



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

Student Name: Abhishek Hasmukhbhai Vagadiya

ID number: Course: 2410890

Course: Motorsport Engineering

Module name: Energy Storage & Control System

Module code: ACEA7007

Assignment: PDI cruise control system

Index

Introduction

Encoding and Decoding of Signals

Controller Design and Implementation

System performance

Effect of Bit Resolution

Effect of Sampling Time

Conclusion

References

1. Introduction

PID controllers are highly popular in the automotive control industry since the control can offer a stable, precise and efficient control of dynamic processes. The aim of the cruise control systems in the case of cruise control is to control the vehicle speed at a desired value in the presence of disturbances like aerodynamic drag, rolling resistance and fluctuations in driver demand. The aim of the project is to model and implement a digital cruise control system in MATLAB and particularly encode and decode signal transmission, design and implement controller and test its performance.

The system replicates a vehicle in which the speed is measured with a sensor, handled through a controller, and fed into actuator control directions of throttle or braking. Binary signals are used to provide communication between system components, and this is similar to a real automotive communication bus. The overall goal is to make sure that there is correct monitoring of demand of the velocity of the driver along with remain stable and efficient of the system. This report has been a critical analysis of the implemented system and the evaluation of its performance in various circumstances.

2. Encoding and Decoding of Signals

Every signal in this system is represented as binary values, and there is a need to transform between the digital and physical worlds. This is done in the MATLAB code, which uses linear scaling associations to do the decoding and encoding.

The result of the velocity sensor is obtained by the raw velocity sensor which is returned by the function `getRawVelocity` and is a binary string that gives the speed of the vehicle.

This binary value is decoded into a physical velocity using the expression:

$$v = \frac{X_{raw}(M_v - m_v)}{R} + m_v$$

Where; $M_v=150$, $m_v=0$, and $R=2^n$. In MATLAB, this is implemented as:

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

This allows the controller to be run on real-world units (mph) that is needed to be meaningfully controlled.

Likewise, the signal of demand of the driver is received with the help of the function `getRawVelocityDemand()`. The raw binary signal is decoded using:

$$d = \frac{X_{raw}(M_d - m_d)}{R} + m_d$$

where the demand range is 30–110 mph. This was one of the major corrections made on the code since an incorrect assignment of a constant was changed with a correct scaling equation. After calculating the control signal, one has to encode it back into a binary form and then send it to the actuator. The inverse mapping is used to do this. This is achieved using the inverse mapping:

$$X_{raw} = \frac{(u - m_u)R}{M_u - m_u}$$

Where, $u \in [-100, 100]$. In MATLAB, this is implemented as:

```
control_raw_dec = ((control - m_ControlSig) * R) / (M_ControlSig - m_ControlSig);
control_raw = dec2bin(round(control_raw_dec), n_bits_actuator_signal);
```

This encoding ensures compatibility with the communication system and accurate actuator response.

3. Controller Design and Implementation

The controller is instantiated in the simulation loop, and it is run with discrete time intervals of the sampling time T_s . The control logic is triggered using:

```
if mod((j-1)*sim_step, Ts) == 0
```

This architecture is a replica of a real-time embedded control system in which periodic updates are taken.

The error signal is defined as:

$$e(k) = d(k) - v(k)$$

where $d(k)$ is the demand and $v(k)$ is the measured velocity. The PID controller computes the control signal as:

$$u(k) = K_p e(k) + K_i \sum e(k) T_s + K_d \frac{e(k) - e(k-1)}{T_s}$$

In MATLAB, this is implemented as:

```
error = demand - velocity;
integral_error = integral_error + (error * Ts);
derivative_error = (error - last_error) / Ts;

control = (P * error) + (I * integral_error) + (D * derivative_error);
last_error = error;
```

The proportional term enhances the responsiveness, the term of the integral guarantees the zero steady-state error and the term of the derivative reduces the oscillations and increases the stability. It is also important to note the inclusion of the integral term as it ensures that the error in the steady-state is near to zero:

$$\lim_{t \rightarrow \infty} e(t) = 0$$

To ensure realistic operation, the control signal is constrained within actuator limits:

```
control = max(min(control, 100), -100);
```

This reflects physical limitations and prevents instability due to excessive control actions.

4. System performance

The system had a low performance, which was caused by poor implementation of key components before correcting the code. The signal of demand was not correctly decoded and the control algorithm was not executed and the control signal resulted in a constant or

unrealistic value. Consequently, the vehicle speed was not responsive to the requirement of the driver and a considerable steady-state error was experienced. The control effort also lacked consistent nature which indicated the lack of an effective feedback mechanism. This shows that proper signal processing and controller realisation is critical in attaining the desired behaviour of the system (Mutambara, 2024).

Graph 1: Velocity response and control effort before implementing correct decoding, PID control, and encoding

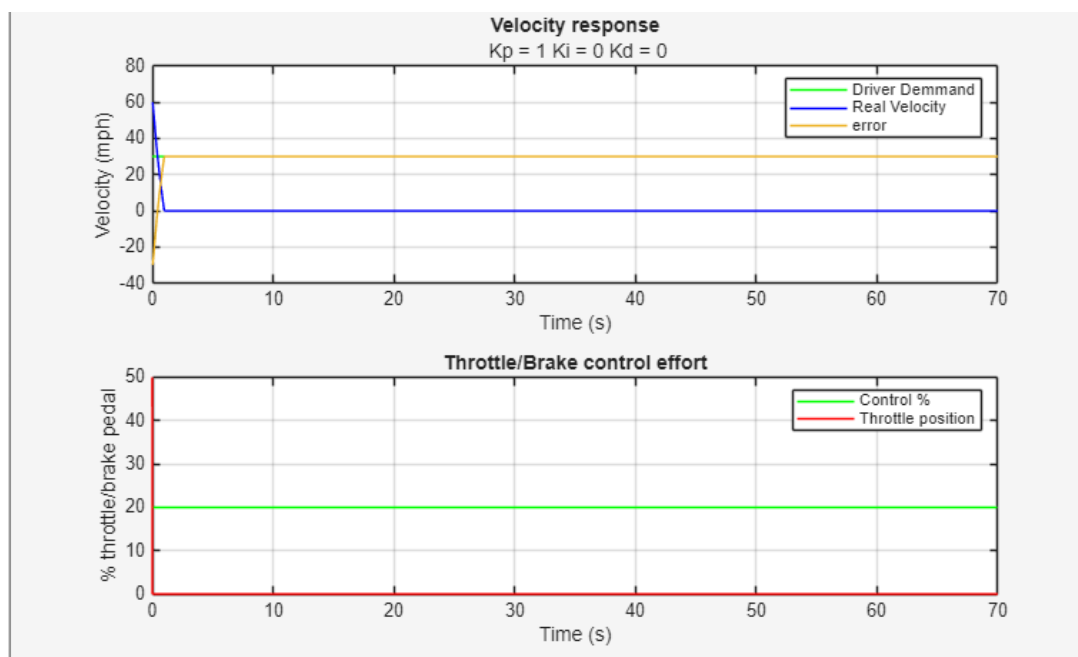


Figure 4.1 Velocity response and control effort before implementing correct decoding, PDI control, and encoding

Once the code was fixed, the system becomes significantly more effective. The velocity reaction is closely correlated with the demand of the driver and has a small delay and inaccuracy in the steady state. The temporary reaction is smooth in nature with small overshoot and gentle movement to targeted value. This proves the efficiency of the PID controller in controlling the system.

The control effort graph indicates that the actuator reacts well to the demand changes with more throttle when there is acceleration and less or negative throttle when there is deceleration. The

control signal is kept between acceptable values and it does not show excessive oscillations, which shows efficient and stable control (Jafaripournimchahi et al., 2022).

These findings indicate the conflict between the velocity response and the control effort. Although more aggressive control action is needed to achieve faster response, the selected controller achieves a balanced performance, which guarantees accuracy and efficiency.

Graph 2: Velocity response and control effort after implementing proper decoding, PID control, and encoding

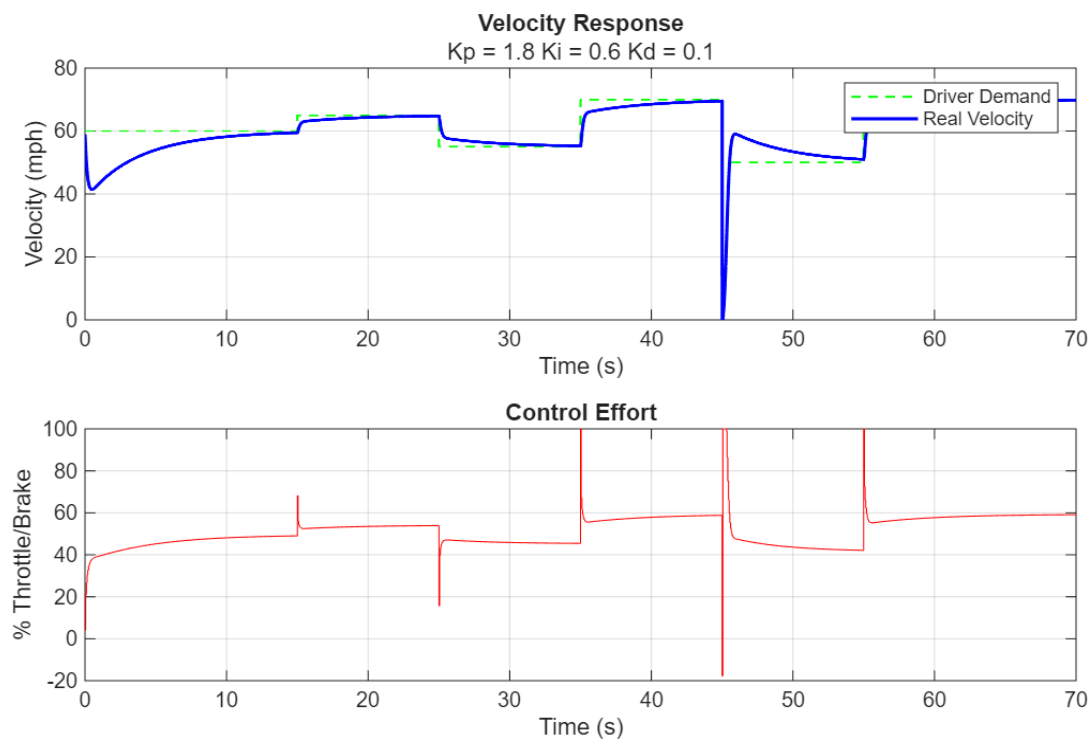


Figure 4.2 Velocity response and control effort after implementing proper decoding, PID control, and encoding

5. Effect of Bit Resolution

The resolution and accuracy of the system is determined by the number of bits that are used to encode the signals. An increased number of bits enables a smaller quantisation error as a result of the ability to finely represent values. The 16-bit encoding allows the system to have much accuracy creating a smooth and accurate control (Wang et al., 2025).

A decrease in the number of bits will also introduce a quantisation effect, which may lower the performance through the introduction of step-like responses and lower accuracy. This could cause oscillation or instability especially in sensitive controls. However, higher bit resolution increases computational load and data transmission requirements. Consequently, accuracy and efficiency have to be comparable. It was noted that reasonable bit resolutions still can deliver satisfactory performance, but values that are too low have a severe impact on the system behaviour.

6. Effect of Sampling Time

The sampling time defines the frequency with which the controller moves its output. Less sampling time becomes more responsive and more accurate when the controller is able to respond faster to the changes in the system. A sampling time of 0.05 seconds is a good compromise between the performance and the computational efficiency in this implementation (Schwenzer et al., 2021).

The longer the sampling time, the slower the frequency of updating, and hence delays in responding, as well as reduced accuracy. This can cause instability in the extreme case since the controller will have inadequate response to changes within the system. On the other hand, extremely small sampling time also demand large computations and might not be feasible in the systems. Moreover, sampling rates may cause noise increase. Thus, sampling time should be selected in accordance with system dynamics as well as practical constraints.

Effect of Communication Delay

The frequency at which the controller turns its output is determined by the sampling time. The controller can react quicker to the changes of the system; the less time is spent on sampling, the more sensitive and precise the sampling process becomes. The performance of such an

implementation is good, but the computational efficiency is poor, especially in cases where the sampling time is less than 0.05.

The more time the sampling, the lower the rate of updating and therefore lag in responding and inaccuracy. This may bring instability in the extreme case as the controller will lack sufficient response to changes in the system. Alternatively, sampling time is very small and this too requires big computations and may not be possible within the systems. Besides, noise can be increased by sampling rates. Sampling time, therefore, must be chosen in accordance to system dynamics besides feasible considerations (Kebria et al., 2018).

7. Conclusion

This project implemented and analysed a digital cruise controller using a PID controller in MATLAB and was successful. The appropriate encoding/decoding functions implementation was the reason of proper communication within the system components whereas the PID controller was used to provide effective control of vehicle velocity. The addition of the integral term made the steady-state error to be zero which is the main requirement of the system.

When the correction of the code was done, the comparison of the system performance before and after the correction revealed the significance of the correct implementation. The fixed system was characterized by the proper tracking, the stable behaviour and the effective control effort. Moreover, bit resolution, sampling time, and communication delay analysis revealed that there are trade-offs during the design of control system.

Comprehensively, the findings prove that in automotive situations careful design and tuning of control systems are critical to optimal performance.

8. References

- Mutambara, A.G., 2024. Design and analysis of control systems: Driving the Fourth Industrial Revolution. CRC press.
- Jafaripournimchahi, A., Cai, Y., Wang, H., Sun, L. and Weng, J., 2022. Integrated-hybrid framework for connected and autonomous vehicles microscopic traffic flow modelling. *Journal of Advanced Transportation*, 2022(1), p.2253697.
- Wang, H., Wang, B., Chen, M. and Wu, L., 2025. A Power Efficient 14-Bit SS ADC With Level Encoding TDC for CMOS Image Sensors. *IEEE Sensors Journal*, 25(24), pp.43956-43964.
- Schwenzer, M., Ay, M., Bergs, T. and Abel, D., 2021. Review on model predictive control: An engineering perspective. *The international journal of advanced manufacturing technology*, 117(5), pp.1327-1349.
- Kebria, P.M., Abdi, H., Dalvand, M.M., Khosravi, A. and Nahavandi, S., 2018. Control methods for internet-based teleoperation systems: A review. *IEEE Transactions on Human-Machine Systems*, 49(1), pp.32-46.