

University of Wales Trinity Saint David

Wales Institute of Science & Art



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Modelling Automotive Control Systems in MATLAB

Energy Storage and Control Systems (ACHA7007)

Anthony Yin Wai Hau

1550559

29/03/2026

1	Contents	
2	Introduction	2
3	Encoding/decoding.....	2
4	Controller design and gain selection	Error! Bookmark not defined.
5	Bit-depth study	Error! Bookmark not defined.
6	Sampling-time study.....	5
7	Network-delay study.....	6
8	Extra implementation	7
9	Conclusion	9
10	References	Error! Bookmark not defined.

2 Introduction

This report analyses a MATLAB-implemented digital cruise controller, focusing on signal quantisation/encoding, discrete-time feedback design, and the effects of sampling, bit-depth, and network delay on closed-loop performance. (ISO, 2018; Åström and Murray, 2008).

In this report, “reasonable performance” is defined as stable tracking without sustained oscillation, moderate control effort, and no severe control chatter.

Performance is assessed using maximum tracking error, integral absolute error (IAE), peak control demand, and the total variation of the control signal, $TV(u)$, as a chatter indicator.

3 Encoding/decoding

The simulation treats the velocity sensor, driver demand and actuator command as digital bus messages rather than continuous signals. At each simulation step the vehicle model returns the measured speed as a binary string. This raw bus value is converted to an integer with `bin2dec(...)` and mapped to mph by linear scaling over the sensor range (0–150 mph). The driver demand is generated as a stepwise profile, transmitted as a binary message, decoded in the same way, and scaled over the demand range (30–110 mph). Working in physical units after decoding allows the tracking error to be formed consistently and the controller gains to be tuned meaningfully. (MathWorks, n.d.).

For the decoded signals the mapping used is:

$$\text{velocity} = m_velocityS + (M_velocityS - m_velocityS) * (v_code / R_sensor)$$

$$\text{demand} = m_demand + (M_demand - m_demand) * (d_code / R_demand)$$

where $R = 2^n - 1$ is the full-scale integer code for an n -bit message (MathWorks, n.d.).

The controller output u is a throttle/brake command limited to $[-100, 100]$. Before transmission it is saturated, mapped to an integer code, rounded/clamped, and encoded as a fixed-width binary string using `dec2bin(code_u, n_bits_actuator_signal)`. `sendControlCommand(...)` decodes the received command back into $[-100, 100]$ and can emulate communication delay by applying a delayed command from stored history (MathWorks, n.d.).

4 Feedback controller design, performance trade-offs, and gain selection

The feedback controller is implemented as a discrete-time PI controller executed every sampling period T_s . At each update the decoded demand $r[k]$ and velocity $y[k]$ form the tracking error $e[k]=r[k]-y[k]$. Zero steady-state error is achieved by integral action, updating $I[k]=I[k-1]+e[k]T_s$, so any persistent offset is accumulated until the control input removes it. The controller output is limited to the actuator range before encoding and transmission, representing physical throttle/brake constraints (Åström and Murray, 2016).

A key trade-off in gain selection is between velocity response and control effort. Increasing K_p reduces rise time and improves tracking but increases peak throttle/brake demand and can introduce overshoot or oscillation. Increasing K_i removes steady-state error more quickly but reduces robustness to delay and quantisation and can make the response more oscillatory. While classical PID literature shows that derivative action can improve transient response, in digitally implemented automotive control loops its practical benefit may be outweighed by sensitivity to quantisation and delay; therefore, derivative action was not used here ($K_d = 0$) and a PI structure was judged more appropriate for robustness and implementation simplicity (Åström and Murray, 2016).

Gains were selected by tuning on a baseline case (16-bit signals, $T_s=0.05$ s, zero delay). K_p was increased until tracking was sufficiently fast without sustained oscillation, then K_i was increased until the steady-state error became negligible while keeping the control effort acceptable. The final gains ($K_p=1.5$, $K_i=0.2$, $K_d=0$) achieved stable tracking with moderate control demand ($\max|u|\approx 73.1$) and IAE after 10 s ≈ 211.5 .

In addition, actuator saturation was handled using a simple anti-windup strategy: the integral state was only updated when the unsaturated controller output remained within the actuator limits. This prevents the integrator from accumulating excessive error during saturation, which would otherwise worsen overshoot and recovery and reduce robustness during large demand changes.

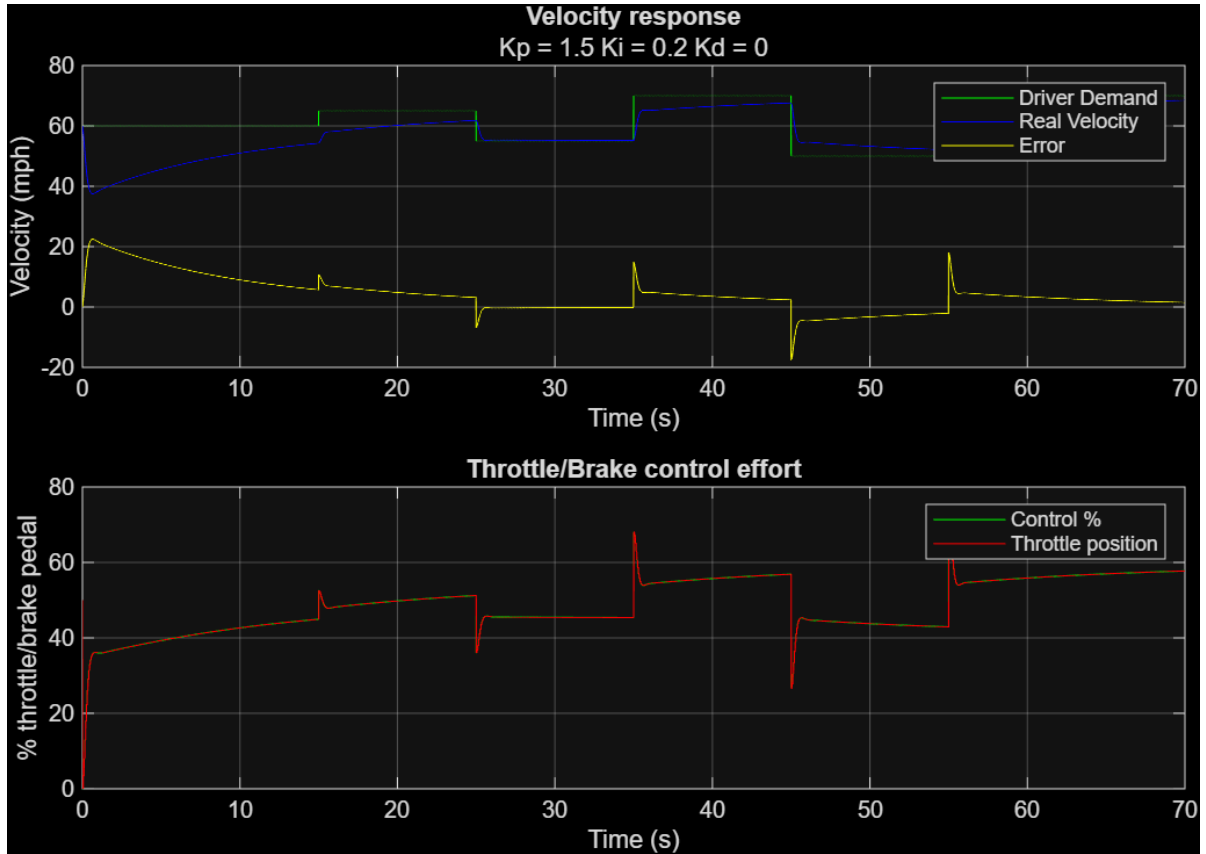


Figure 1. Closed-loop response of the digital cruise controller (top: demand, velocity and error; bottom: control effort and throttle position) for $K_p=1.5$, $K_i=0.2$, $K_d=0$, $T_s=0.05$ s, 16-bit signals and zero delay.

5 Bit-depth study

The velocity sensor, driver demand, and actuator command are exchanged as quantised n -bit bus messages. Reducing the bit depth increases the quantisation step size (approximately $\Delta \approx \text{range}/(2^n - 1)$), which reduces measurement and actuation resolution. In closed loop, coarse quantisation can introduce small limit cycles and increased actuator switching (“control chatter”), even when average tracking error remains similar. (Phillips, 1973).

A bit-depth sweep was performed from 16 bits down to 4 bits with fixed controller gains ($K_p=1.5$, $K_i=0.2$, $K_d=0$), sampling time $T_s=0.05$ s, and zero delay. Performance was assessed using peak tracking error, integral absolute error (IAE), and the total variation of the control signal $TV(u)=\sum|\Delta u|$ as a chatter indicator. Tracking metrics changed only modestly down to 8 bits, but $TV(u)$ increased rapidly below 8 bits, indicating excessive switching. Therefore, the minimum bit depth that still guarantees reasonable performance was 8 bits.

bits	maxErr (mph)	maxU (%)	IAE	TV(u)
16	17.937	73.123	211.55	187.13
12	17.943	73.138	211.67	200.15
10	17.924	73.216	211.85	233.63
8	18.073	73.333	214.70	358.43
6	18.764	74.603	218.38	1031.70
4	19.302	86.667	211.05	3453.30

Table 1. Bit-depth sweep results (metrics computed for $t \geq 10$ s).

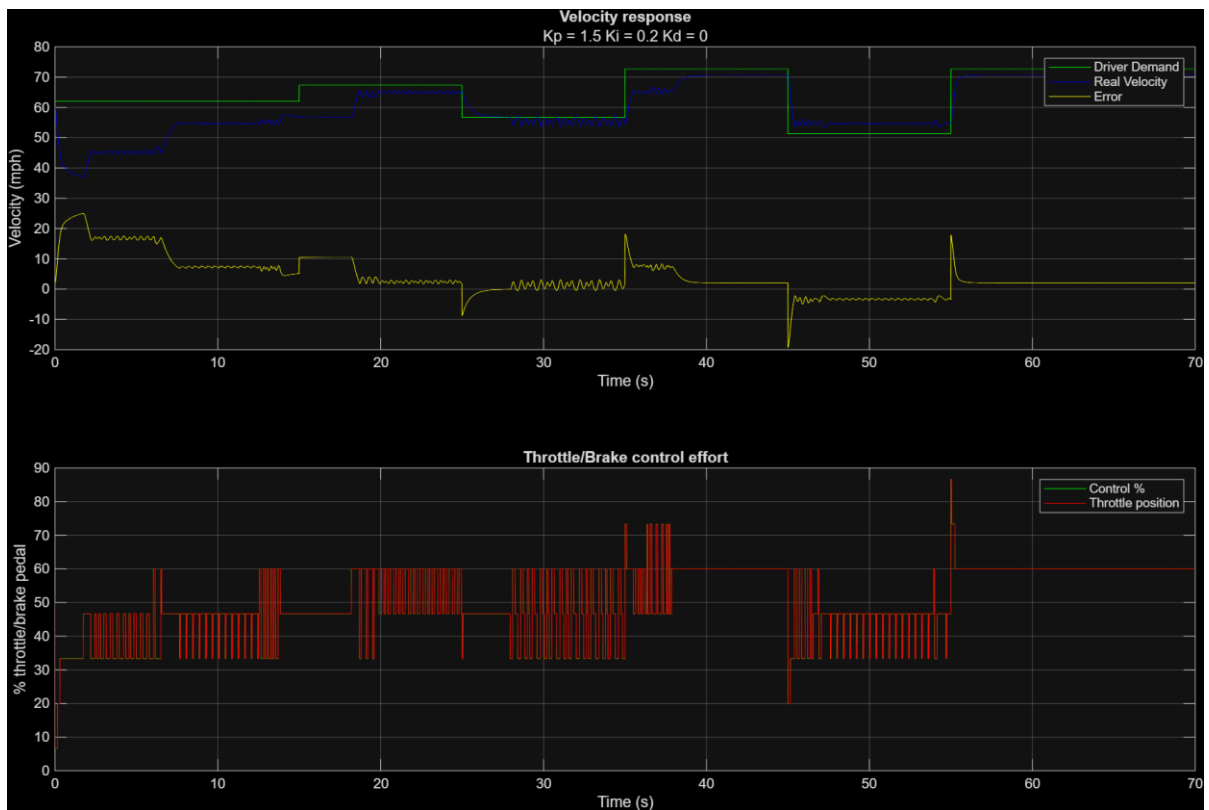


Figure 2. 4-bit response showing quantisation-induced control chatter ($K_p = 1.5$, $K_i = 0.2$, $K_d = 0$, $T_s = 0.05$ s).

6 Sampling-time study

The controller is executed in discrete time and the actuator command is updated every sampling period T_s , while the vehicle dynamics are simulated at a smaller step (sim_step). Increasing T_s reduces the update rate, which typically increases effective delay and reduces stability margin; however, smaller T_s increases CPU and bus load in a real ECU. The choice of T_s is therefore a practical trade-off between closed-loop responsiveness, computational burden and communication bandwidth (Åström and Murray, 2008).

A sampling-time sweep was performed from $T_s=0.01$ s to $T_s=0.40$ s with fixed gains ($K_p=1.5$, $K_i=0.2$, $K_d=0$), 16-bit signals and zero network delay. Over this range, tracking metrics changed only slightly: $\max|e|$ remained close to 18 mph and peak control remained below 75%. Control variation $TV(u)$ increased gradually with T_s (less smooth control due to longer hold between updates). The longest sampling time that still guaranteed reasonable performance in the tested range was $T_s=0.40$ s (largest tested).

T_s (s)	maxErr (mph)	maxU (%)	IAE	TV(u)
0.01	17.930	72.967	212.70	184.87
0.02	17.932	73.007	212.43	185.19
0.05	17.937	73.123	211.55	187.13
0.10	17.947	73.312	210.07	191.58
0.20	17.966	73.693	207.33	204.11
0.30	18.025	74.047	205.56	220.19
0.40	17.997	74.456	202.50	242.59

Table 2. Sampling-time sweep results (metrics computed for $t \geq 10$ s).

7 Network-delay study

Communication or actuator delay introduces dead-time into the feedback loop, reducing phase margin and making the controller react to outdated information. As delay increases, the response becomes less well-damped and control effort becomes more variable; beyond a critical delay, oscillation and saturation can occur. Consistent with dead-time control theory, delay primarily erodes dynamic robustness rather than steady-state accuracy, which is why instability-like behaviour first appears as oscillation and rising control variation before complete loss of tracking (Bemporad, Heemels and Johansson, 2010; Normey-Rico and Camacho, 2007).

A delay sweep was performed with $T_s = 0.05$ s, 16-bit signals and fixed gains ($K_p = 1.5$, $K_i = 0.2$, $K_d = 0$). Up to about 0.25 s delay the tracking metrics changed little, but $TV(u)$ increased steadily, indicating reduced smoothness. A more pronounced deterioration occurred between 0.50 s and 0.56 s: $\maxErr$ and IAE rose rapidly and peak control demand moved closer to saturation. Using the acceptance criteria

above, satisfactory performance was maintained up to approximately 0.50–0.54 s of delay, with the 0.54 s case considered borderline rather than cleanly acceptable.

The original teaching code is useful but simplified: the plant is idealised, delay is deterministic, and bus signals are noise-free. Gains were tuned empirically, so the results demonstrate sound simulation behaviour rather than production-ready calibration.

delay (s)	maxErr (mph)	maxU (%)	IAE	TV(u)
0.00	17.937	73.123	211.55	187.13
0.05	17.949	73.306	211.94	196.59
0.10	17.961	73.492	212.69	213.75
0.25	17.994	74.051	217.73	313.34
0.50	18.627	75.647	276.94	972.68
0.52	18.447	69.741	278.37	1072.90
0.54	20.528	73.150	284.28	1160.60
0.56	23.839	82.281	328.84	1404.30
0.58	28.960	90.228	436.15	1916.40
0.60	33.950	91.144	516.95	2242.30

Table 3. Network-delay sweep results (metrics computed for $t \geq 10$ s).

8 Extra implementation

Repeatability required resetting persistent states inside the plant and controller functions. This was handled by clearing function memory at the start of each run (clear functions) and by using integer-step timing ($N_s = \text{round}(T_s/\text{sim_step})$) to avoid floating-point sampling errors. These implementation details are important because they ensure that the bit-depth, sampling-time and delay sweeps are compared on a consistent basis rather than being influenced by residual controller state or timing drift.

Comfort metrics (acceleration and jerk): To assess smoothness beyond tracking error, longitudinal acceleration $a(t)$ and jerk $j(t)$ were estimated from the simulated velocity. Velocity was converted from mph to m/s, smoothed with a 0.10 s moving-average window, and differentiated using MATLAB's gradient operator. For the baseline case ($K_p=1.5$, $K_i=0.2$, $K_d=0$, $T_s=0.05$ s, 16-bit, delay=0) and for $t \geq 10$ s:

$\text{rms}(a)=1.98 \text{ m/s}^2$, $\text{p95}(|a|)=0.57 \text{ m/s}^2$, $\text{rms}(j)=12.54 \text{ m/s}^3$, and $\text{p95}(|j|)=8.29 \text{ m/s}^3$. RMS/percentile jerk was used because absolute maxima are dominated by short spikes at demand transitions. Higher TV(u) cases (low bit depth or large delay) are expected to increase jerk and reduce comfort. (ISO, 1997; MathWorks, n.d.).

Metric	Value ($t \geq 10 \text{ s}$)
$\text{rms}(a)$	1.98 m/s^2
$\text{p95}(a)$	0.57 m/s^2
$\text{rms}(j)$	12.54 m/s^3
$\text{p95}(j)$	8.29 m/s^3

Table 4. Comfort metrics derived from velocity (smoothed + gradient).

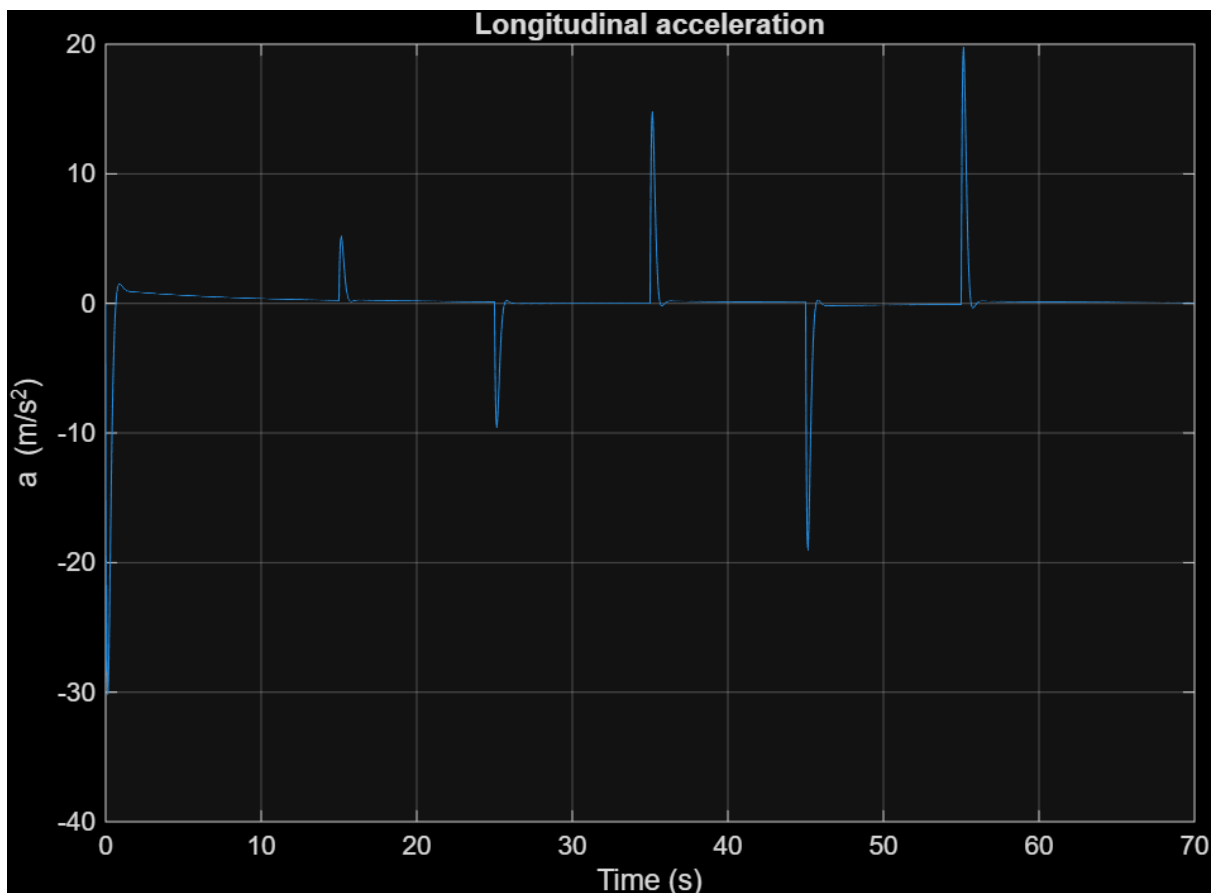


Figure 3. Estimated longitudinal acceleration from the simulated velocity trace (mph→m/s), using a 0.1 s moving-average and gradient-based differentiation.

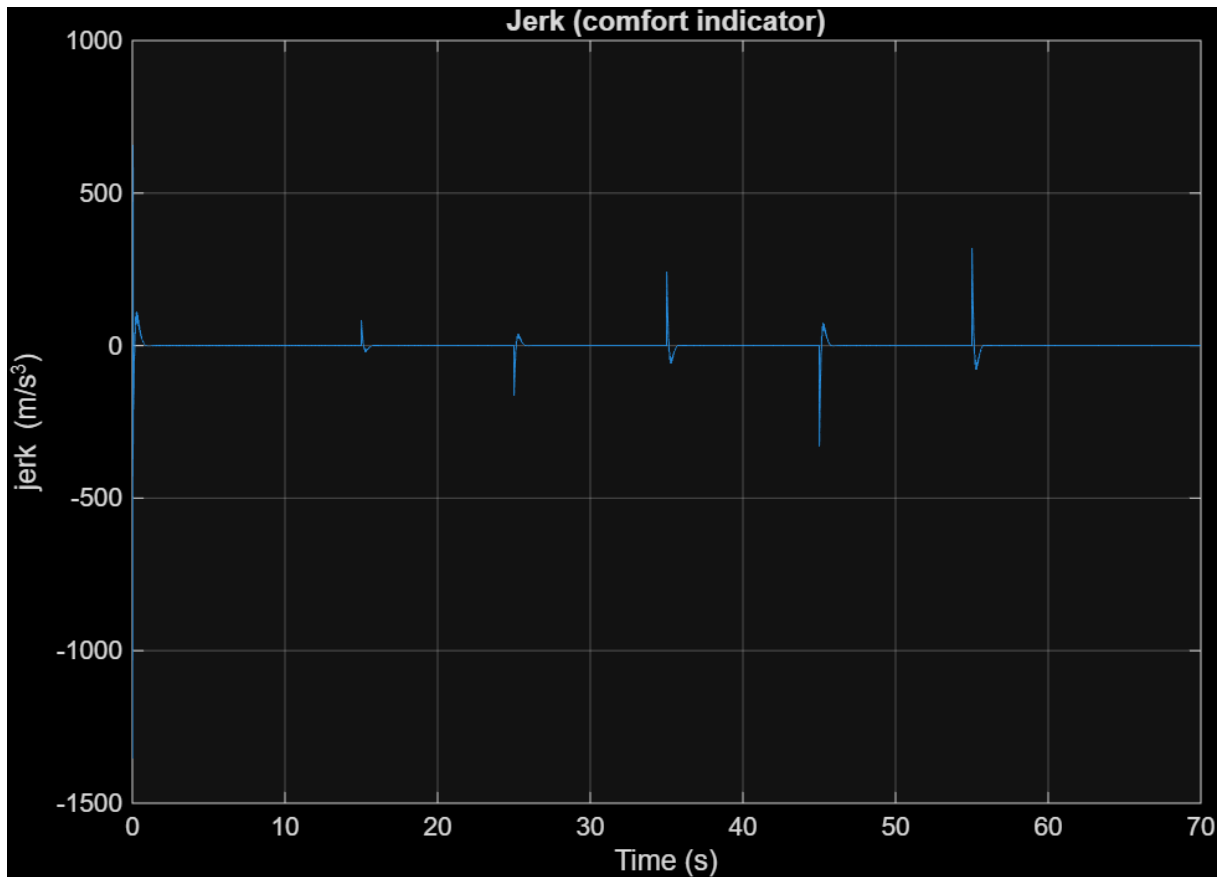


Figure 4. Estimated jerk from the simulated velocity trace (mph→m/s), using a 0.1 s moving-average and gradient-based differentiation.

9 Conclusion

A discrete-time PI cruise controller was implemented using binary encoding and decoding of the sensor, demand and actuator signals. With the tuned gains ($K_p = 1.5$, $K_i = 0.2$, $K_d = 0$), the system achieved stable tracking with moderate control effort and zero steady-state error. The quantisation study indicated that 8 bits was the minimum practical resolution before control chatter increased markedly. The sampling-time study showed acceptable behaviour up to $T_s = 0.40$ s within the tested range. The delay study showed that satisfactory performance was maintained up to roughly 0.50–0.54 s, beyond which tracking degradation and control variation increased rapidly. Overall, the results demonstrate the trade-offs between implementation constraints, robustness and closed-loop performance in digital automotive control systems.

10References

1. ISO (2018) ISO 15622:2018 Intelligent transport systems — Adaptive cruise control systems — Performance requirements and test procedures. Geneva: International Organization for Standardization. Available at: <https://www.iso.org/standard/71515.html> (Accessed: 7 April 2026).
2. ISO (2009) ISO 22179:2009 Intelligent transport systems — Full speed range adaptive cruise control (FSRA) systems — Performance requirements and test procedures (Withdrawn; replaced by ISO 15622:2018). Geneva: International Organization for Standardization. Available at: <https://www.iso.org/standard/40753.html> (Accessed: 7 April 2026).
3. ISO (1997) ISO 2631-1:1997 Mechanical vibration and shock — Evaluation of human exposure to whole-body vibration — Part 1: General requirements. Geneva: International Organization for Standardization. Available at: <https://www.iso.org/standard/7612.html> (Accessed: 7 April 2026).
4. Åström, K.J. and Murray, R.M. (2008) Feedback Systems: An Introduction for Scientists and Engineers. Princeton, NJ: Princeton University Press. Available at: https://www.cds.caltech.edu/~murray/books/AM08/pdf/fbs-public_15Oct16.pdf (Accessed: 7 April 2026).
5. Åström, K.J. and Murray, R.M. (2016) 'PID Control'. In: Feedback Systems (online chapter). Available at: https://www.cds.caltech.edu/~murray/books/AM05/pdf/fbs-pid_08Nov16.pdf (Accessed: 7 April 2026).
6. Bemporad, A., Heemels, W.P.M.H. and Johansson, M. (eds.) (2010) Networked Control Systems. London: Springer. Available at: https://ceid.utsa.edu/ataha/wp-content/uploads/sites/38/2017/10/NCS_Book.pdf (Accessed: 7 April 2026).
7. Normey-Rico, J.E. and Camacho, E.F. (2007) Control of Dead-time Processes. London: Springer. (ISBN 978-1-84628-829-6).
8. Phillips, C.L. (1973) 'Limit cycles resulting from quantization in digital control systems'. NASA Technical Reports Server. Available at: <https://ntrs.nasa.gov/citations/19730050840> (Accessed: 7 April 2026).

9. MathWorks (n.d.) MATLAB documentation: bin2dec, dec2bin, movmean and gradient. Available at: <https://www.mathworks.com/help/matlab/> (Accessed: 7 April 2026).
10. Ogata, K. (2010) Discrete-Time Control Systems. 2nd edn. Upper Saddle River, NJ: Pearson.