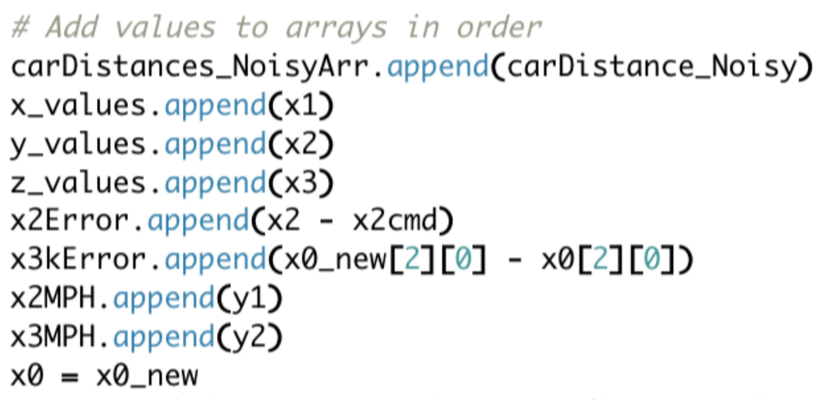


```
radar_measurement_range *= 3.2808
radar_control_range *= 3.2808
# define all the variables
A21 = 29.6
A22 = 0.0000559
A31 = 1459.327
A32 = 0.0189
d3 = 14.3201
B = 0.0225
rw = 15.37 / 12.0
lpk = 0.17
xK = 0
x3_k = 0
x3_k1 = 0
alOLD = 0
x2_k1 = 0
xKNoise = 0
x3_kNoise = 0
x3_k1Noise = 0
alOLDNoise = 0
x2_k1Noise = 0
cutoff_frequency = 1.0  # Adjust the cutoff frequency as needed
sampling_rate = 1 / ts  # Assuming uniform sampling
x2 = x2 * 5280 / 3600
x3 = x2 / rw  # (rw*(1-lpk))
x2cmd = (x2cmd * 5280) / 3600
x3cmd = x2cmd / rw
x3cmd2 = 45.80


def calc_al(l_val):
    al_local = 2 * (l_val / lpk) / (1 + (l_val / lpk) ** 2)
    # -1<= a_local <=1
    al_local = max(-1, al_local)
    al_local = min(1, al_local)
    return al_local
```

```
def calc_lamda(v, w):
    l_local = 0
    if v <= rw * w:
        if w != 0:
            l_local = 1 - v / (rw * w)
    else:
        if v != 0:
            l_local = (rw * w) / v - 1
    # -lpk<=l <=lpk
    l_local = min(l_local, lpk)
    l_local = max(l_local, -lpk)
    return l_local


def f(x, TconCurrent):
    l_x = calc_lamda(x[1][0], x[2][0])
    al_x = calc_al(l_x)

    x1dot = x[1][0]
    x2dot = A21 * al_x - A22 * (x[1][0] ** 2)
    x3dot = (-A31 * al_x) - A32 * (x[1][0] ** 2) - d3 + B * TconCurrent
    return np.array([[x1dot], [x2dot], [x3dot]])


def add_gaussian_noise(signal, mean=0, std=10):
    noise = np.random.normal(mean, std, signal.shape)
    noisy_signal = signal + noise
    return noisy_signal


# Function to apply a higher-order low-pass filter
def high_order_low_pass_filter(
    signal, cutoff_frequency, sampling_rate, order, ripple=1, stop_atten=30
):
    nyquist = 0.5 * sampling_rate
    normal_cutoff = cutoff_frequency / nyquist
    # Design the Chebychev Type II Filter using scipy
    b, a = cheby2(order, stop_atten, cutoff_frequency, "low", fs=sampling_rate)
    # Apply the filter
    filtered_signal = filtfilt(b, a, signal)
    return filtered_signal


def calc_torques_sliding(x2, x2_cmd, x3, safe_dis_error):
    lamda = calc_lamda(x2, x3)
    alpha_lamda = calc_al(lamda)
    Tcon1 = 1 / B * ((A31 - k4 * A21) * alpha_lamda + (A32 + k4 * A22) * (x2**2)
        + d3)
    Tcon2 = 1 / B * (-(k3 * k4) * (x2 - x2_cmd) - k3 * (x3 - x3cmd))
    Tcon_v = Tcon1 + Tcon2
    if safe_dis_error != None:
        Tcon3 = 1 / B * (-(k5 * k6) * safe_dis_error - k6 * (x3 - x3cmd2))
        Tcon_d = Tcon1 + Tcon3
    else:
        Tcon_d = None
    return Tcon_v, Tcon_d


x3Old = x3  # x3 one time step before
Tcon = 0.0
TconNoise = 0.0  # temporary control torque value
TconFiltered = 0.0
# defining the gaussian noise arrays that will affect the data
gaussianNoiseVelocity = np.random.normal(0, 5, tn)
gaussianNoiseAngularVelocity = np.random.normal(0, 5, tn)
x_values = []
y_values = []
z_values = []
Tcon_values = []
TconNoise_values = []
TconFiltered_values = []
x0 = [[x1], [x2], [x3]]
x0Noise = [[x1], [x2], [x3]]
x0Filtered = [[x1], [x2], [x3]]
time = []
x2Error = []
x3kError = []
```



```
x2MPH = []
x3MPH = []
constantCar_NoisyPosValues = []
carDistances_NoisyArr = []
car_vsNoisy_pos = 0  # going to start ahead of the car by this much
carDistance_Noisy = 0  # will calculate how much each time they are apart
x2cmd_noisy = x2cmd
car2_start_pos = 200
safe_dis_error_values = []
for i in range(0, (tn + 1)):
    time.append(i * ts)
    # Calculating Data with Noise (first time through will calculate the noisy data)
    car_vsNoisy_pos = car2_start_pos + x2_2 * ts * i
    constantCar_NoisyPosValues.append(car_vsNoisy_pos)
    x1 = x0[0][0]
    x2 = x0[1][0]
    x3 = x0[2][0]
    y1 = 3600 / 5280 * x2
    y2 = (60 / (2 * math.pi)) * x3
    carDistance_Noisy = car_vsNoisy_pos - x1
    if carDistance_Noisy <= radar_measurement_range:
        safeDistError = carDistance_Noisy - e4_cmd
    else:
        safeDistError = 1000
    safe_dis_error_values.append(safeDistError)
    # Calculating Data
    Tcon_v, Tcon_d = calc_torques_sliding(x2, x2cmd, x3, safeDistError)
    if safeDistError < radar_control_range:
        alpha = safeDistError / radar_control_range
        Tcon = (1 - alpha) * Tcon_d + alpha * Tcon_v
    else:
        Tcon = Tcon_v
    Tcon_values.append(Tcon)
    K1 = f(x0, Tcon)
    K2 = f(x0 + 0.5 * ts * K1, Tcon)
    K3 = f(x0 + 0.5 * ts * K2, Tcon)
    K4 = f(x0 + ts * K3, Tcon)
    x0_new = x0 + ts * K1
```



```
# Add values to arrays in order
carDistances_NoisyArr.append(carDistance_Noisy)
x_values.append(x1)
y_values.append(x2)
z_values.append(x3)
x2Error.append(x2 - x2cmd)
x3kError.append(x0_new[2][0] - x0[2][0])
x2MPH.append(y1)
x3MPH.append(y2)
x0 = x0_new
```

Figure 15. Python 3 Program Demonstrating the Implementation of Simulated Experiment

Conclusion

In conclusion, this review demonstrates the effective application of adaptive cruise control in a simulated autonomous vehicle using radar measurements subject to Gaussian noise. By employing the parameters of a 2024 Mazda CX-90 and utilizing advanced control techniques, such as Sliding Mode Control laws and Chebyshev Type II filters, the vehicle successfully maintains desired speeds and safe distances from preceding vehicles while filtering out degraded data. The integration of these unique control laws and digital filters showcases the robustness of the system against degraded sensor measurements, thereby affirming the importance of measurement filtering algorithms in enhancing autonomous vehicle performance.

The application of Neural Networks can be further explored to enhance the adaptive cruise control system. Neural Networks are a popular form of approximate learning solutions. The learning process of most learning strategies starts off by creating randomized weights and biases, and then going back through multiple loops, or epochs, to adjust these weights and biases with different learning strategies in order to minimize the error. In order to properly correct the weights and biases, most learning strategies calculate the gradient, a derivative for a multi-dimensional function, of a loss function to measure how far off the model's predictions were from the actual values of the dataset. The iterative process of updating the weights and biases, called backpropagating, is repeated until the error function converges to a specified value (Abdallaoui, Ikaouassen, Kribèche, Chaibet, Aglzim, 2023). By training on large datasets of driving scenarios, neural networks learn to predict the vehicle's behavior in response to different control inputs. This allows the autonomous system to anticipate how the vehicle will move and adjust its controls to maintain stability and comfort. Also, their adaptability to new situations is crucial for handling unpredictable events such as sudden stops by other vehicles, pedestrians entering the road, or changes in road surface conditions. Neural networks can continuously learn and update their models based on new data, improving their performance over time.

References

Abdallaoui, S., Halima Ikaouassen, Kribèche, A., Chaibet, A., & El‐Hassane Aglzim. (2023). Advancing autonomous vehicle control systems: An in‐depth overview of decision‐making and manoeuvre execution state of the art. *The Journal of Engineering*, *2023*(11). https://doi.org/10.1049/tje2.12333

Associated Press. (2022). Nearly 400 car crashes in 11 months involved automated tech, companies tell regulators. *NPR*. https://www.npr.org/2022/06/15/1105252793/nearly-400-car-crashes-in-11-months-involved-automated-tech-companies-tell-regul.

AUTOCRYPT. The State of Level 3 Autonomous Driving in 2023: Ready for the Mass Market?, Autocrypt, 2023. https://autocrypt.io/the-state-of-level-3-autonomous-driving-in-2023.

Automated and unmanned vehicles. SAE International. https://www.sae.org/what-is-automated-and-unmanned

Automated driving revolution: Mercedes-Benz announces U.S. availability of DRIVE PILOT – the world's first certified SAE Level 3 system for the U.S. market. (2023, September 28). MBUSA Newsroom. https://media.mbusa.com/releases/automated-driving-revolution-mercedes-benz-announces-us-availability-of-drive-pilot-the-worlds-first-certified-sae-level-3-system-for-the-us-market.

Burden, Richard L., Fairies, J. Douglas. Numerical Analysis, 5th ed., PWS Publishing Company, Boston, 1993,pp.130-142.

Cheever, E. (2019). *Fourth Order Runge-Kutta*. Swarthmore.edu. https://lpsa.swarthmore.edu/NumInt/NumIntFourth.html

Digital Signal Processing: A Practical Guide for Engineers and Scientists. 2003, www.analog.com/media/en/technical-documentation/dsp-book/dsp_book_Ch20.pdf.

Jazar, Reza N.. Vehicle Dynamics: Theory and Application, Springer, 2008.

Kachroo, P., & Tomizuka, M. (1994). Vehicle Traction Control and its Applications. *UC Berkeley*, California PATH Research Paper UCB-ITS-PRR-94-08. https://escholarship.org/uc/item/6293p1rh.

L. Ge, Y. Zhao, S. Zhong, Z. Shan, & K. Guo, Eds. (2023). Efficient nonlinear model predictive motion controller for autonomous vehicles from standstill to extreme conditions based on split integration method]. ScienceDirect. https://www.sciencedirect.com/science/article/abs/pii/S0967066123002897

Marshall, Aarian. How Self-Driving Cars Navigate Construction Zones, Wired, 2017.

NPR. Nearly 400 car crashes in 11 months involved automated tech, companies tell regulators, 2022.

Rajamani, Rajesh. Vehicle Dynamics and Control, 2nd ed., Springer Publishing Company, New York, 2006.

Rudolph, Gert, Veolzke, Uwe. Three Sensor Types Drive Autonomous Vehicles, Fierce Electronics, 2017. https://www.fierceelectronics.com/components/three-sensor-types-drive-autonomous-vehicles

SAE International. (2021). Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-road Motor Vehicles. SVRP J3016. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. SAE International*. https://www.sae.org/standards/content.

Unsal, C., Kachroo, P.. Sliding Mode Measurement Feedback Control for Antilock Braking Systems, IEEE Transactions on Control Systems Technology, 1999.

Vargas, J., Alsweiss, S., Toker, O., Razdan, R., & Santos, J. (2021). An overview of autonomous vehicles sensors and their vulnerability to weather conditions. *Sensors*, *21*(16), 5397. https://doi.org/10.3390/s21165397.