EEE 332 / 413 Project Report

Term	☐ Spring ☐ Summer ☑ Autumn	**Year**	2020

Students' Name	**Students' ID**	**E-mail Address**
Faiyaz Ahmed	1820545	1820545@iub.edu.bd
Jarin Tasnim Supti	1820759	1820759@iub.edu.bd

Instructor's Name	Dr. Abdur Razzak	**Designation**	Professor

1. Project Title

Designing a Simple Cruise Control System Using Unity Closed-Loop Feedback & PI controller

2. Introduction

The role of the cruise control mechanism is to precisely maintain the desired speed of a vehicle, without the intervention of a driver. A state-of-the-art auto cruise control is a closed control loop that takes over control of the engine throttle using a PI controller. Conventional cruise control takes over the accelerator operation at speeds over 48 km/h (30 mph) when it is engaged [1].

Activation requires that the ON button be pressed and the desired speed set. The driver must press the ON button to activate the system each time the engine is started. Once the cruise control is ON, the driver can set a speed by driving at the desired speed and then pressing the SET button.

Conventional Cruise Control (CCC) systems adjust vehicle velocity by automatically controlling the throttle and/or the brake. Adaptive Cruise Control (ACC) technologies are now commercially available in a wide variety of passenger cars and are the subject of testing, automakers, policymakers, and customers around the world. The first generation of ACC systems was included in some luxury vehicles by automakers and their suppliers primarily from the viewpoint of enhancing driving comfort and convenience with some additional potential increase in safety.

3. Literature Review, Background and Motivation

For the following design, a variety of tutorials and relevant publications have been studied to get an understanding of the existing methodology and are thus referenced in the reference section at the end. In [2] the controller architecture designed was composed of upper and lower levels controller. ACC controller is the upper-level controller, while the lower level is called the longitudinal controller. The ACC controller determines the desired acceleration that is transmitted to the longitudinal controller. The longitudinal controller oversees the throttle and/or brake command signal, which is required to track the desired acceleration [2,3] and returns the Error signal to the ACC controller [2]. [4,5,6] has shown that PID controllers work very well as longitudinal controllers.

All the literatures reviewed were highly sophisticated and complicated beyond the scope of EEE 411- Control System Course. Therefore, for a project of complexity within this course, example controller designs offered by Control Tutorials for MATLAB and Simulink (CTMS) [7] and MathWorks [8] were studied and one of the relatively simple system was reproduced with further attempt at analyzing the system via root locus and step response.

Background

A PI controller is an instrument used in industrial control applications to regulate temperature, flow, pressure, speed, and other process variables. PI (proportional integral) controllers use a control loop feedback mechanism to control process variables and are one of the most accurate and stable controllers. The proportional component gain (Kc) determines the ratio of output response to the error signal. In general, increasing the proportional gain will increase the speed of the control system response. However, if the proportional gain is too large, the process variable will begin to oscillate. If Kc is increased further, the oscillations will become larger and the system will become unstable and may even oscillate out of control. The integral component sums the error term over time. The result is that even a small error term will cause the integral component to increase slowly. The integral response will continually increase over time unless the error is zero, so the effect is to drive the Steady-State error to zero.

Steady-State error is the final difference between the process variable and set point. A phenomenon called integral windup results when integral action saturates a controller without the controller driving the error signal toward zero. Rise Time is the time it takes for the step response to rise from 10% to 90% of the steady-state response. Settling Time is the time it takes for the error $|y(t) - y_{final}|$ between the response y(t) and the steady-state response y_{final} to fall to within 2% of y_{final}.

Motivation

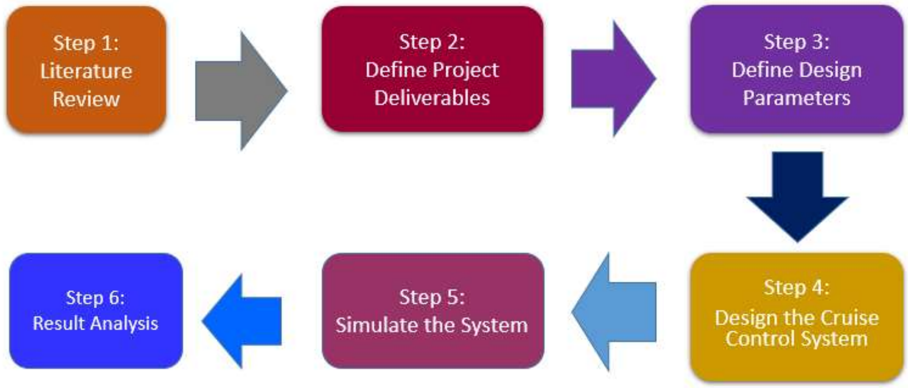
Cruise Control makes it easier and more comfortable to drive with less chance of accidents, reduces overall fuel consumption and tire wear. It also effectively deters drivers from speeding in a speed limit zone subconsciously.

4. Objectives, Methodology and Proposed System

The objective or this project is to build a PI (proportional integral) controller which uses a Unity Closed-Loop Feedback control mechanism to control the speed of the vehicle and maintain it at a desired speed r, here r = 30 m/s with the system being a stable one. The Settling time should be less than 15s, Rise time less than 5s, and Overshoot less than 5%.

Methodology

The following methodology was implemented for this project:



Proposed System

In the free-body diagram below, the vehicle, of mass m, is acted on by a control force, u. The force u represents the force generated at the road/tire interface. The resistive forces, bv, due to rolling resistance and wind drag, etc., are assumed to vary linearly with the vehicle velocity, v, and act in the direction opposite the vehicle's motion.

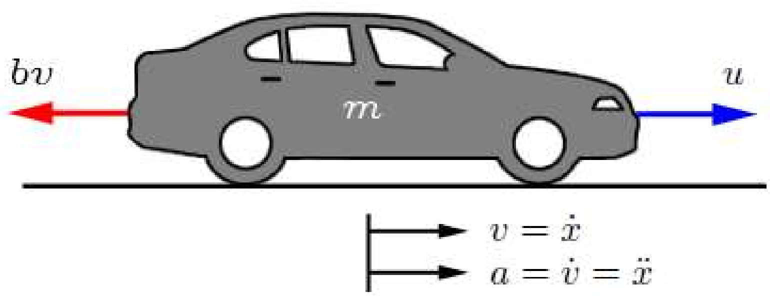


Figure: Free-body diagram

With these assumptions a first-order mass-damper system is produced. Summing forces in the x-direction and applying Newton's 2nd law, following system equation is derived:

$$m\dot{v} + bv = u$$

Since the speed of the vehicle is to be controlled, the output equation is chosen as follows:

$$y = v$$

The state-space representation is therefore:

$$\dot{\mathrm{x}} = [\dot{v}] = \left[\frac{-b}{m}\right][v] + \left[\frac{1}{m}\right][u]$$

$$y = [1][v]$$

Taking the Laplace transform of the governing differential equation and assuming zero initial conditions, the transfer function of the cruise control system is:

$$P(s) = \frac{1}{ms + b} \quad \left[\frac{m/s}{N}\right]$$

The transfer function of a PI controller is:

$$C = K_p + \frac{K_i}{s}$$

The block diagram of a typical unity feedback system is shown below:

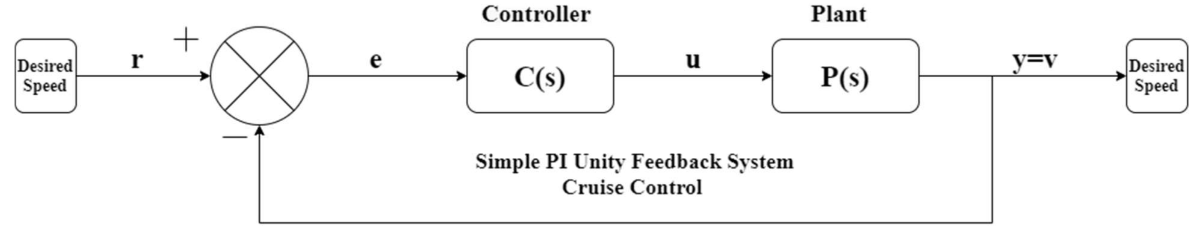


Figure: Block diagram of the control system

The closed-loop transfer function for this cruise control system with a PI controller

$$T(s) = \frac{Y(s)}{R(s)} = \frac{P(s)C(s)}{1 + P(s)C(s)} = \frac{K_p s + K_i}{ms^2 + (b + K_p)s + K_i}$$

5. Simulation Circuit / Hardware Prototype

The following design specifications, and parameters were followed:

i. *Transient Parameters:*
   - Rise time < 5 s
   - Settling time < 15 s
   - Overshoot < 5%
   - Steady-state error < 1%

ii. *System parameters:*
   - m = vehicle mass = 1200 kg;
   - b = damping coefficient = 35 N.s/m;
   - r = reference speed = 30 m/s;

The Simulink model representation of the system is shown below:

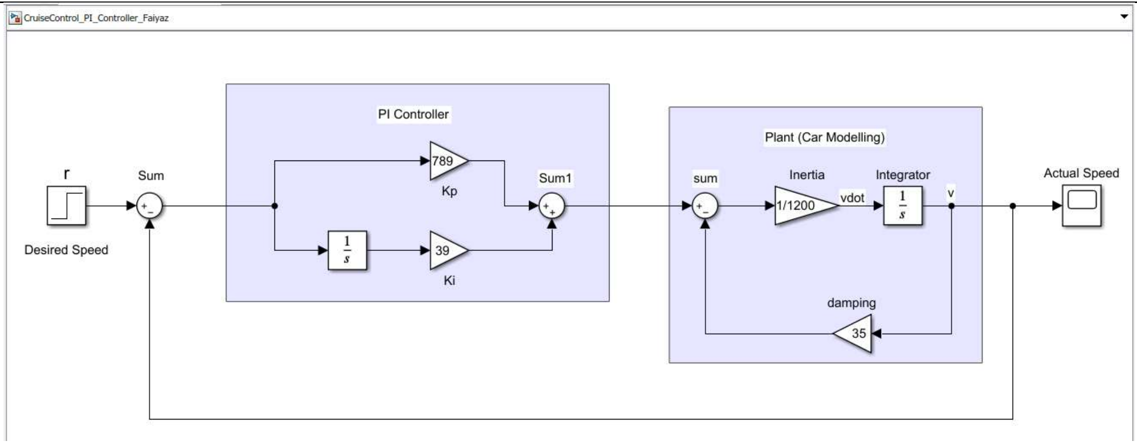


Figure: Simulated model

The transfer function obtained via the linear analysis of the system is shown below:

```
>> CruiseControlSystem

CruiseControlSystem =

     23670 s + 1170
  ---------------------
  1200 s^2 + 824 s + 39
```

Figure: Continuous time transfer function

6. Results and Discussions

Characteristic equation:

$$1200s^2 + 824s + 39 = 0$$

$$s^2 + \frac{824}{1200}s + \frac{39}{1200} = 0$$

Here $\mathbf{a} = \frac{\mathbf{824}}{\mathbf{1200}}$, $\mathbf{b} = \frac{\mathbf{39}}{\mathbf{1200}}$

Damping factor $= \frac{a}{2} =$ ***0.343*** N. m/s

Natural frequency $= \sqrt{b} =$ ***0.180 rad/s***

Damping ratio $= \frac{\textit{Damping factor}}{\textit{Natural frequency}} =$ ***1.906***

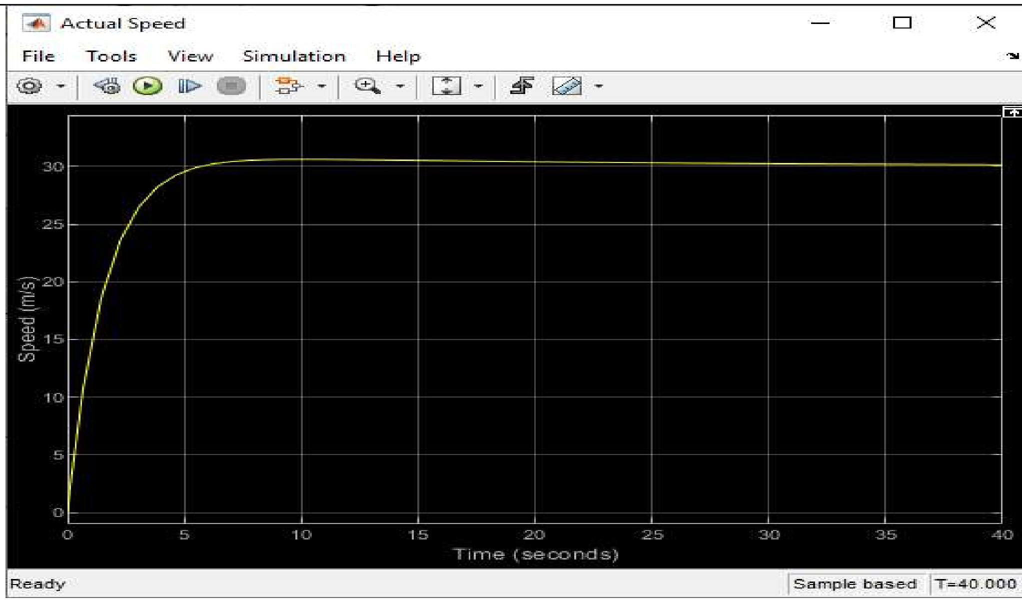


Figure: Output speed waveform

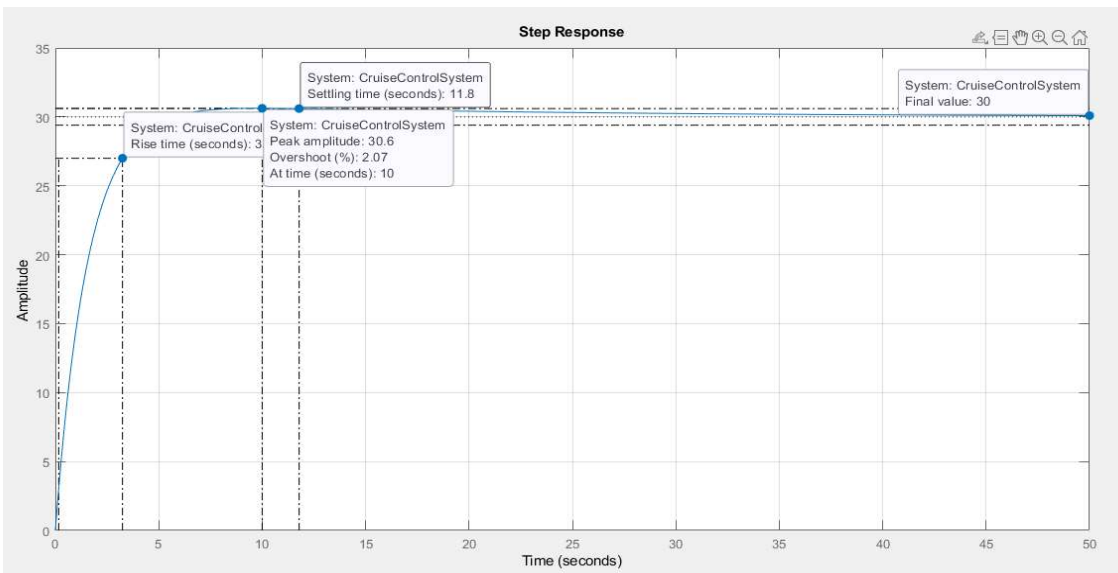


Figure: Step response of the system

Transient Parameters (from step response):

Characteristics	Value
Rise Time	3.09s
Settling Time	11.8s
Overshoot	2.07%
Peak Amplitude	30.6
Peak Time	10s
Steady State Speed	30 m/s
Steady State Error	0 m/s

Stability analysis:

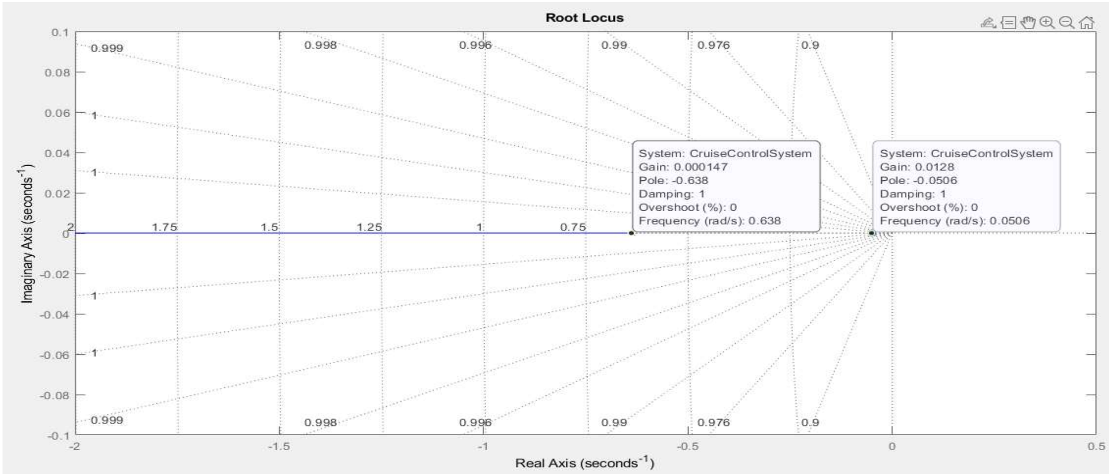


Figure: Root Locus Plot

Code to Verify Simulink Output

```
m = 1200;
b = 35;
r = 30;

Kp = 789;
Ki = 39;
Kd = 0;

t = 0:0.1:50;

s = tf('s');
C = pid(Kp,Ki,Kd);
P_cruise = 1/(m*s + b);
T = feedback(C*P_cruise,1);
CruiseControlSystem  = r*T;

figure('Name','Cruise Control','NumberTitle','off');
step(CruiseControlSystem ,t)
axis([0 50 0 r+5])
grid on; grid minor;

figure('Name','Cruise Control','NumberTitle','off');
rlocus(CruiseControlSystem)
grid on; grid minor;

CruiseControlSystem
```

Both *poles are real and distinct* (at positions -0.6 and -0.05 on the real axis), as seen from the root locus. As a result, the system is an overdamped system.

The system is stable, as there are *no poles on the right half plane*. There is one zero (at the position -0.05 on the real axis) as well on the Left Half Plane. The step response verifies that the system has zero steady state error, a *rise time* of 3.09s, a *settling time* of 11.8s, *overshoot* of 2.07 %, *peak amplitude* of 30.6 m/s, *peak time* at 10s, and the expected *steady state value* of 30 m/s.

The *characteristic equation* for pole location ($1200s^2 + 824s + 39$) was a second order system, where the *damping factor* is 0.343, *natural frequency* is 0.180, and the *damping ratio* is 1.906. From the damping ratio, it can also be seen that the system is overdamped as it is over 1.

For the PI controller, K_p = 789 and K_i = 39 were set, using the PID Tuner (SIMULINK), after carefully manual tuning the Settling Time and Overshoot values to be within the design parameters as defined before. In this simulation, the vehicle mass was assumed to be 1200kg, damping coefficient, b was 35 N.s/m, and the desired speed, r was 30 m/s.

7. Impact of Project on The Environment and Sustainability, and Society, Health, and Legal Issues

Cruise Control is useful for long journeys (reducing driver fatigue, improving convenience, and making it easy to place changes more safely) around highways and sparsely inhabited routes. It makes it easier for the driver to relax as it helps drivers to maintain the same pace without having to keep their right foot on the accelerator. Cruise control also helps decrease the tendency of the driver to accelerate, which can help improve safety and lower the risk of collisions and getting pulled over.

Keeping the engine at a constant pace dramatically increases fuel economy. Less gas is necessary to maintain a constant pace and hence less exhaust). Lower exhaust gas emission and lower fuel consumption makes Cruise Control an environment friendly System which is also sustainable.

8. Summary

Both *poles are real and distinct* (at positions -0.6 and -0.05 on the real axis), as seen from the root locus. As a result, the system is an overdamped system.

The system is stable, as there are *no poles on the right half plane*. There is one zero (at the position -0.05 on the real axis) as well on the Left Half Plane. The step response verifies that the system has zero steady state error, a *rise time* of 3.09s, a *settling time* of 11.8s, *overshoot* of 2.07 %, *peak amplitude* of 30.6 m/s, *peak time* at 10s, and the expected *steady state value* of 30 m/s.

The *characteristic equation* for pole location ($1200s^2 + 824s + 39$) was a second order system, where the *damping factor* is 0.343, *natural frequency* is 0.180, and the *damping ratio* is 1.906. From the damping ratio, it can also be seen that the system is overdamped as it is over 1.

For the PI controller, K_p = 789 and K_i = 39 were set, using the PID Tuner (SIMULINK), after carefully manual tuning the Settling Time and Overshoot values to be within the design parameters as defined before. In this simulation, the vehicle mass was assumed to be 1200kg, damping coefficient, b was 35 N.s/m, and the desired speed, r was 30 m/s.

While tuning the PI Controller, it was seen that a proportional controller, K_p, decreases the rise time, which is desirable in this case. Choosing a lower proportional gain, K_p, will give a reasonable rise time, and adding an integral controller, K_i, will eliminate the steady-state error. The simulated design satisfies the design requirements as seen by the unit step response.

9. References

[1] Lingyun Xiao & Feng Gao (2010) A comprehensive review of the development of adaptive cruise control systems, Vehicle System Dynamics: International Journal of Vehicle Mechanics and Mobility, 48:10, 1167-1192

[2] R. Rajamani, Vehicles Dynamics and Control, Springer, NewYork, 2006.

[3] C. Liang, Traffic-friendly adaptive cruise control design, Ph.D. Diss., University of Michigan, 2000.

[4] P. Ioannou, Z. Xu, S. Eckert, D. Clemons, and T. Sieja, Intelligent cruise control: Theory and experiment, Proceedings of the 32nd Conference on Decision and Control, San Antonio, Texas, 1993.

[5] P. Ioannou and C. Chien, Autonomous intelligent cruise control, IEEE Trans. Veh. Technol. 42 (1993), pp. 657–672.

[6] D.Yanakiev and I. Kanellakopoulos, Speed tracking and vehicle follower control design for heavy-duty vehicles, Veh. Syst. Dyn. 25 (1996), pp. 251–276.

[7] Control Tutorials for MATLAB and Simulink (CTMS) https://ctms.engin.umich.edu/CTMS/index.php?example=CruiseControl&section=SimulinkSimscape

[8] Ulusoy M, “Simulating Disturbance Rejection”. Mathworks 2019. [Online]. https://www.mathworks.com/videos/understanding-control-systems--part-4--simulating-disturbance-re-1480629735127.html