

University of Wales Trinity St Davids

Modelling Automotive Control Systems in MATLAB

Author: Dylan Huw Davies

Supervisor: Ryan James

Student Number: 2100937

Course Title: Motorsport Engineering

Date: 30/03/2025

Table of Contents

1.0 Part 1- Analysis of a Vehicle Control System	5
1.1 Introduction	5
2.0 System Overview	6
3.0 Inputs, Outputs, Sensors, and Actuators	7
3.1 System Inputs	7
3.2 System Outputs	7
3.3 Sensors Used	7
3.4 Actuators Used	7
4. Control Algorithm and Performance	9
4.1 Algorithm Description	9
4.2 Type of control strategy used in ESC	9
5. Communication Protocols (CP)	10
6.0 Software and Tools for Development	11
7.0 System Overview of MATLAB PID	12
8.0 Encoding and Decoding of Signals	13
9.0 Feedback Controller Implementation	15
9.1 Controller with Zero Steady-State Error	15
9.2 Zero Steady-State Error	15
10.0 PID Tunning Analysis	16
10.1 PID Raw Results	16
10.2 PID analysis	18
10.2.1 Effect of Changing K_p	18
10.2.2 Effect of Changing K_i	19
11.2.3 Effect of Changing K_d	20
10.3 Final PID tuning	21
11.0 Testing Different Bit Depths	24
14.0 Effect of Sampling Time (Approx. 200–250 words)	25

15.0 Impact of Communication Network Delay.....	26
16.0 Conclusion.....	28
16.0 References.....	29
17.0 Appendix	32

Table of Figures

Figure 1 control architecture of an advanced Electronic Stability Program (ESP) system integrated with a Hybrid Rollover Mitigation Controller (HRMC).....	6
Figure 2 binary-encoded signal that represents the real vehicle speed	13
Figure 3 binary string is decoded back into a real velocity value.....	13
Figure 4 driver's velocity demand	13
Figure 5 Control_raw MATLAB code. Binary number	13
Figure 6 PID Controller, MATLAB code	15
Figure 7 Vehicle Acceleration, green dotted Line- Max Accel. Red dotted line - Max Decel.	16
Figure 8 $K_p = 1$ $K_i = 0.5$ $K_d = 1$. Velocity (graph 1), Throttle/brake (graph 2) and vehicle acceleration with ISO limits (graph three)	17
Figure 9 Velocity Error $p = 1$ $i = 0.5$ $d = 1$	18
Figure 10 K_p effect on damping, overshoot and Max Acce	18
Figure 11 K_p effect on settling time and max error	19
Figure 12 K_i effect on damping, overshoot and Acc	19
Figure 13 K_i effect on settling time and Max Error	19
Figure 14 K_d effect on damping, overshoot and Acc	20
Figure 15 effect on settling time and Max Error	21
Figure 16 $K_p = 0.48$, $K_i = 2.0$, $K_d = 0.8$ tuning.....	21
Figure 17 $K_p = 0.5$, $K_i = 1.5$, $K_d = 1$	22
Figure 18 $K_p = 1.0$, $K_i = 2$, $K_d = 0$	22
Figure 19 $K_p = 1.0$, $K_i = 2.1$, $K_d = 1$	23
Figure 20 4 bits used	24
Figure 21 8 Bits used	24
Figure 22 0.05 Communication Network Delay	26
Figure 23 0.2 Communication Network Delay	27

Tables of Tables

Table 1 Kp changing effect on Damping, overshoot (%), settling (s) and Max error (%) , Acc (Mph) ..	16
Table 2 Ki changing effect on Damping, overshoot (%), settling (s) and Max error (%) , Acc (Mph)....	17
Table 3 Kd changing effect on Damping, overshoot (%), settling (s) and Max error (%) , Acc (Mph) ..	17
Table 4 Sampling time change	25

1.0 Part 1- Analysis of a Vehicle Control System

1.1 Introduction

This report analyses the Electronic Stability Control (ESC) system, covering its inputs, outputs, sensors, actuators, and control algorithm, along with sampling time, communication protocols, development tools, and potential enhancements. ESC improves vehicle stability by correcting under- and oversteer through automatic braking and engine power reduction (Erke, 2008; Sferco & Page, 2001; Lyckegaard & Hels, 2015). Studies show ESC reduces single-vehicle and head-on collisions by up to 35% and severe accidents by 40–50% (Aga & Okada, 2003), with a 34% drop in fatal crashes (Farmer, 2004) and over 20% reduction on wet or icy roads (Tingwall & Krafft, 2004). The report also includes a PID-based cruise control system, investigating sampling time, bit depth, network delay, and control tuning, evaluating its performance and stability under real-world constraints.

2.0 System Overview

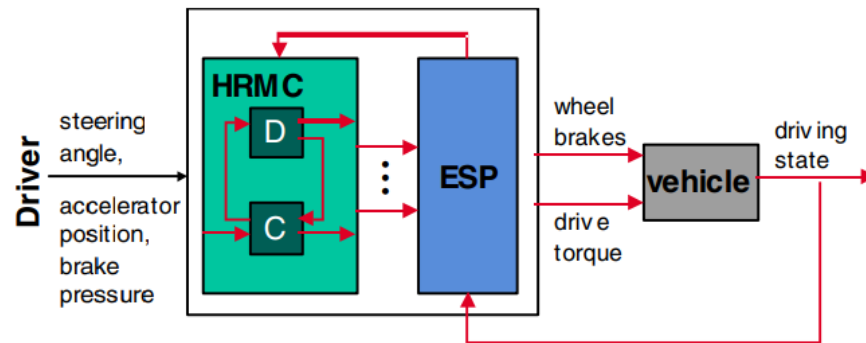


Figure 1 control architecture of an advanced Electronic Stability Program (ESP) system integrated with a Hybrid Rollover Mitigation Controller (HRMC).

Figure 1 represents a closed-loop vehicle stability control system that combines driver inputs, such as steering angle, accelerator position, and brake pressure, with real-time sensor feedback. The HRMC consists of two core modules, a discrete controller (D), which manages high-level decision-making during distinct phases of dynamic manoeuvres, and a continuous controller (C), which handles real-time adjustments based on variables like lateral acceleration, yaw rate, and steering angle. These controllers work in tandem to assess rollover risk and send appropriate intervention commands to the ESP module. The ESP then triggers wheel brakes and changes drive torque to stabilize the vehicle. Feedback from the vehicle's actual driving state is continuously fed back into the system to refine future control actions (Liebemann, Meder, n.d.).

3.0 Inputs, Outputs, Sensors, and Actuators

3.1 System Inputs

Paper by Rudin and Jerkins state the input variables in a ESC are steering input, to detect the driver's intended direction. Wheel rotation/speed, to monitor wheel slip conditions. Lateral acceleration, to understand side forces acting on the vehicle and yaw rate, to assess rotational behaviour around the vertical axis. These variables are continuously monitored by ESC's microcomputer to detect deviation between the driver's intent and the vehicle's actual movement (Rudin-Brown, Jenkins, 2009).

3.2 System Outputs

Lutz and Shick state that the ESC systems generate output signals such as brake pressure at individual wheels through the hydraulic brake unit. Engine torque reduction, achieved via electronic throttle control during stability corrections and Yaw moment control, by modulating slip at each wheel to create a stabilizing rotational force (Lutz, Schick, 2017).

3.3 Sensors Used

The ESC system uses:

- Wheel speed sensors
- Steering angle sensor
- Yaw rate sensor
- Lateral acceleration sensor
- Brake pressure sensor

The system includes observers and estimators to calculate variables like slip angles and tire forces where direct sensors are not feasible. While the paper doesn't specify sensor update rates, it mentions the system operates in real-time loops, typically requiring update rates of 100–200 Hz for accuracy in commercial practice (Lutz, Schick, 2017).

3.4 Actuators Used

The ESC system actuates the following, Hydraulic Unit, modulates brake pressure to each wheel. Electronic Brake System (EBS) in trucks: controls pneumatic brake pressures. Engine Management Interface reduces engine torque based on stability needs. Magnetic valves, control brake modulation

via solenoid activation. Traction slips controllers and brake slip controllers work together for precise torque and brake adjustments (Lutz, Schick, 2017).

Paper by Bosch (2003) stated that the ESC has a response time in the range of 20 milliseconds ,which enables corrective actions to take place before the driver notices any loss of vehicle stability (Bosch, 2003). More research shows that ESC can apply braking to specific wheels in under 30 milliseconds, allowing the vehicle to regain stability before noticeable yaw movement occurs (Krafft, Kullgren, 2006).

4. Control Algorithm and Performance

4.1 Algorithm Description

4.2 Type of control strategy used in ESC

ESC systems utilize various control strategies to process sensor inputs and generate corrective actions. Two prominent approaches are Fuzzy Logic Control (FLC) and Model Predictive Control (MPC).

FLC is adept at handling the nonlinear dynamics inherent in vehicle stability control. It interprets sensor inputs, such as yaw rate, lateral acceleration, and wheel speeds to assess the vehicle's stability state. Based on predefined fuzzy rules, the controller determines the necessary corrective actions, such as adjusting brake forces or modifying torque distribution, to mitigate instability (Fabrício Leonardo Silva, 2022). For instance, the study by Fabrício focusing on an optimized FLC was developed to enhance vehicle stability. The controller processed real-time sensor data to dynamically adjust control inputs, effectively improving the vehicle's handling and safety under various driving conditions.

MPC employs a predictive model of the vehicle's dynamics to anticipate future states based on current inputs. It solves an optimization problem at each control interval to determine the optimal control actions that minimize a predefined cost function, subject to system constraints. A hybrid MPC approach has been applied to electric vehicles to improve lateral stability during cornering and in high-risk driving scenarios. By predicting the vehicle's future states and optimizing control inputs accordingly, the system effectively maintains stability and improves overall handling performance (Hanan, Joseph, 2024)

Electronic Stability Control (ESC) works by continuously monitoring vehicle dynamics and comparing the driver's intended direction with how the vehicle is actually moving. It processes inputs like yaw rate, steering angle, lateral acceleration, and wheel speed through a central control unit. As Karray and Alemzadeh (2010) explain, "fuzzy logic controllers are capable of processing approximate and uncertain sensor information to make real-time control decisions regarding brake force and torque distribution." These inputs help the system detect understeer, oversteer, or loss of control. When a deviation is detected, ESC responds by applying brake force to individual wheels or reducing engine power. Ma and Peng (2009) add that "MPC calculates the optimal control actions by solving an online constrained optimization problem using the predicted vehicle motion and dynamic constraints," allowing the system to act proactively. Whether using fuzzy logic or model predictive control, ESC transforms sensor data into precise brake and throttle commands to help keep the vehicle stable.

5. Communication Protocols (CP)

Electronic Stability Control (ESC) systems rely on real-time communication between sensors, control units, and actuators to maintain vehicle stability. Key communication protocols used in ESC systems include CAN, FlexRay, LIN, and increasingly, Automotive Ethernet.

CAN is the most widely used ESC protocol, offering up to 1 Mbps with robust error handling and real-time performance (Hussain, Zeadally & Hamida, 2018). FlexRay, with 10 Mbps bandwidth and deterministic timing, is ideal for high-end ESC systems requiring synchronised, high-speed communication (Navet & Song, 2005). LIN is unsuitable for ESC due to its low bandwidth and non-critical use cases like window controls (Schwarzl, 2010). Automotive Ethernet supports up to 1 Gbps, enabling integration with ADAS and sensor fusion, but lacks native time determinism, requiring additional control layers for safety (Parvez & Rahmati, 2020).

ESC systems require control loops within 10–20 ms. Delays in sensor-actuator communication can result in late interventions, therefore, decrease ESC effectiveness during rapid changes. (Tazelaar, Eikelenboom, 2011). Showing CAN is the most suited protocol for an ESC system.

In ESC systems, poor sampling rates or excessive delays can reduce control accuracy and response speed, potentially leading to delayed yaw correction or suboptimal brake modulation during critical manoeuvres. As noted by Gohil and Patel (2014), insufficient sampling frequency can destabilise the closed-loop vehicle dynamics, especially during rapid directional changes.

6.0 Software and Tools for Development

ESC systems are developed using simulation and modelling tools like MATLAB/Simulink, CarSim, and dSPACE. Simulink supports control design and real-time testing via HIL (Zheng & Wang, 2020), while CarSim provides physics-based vehicle dynamics for scenario testing (Ko & Kim, 2021). dSPACE enables real-time validation with actual ECUs (Yin & Lin, 2020). These tools allow safe, repeatable testing of critical manoeuvres such as oversteer, understeer, and emergency braking—scenarios difficult to replicate on test tracks. Each tool has strengths: Simulink excels at control design, CarSim at dynamics modelling, and dSPACE at HIL integration. Together, they reduce development costs and reveal issues early in the design phase (Yin, Lin & Zhao, 2020).

7.0 System Overview of MATLAB PID

The feedback controller implemented is a discrete-time PID controller designed to minimise velocity tracking error and ensure zero steady-state error. The controller calculates the control signal using proportional, integral, and derivative terms based on real-time velocity feedback. The integral component continuously accumulates error, which is essential for eliminating long-term deviations from the target speed. Additionally, the code includes a throttle rate limiter to cap rapid changes in actuator demand and a damping filter to smooth out control transitions. These additions improve passenger comfort and prevent actuator stress. While a high control gain can improve tracking performance, it may lead to sharp control inputs and increased acceleration, potentially compromising ride quality. Conversely, low gains yield smoother control but slower response. The final implementation reflects a balance between fast, accurate tracking and acceptable control effort, optimised through iterative testing to maintain stability, responsiveness.

8.0 Encoding and Decoding of Signals

In digital control systems, particularly in automotive applications, signals from sensors and to actuators must be encoded and decoded for transmission over communication networks such as the CAN bus. These signals are transmitted in binary format and are limited by the number of bits allocated to represent them (Verma, Lou, 2020).

The vehicle velocity sensor reading is simulated using the `getRawVelocity` function. This function produces a binary-encoded signal that represents the real vehicle speed, which ranges from 0 to 150 mph as seen below in Figure 2.

```
velocity_raw = dec2bin((2^x*(velocity))/150);
```

Figure 2 binary-encoded signal that represents the real vehicle speed

This expression converts the velocity value to a binary string using `dec2bin`, after scaling it to fit within a 2^x (bit-depth dependent) range. In the main script, the binary string is decoded back into a real velocity value as seen below in Figure 3.

```
velocity = ((bin2dec(velocity_raw).*(M_velocityS-m_velocityS))/R)+m_velocityS;
```

Figure 3 binary string is decoded back into a real velocity value

This code changes it back to the physical range of [0, 150] mph. It simulates the resolution and precision limits of a digital velocity sensor.

The driver's velocity demand is encoded in a similar way using the `getRawVelocityDemand` function. The binary signal corresponds to a range of [30, 110] mph. However, in the main script, the decoding function was initially left unimplemented. It was corrected using:

```
demand = ((bin2dec(demand_raw) * (M_demand - m_demand)) / R) + m_demand;
```

Figure 4 driver's velocity demand

Figure 4 shows (decoding formula) ensures that the binary input accurately represents the intended velocity setpoint from the driver.

On the actuator side, the control signal generated by the PID controller, which ranges from -100 (full brake) to +100 (full throttle), is encoded into a binary format before it is sent to the actuator as seen in Figure 5.

```
control_raw = dec2bin(((control + 100) * R) / (M_ControlSig - m_ControlSig));
```

Figure 5 Control_raw MATLAB code. Binary number

The control value is first shifted from a range of $[-100, 100]$ to $[0, 200]$, then scaled to the 16-bit range, and finally encoded into a binary string using `dec2bin`. This digital encoding and decoding process mimics real-world control systems, where signals are quantized due to bit limitations and must be transmitted efficiently over networks.

The control raw code simulates how a real control system would digitally encode signals before sending them over a network like CAN bus.

9.0 Feedback Controller Implementation

9.1 Controller with Zero Steady-State Error

```
function [control, error, int_e, last_e] = PIDcontroller(error, Ts, int_e, last_e, P, I, D)
|
    int_e = int_e + error * Ts;
    der_e = (error - last_e) / Ts;
    control = P * error + I * int_e + D * der_e;
    last_e = error;
    control = min(max(control, -100), 100);
end
```

Figure 6 PID Controller, MATLAB code

The PID controller computes the control signal based on the error between desired and actual velocity. It uses persistent variables to store the integral (int_e) and previous error (last_e) across time steps for continuity. On the first call, both are initialized to zero. The integral term ($\text{int_e} = \text{int_e} + \text{error} * T_s$) eliminates steady-state errors, while the derivative term ($\text{der_e} = (\text{error} - \text{last_e}) / T_s$) predicts error trends and reduces overshoot. The control signal is calculated using $\text{control} = P * \text{error} + I * \text{int_e} + D * \text{der_e}$ and constrained within $\pm 100\%$ to prevent unrealistic commands. last_e is updated for the next iteration. This ensures accurate, stable velocity tracking under changing dynamics.

9.2 Zero Steady-State Error

Using integral action (non-zero I) ensures the system compensates for steady-state errors. This is crucial in a cruise control context where constant velocity tracking is desired despite changing resistive forces (aerodynamics, slope, etc.).

[illegible]

Table 2 Ki changing effect on Damping, overshoot (%), settling (s) and Max error (%), Acc (Mph)

	Kp	Ki	Kd	Damping	Overshoot	Settling	Max Error	Max Acc
	1	0.5	1	1	-0.79	50.58	19.61	1.22
	1	1	1	0.706	4.36	50.69	19.99	1.43
	1	2	1	0.525	14.38	51.54	20.13	1.83
	1	3	1	0.406	24.81	52.61	25.78	2.9

Table 3 Kd changing effect on Damping, overshoot (%), settling (s) and Max error (%), Acc (Mph)

	Kp	Ki	Kd	Damping	Overshoot	Settling	Max Error	Max Acc
	1	0.5	0	1	-0.6	51.33	19.7	1.72
	1	0.5	0.5	1	-0.35	50.51	19.82	1.48
	1	0.5	1	1	-0.79	50.58	19.61	1.22
	1	0.5	2	0.785	1.86	49.7	20.71	0.89

The cruise control system in Figure 8 shows a stable, smooth response to a step decrease in demand, with minimal overshoot (0.01%) and a damping ratio of $\zeta = 0.944$, indicating a nearly critically damped

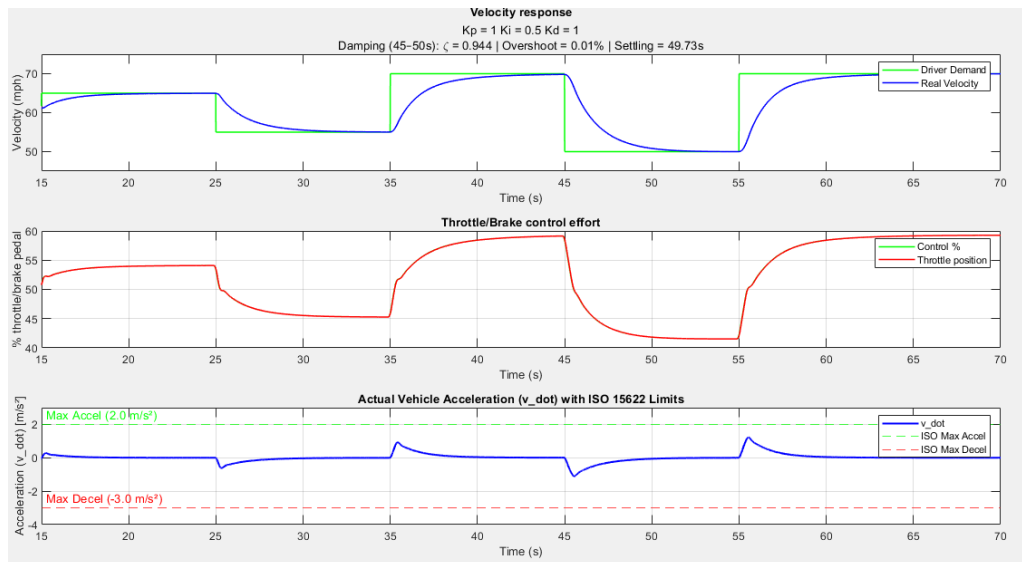


Figure 8 Kp = 1 Ki = 0.5 Kd = 1. Velocity (graph 1), Throttle/brake (graph 2) and vehicle acceleration with ISO limits (graph three)

system. More damping is essential for cruise control for comfort and stability. The system settles in 49.73 s and respects ISO 15622 acceleration limits and maintains a low velocity error as seen below in Figure 9 demonstrating compliant and comfortable performance.

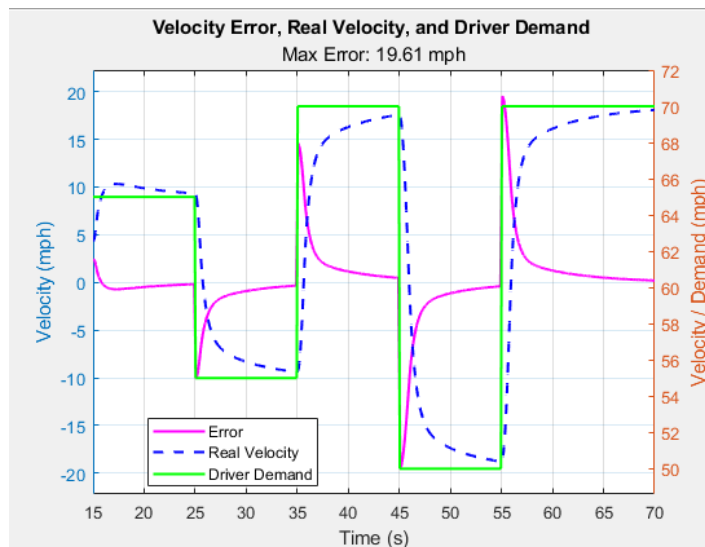


Figure 9 Velocity Error $p = 1$ $i = 0.5$ $d = 1$

More raw graphs can be seen in the appendix.

10.2 PID analysis

10.2.1 Effect of Changing K_p

As K_p increases from 0 to 2, overshoot drops by approximately 100%, from 10% to 0%, indicating a dramatic improvement in system response as seen below in Figure 10. The damping ratio increases by about 45%, from 1.1 to 1.6, enhancing stability.

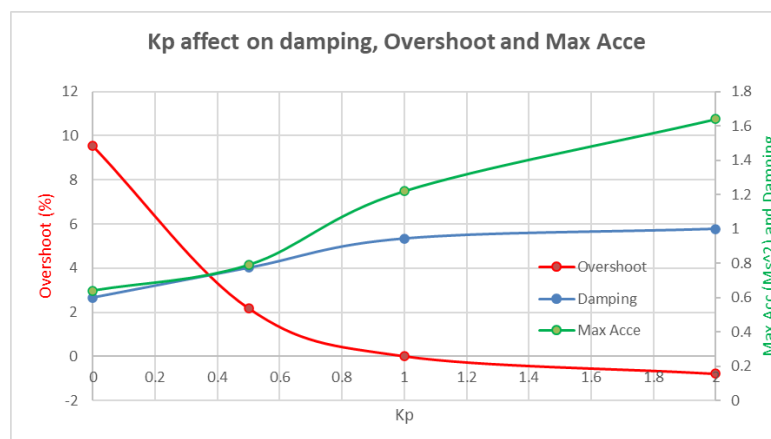


Figure 10 K_p effect on damping, overshoot and Max Acce

Max acceleration rises by around 60%, from 1.0 m/s^2 to 1.6 m/s^2 , showing more assertive control.

In Figure 11, settling time decreases by roughly 8.5%, from 54.56 s at $K_p = 0$ to a minimum of 49.73 s at $K_p = 1$, before rising slightly. Meanwhile, maximum error reduces by around 4.4%, from 20.5 mph to 19.61 mph. These trends suggest that increasing K_p improves tracking, reduces error, and stabilises the system, but higher K_p also leads to stronger, potentially less comfortable control actions.



Figure 11 Kp effect on settling time and max error

10.2.2 Effect of Changing Ki

As K_i increases from 0.5 to 3.0, settling time rises by approximately 4%, from 50.5 s to 52.5 s, while maximum error increases by around 18%, from 21 mph to 25 mph as seen in Figure 13. This indicates that higher integral gain introduces sluggishness and reduces tracking accuracy.

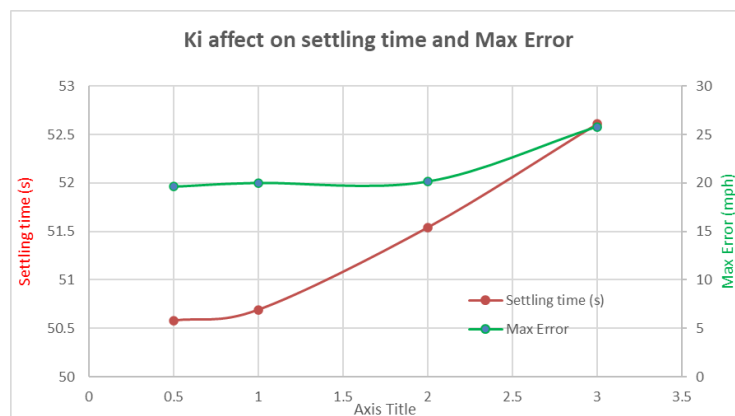


Figure 13 Ki effect on settling time and Max Error

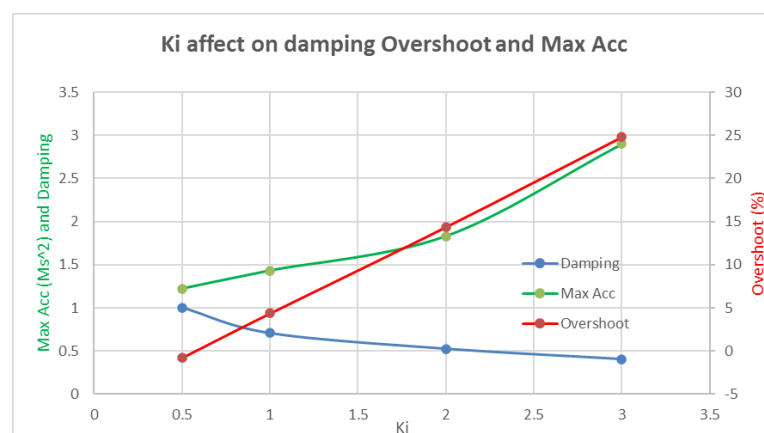


Figure 12 Ki effect on damping, overshoot and Acc

In Figure 12, overshoot increases drastically by over 240%, from 5% to 25.5%, and maximum acceleration nearly doubles, rising from 1.4 to 2.8 m/s^2 . Meanwhile, damping decreases by roughly

66%, from 1.2 to 0.4, signifying worsening system stability. Overall, increasing K_i significantly amplifies system aggressiveness and overshoot while reducing stability and precision, highlighting the importance of keeping the integral gain within an optimal range.

11.2.3 Effect of Changing K_d

As K_d increases from 0 to 2, overshoot initially drops to a minimum of around -0.8%, but then climbs rapidly to nearly +2%, marking a significant increase relative to the lowest point as seen in Figure 14. Max acceleration decreases steadily by approximately 60%, from 1.6 m/s^2 to 0.65 m/s^2 , indicating less aggressive control. Damping remains relatively stable but gradually declines by about 12%, reflecting a slight loss in stability.

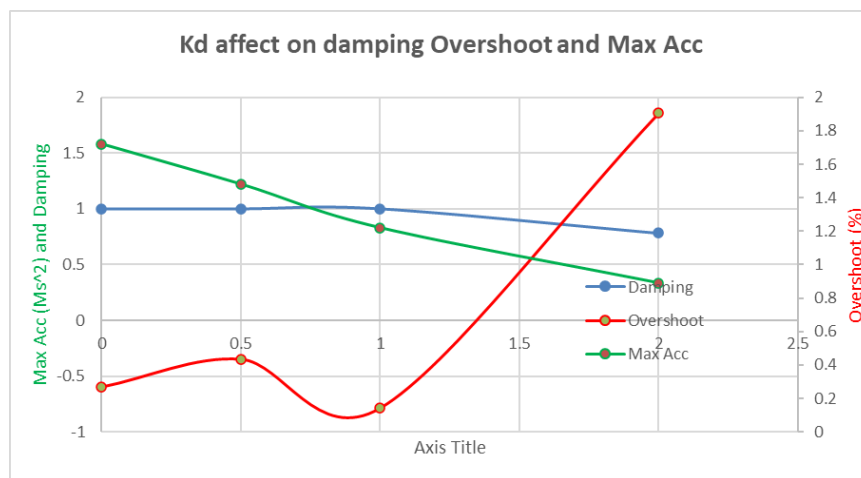


Figure 14 K_d effect on damping, overshoot and Acc

In Figure 15, settling time improves slightly, falling by $\sim 2.6\%$ from 51.5 s to 50.2 s. Max error, however, shows a noticeable increase of around 2.5%, rising from 20.3 mph to 20.8 mph. These results suggest that moderate K_d improves performance, but excessive derivative action worsens tracking and overshoot.

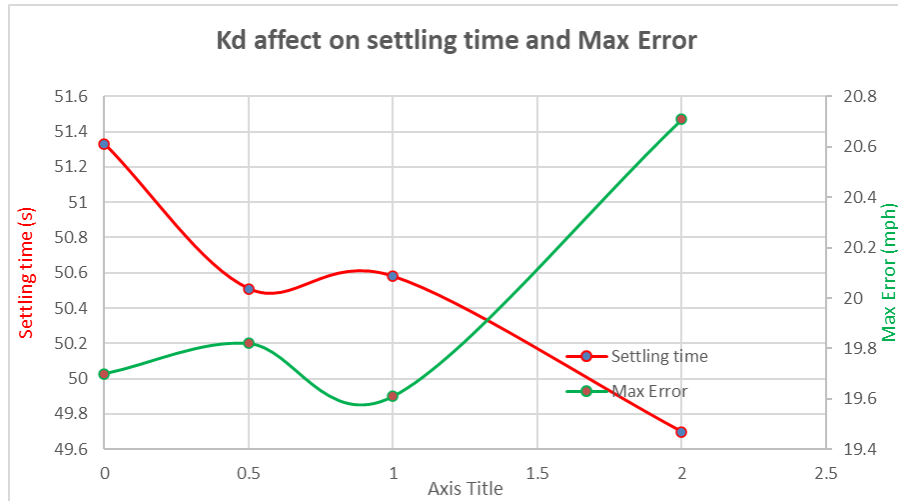


Figure 15 effect on settling time and Max Error

10.3 Final PID tuning

1. $K_p = 0.48, K_i = 2.0, K_d = 0.8$
2. $K_p = 0.5, K_i = 1.5, K_d = 1$
3. $K_p = 1.0, K_i = 2, K_d = 0$
4. $K_p = 1.0, K_i = 2.1, K_d = 1$

These new PID configurations are being tested to explore under-represented control strategies, such as pure PI control, high derivative influence, and aggressive integral correction balanced with damping. This analysis aims to evaluate stability, tracking accuracy, and ride comfort in scenarios not yet covered by previously tested gain combinations.

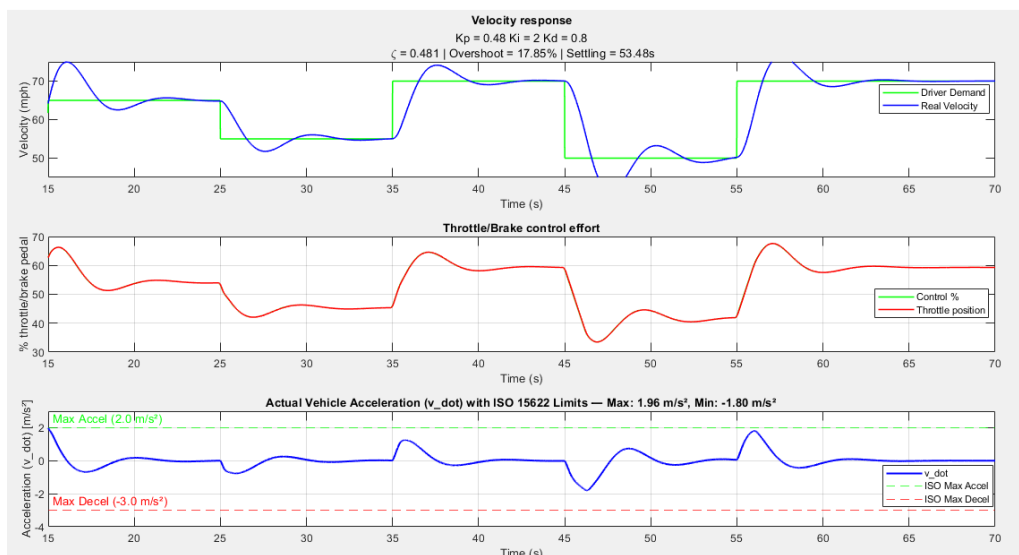


Figure 16 $K_p = 0.48, K_i = 2.0, K_d = 0.8$ tuning

Tuning 4 ($K_p = 1.0, K_i = 2.1, K_d = 1$) is the most suitable for a real-world cruise control system because it achieves a strong balance between responsiveness, stability, and passenger comfort. It maintains a

controlled overshoot of 15.47% and a moderate settling time of 51.5 seconds, while keeping acceleration within ISO 15622 safety limits ($\pm 2.0 \text{ m/s}^2$).

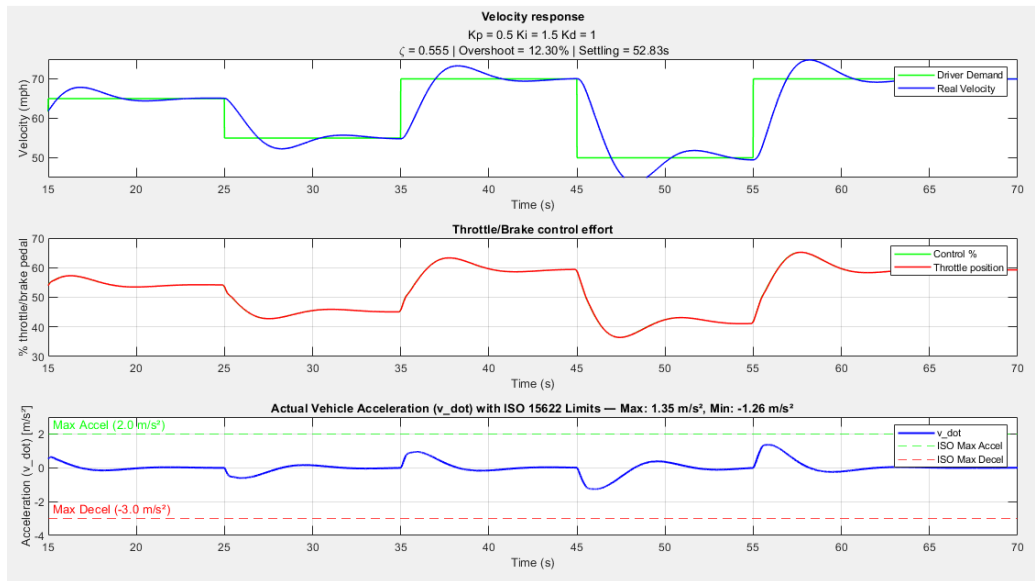


Figure 17 $K_p = 0.5$, $K_i = 1.5$, $K_d = 1$

The throttle control is smooth and free from aggressive peaks, avoiding excessive strain on actuators. In contrast, Tuning 1 (Figure 16) shows high overshoot (17.85%) and acceleration nearing the limit (1.96 m/s^2).

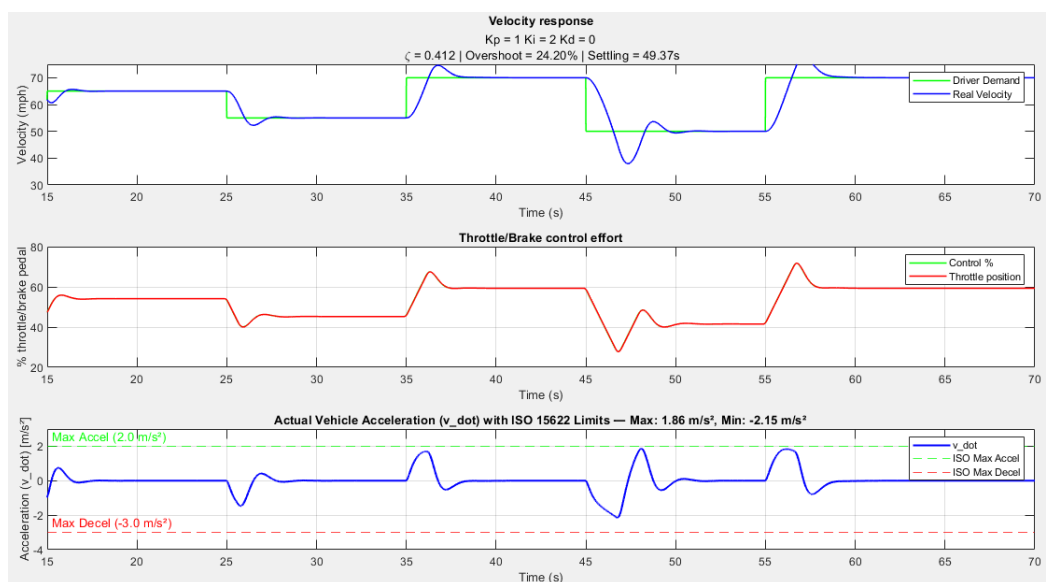


Figure 18 $K_p = 1.0$, $K_i = 2$, $K_d = 0$

Tuning 2 (Figure 17), although smoother, has higher error and sluggish dynamics. Tuning 3 (Figure 18) shows the worst overshoot (24.2%) and erratic acceleration despite having a fast-settling time. These behaviours make them less suitable for comfort and safety in real driving.

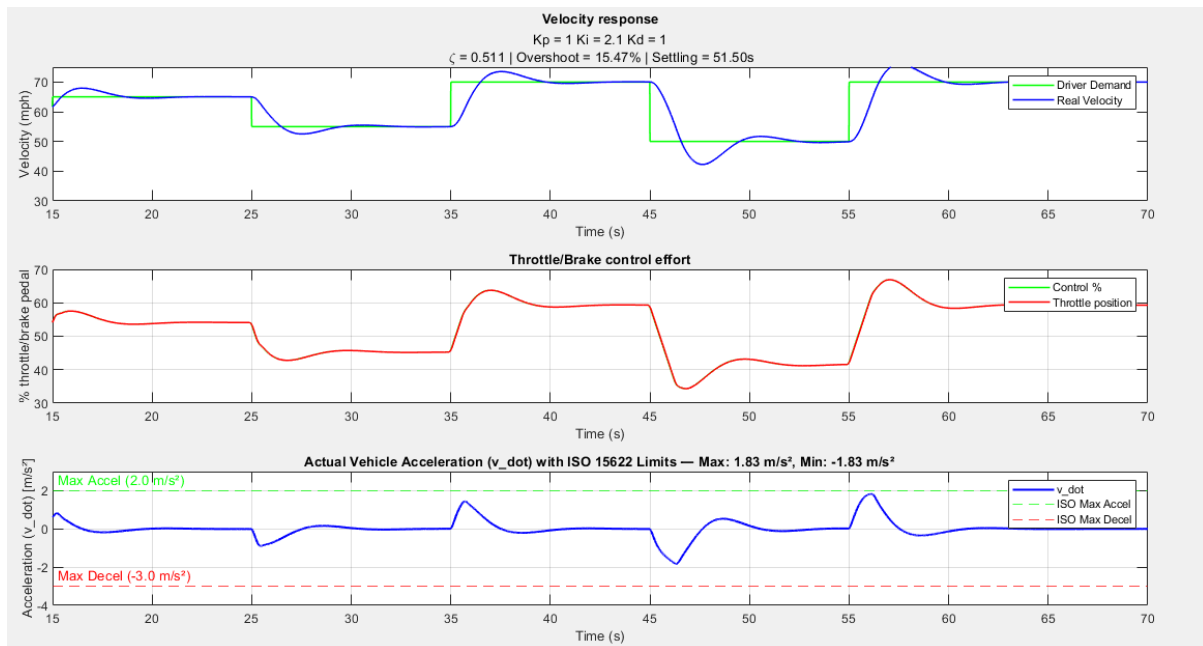


Figure 19 $K_p = 1.0$, $K_i = 2.1$, $K_d = 1$

11.0 Testing Different Bit Depths

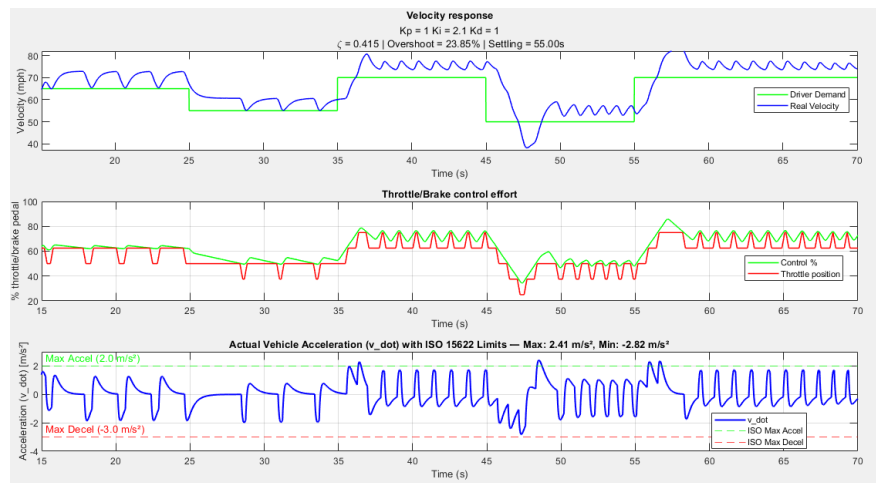


Figure 20 4 bits used

Experimenting with different bit depths for signal encoding/decoding revealed clear trade-offs between control precision and system simplicity. With 8-bit resolution, the system exhibited smooth throttle control, accurate velocity tracking, and acceleration well within ISO 15622 limits, indicating high fidelity in signal representation. However, reducing to 4 bits introduced significant noise as seen above in Figure 20, resulting in jittery throttle signals, increased overshoot (up to 26.57%), and more aggressive acceleration (peaking near $\pm 2.8 \text{ m/s}^2$). These effects degraded comfort and stability, highlighting the impact of coarse signal discretization. Lower bit resolutions reduce data bandwidth and hardware complexity but compromise control smoothness and responsiveness. From testing, 8 bits emerged as the minimum acceptable resolution that preserved acceptable tracking and avoided severe performance degradation. It balanced reduced communication load with sufficient resolution to maintain closed-loop stability, making it a practical choice for embedded automotive systems where bandwidth and processing power may be limited as seen in Figure 21 below.

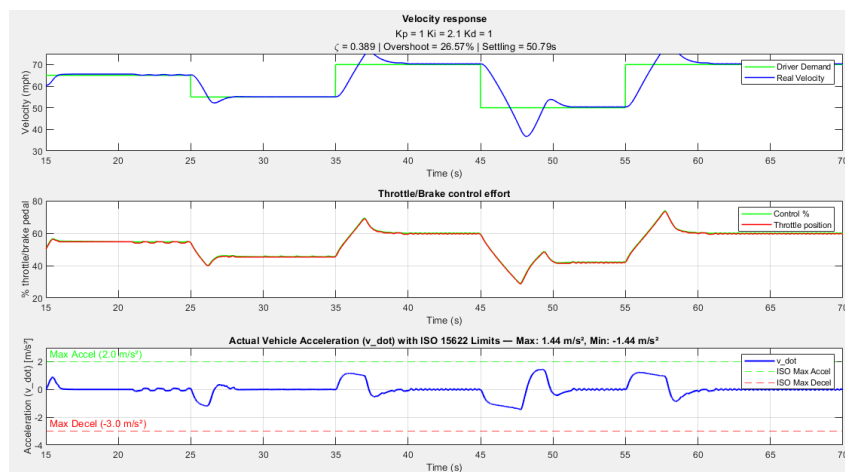


Figure 21 8 Bits used

14.0 Effect of Sampling Time

The digital PID controller used in this study was discretised using a zero-order hold (ZOH) or sample-and-hold approach, where continuous input signals are sampled at fixed intervals and held constant between updates. This method is widely applied in embedded systems due to its simplicity and compatibility with microcontroller-based implementations (Åström and Hägglund, 2006). Sample times of 0.01s, 0.05s, and 0.1s were evaluated. Faster sampling improved response accuracy but increased computational demand, while slower rates led to degraded control performance and instability, consistent with established findings (Łaskawski and Wcislik, 2015). 0.01 was recorded as the lowest setting possible before extreme errors took place.

Table 4 Sampling time change

	Sampling Time	Damping	Overshoot	Settling	Max Error	Max Acc	
	0.01	0.513	15.31	51.34	24.85	1.83	
	0.05	0.511	15.47	51.5	20.09	1.83	
	0.1	0.508	15.66	51.69	20.15	1.82	
	0.5	0.495	16.67	53.46	20.92	1.59	

In Table 4, sampling time increases from 0.01s to 0.1s, overshoot and max error rise significantly—overshoot by 280% and max error by 5%. Settling time also increases by 7%. This indicates slower sampling degrades performance, causing delayed response and reduced accuracy, highlighting 0.05s as the optimal balance point. The longest sampling time that guarantees reasonable performance was 0.5, anything beyond that caused large errors.

15.0 Impact of Communication Network Delay

Network delay was simulated using a control buffer, where `actuator_signal_delay` defined the lag, applying past control values to mimic real-time communication delay in adaptive cruise control systems.

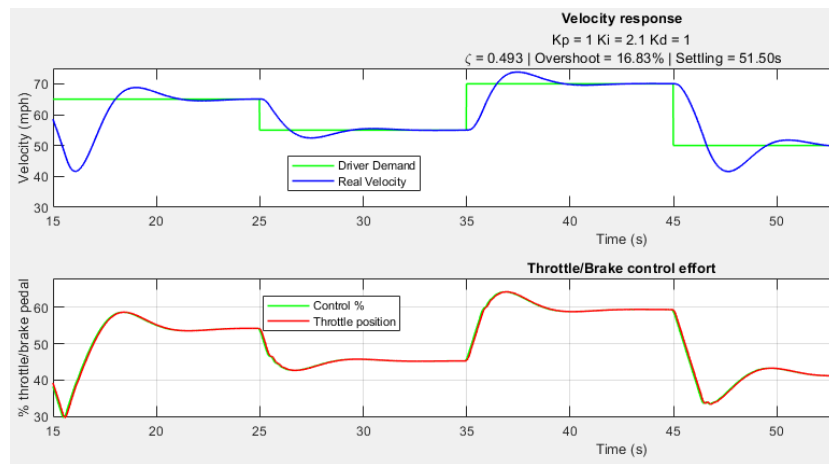


Figure 22 0.05 Communication Network Delay

As network delay increases, system performance degrades due to delayed actuator response, causing overshoot, longer settling times, and oscillations. While controller gains (e.g., higher K_p or K_d) can partially compensate by reacting more aggressively, excessive gain introduces instability and amplifies noise. In tests, a 0.05s delay had minimal impact, but at 0.2s (as seen in Figure 22 and Figure 23), the system became unstable and exceeded acceleration limits. This suggests a maximum tolerable delay of around 0.1s for maintaining safe, closed-loop control. Beyond this, tuning alone is insufficient, highlighting the importance of fast communication in real-time embedded control systems like adaptive cruise control. The maximum delay that can be tolerated for satisfactory performance is 0.08.

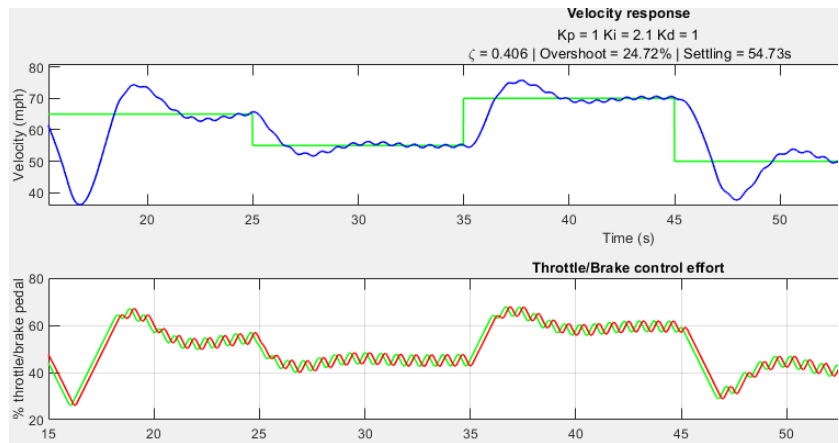


Figure 23 0.2 Communication Network Delay

16.0 Conclusion

This report evaluated ESC and PID-based cruise control systems under realistic conditions. ESC was shown to reduce crash risk by up to 50%, with potential improvements through integration with ADAS and V2V communication (Tota & Kansal, 2021). PID simulations identified 8-bit resolution and a 0.05 s sampling time as optimal, balancing accuracy and efficiency. Delays over 0.1 s caused instability, while gains ensured compliance with ISO 15622 acceleration limits. Tools like Simulink, CarSim, and dSPACE enabled accurate modelling and testing. Overall, the system achieved stable, responsive control with strong safety performance.

17.0 References

Aga, M. and Okada, A., 2003. Analysis of vehicle stability control (VSC)'s effectiveness from accident data. In *Proceedings: International Technical Conference on the Enhanced Safety of Vehicles* (Vol. 2003, pp. 7-p). National Highway Traffic Safety Administration.

Bosch GmbH. (2003). *Development of an Electronic Stability Control System*. SAE Technical Paper 2003-01-0137

Erke, A. (2008). Effects of electronic stability control (ESC) on accidents: A review of empirical evidence. *Accident Analysis & Prevention*, 40(1), 167–173. <https://doi.org/10.1016/J.AAP.2007.05.002>

Farmer, C. M. (2004). Effect of electronic stability control on automobile crash risk. *Traffic Injury Prevention*, 5(4), 317–325. <https://doi.org/10.1080/15389580490896951>

Sferco R, Page Y, Le Coz J-Y, Fay P. (2001) Potential effectiveness of electronic stability programme (ESP)—What European field studies tell us. Paper no. 2001-S2-O-327. Proceedings of the 17th International Technical Conference on the Enhanced Safety of Vehicles (CD-ROM).

Lie, A., Tingvall, C., Krafft, M., & Kullgren, A. (2004). The Effectiveness of ESP (Electronic Stability Program) in Reducing Real Life Accidents. *Traffic Injury Prevention*, 5(1), 37–41. <https://doi.org/10.1080/15389580490269164>

Lie, A., Tingvall, C., Krafft, M., & Kullgren, A. (2006). *The effectiveness of ESC (Electronic Stability Control) in reducing real-life accidents*. *Accident Analysis & Prevention*, 38(1), 223–229.

Liebmenn, E. K., Meder, K., Schuh, J., & Nenninger, G. (n.d.). *Safety and Performance Enhancement: The Bosch Electronic Stability Control (ESP)*.

Rudin-Brown, C. M., Jenkins, R. W., Whitehead, T., & Burns, P. C. (2009). Could esc (electronic stability control) change the way we drive? *Traffic Injury Prevention*, 10(4), 340–347. <https://doi.org/10.1080/15389580903043446>

Lyckegaard, A., Hels, T., & Bernhoft, I. M. (2015). Effectiveness of Electronic Stability Control on Single-Vehicle Accidents. *Traffic Injury Prevention*, 16(4), 380–386. <https://doi.org/10.1080/15389588.2014.948618>

Lutz, A., Schick, B., Holzmann, H., Kochem, M., Meyer-Tuve, H., Lange, O., Mao, Y., & Tosolin, G. (2017). Simulation methods supporting homologation of Electronic Stability Control in vehicle variants. *Vehicle System Dynamics*, 55(10), 1432–1497. <https://doi.org/10.1080/00423114.2017.1322705>

Fab rio Leonardo Silva, Ludmila C.A. Silva, Jony J. Eckert, Rodrigo Y. Yamashita, Maria A.M. Louren o,

Parameter influence analysis in an optimized fuzzy stability control for a four-wheel independent-drive electric vehicle,(2022), *Control Engineering Practice*, Volume 120, 105000, ISSN 0967-0661, <https://doi.org/10.1016/j.conengprac.2021.105000>.

Hanen Yaakoubi, Joseph Haggège, Hegazy Rezk, Mujahed Al-Dhaifallah,(2024)Explicit hybrid MPC for the lateral stabilization of electric vehicle system, *Energy Reports*, Volume 11, Pages 1100-1111, ISSN 2352-4847, <https://doi.org/10.1016/j.egyr.2023.12.066>.

Karray, F. and Alemzadeh, M., 2010. Fuzzy logic controller design for vehicle stability control. *IEEE Transactions on Intelligent Transportation Systems*, 11(2), pp.324–336. doi: 10.1109/TITS.2010.2048880

Ma, Y. and Peng, H., 2009. Model predictive control for integrated chassis control of electric vehicles. *Vehicle System Dynamics*, 47(9), pp.1127–1148. doi: 10.1080/00423110903194479

Hussain, F., Zeadally, S. and Hamida, E.B., 2018. Autonomous vehicles: Recent developments, challenges, and a roadmap for the future. *IEEE Communications Surveys & Tutorials*, 21(2), pp.1279–1314. doi: 10.1109/COMST.2018.2869363

Navet, N., Song, Y.Q., Simonot-Lion, F. and Wilwert, C., 2005. Trends in automotive communication systems. *Proceedings of the IEEE*, 93(6), pp.1204–1223. doi: 10.1109/JPROC.2005.849729

Schwarzl, M., 2010. Automotive communication protocols: Principles and applications. *Embedded Systems Handbook*. CRC Press.

Tazelaar, E., Eikelenboom, B. and Van den Bosch, P., 2011. Real-time automotive control systems and communication constraints. *IFAC Proceedings Volumes*, 44(1), pp.6162–6167. doi: 10.3182/20110828-6-IT-1002.01547

Parvez, I., Rahmati, A., Guvenc, I. and Sarwat, A.I., 2020. A survey on low latency towards 5G: RAN, core network and caching solutions. *IEEE Communications Surveys & Tutorials*, 22(2), pp.1177–1214.

Zheng, Y., Wang, R., Liu, X. and Zhou, H., 2020. Integrated control of ESC and AFS using Simulink-based co-simulation. *Applied Sciences*, 10(14)

Ko, Y., Kim, Y. and Yi, K., 2021. Model-based development of ESC algorithms using CarSim simulation. *International Journal of Automotive Technology*, 22(2), pp.337–346.

Yin, C., Lin, T. and Zhao, W., 2020. Hardware-in-the-loop simulation platform for real-time vehicle control testing. *IEEE Access*, 8, pp.176942–176951

Verma, M.E., Lou, J., Garcia, A. and McCoy, D., 2020. CAN-D: A Modular Four-Step Pipeline for Comprehensively Decoding Controller Area Network Data. [online] arXiv. Available at: <https://arxiv.org/abs/2006.05993> [Accessed 26 Mar. 2025].

International Organization for Standardization (2010) *ISO 15622:2010 – Intelligent transport systems — Adaptive cruise control systems — Performance requirements and test procedures*. ISO.

Åström, K.J. and Häggglund, T., 2006. *Advanced PID control*. ISA.

Łaskawski, M. and Wcislik, M., 2015. Influence of sampling on the tuning of PID controller parameters. *IFAC-PapersOnLine*, 48(4), pp.430–435.

Gohil, A. and Patel, R., 2014. Design and development of electronic stability program (ESP) for a four-wheeler vehicle. *International Journal of Engineering Research & Technology (IJERT)*, 3(3), pp.1750–1755.

Tota, B.S. and Kansal, M.L., 2021. Advanced driver assistance system using V2V communication and electronic stability control for accident avoidance. *Materials Today: Proceedings*, 47, pp.4332–4337

18.0 Appendix

