

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 & Control Systems, ACHA7007

Anthony Cook

2515979

Date (10/04/2026)

Contents

1. Decoding & Encoding.....	4
2. PID Controller	5
2.1. Control Function	7
2.2. Proportional Gain Tuning.....	8
2.3. Integral Gain Tuning.....	12
2.4. Differential Gain Tuning.....	16
3. Bit Rate Study.....	20
4. Sampling Time Study	24
5. Delay Study.....	33
6. Discussion & Conclusions	35
6.1. Lack of Anti-Windup	35
6.2. Control Function Problems.....	35
7. Future Work	36
7.1. Tuning Automation.....	36
8. References	37
9. Appendices.....	38

Figure 1 - Control System Response.....	6
Figure 2 - Control Function Code	7
Figure 3 - Velocity & error response for proportional (P) tuning at $t = 2$ s	9
Figure 4 - Velocity & control effort response for proportional (P) tuning at $t = 2$ s	9
Figure 5 - Velocity & error response for proportional (P) tuning at $t = 10$ s	10
Figure 6 - Velocity & control effort response for proportional (P) tuning at $t = 10$ s	10
Figure 7 - Velocity & error response for proportional (P) tuning at $t = 70$ s	11
Figure 8 - Velocity & control effort response for proportional (P) tuning at $t = 70$ s	11
Figure 9 - Velocity & error response for Integral (I) tuning at $t = 2$ s	13
Figure 10 - Velocity & control effort response for Integral (I) tuning at $t = 2$ s	13
Figure 11 - Velocity & error response for Integral (I) tuning at $t = 10$ s	14
Figure 12 - Velocity & control effort response for Integral (I) tuning at $t = 10$ s	14
Figure 13 - Velocity & error response for Integral (I) tuning at $t = 70$ s	15
Figure 14 - Velocity & control effort response for Integral (I) tuning at $t = 70$ s	15
Figure 15 - Velocity & error response for Differential (D) tuning at $t = 2$ s	16
Figure 16 - Velocity & control effort response for Differential (D) tuning at $t = 2$ s	17
Figure 17 - Velocity & error response for Differential (D) tuning at $t = 10$ s	17
Figure 18 - Velocity & control effort response for Differential (D) tuning at $t = 10$ s	18
Figure 19 - Velocity & error response for Differential (D) tuning at $t = 70$ s	18
Figure 20 - Velocity & control effort response for Differential (D) tuning at $t = 70$ s	19
Figure 21 - Velocity & error response for Bit Rate at $t = 10$ s	21
Figure 22 - Velocity & control effort response for Bit Rate at $t = 10$ s	21
Figure 23 - Velocity & error response for Bit Rate (without 1, 2, & 4) at $t = 10$ s	22
Figure 24 - Velocity & control effort response for Bit Rate (without 1, 2, & 4) at $t = 10$ s	23
Figure 25 - Velocity & error response for Sampling Time at $t = 2$ s	24
Figure 26 - Velocity & control effort response for Sampling Time at $t = 2$ s	25
Figure 27 - Velocity & error response for Sampling Time at $t = 10$ s	26
Figure 28 - Velocity & control effort response for Sampling Time at $t = 10$ s	26
Figure 29 - Velocity & error response for Sampling Time at $t = 70$ s	27
Figure 30 - Velocity & control effort response for Sampling Time at $t = 70$ s	28
Figure 31 - Velocity & error response for Sampling Time at $t = 28 - 44$ s	29

Figure 32 - Velocity & control effort response for Sampling Time at $t = 28 - 44$ s.....	29
Figure 33 - Velocity & error response for Sampling Time at $t = 33 - 39$ s.....	30
Figure 34 - Velocity & control effort response for Sampling Time at $t = 33 - 39$ s.....	30
Figure 35 - Velocity & error response for Sampling Time at $t = 40 - 54$ s.....	31
Figure 36 - Velocity & control effort response for Sampling Time at $t = 40 - 54$ s.....	31
Figure 37 - Tuned System, Delay = 0s.....	33

Table 1 - PID Variables	5
Table 2 - Tested P Values	8
Table 3 - Tested D Values.....	12
Table 4 - Tested D Values.....	16
Table 5 - 0s Delay Values.....	33
Table 6 - Tuned System, Delay = 0.02s.....	34
Table 7 - 0.02s Delay Values	34

1. Decoding & Encoding

It is necessary to decode the raw values for both the velocity and demand signals, to achieve this, the following equation was used:

$$X_{Physical} = \frac{X_{Raw} * (M - m)}{R} + m$$

Equation 1 - Decode

Where:

$X_{Physical}$ = Decoded "Real World" Signal

X_{Raw} = Encoded Signal

M = Max Sensor Value

m = Min Sensor Value

R = Resolution = $2^n - 1$

n = Number of Bits of Signal

As well as this equation, a conversion from binary to decimal is required to enable the code to perform calculations with the physical values, this gave the MATLAB equations:

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

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

In addition to decoding the velocity and demand signals, encoding the physical control signal is required, this was achieved using:

$$X_{Raw} = \frac{(X_{Physical} - m) * R}{M - m}$$

Equation 2 - Encode

Converting the new value to binary and then to string is necessary to allow the "sendControlCommand" function to work, giving the MATLAB equation:

`RawCon = num2str(dec2bin(((control-m_ControlSig).*R)/(M_ControlSig-m_ControlSig)))`

2. PID Controller

Table 1 - PID Variables

Variable	Definition	Tuning Effect
Proportional (P)	Produces a proportional control action to the error signal. Larger error value will generate larger output response	Typically produces a steady-state offset. As P increases, overshoot and oscillations can occur but rise time reduces
Integral (I)	Produces a proportional control action to the error accumulated over time	Reduces the steady-state error, however, response may slow with increased overshoot and stability issues
Derivative (D)	Produces a proportional control action to the error rate of change	Increases damping and stability, however, is noise sensitive. Settling time and overshoot is reduced as D increases
Rise Time (t_r)	The time taken for the response to increase from 10-90% of the steady-state value	Decreased I or increased P are likely to produce lower rise, however, extreme D values may slow it
Settling Time (t_s)	The time taken for the response to settle within a predetermined percentage of the steady-state value (typically 2% or 5%)	High P & I values can produce oscillations and so longer settling time. With the correct value for D, settling time decreases
Peak Overshoot (%)	Max amount passed the desired value that the response exceeds	Higher P&I values yield increased peak, with

		increases in D reducing the peak
Steady-State Error	The difference between the desired value and the actual when the system has settled	As P increases error goes down, with the addition of some I value eliminating the majority. D has very little effect

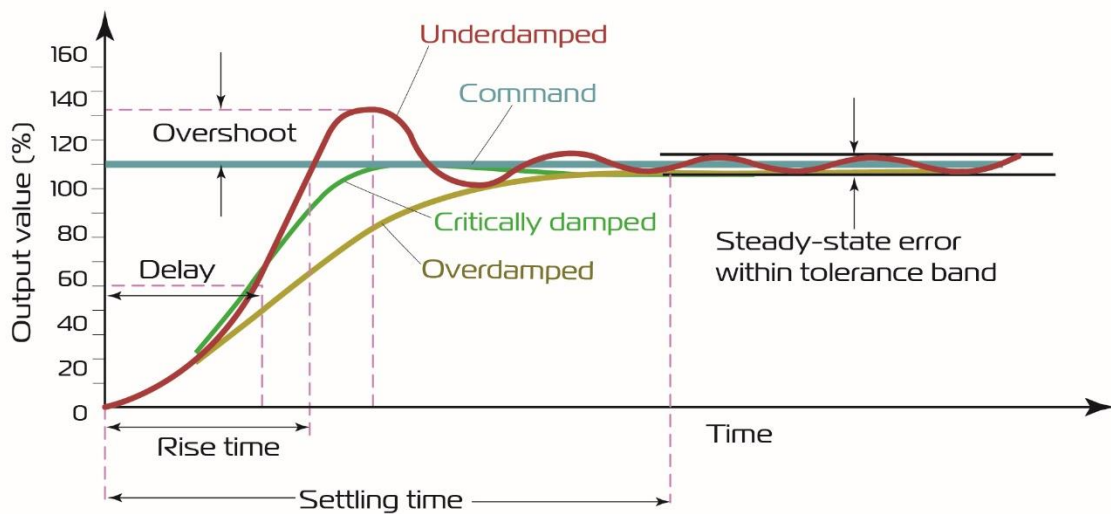


Figure 1 - Control System Response

For the tuning of the PID, it was decided to use the basic “Trial & Error” method, this is a manual tuning with gain values being adjusted in an iterative manor based upon the system response. The reason this method was chosen was due to it’s simplicity, intuitiveness, and due to the nature of manually altering gain values, it was felt that more could be learned about the system as relationships were discovered throughout the process.

- 1) To begin all 3 gains are set to 0
- 2) P is altered until the systems response is quick but can contain some oscillation.
- 3) I is then modified with the goal of removing the steady-state error.
- 4) D is increased to reduce any overshoot and stabilise the response.

While this is simple, it can be very time consuming and is rarely repeatable

2.1. Control Function

```
function [control,error] = calcController(P,I,D,demand,velocity_measured,Ts,M_ControlSig,m_ControlSig)

% ---Definition---
% P = Proportional Gain
% I = Integral Gain
% D = Differential Gain
% demand = Desired speed
% velocity = Actual speed (as seen by sensor)
% Ts = Sampling time
% M_ControlSig = Maximum actuator value (Full Throttle)
% m_ControlSig = Minimum actuator value (Full Brake)

persistent e_Previous e_integral
% These persistent variables store the named values between each time step
% They are also required for both the derivative term and accumulation of the integral

if isempty(e_Previous)
    e_Previous = 0;
    e_integral = 0;
end
% This sets the previous error and integral to 0 in the event a previous
% error value is absent (at the start of the simulation)

% ---Error---
error = demand - velocity_measured;
% Basic error calc

e_integral = e_integral + error*Ts;
% Integral term, this accumulates the past error to remove steady-state offset

e_derivative = (error-e_Previous)/Ts;
% Derivative term, this approximates the rate of change error
% providing damping & improved stability

control = P*error+I*e_integral+D*e_derivative;
% Unsaturated control value, combines the P,I,and D with their gain

control = max(min(control,M_ControlSig),m_ControlSig);
% Saturates the control value
% limiting the output to the physical limits of the actuator

e_Previous = error;
% Stores new error as previous error for the next iteration

end
```

Figure 2 - Control Function Code

2.2. Proportional Gain Tuning

Table 2 - Tested P Values

I=0, D=0		
P	Error Max	Error Mean
0	70	62.2067
0.1	70	62.2067
0.2	70	62.2067
0.3	70	62.2067
0.4	70	62.2067
0.5	70	56.1896
0.6	70	47.7152
0.7	60.1237	41.468
0.8	54.9762	37.3072
0.9	51.4262	34.1348
1	48.7025	31.5745
1.1	46.5053	29.438
1.2	44.6696	27.6151
1.3	43.1064	26.0335
1.4	41.7491	24.6437
1.5	40.5612	23.4095
1.6	39.5061	22.3043
1.7	38.5631	21.3068
1.8	37.7139	20.4013
1.9	36.9426	19.575
2	36.2399	18.8173
2.1	35.5945	18.1614
2.11	35.535	18.1154
2.12	35.4732	18.0717
2.121	35.4663	18.0676
2.122	35.4617	18.0634
2.123	35.4549	18.0592
2.124	35.448	18.0551
2.125	35.5598	18.0507
2.13	39.2197	18.0306
2.14	46.711	17.9942
2.15	50	17.9473
2.2	50	17.6296

These values of P were input and the max and mean error values for each recorded. At P=2.124 the system reached the lowest error values before large spikes began to appear and so it was decided to define this value going forward. The graphs created contain a shortened version of the list of values in order to improve clarity.

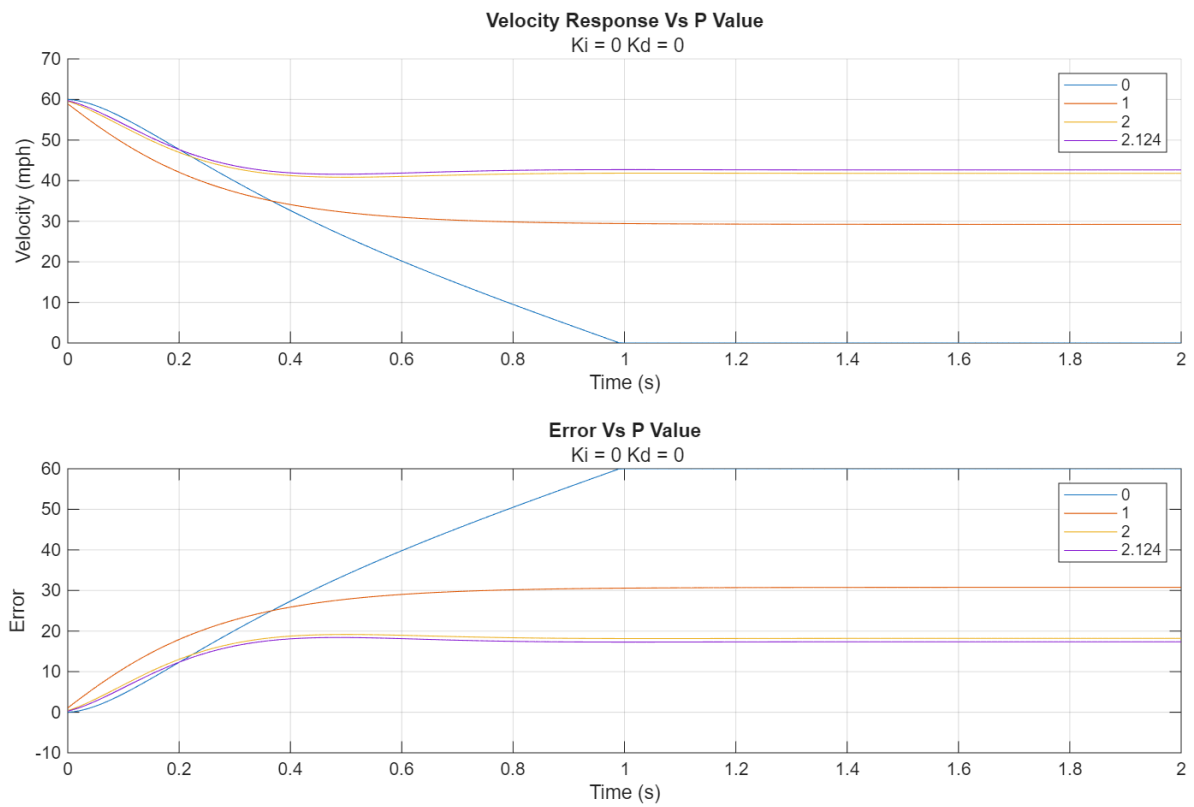


Figure 3 - Velocity & error response for proportional (P) tuning at $t = 2$ s

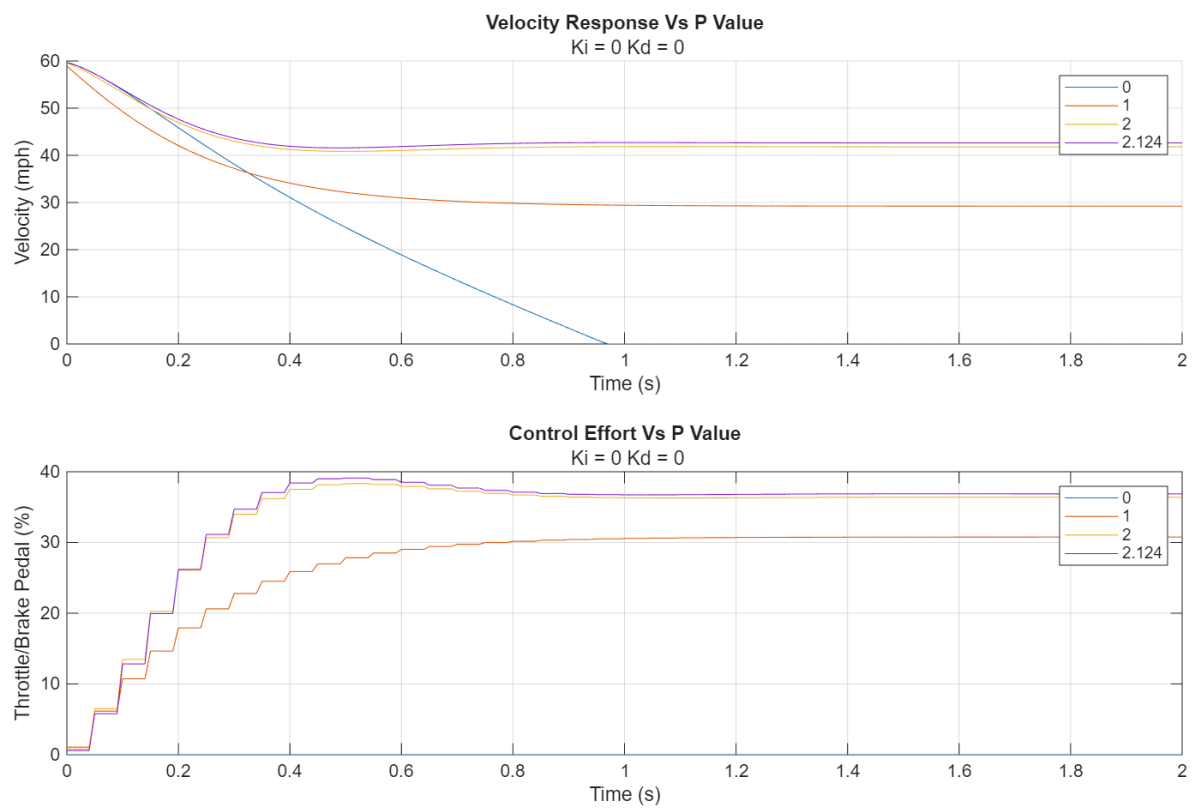


Figure 4 - Velocity & control effort response for proportional (P) tuning at $t = 2$ s

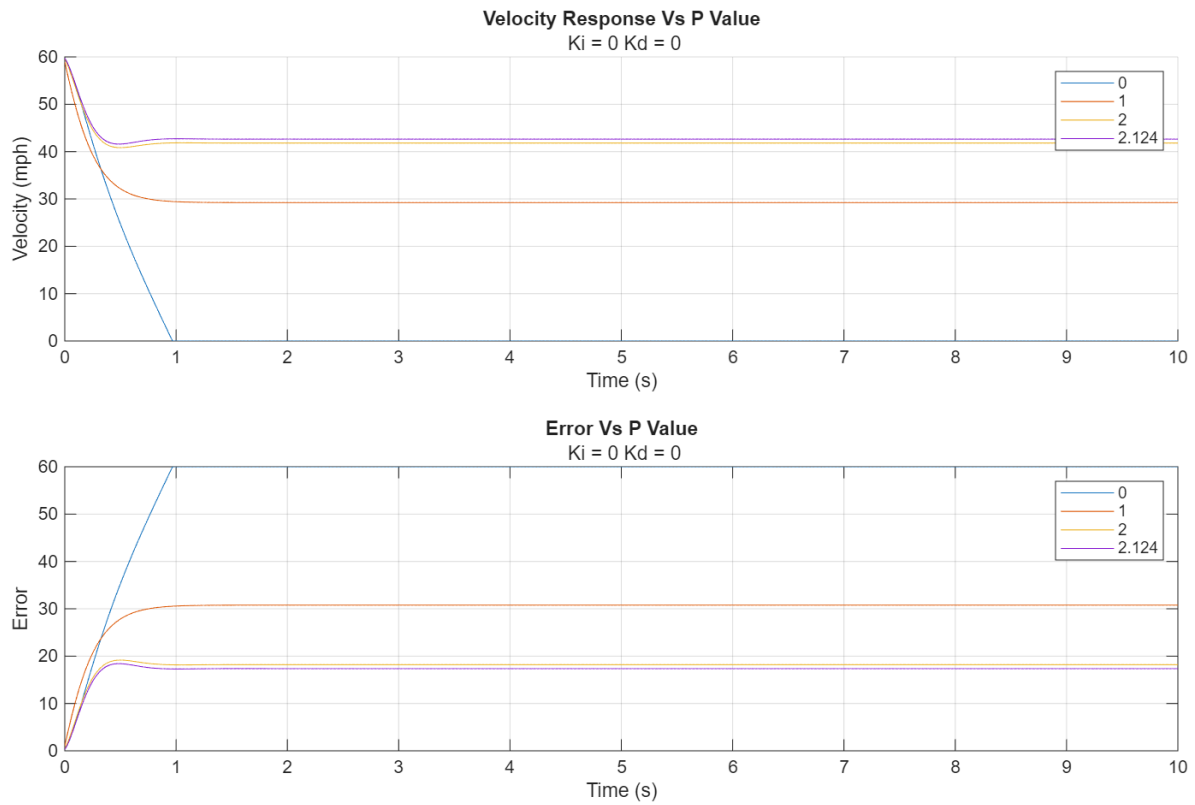


Figure 5 - Velocity & error response for proportional (P) tuning at $t = 10$ s

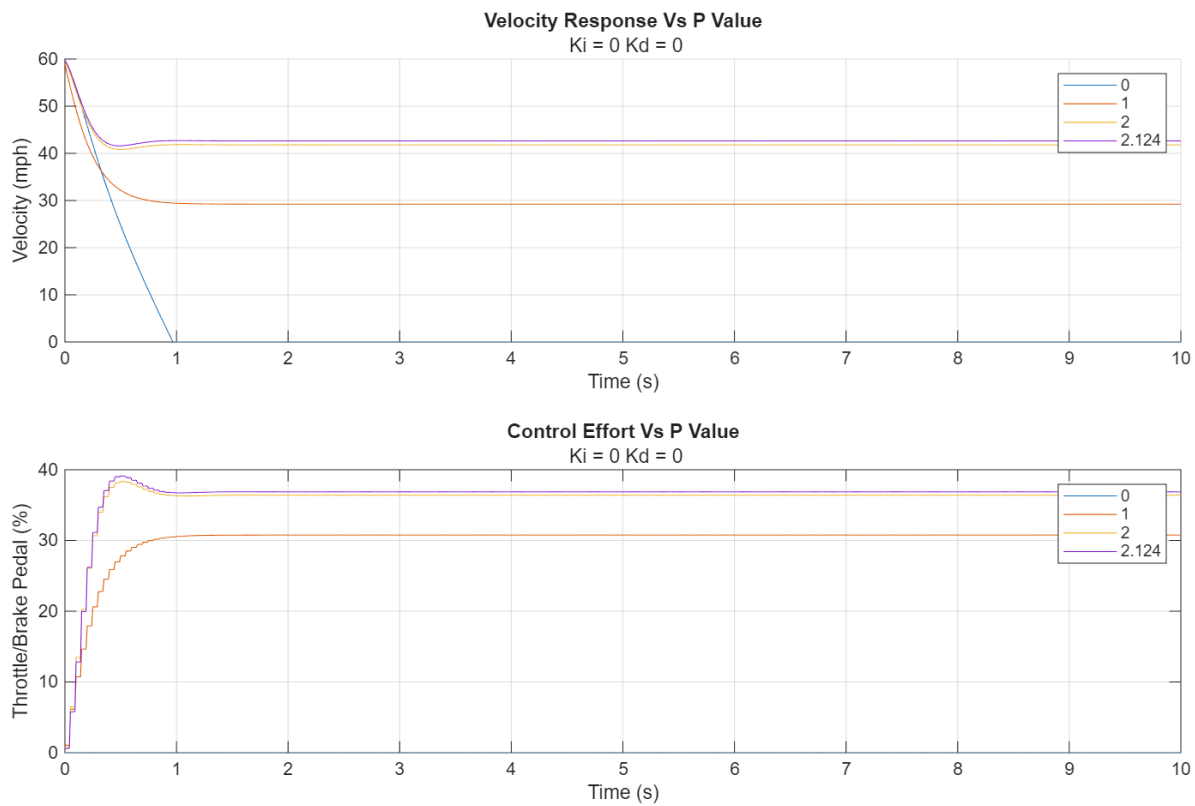


Figure 6 - Velocity & control effort response for proportional (P) tuning at $t = 10$ s

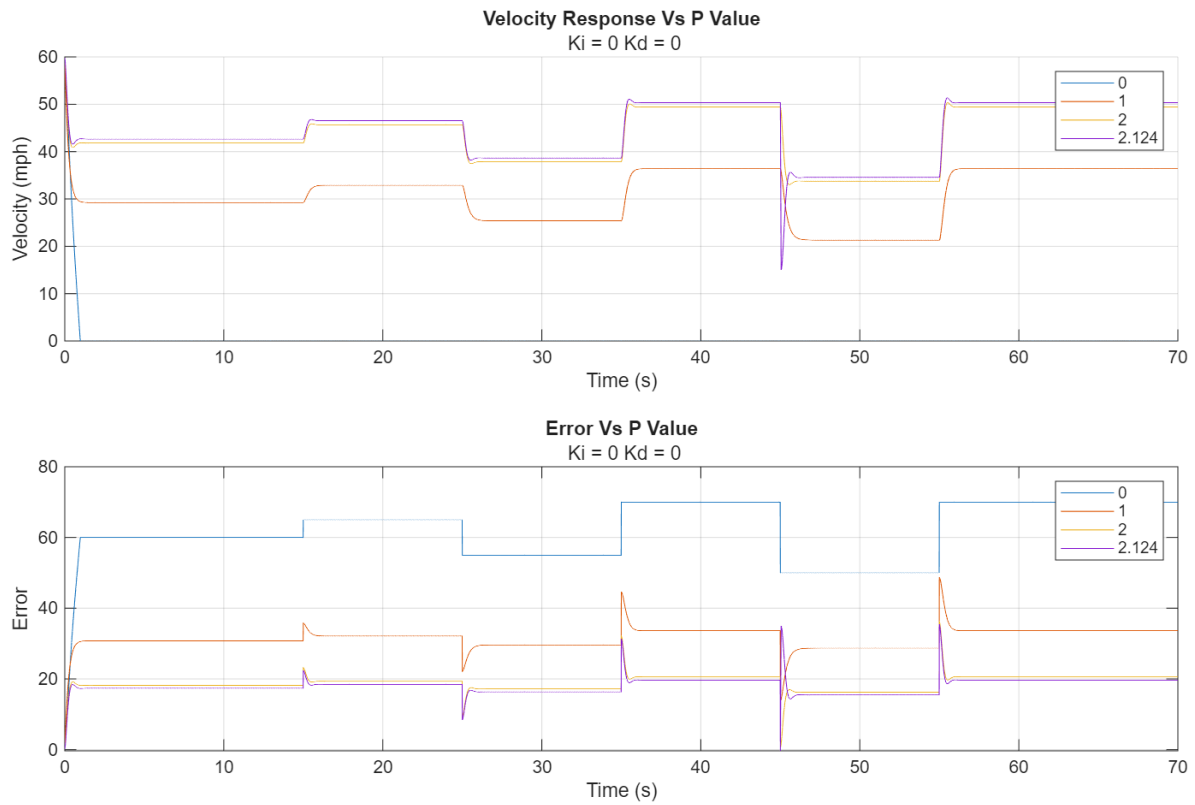


Figure 7 – Velocity & error response for proportional (P) tuning at $t = 70$ s

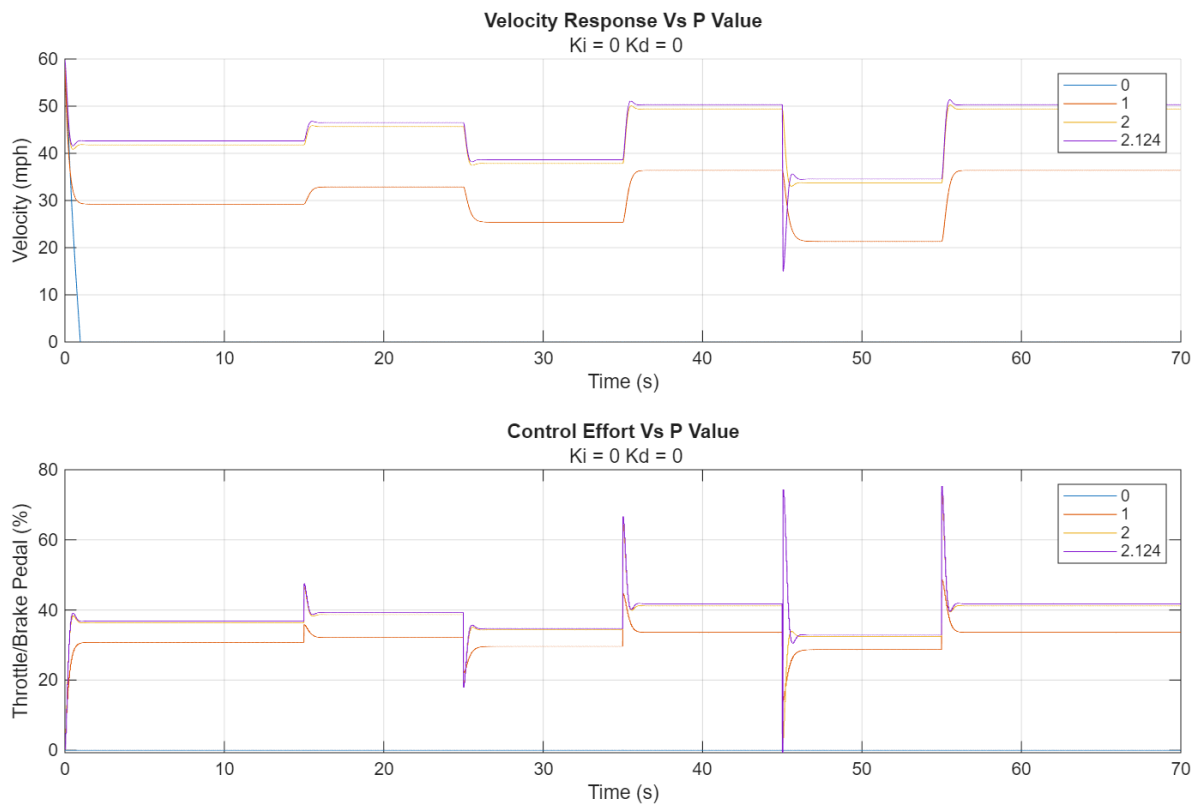


Figure 8 - Velocity & control effort response for proportional (P) tuning at $t = 70$ s

2.3. Integral Gain Tuning

Table 3 - Tested D Values

P=2.124, D=0		
I	Error Max	Error Mean
0	35.448	18.0551
0.1	20.145	6.6285
0.2	18.378	3.536
0.3	18.5016	2.299
0.4	18.8495	1.6602
0.5	19.1631	1.2715
0.6	19.4034	1.011
0.7	19.5842	0.8246
0.8	19.7124	0.6853
0.9	19.8039	0.578
1	19.868	0.4936
1.1	19.9115	0.4257
1.2	19.9413	0.37
1.3	19.9619	0.3243
1.4	19.9756	0.2861
1.5	19.9847	0.2539
1.6	19.9893	0.2266
1.7	19.9939	0.2031
1.8	19.9962	0.183
1.9	19.9985	0.1655
2	19.9985	0.1503
2.1	19.9985	0.137
2.2	19.9985	0.1254
2.3	20.0008	0.1151
2.4	19.9985	0.106
2.5	20.0008	0.098
2.6	19.9985	0.0908
2.7	20.0008	0.0845
2.8	20.0008	0.0788
2.9	20.0008	0.0736
3	20.0008	0.0691
3.1	19.9985	0.065
3.2	20.0008	0.0612
3.3	20.0008	0.0579
3.4	20.0008	0.0549
3.5	20.0008	0.0521
3.6	19.9985	0.0496
3.7	19.9985	0.0473
3.8	20.0008	0.0453
3.9	20.0008	0.0434
4	20.0008	0.0416

These values of D were input and the max and mean error values for each recorded. At

D=3.6 the system reached the lowest error values before large overshoot began to appear and so it was decided to define this value going forward. The graphs created contain a shortened version of the list of values to improve clarity.

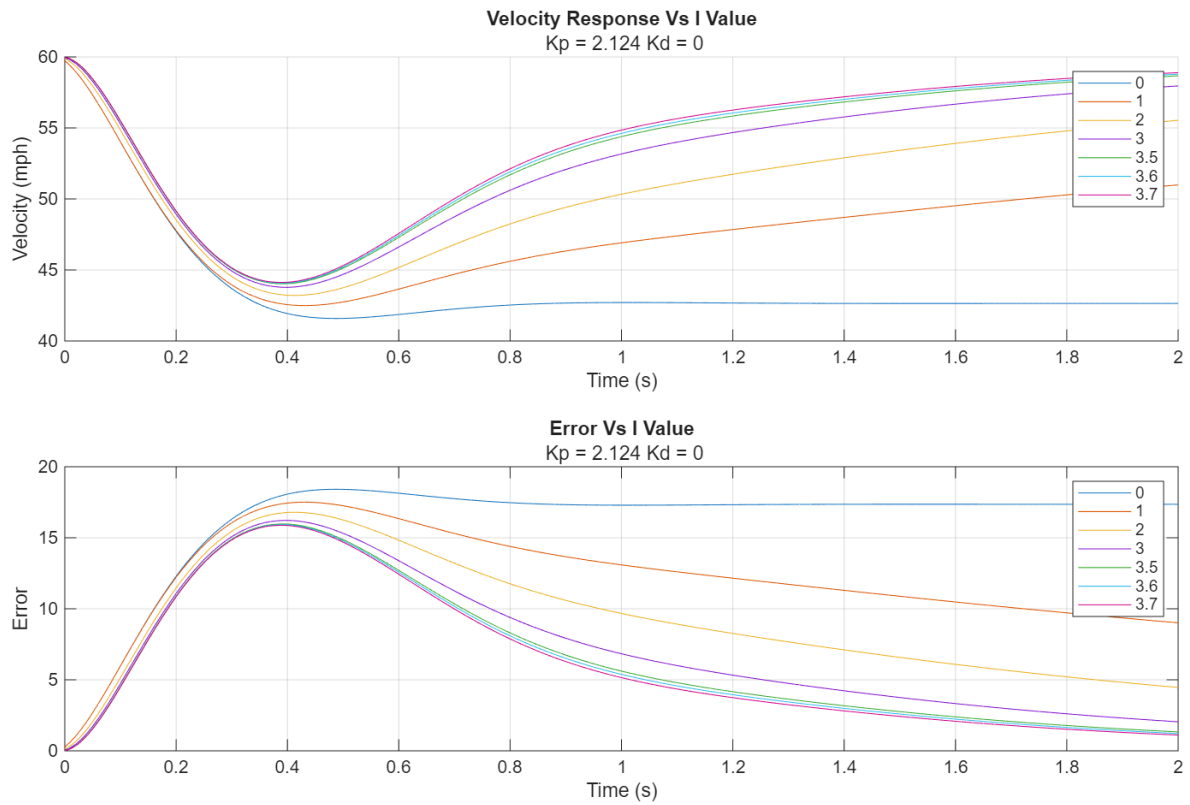


Figure 9 - Velocity & error response for Integral (I) tuning at $t = 2$ s

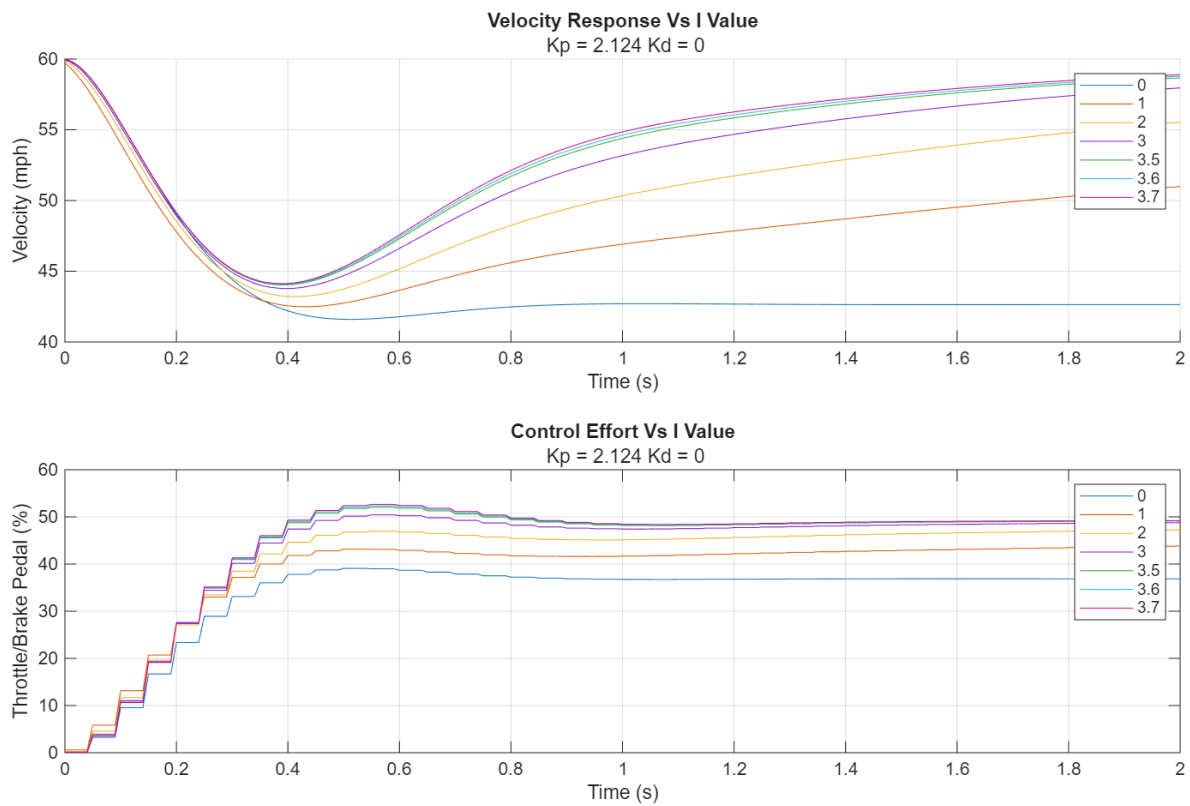


Figure 10 - Velocity & control effort response for Integral (I) tuning at $t = 2$ s

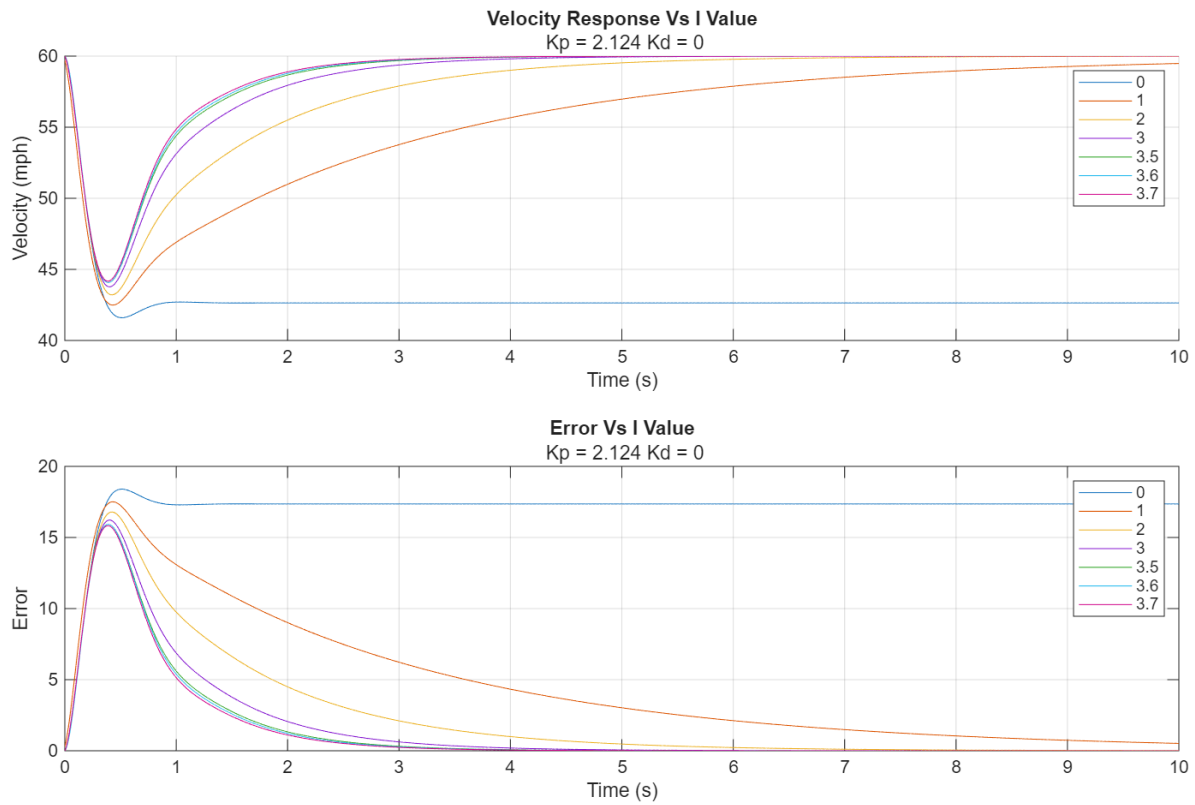


Figure 11 - Velocity & error response for Integral (I) tuning at $t = 10$ s

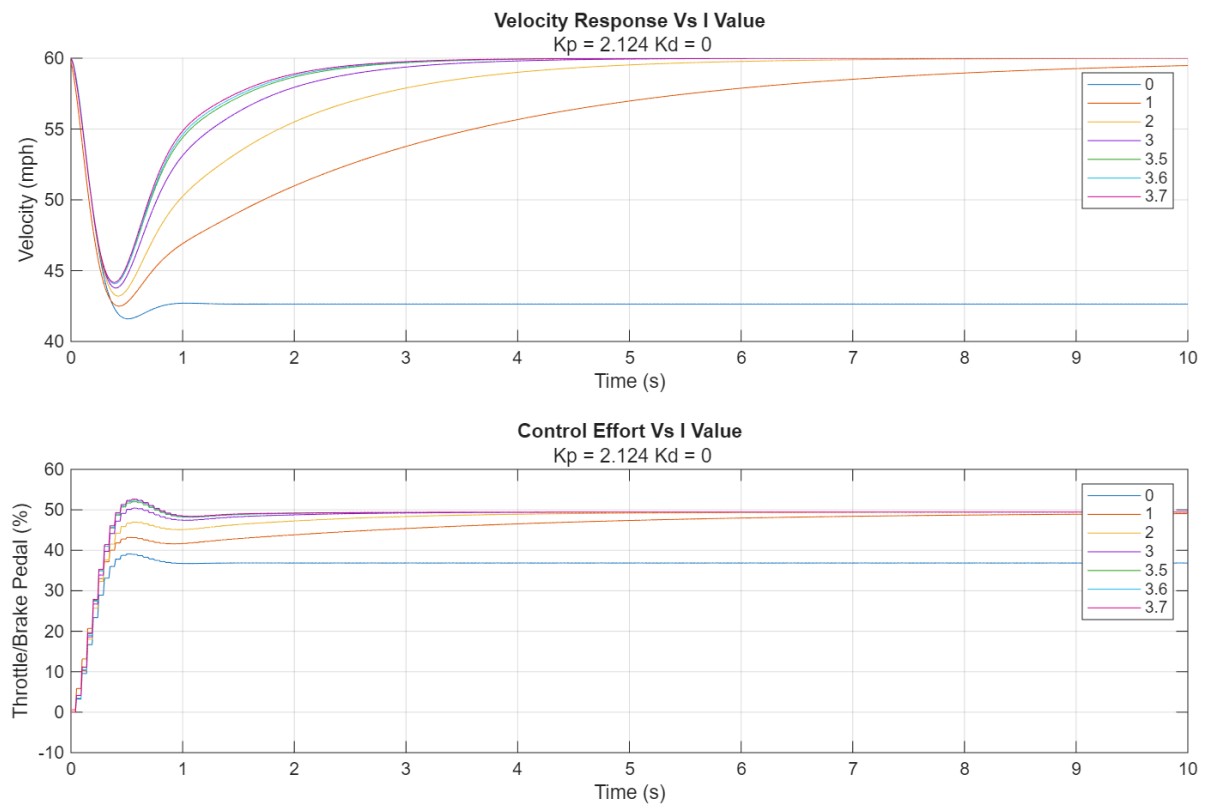


Figure 12 - Velocity & control effort response for Integral (I) tuning at $t = 10$ s

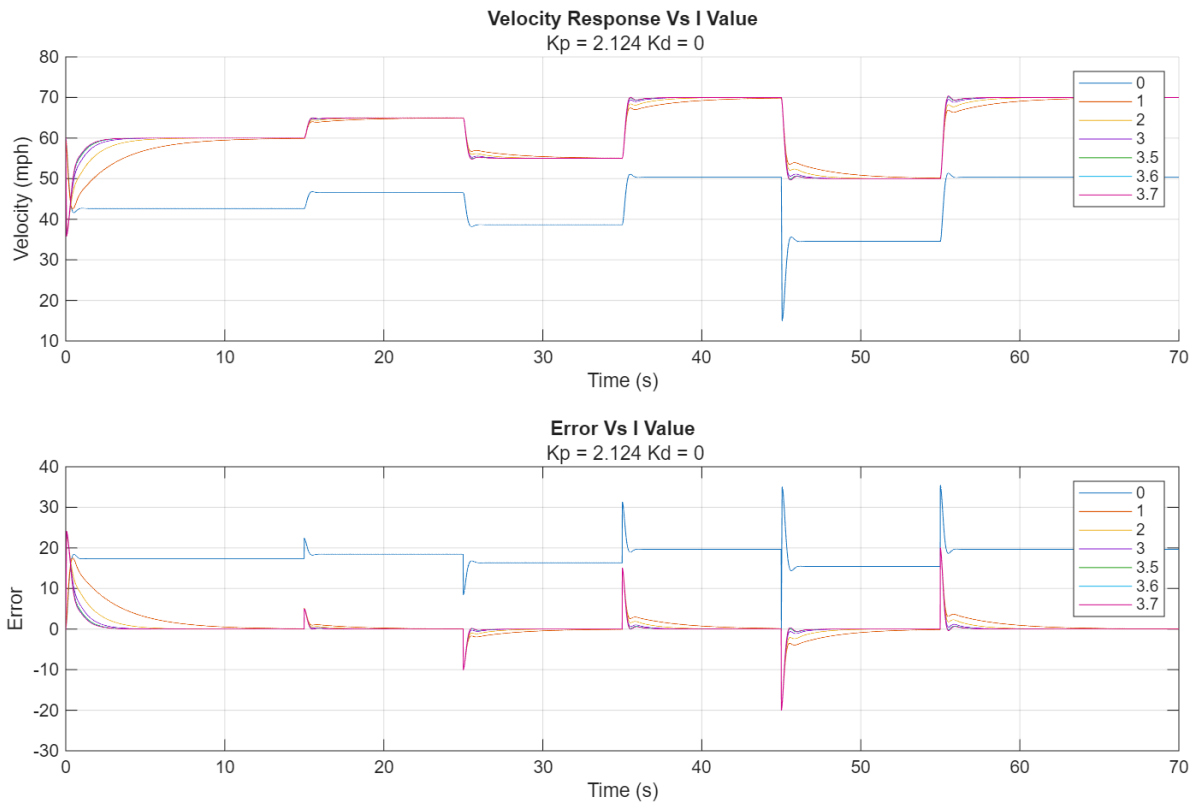


Figure 13 - Velocity & error response for Integral (I) tuning at $t = 70$ s

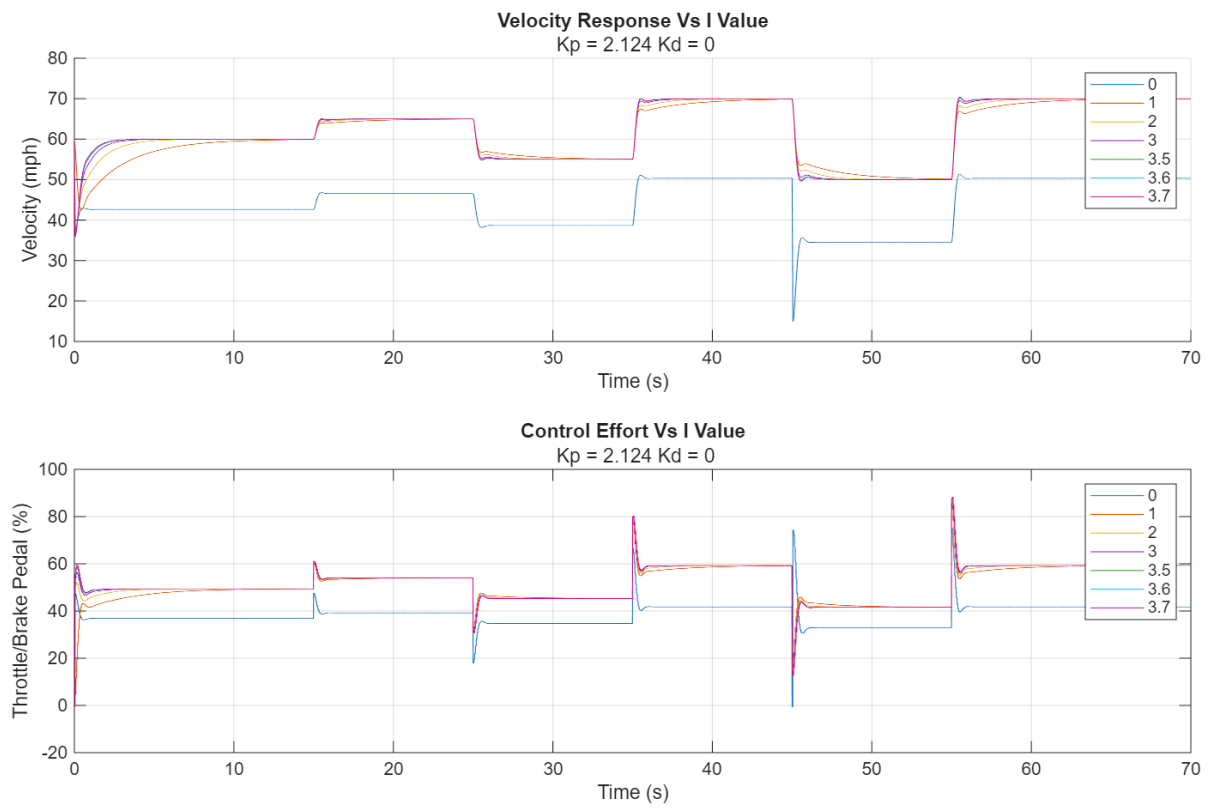


Figure 14 - Velocity & control effort response for Integral (I) tuning at $t = 70$ s

2.4. Differential Gain Tuning

Table 4 - Tested D Values

P=2.124, I=3.6		
D	Error Max	Error Mean
0	19.9985	0.0496
0.001	20.0488	0.0486
0.01	20.0488	0.0485
0.02	20.0488	0.0483
0.03	19.9023	0.0462
0.04	50	0.0458

Compared with P and I, changes in D made very minor changes, however, did decrease overshoot percentage. From D=0.04 onward, large spikes in the velocity response and error appeared before becoming extreme oscillations, for this reason the final D value was defined as 0.03

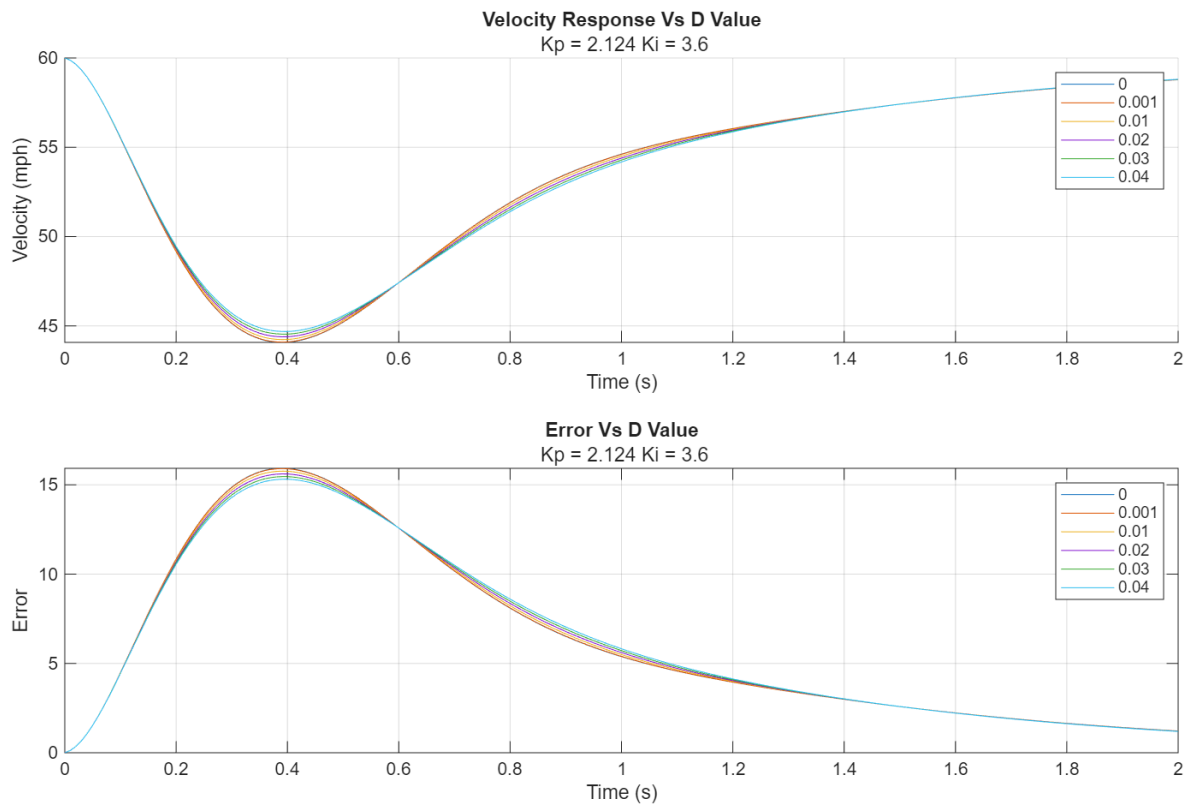


Figure 15 - Velocity & error response for Differential (D) tuning at $t = 2$ s

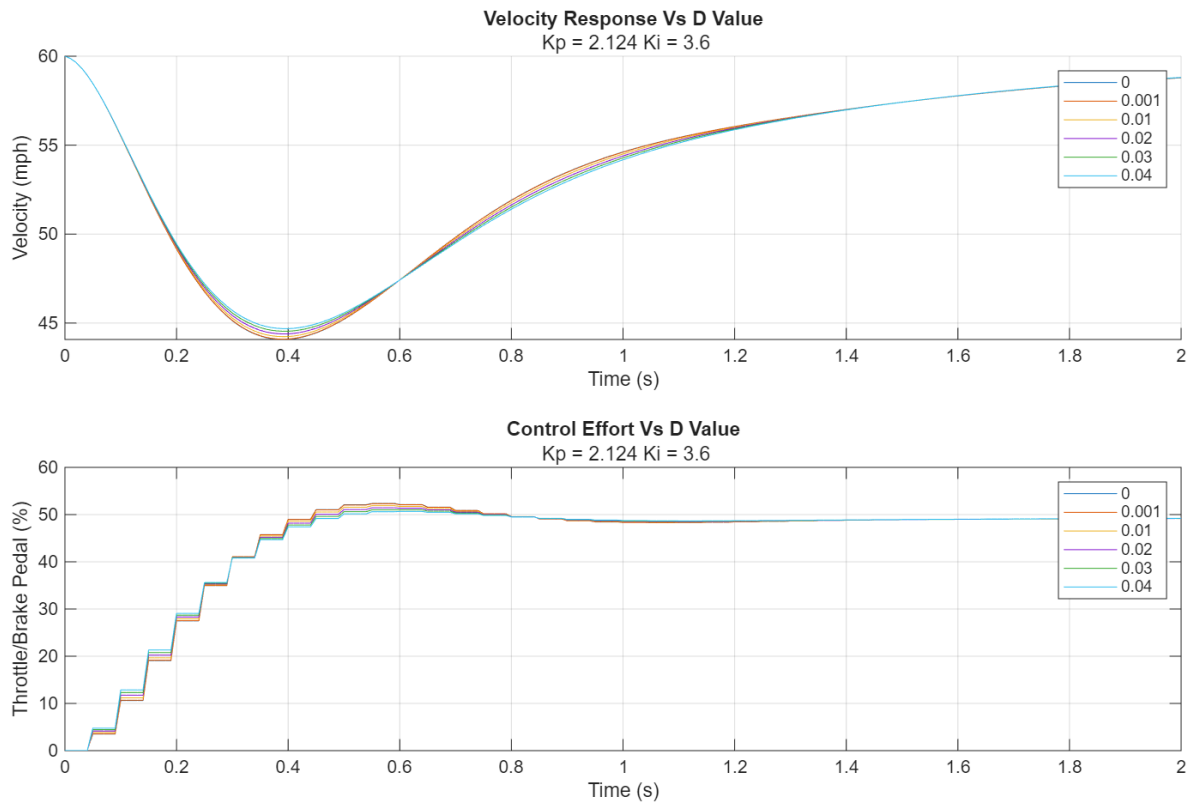


Figure 16 - Velocity & control effort response for Differential (D) tuning at $t = 2$ s

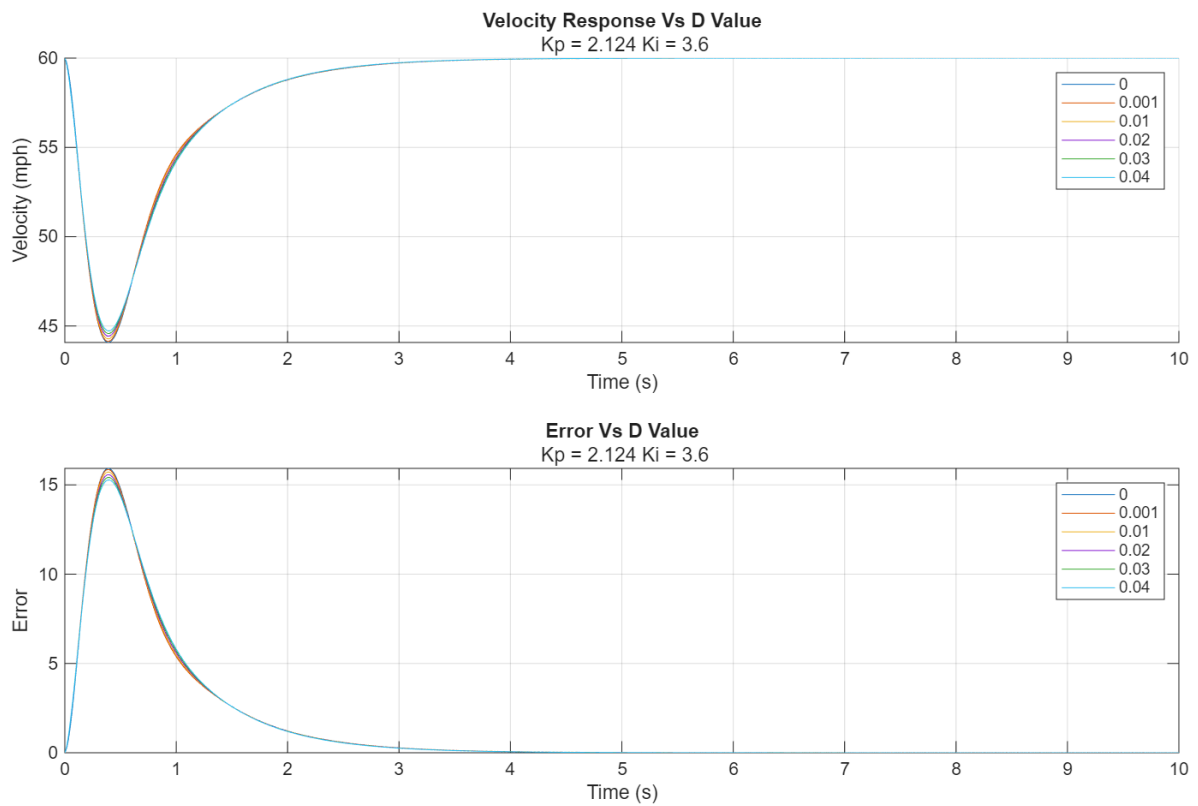


Figure 17 - Velocity & error response for Differential (D) tuning at $t = 10$ s

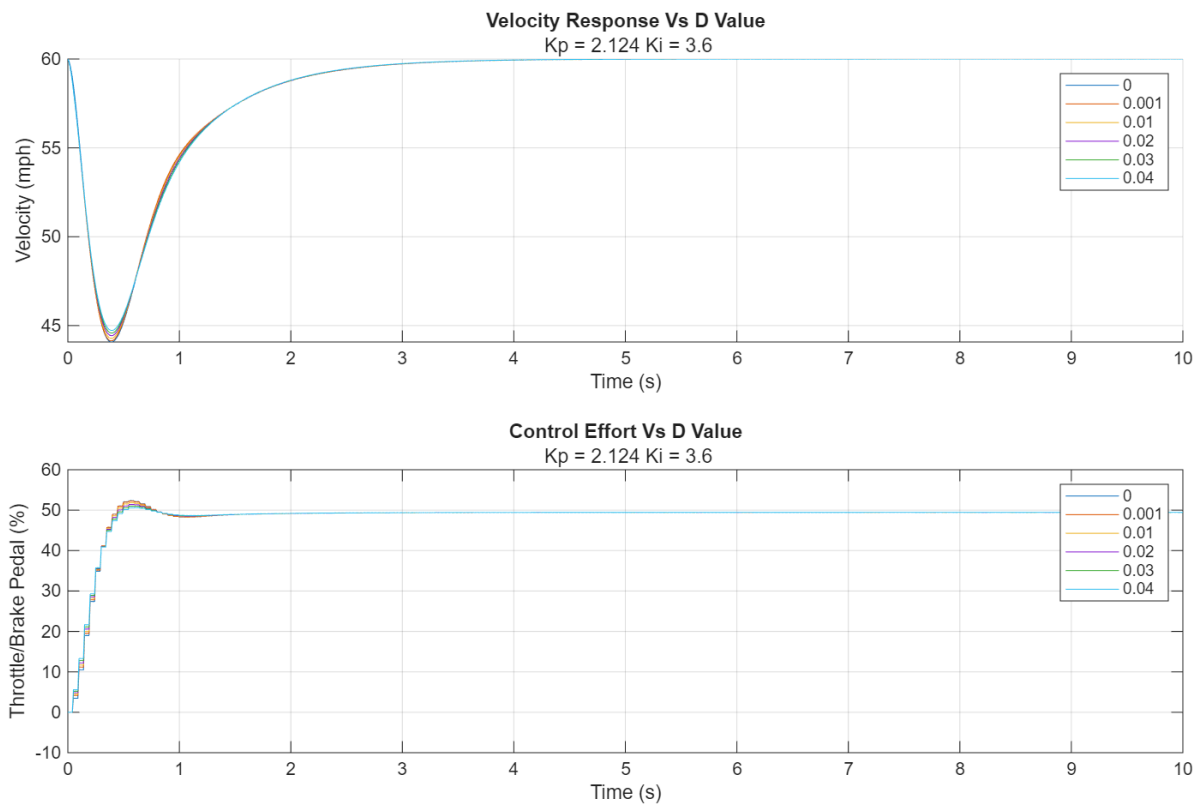


Figure 18 - Velocity & control effort response for Differential (D) tuning at $t = 10$ s

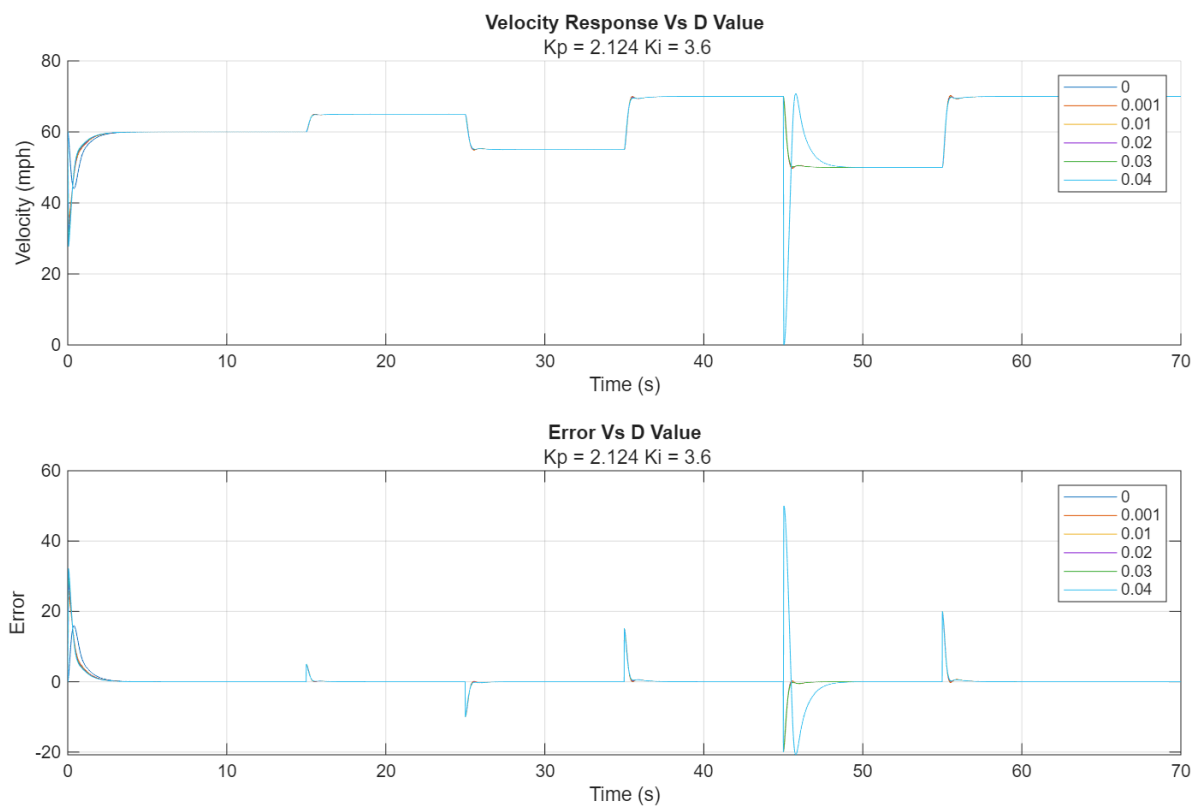


Figure 19 - Velocity & error response for Differential (D) tuning at $t = 70$ s

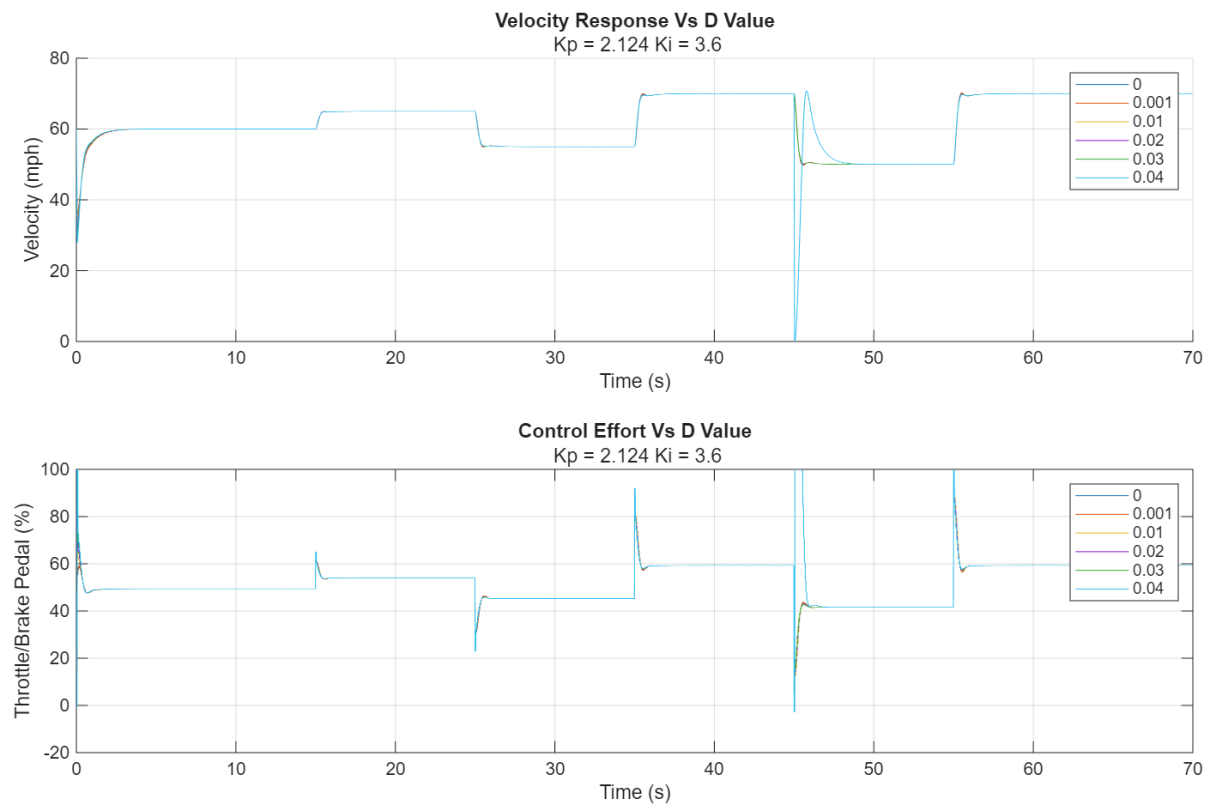


Figure 20 - Velocity & control effort response for Differential (D) tuning at $t = 70$ s

3. Bit Rate Study

For the bit rate sensitivity study, the values chosen were the powers of 2 from 1 to 64, this was chosen as many systems such as microcontrollers and CAN signals are based in binary and so naturally work in powers of 2. This also provided a uniform scaling of the resolution as each step doubles the last, which produces a clean and systematic study of the sensitivity. Using powers of 2 also avoids non-linear spacing of values, facilitating simpler analysis of trends.

The change of bit rate affects the resolution as:

$$Resolution = \frac{Max - Min}{2^n}$$

Equation 3 - Resolution

Where $n = \text{Bit rate}$ so,

An increase in bit rate produces smaller, finer step sizes, so with a sensor measuring

0-150mph, the step sizes are as follows:

2-bit → 4 levels → 37.5mph step size

8-bit → 256 levels → 0.6mph step size

16-bit → 65,536 levels → 0.002mph step size

Therefore, lower bit rates produce larger error, which affects the measured velocity and error signal of the PID directly.

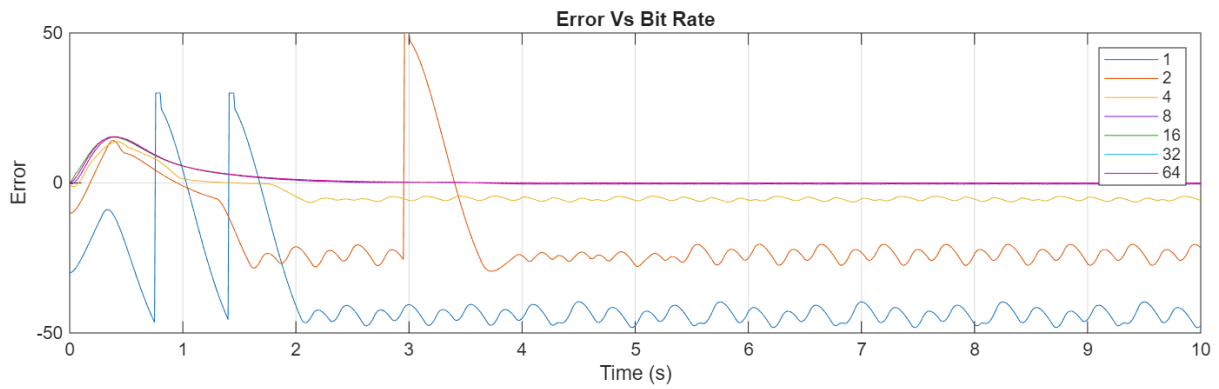
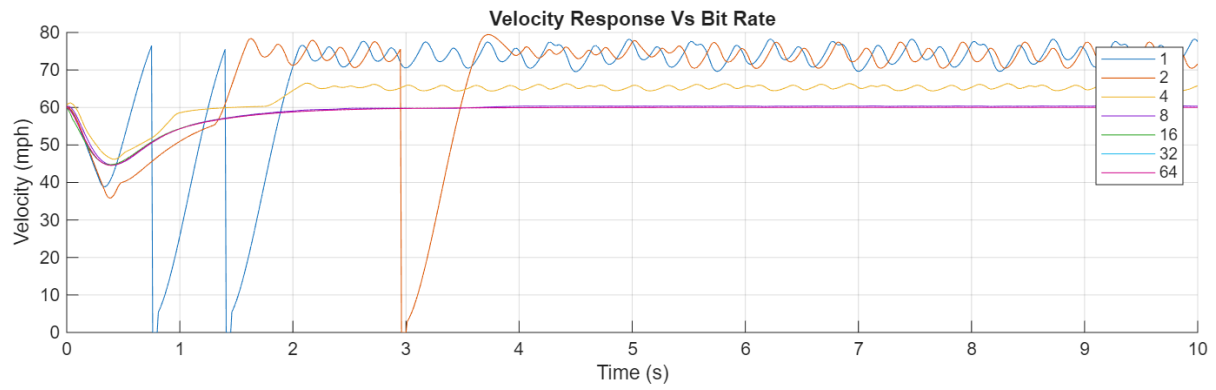


Figure 21 - Velocity & error response for Bit Rate at $t = 10$ s

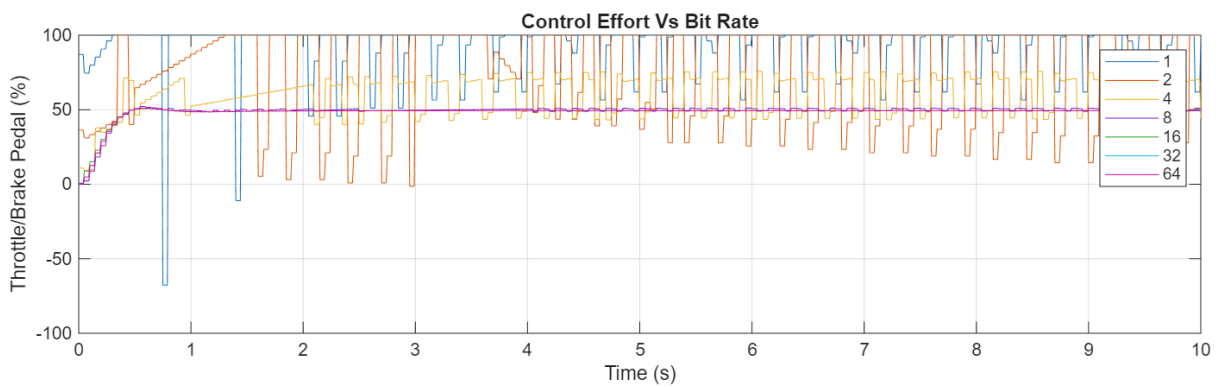
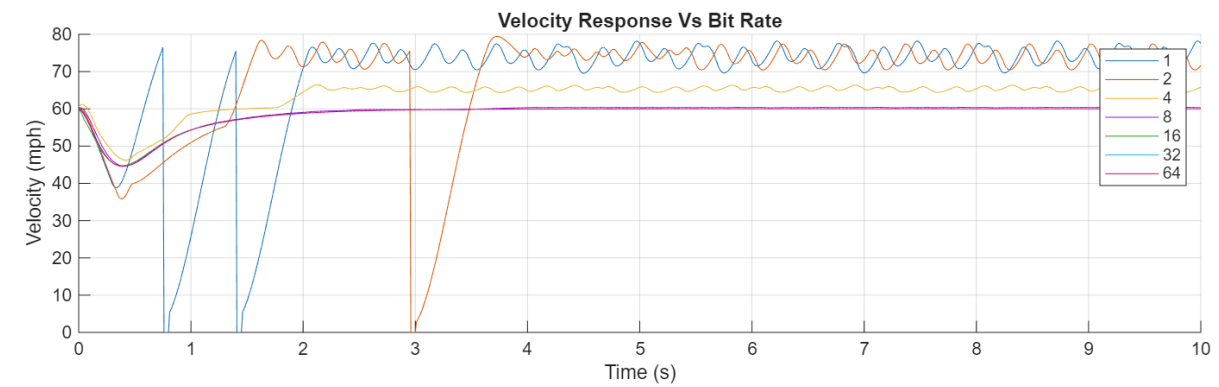


Figure 22 - Velocity & control effort response for Bit Rate at $t = 10$ s

After an initial run of the baseline values, steady-state values outside of the satisfactory 2% tolerance were observed clearly from 1 to 4 bits (figure 21 & 22), as well as the oscillations and sudden jumps expected from a low bit rate, and so these values were removed from the following simulation:

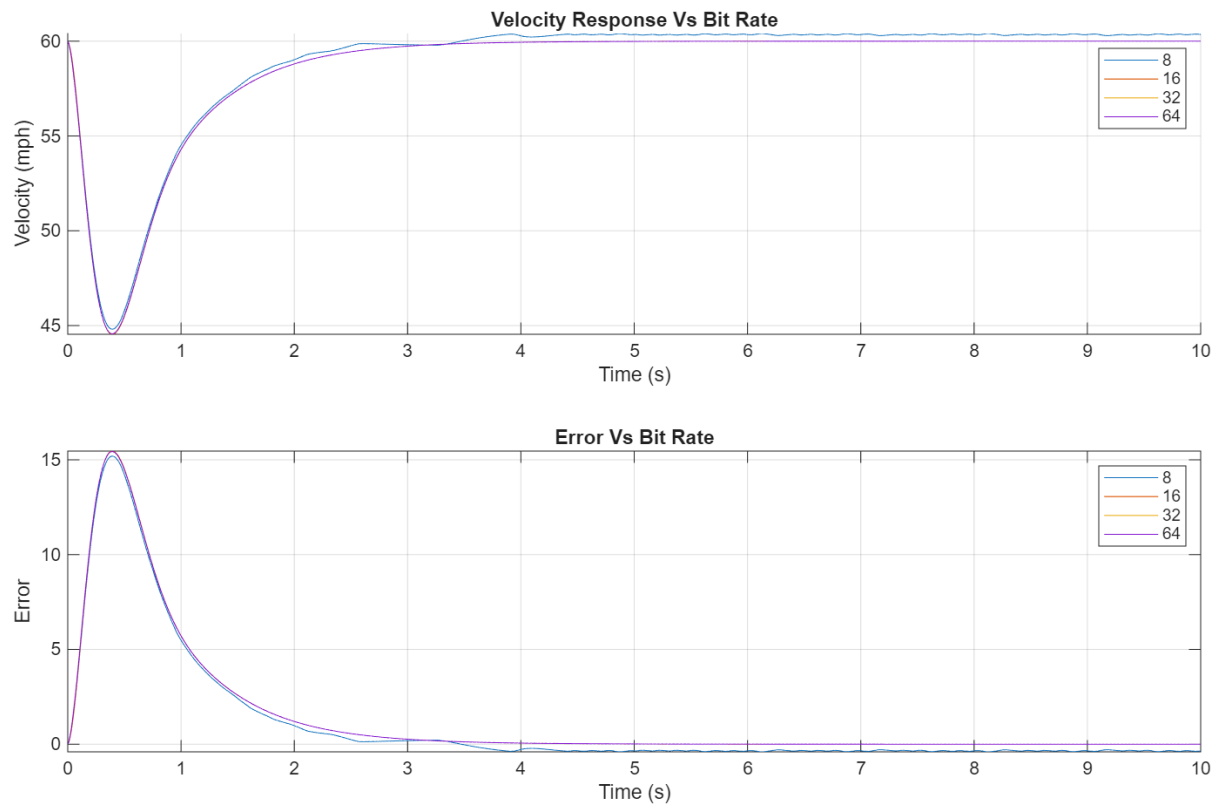


Figure 23 - Velocity & error response for Bit Rate (without 1, 2, & 4) at $t = 10$ s

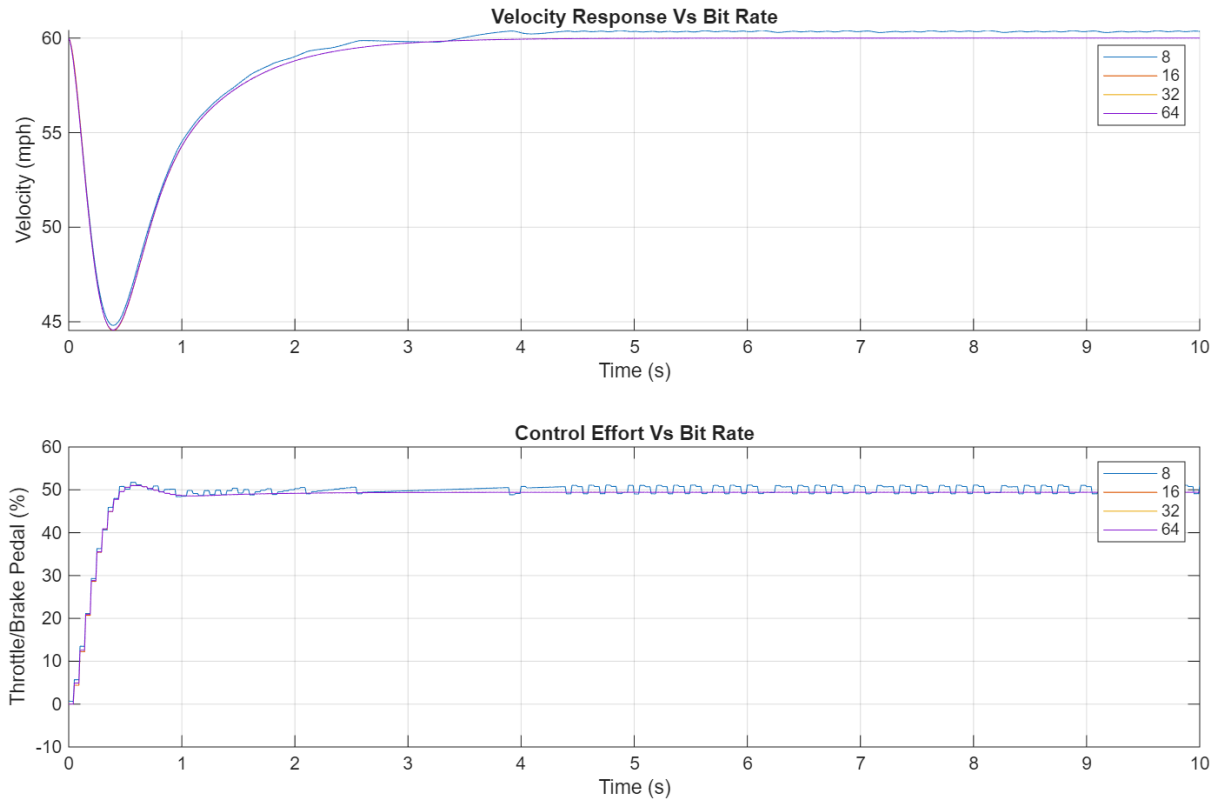


Figure 24 - Velocity & control effort response for Bit Rate (without 1, 2, & 4) at $t = 10$ s

In the simulations run with bit rates 8-64 (figure 23 & 24), steady-state error outside the 2% tolerance was found in the 8-bit response, this eliminated it, leaving 16 as the lowest bit rate that would still guarantee acceptable performance metrics. This however the lowest bit rate of the powers of 2, with sensors typically including 10-bit and 12-bit in between 8 and 16, for the purpose of this study it was decided to remain with 2^n bits meaning a rate of 16-bits was determined.

If a 10-bit or 12-bit rate could produce acceptable performance, it would be advantageous to run with that lower rate, this is due to the increased computational load, memory usage, and double load from 8-bit to 16-bit which can cause signal delays and saturation on systems like CAN bus.

4. Sampling Time Study

As the sampling time must be an integer multiple of the sim step value 0.01, the values chosen were 2x, 5x, 10x, 15x, and 20x that of the sim step. The velocity response, error, and control effort were plotted for analysis.

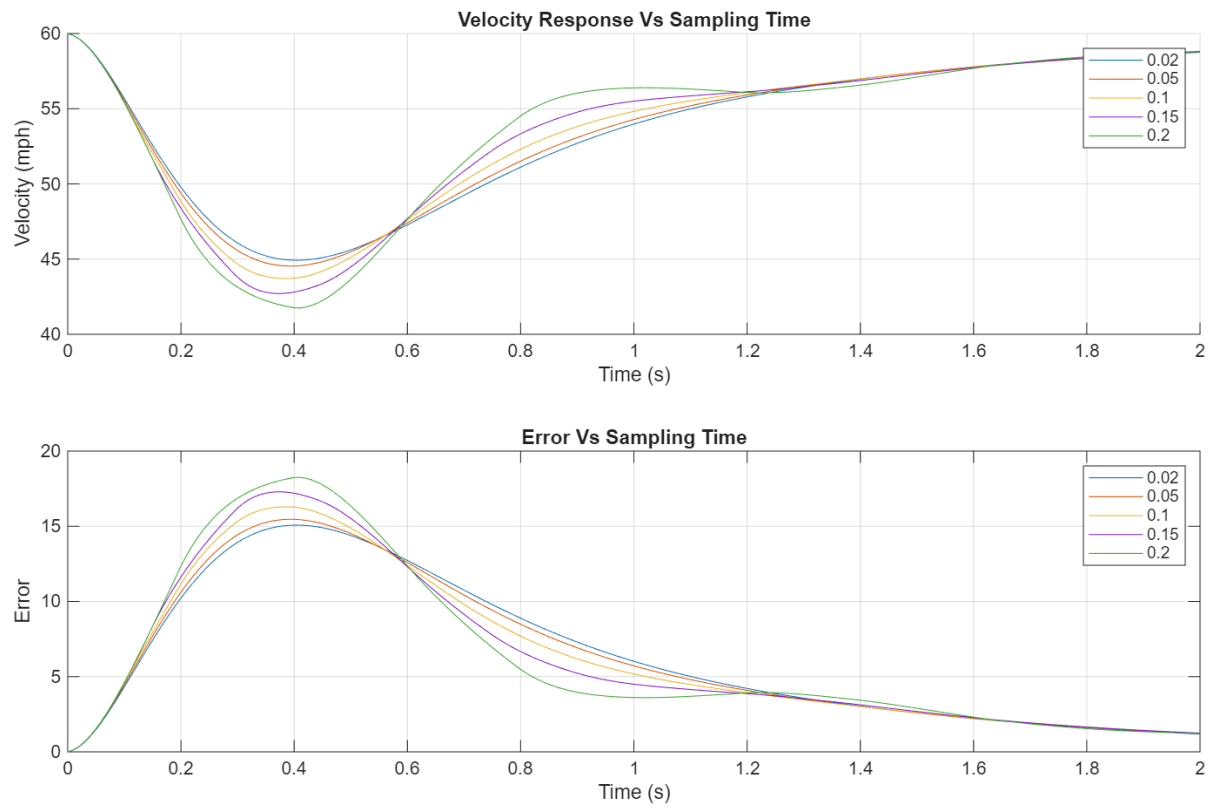


Figure 25 - Velocity & error response for Sampling Time at $t = 2$ s

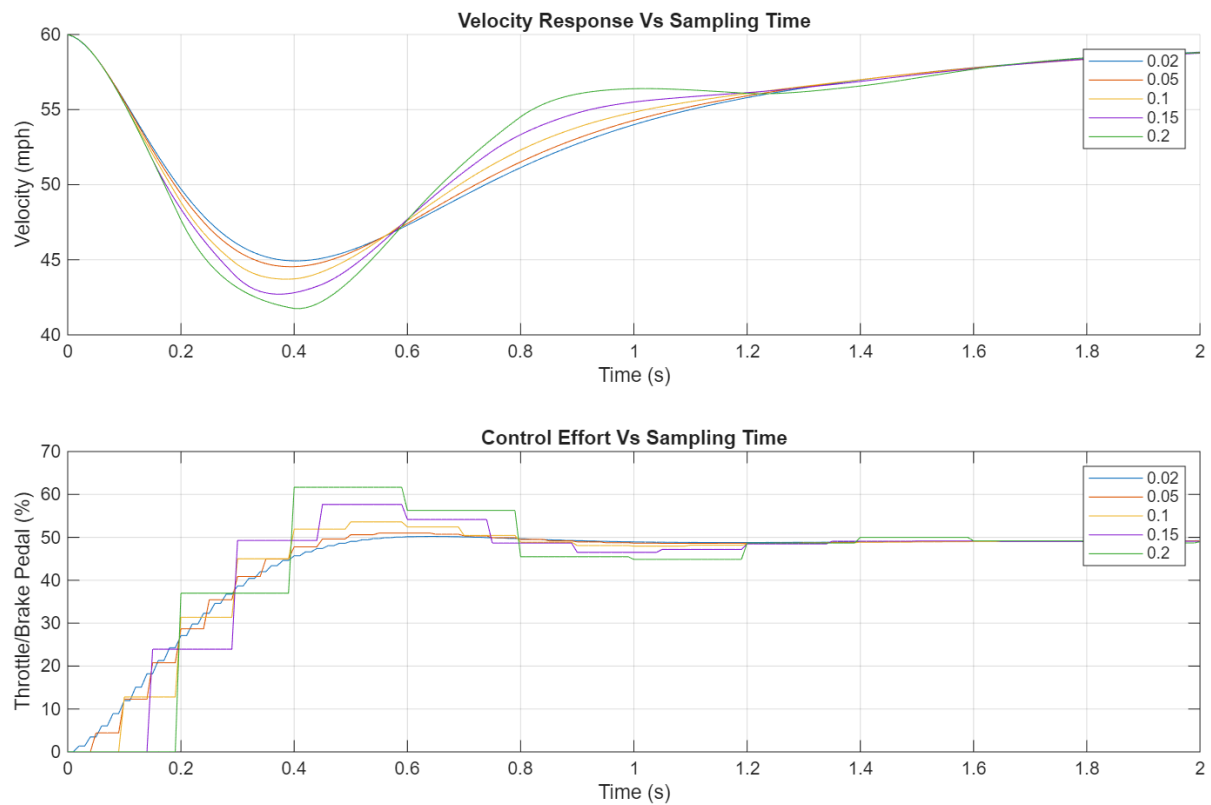


Figure 26 - Velocity & control effort response for Sampling Time at $t = 2$ s

At a sim length of 2s (figure 25 & 26), it is difficult to determine any definite outliers, however, like expected, the lower sampling time values do produce a shorter settling time as the controller can respond faster with reduced delay in error correction.

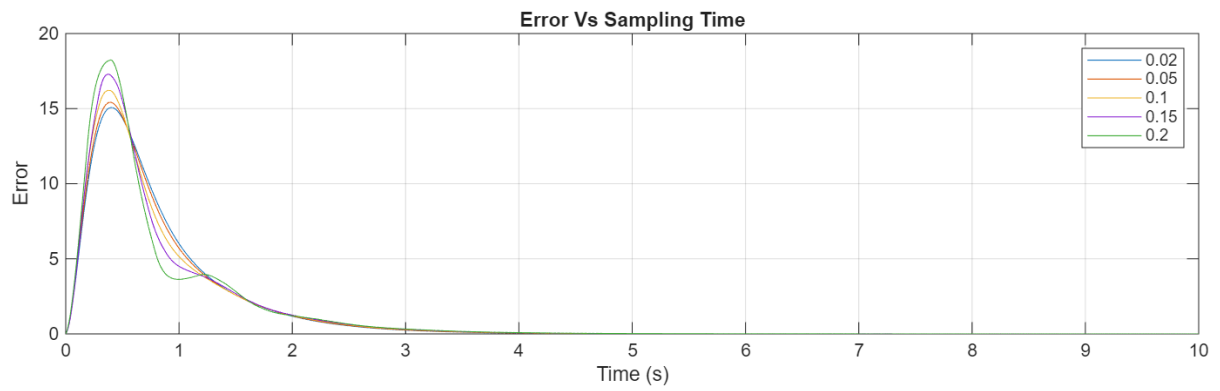
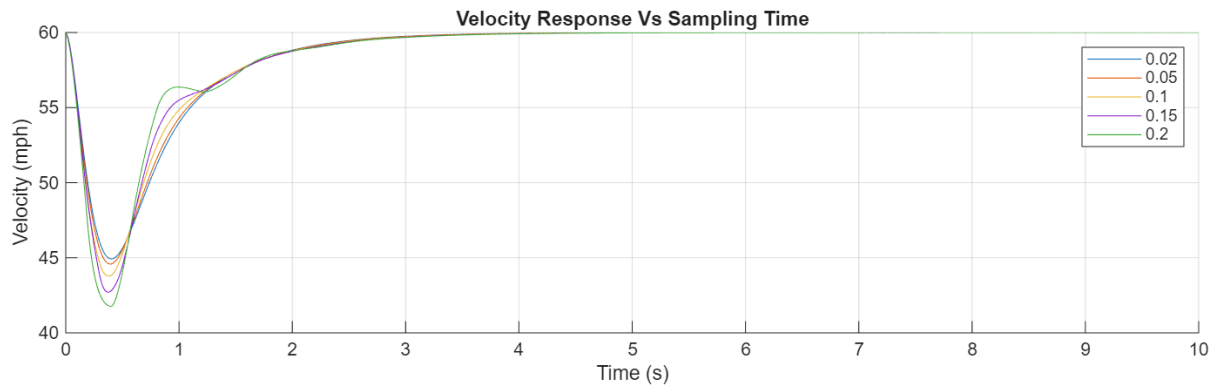


Figure 27 - Velocity & error response for Sampling Time at $t = 10$ s

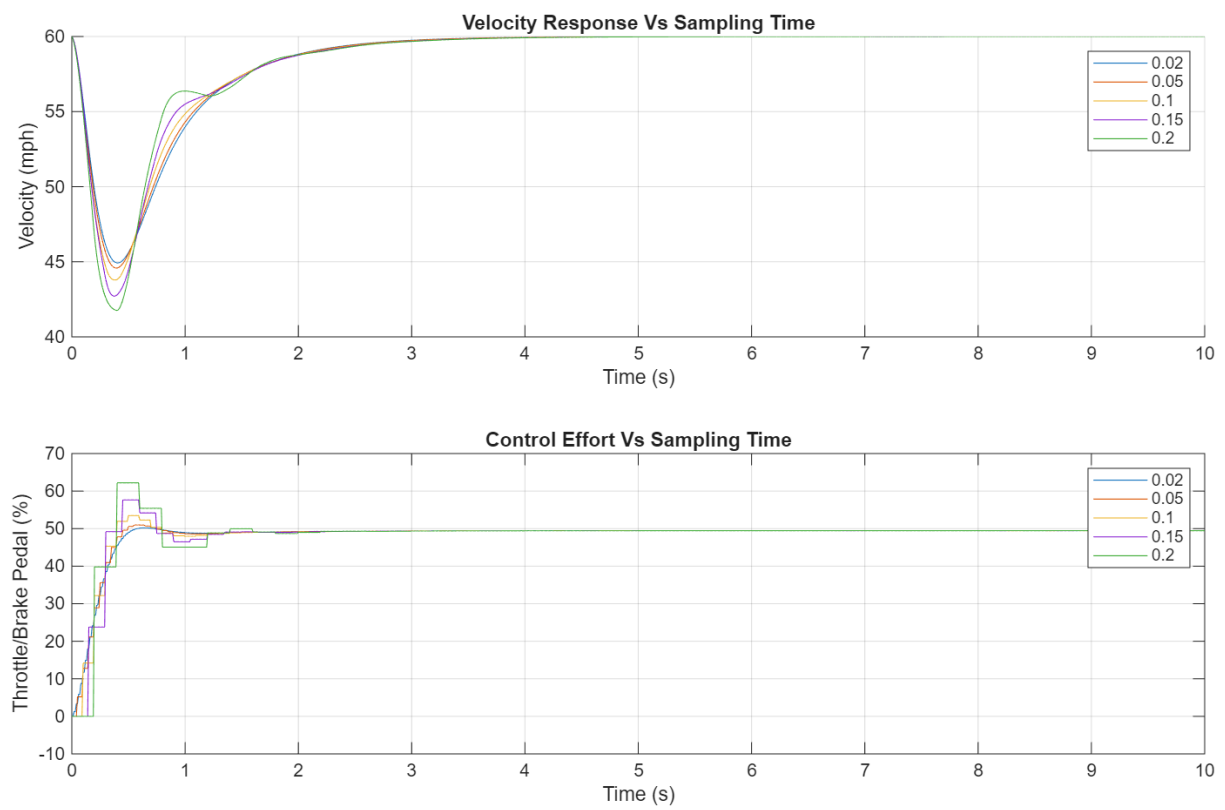


Figure 28 - Velocity & control effort response for Sampling Time at $t = 10$ s

At a sim length of 10s (figure 27 & 28), slightly clearer observations can be made, namely the noticeably larger oscillations produced by 0.2s in the first 1.5 seconds of the simulation, however, it still appears to settle quickly after 1.5 seconds. While 0.2 cannot be ruled out yet, these observations suggest it may be unsuitable over the full length of 70 seconds.

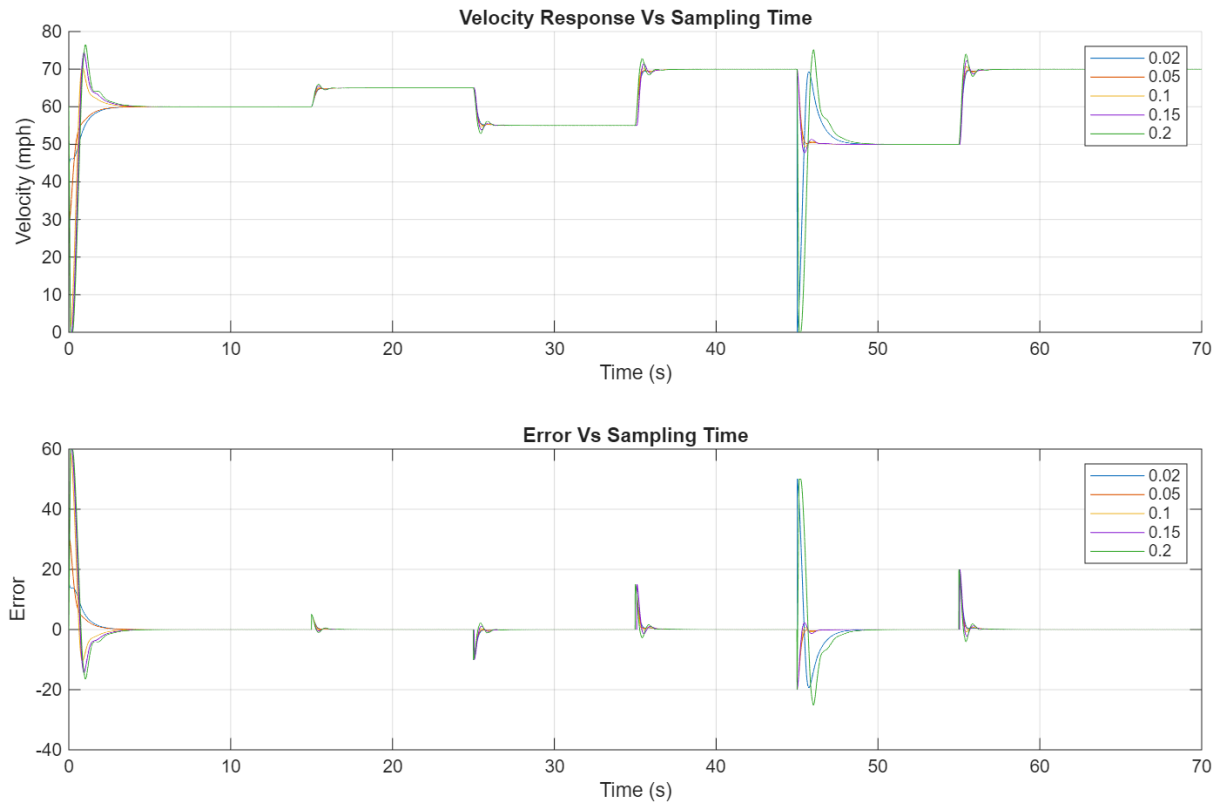


Figure 29 - Velocity & error response for Sampling Time at $t = 70$ s

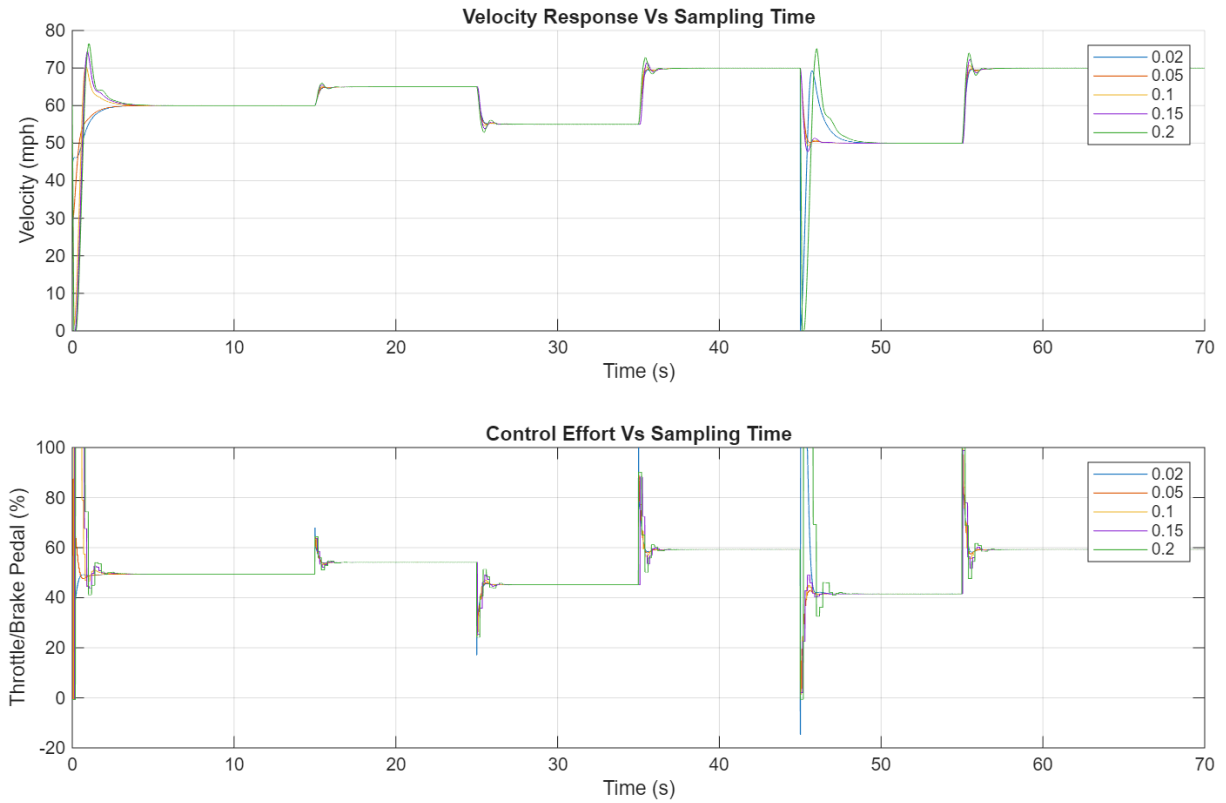


Figure 30 - Velocity & control effort response for Sampling Time at $t = 70$ s

At full sim length of 70s (figure 29 & 30), extremely large spikes are observed around 45s for both 0.2s and 0.02, the highest and lowest values. Over the rest of the simulation 0.02 displays the best performance as expected, the sudden spike is likely the cause of the sudden 20mph drop in demand, and, while the 0.2 sample time is experiencing the drop due its delayed reaction to the change, the 0.02 sample time is also experiencing the same drop but in contrast, because the reaction time of the controller is almost immediate. This would appear to be desirable, however, it can make the controller overly aggressive, which can be compounded when the derivative action is considered, as it can be approximated as:

$$\frac{e(k) - e(k - 1)}{T_s}$$

Equation 4 - Derivative

So, a low T_s value can greatly increase the rate of change term in comparison to the same error jump, possibly causing derivative kick, aggressive control spike, and rebound overshoot before settling.

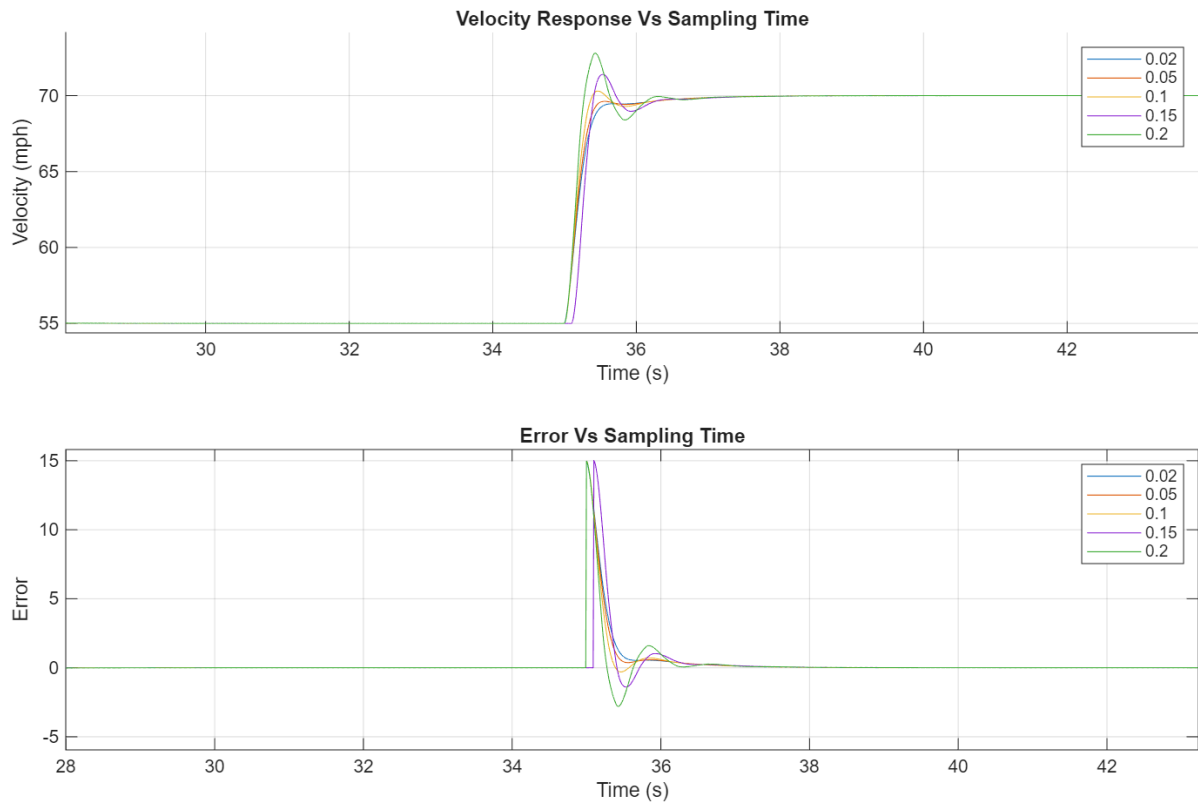


Figure 31 - Velocity & error response for Sampling Time at $t = 28 - 44$ s

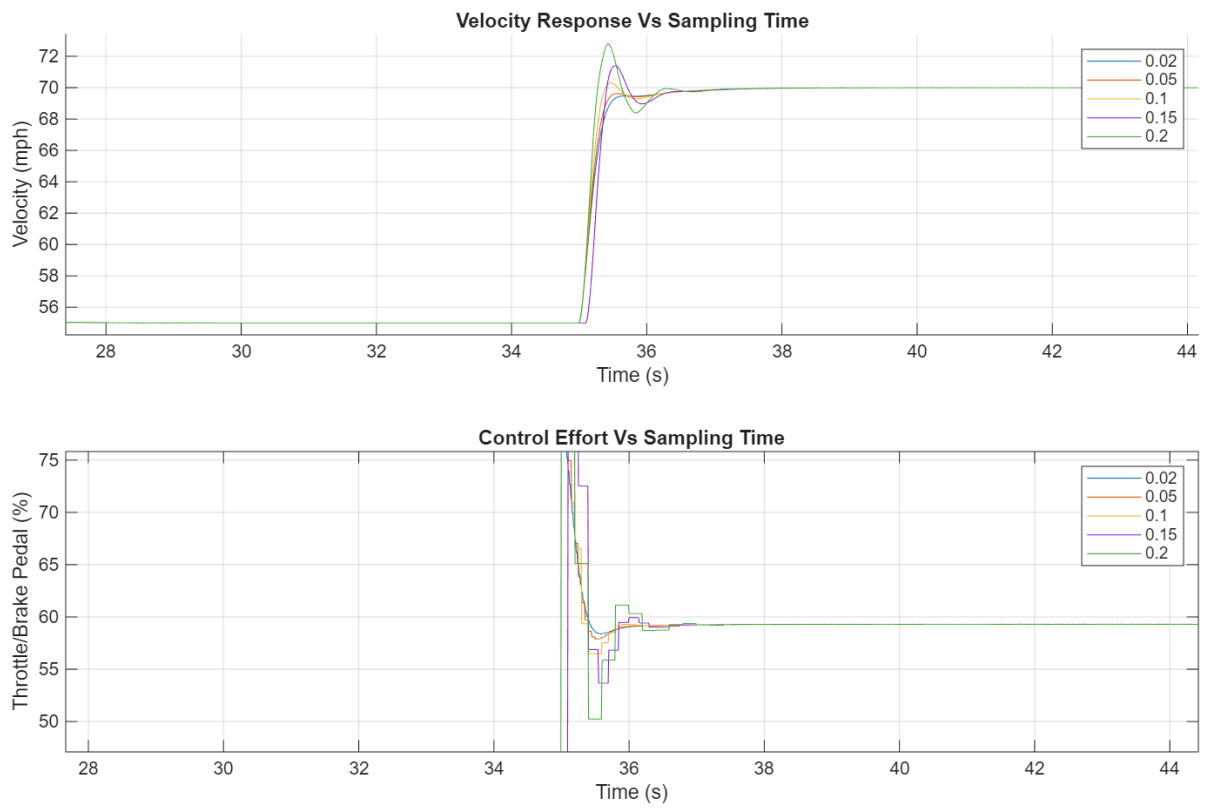


Figure 32 - Velocity & control effort response for Sampling Time at $t = 28 - 44$ s

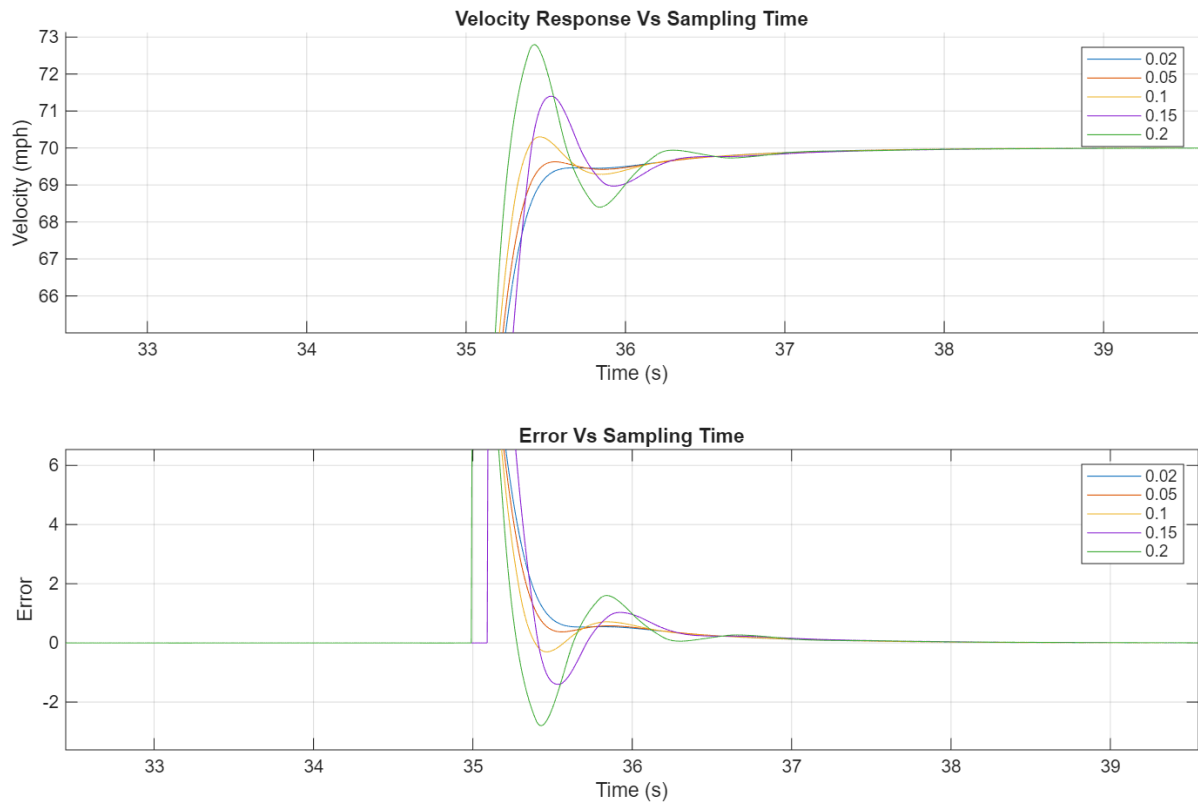


Figure 33 - Velocity & error response for Sampling Time at $t = 33 - 39$ s

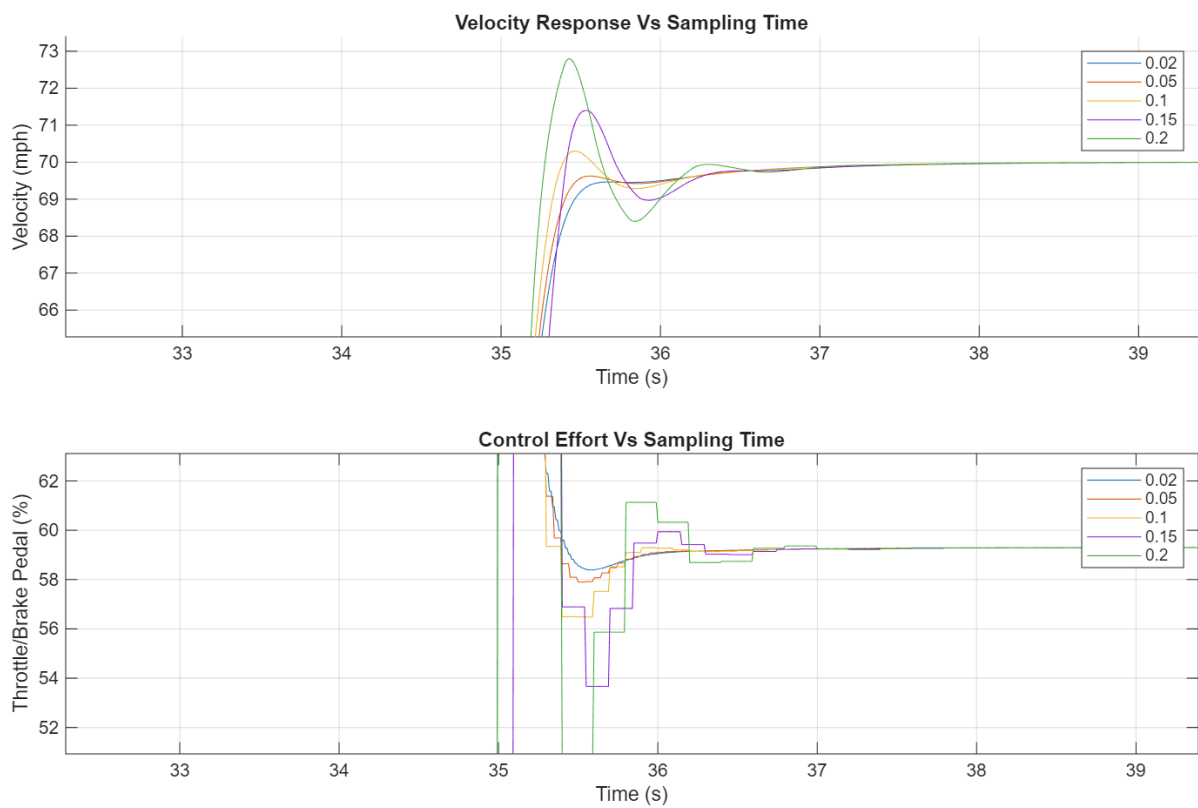


Figure 34 - Velocity & control effort response for Sampling Time at $t = 33 - 39$ s

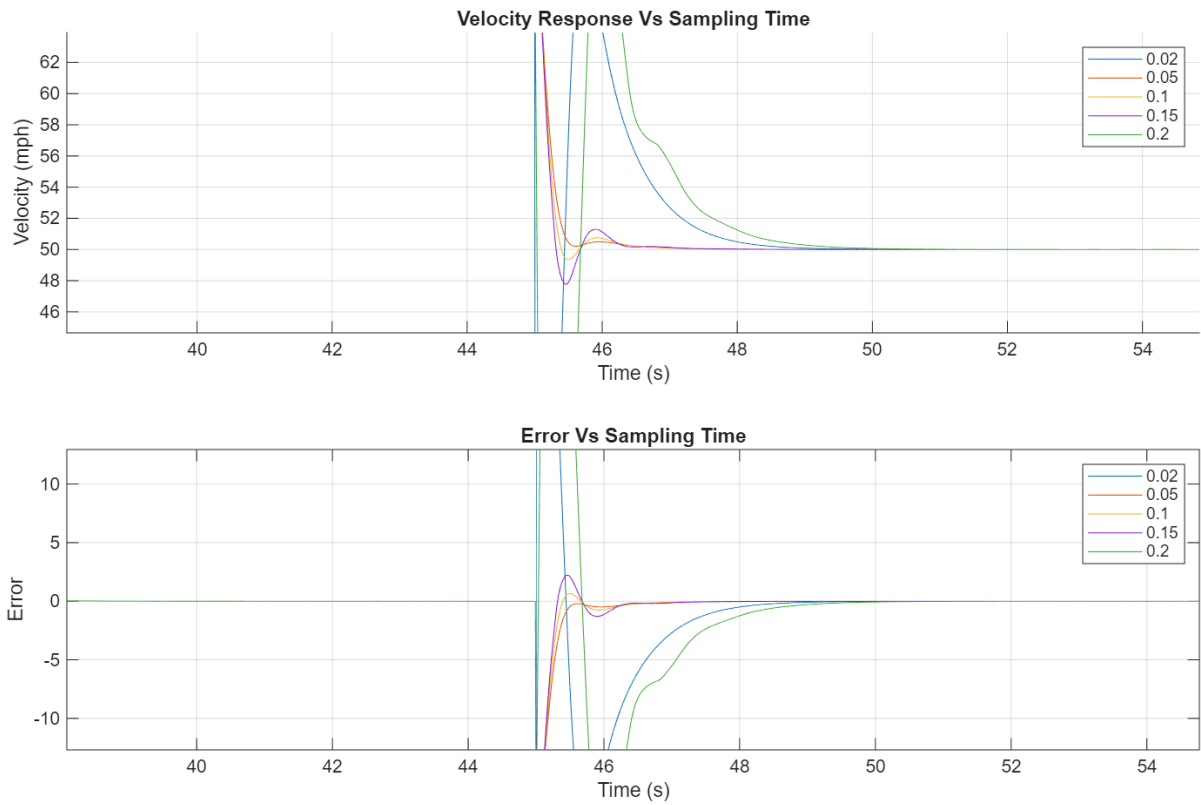


Figure 35 - Velocity & error response for Sampling Time at $t = 40 - 54$ s

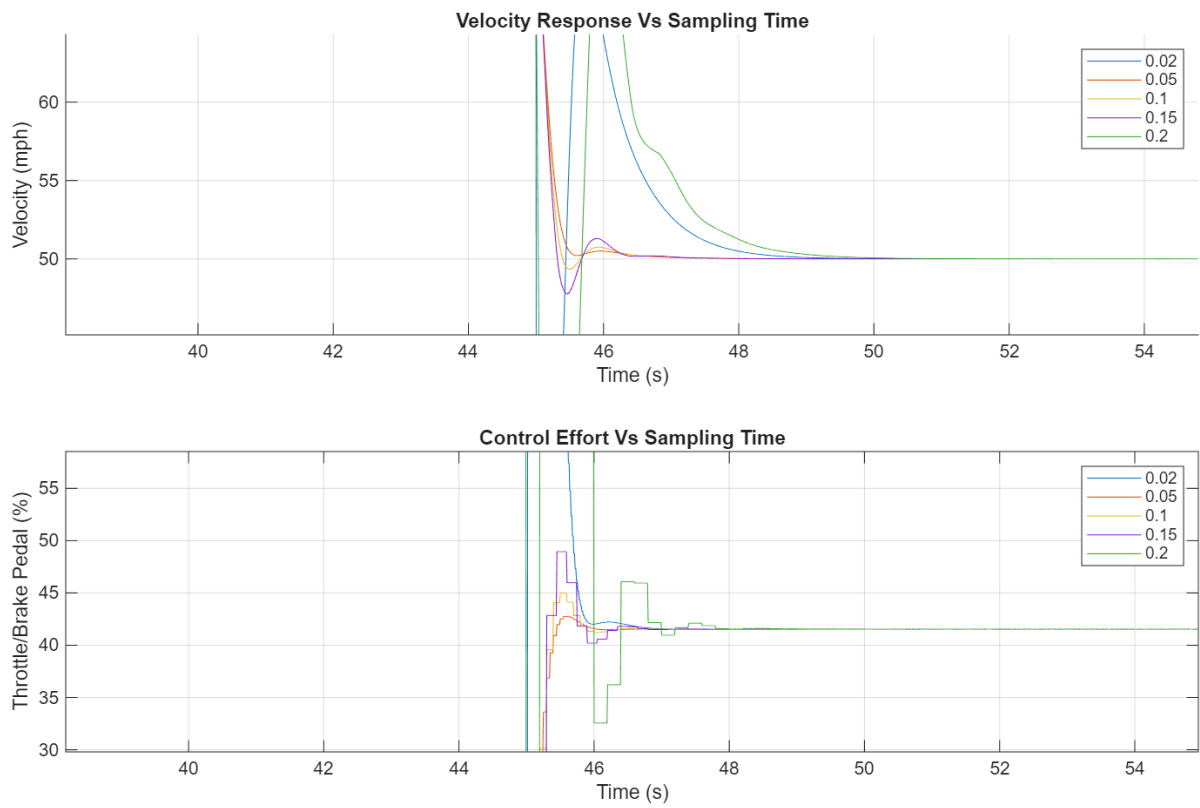


Figure 36 - Velocity & control effort response for Sampling Time at $t = 40 - 54$ s

As observed earlier, the spike experienced by $T_s = 0.2$ and $T_s = 0.02$ (figure 35 & 36), is such that the peak overshoot of both values exceeds the 5% deemed acceptable and such, 0.15s is the highest value of T_s that yields acceptable performance metrics. While using $T_s < 0.15$ would generate a faster response and improve tracking, much like using a higher bit rate, this comes at the cost of increased computational load, sensitivity to noise, and ongoing control activity even while the steady-state error is within acceptable bounds as well as experiencing derivative kick.

5. Delay Study

Standard delay (signal lag) of a cruise control system is ~20ms or 0.02s

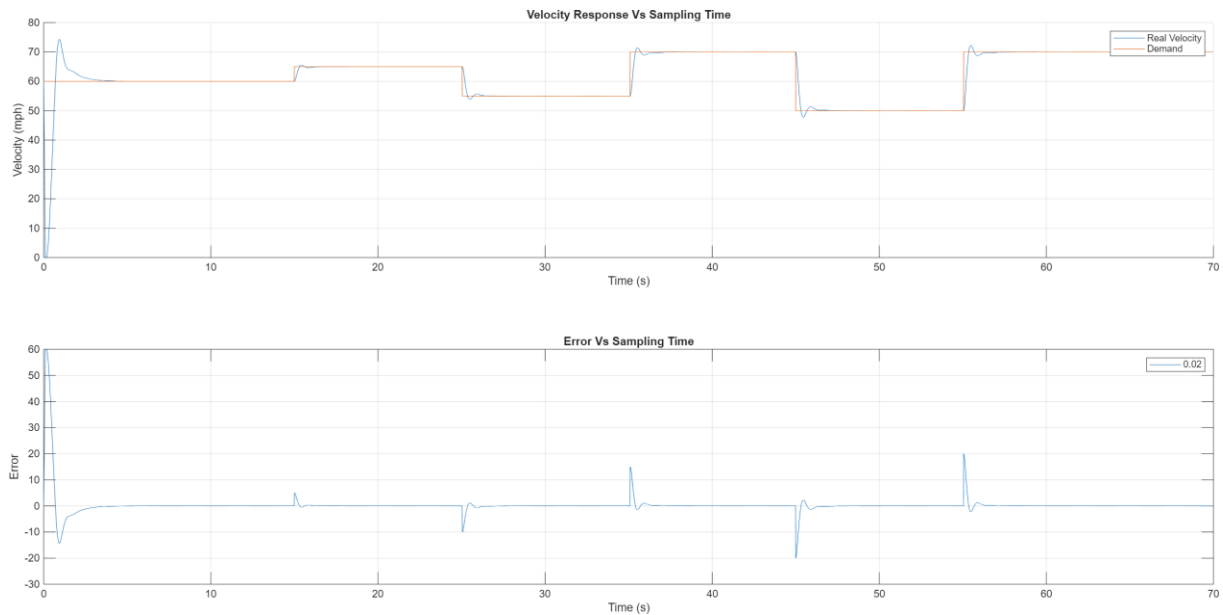


Figure 37 - Tuned System, Delay = 0s

Table 5 - 0s Delay Values

Tuned PID, Delay=0s						
Sim Time (s)	Desired Speed (mph)	Settling Time (s)	Overshoot (mph)	Overshoot (%)	Max Overshoot (%)	Mean Overshoot (%)
15	65	0.24	0.5	0.769230769	4.5	2.555144855
25	55	0.5	1.3	2.363636364		
35	70	0.28	1.4	2		
45	50	0.6	2.25	4.5		
55	70	0.54	2.2	3.142857143		

The tuned system with 0 delay has a mean 2% settling time of 0.43s & max of 0.6s, standard acceptable is between 2-5s (Åström and Murray, 2008). The system also has a mean overshoot of 2.555% & max of 4.5%, standard acceptable is <5% (Oguntosin and Olasina, 2021).

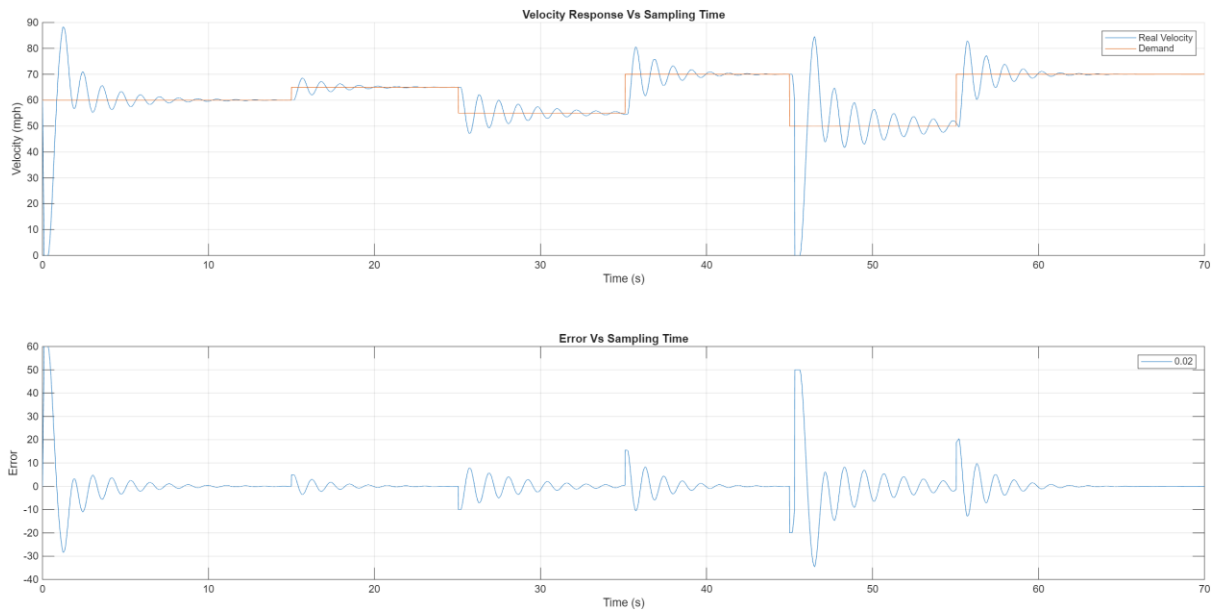


Table 6 - Tuned System, Delay = 0.02s

Table 7 - 0.02s Delay Values

Tuned PID, Delay=0.02s						
Sim Time (s)	Desired Speed (mph)	Settling Time (s)	Overshoot (mph)	Overshoot (%)	Max Overshoot (%)	Mean Overshoot (%)
15	65	3	3.52	5.415384615	100	30.36333666
25	55	7.21	2.9	5.272727273		
35	70	4.13	10.5	15		
45	50	12	50	100		
55	70	4.66	18.29	26.12857143		

With massive overshoot issues at the 45 second mark, for a re-tune to be viable, the solves identified for the control function issue should be implemented before updating to ensure accuracy and reliability of any results.

6. Discussion & Conclusions

6.1. Lack of Anti-Windup

The control function created, “calcController”, contains saturation for the actuator, however, does not include an anti-windup term, this means that it is possible for the integral to continue accumulating, even after full saturation of the actuator, thus likely causing increased overshoot and slowing recovery.

6.2. Control Function Problems

Large spikes in the velocity response, error, and control effort often show up at ~45s simulation time, this is seen across the P, I, and D tuning, the bit rate and sample time studies and can also be observed in the 0.02s delay graph. There is a possibility this is due to the control function seen in figure 2, with the derivative term:

$$e_derivative = (error - e_Previous) / Ts$$

being based upon the error opposed to the measured velocity value. With the derivative being based on error, the sudden drop in demand of 20mph is creating a derivative kick. This “kick” is discussed in the sample time study as it effects both the highest and lowest values but not the middle.

A possible fix could be changing the derivative term to:

$$v_derivative = (velocity_measured - v_Previous) / Ts;$$

This would require 3 other small changes to work with the new term, notably, updating the control equation to:

$$control = P * error + I * e_integral - D * v_derivative;$$

The derivative component is subtracted from the others due to the increasing velocity reducing error. The PID would then ideally be re-tuned to increase the accuracy of results.

7. Future Work

7.1. Tuning Automation

With the updated derivative term requiring a re-tune of the system, it would be advisable to implement an automated tuning process for speed and accuracy. Perhaps using the Cohen-Coon method would suit as it has increased accuracy for systems that include a delay

8. References

- ❖ International Organization for Standardization (ISO) (2009) *ISO 22179:2009 Intelligent transport systems — Full speed range adaptive cruise control (FSRA) systems — Performance requirements and test procedures*. Geneva: ISO. Available at: [<ISO 22179 PDF>](#) (Accessed: 10 April 2026).
- ❖ Åström, K.J. and Murray, R.M. (2008) *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton: Princeton University Press. Available at: [<Feedback Systems PDF>](#) (Accessed: 10 April 2026).
- ❖ International Organization for Standardization (ISO) (2018) *ISO 15622:2018 Intelligent transport systems — Adaptive cruise control systems — Performance requirements and test procedures*. Geneva: ISO. Available at: [<ISO 15622 page>](#) (Accessed: 10 April 2026).
- ❖ de Winkel, K.N., Irmak, T., Happee, R. and Shyrokau, B. (2023) *Standards for passenger comfort in automated vehicles: Acceleration and jerk*. **Applied Ergonomics**, 106, 103881. Available at: <https://doi.org/10.1016/j.apergo.2022.103881> (Accessed: 10 April 2026).
- ❖ Oguntosin, V. and Olasina, J. (2021) *Control design and analysis of cruise control system*. Journal of Research in Engineering and Applied Sciences. Available at: <https://www.researchgate.net/publication/353925596> (Accessed: 10 April 2026).
- ❖ Nise, N.S. (2020) *Control Systems Engineering*. 8th edn. Hoboken: Wiley.

9. Appendices