Design of PID Controller for Automobile Cruise Control System

Kiran M V
122101019@smail.iitpkd.ac.in
Indian Institute of Technology Palakkad

***Abstract*—The course project discusses on the design and simulation of a Proportional-Integral-Derivative (PID) controller for automobile cruise control system. Most of the automobiles are now featured with cruise control and the development of control systems become essential for improving fuel efficiency and driving experience. The PID controller is utilised to regulate the vehicle's speed and maintain it at constant level. The PID controller parameters, i.e, proportional (K_p), derivative (K_d) and integral (K_i) gains, are designed to achieve desired system performance parameters. Through modeling using transfer function blocks and controllers and through simulations (in MATLAB-MATLAB), the effectiveness of the proposed PID controller is demonstrated.**

***Index Terms*—PID Control, Cruise Control, MATLAB**

I. Objectives

The cruise control is a control system which allows the driver to set a certain speed, which will make the vehicle to move in a set speed, without the need for constant manual input on the accelerator pedal. It is a closed-loop control system that adjusts throttle or brakes to counteract deviations from the desired or set speed. The disturbances can be road direction changes, climbing slopes or wind speed. As automotive industry progress with help of automation and control systems, design of cruise control systems become essential. We will be using a PID control for implementing the cruise control system. This study aims to optimize the PID controller parameters, including the proportional gain (Kp), integral gain (Ki), and derivative gain (Kd) to achieve a balance between stability and response, which will enhance the overall driving experience and fuel efficiency. We also aim to analyse transient response parameters such as settling time, rise time, peak time, maximum overshoot and steady-state error, etc. The objectives of this paper also include to simulate the performance of the designed PID controller in MATLAB-MATLAB, by using transfer function blocks to model the cruise control system and controllers.

II. Motivation

This idea of this project came up from a genuine interest in understanding how cars regulate their speed automatically. Even if cruise control is a common feature in car nowadays, the implementation using simple PID control is appreciable. The ability to maintain steady speed and reduce disturbances is essential for ensuring a safe and comfortable driving experience. By studying the principles behind how cruise control works will help in understanding how technology can make driving safer and more efficient.

III. State of the Art

One of the most crucial challenges is the optimization of the cruise control system.
Despite its advantages, the optimal performance of the cruise control system relies heavily on the effectiveness of the feedback control mechanism. Various control systems have been proposed to address this challenge, including fuzzy logic, conventional PID, state space, hybrid control, ant lion optimization, particle swarm optimization, genetic algorithm-based PID control, adaptive control, etc. Recent studies have demonstrated the successful application of the new PID controller design methodologies which showed improved performance over conventional PID controllers.

IV. Methodology

A. System Overview

We chose PID as the control system due to simplicity, ease of implementation and effectiveness. The primary focus is the systematic tuning of PID

controller parameters, namely the proportional gain (Kp), derivative gain (Kd) and integrator gain (Ki), to attain an optimal balance between stability and response. We will look in detail about the above mentioned.

B. Modelling the Cruise Control System

We will first look at free body diagram of the model.

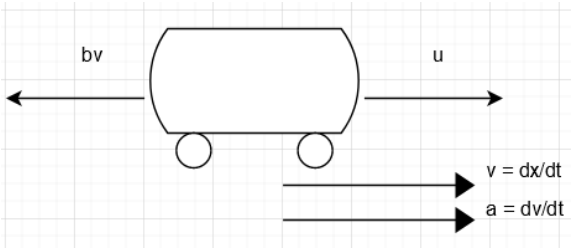

Fig. 1. FBD of model

Given the assumptions, we are dealing with a basic mass-damper system of first order. By adding up forces along the x-direction and utilizing Newton's second law, we derive the subsequent equation for the system.

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

The speed of the vehicle is given by

$$y = v$$

We assume system parameters as :

- Vehicle mass, m = 1000kg
- Damping coefficient, b = 50 N.s/m
- Reference speed, r = 10 m/s

For State-space model, state variables are required. Since our proposed model is first-order system, it will have only one state variable.

$$x = \begin{bmatrix} \dot{v} \end{bmatrix} = \begin{bmatrix} \frac{-b}{m} \end{bmatrix} \begin{bmatrix} v \end{bmatrix} + \frac{1}{m} \begin{bmatrix} \dot{u} \end{bmatrix}$$

$$y = \begin{bmatrix} 1 \end{bmatrix} \begin{bmatrix} v \end{bmatrix}$$

We will apply Laplace transform to the governing differential equation and assuming zero initial conditions, we obtain the transfer function of the cruise control system as follows:

$$P(s) = \frac{V(s)}{U(s)} = \frac{1}{ms+b}$$

C. Controller Design

Now we will use the transfer function and system parameters to design the controller. Below is the depicted block diagram of a standard unity feedback system.

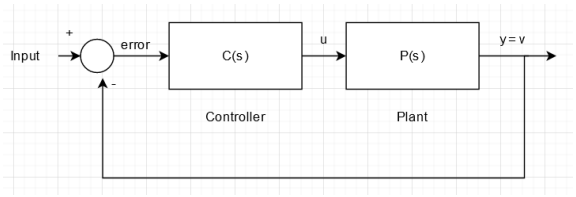

Fig. 2. Block diagram of model

We also know that, the transfer function of a general PID controller is given by:

$$C(s) = K_p + \frac{K_i}{s} + K_d s$$

The basic block diagram of a PID controller is given below

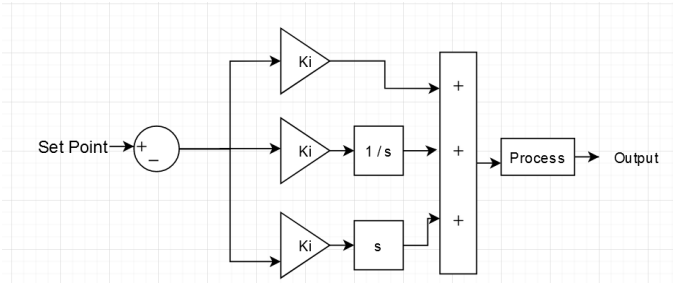

Fig. 3. Block Diagram of a PID controller

1) Characteristics of the P, I and D Terms:

- Increasing the proportional gain (K_p) makes the control signal increase proportionally for the same error level. This means the controller pushes harder for a given error, causing the closed-loop system to react more quickly(but also to overshoot more). Another effect of increasing K_p is that it tends to reduce the steady-state error.
- When you add a derivative term (K_d) to the controller, it gives the controller the ability to anticipate errors. In basic proportional control, the control signal only increases when the error increases, assuming a fixed K_p. But with derivative control, the control signal can increase if the error starts to trend upward, even if the error magnitude is still small. This anticipation helps to dampen the system, reducing overshooting. However, adding a derivative term doesn't affect the steady-state error.

- Adding an integral term to the controller (K_i) helps in reducing steady-state error. When there's a consistent, ongoing error, the integrator gradually builds up, increasing the control signal and minimizing the error. However, a downside of the integral term is that it can make the system slower and more prone to oscillations, as it may take time for the integrator to reset when the error signal changes direction.

2) Proportional Control: In a proportional (P) control system, the control action is directly proportional to the error signal, which is the difference between the desired setpoint and the actual process variable measurement.

The closed-loop transfer function of our unity-feedback system with a proportional controller is the following, where $X(s)$ is our output (equals $Y(s)$) and our reference $R(s)$ is the input:

$$T(s) = \frac{Y(s)}{R(s)} = \frac{K_p}{ms+(b+K_p)}$$

3) Proportional Integral Control: An addition of an integral controller to the system eliminates the steady-state error. The closed-loop transfer function of this cruise control system with a PI controller ($C = K_p + K_i/s$) is:

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

4) Proportional Integral Derivative Control: The derivative action introduces a component that is proportional to the rate of change of the error signal. It helps predict future changes in the error and can improve transient response and stability in systems.

The closed-loop transfer function for this cruise control system with a PID controller ($C = K_p + K_i/s + K_d s$) is:

$$T(s) = \frac{Y(s)}{R(s)} = \frac{K_d s^2+K_p s+K_i}{(m+K_d)s^2+(b+K_p)s+K_i}$$

5) Steps for Designing a PID Controller: When designing a PID controller for a specific system, we can achieve the desired response by following the steps outlined below.

- Obtain an open-loop response for the system
- Add a proportional control to improve the rise time
- Add an integral control to reduce the steady-state error
- Add a derivative control to reduce the overshoot
- Adjust gains K_p, K_i, and K_d until we obtain the desired response.

V. Results and Discussion

A. Open-Loop System Analysis

The system parameters are set as follows

- Vehicle mass, m = 1000kg
- Damping coefficient, b = 50 N.s/m
- Nominal control force, u = 500 N

In this scenario, the automobile should be capable of reaching a specified speed within a time frame of less than 5 seconds. In this application, a 10% overshoot and 2% steady-state error on the velocity are sufficient. So we set the design requirement as

- Rise time $< 5s$
- Overshoot $< 10\%$
- Steady-state error $< 2\%$

The transfer function model for the cruise control problem is

$$P(s) = \frac{V(s)}{U(s)} = \frac{1}{ms+b}$$

1) Open-Loop Step Response: A MATLAB simulation was conducted to observe the open-loop response of the system to a step input force of 500 Newtons, without any feedback control.

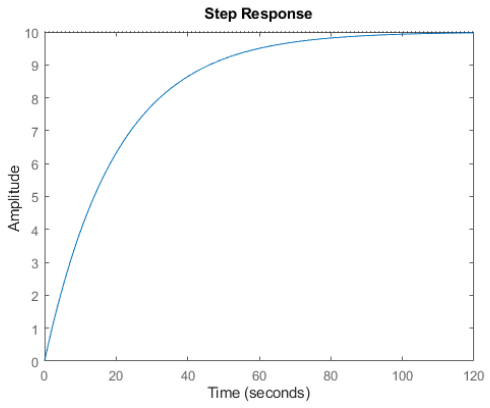

Fig. 4. Open-Loop Step Response

The open-loop system behaves as expected for a first-order system, with no overshoot or oscillations. However, the rise time is too slow, taking approximately 60 seconds to reach the desired steady-state speed of 10 m/s. A feedback controller needs to be designed to significantly speed up the system's response.

2) Open-Loop Poles/Zeros: The proposed system has a single pole at s = -b/m. MATLAB simulation of the same is shown below

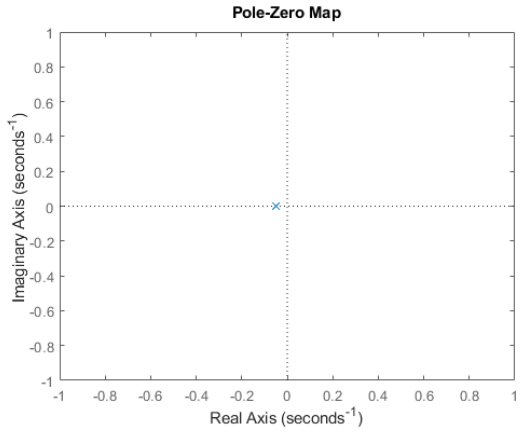

Fig. 5. Open-Loop Poles/Zeros

The open-loop system's stability is assured by the real and negative nature of the pole (left half plane pole). Absence of oscillations indicates stable behavior. Moreover, the system's response speed is determined by the pole's magnitude, $\frac{b}{m}$: a larger magnitude leads to quicker attainment of the steady-state value.

3) Open-Loop Bode Plot: The open-loop frequency response of the system is plotted in MATLAB.

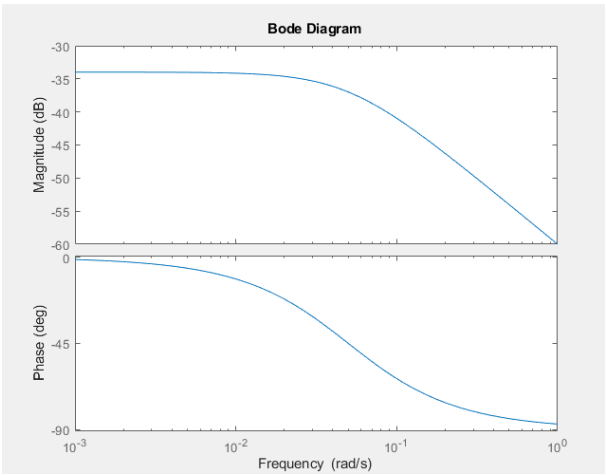

Fig. 6. Open-Loop Bode Plot

The Bode plots exhibit typical characteristics of first-order systems, such as a -3 dB magnitude and -45 degrees phase shift at the corner frequency $\omega = \frac{b}{m} = 0.05$ rad/s. Additionally, a -20 dB/dec roll-off is observed at high frequencies.

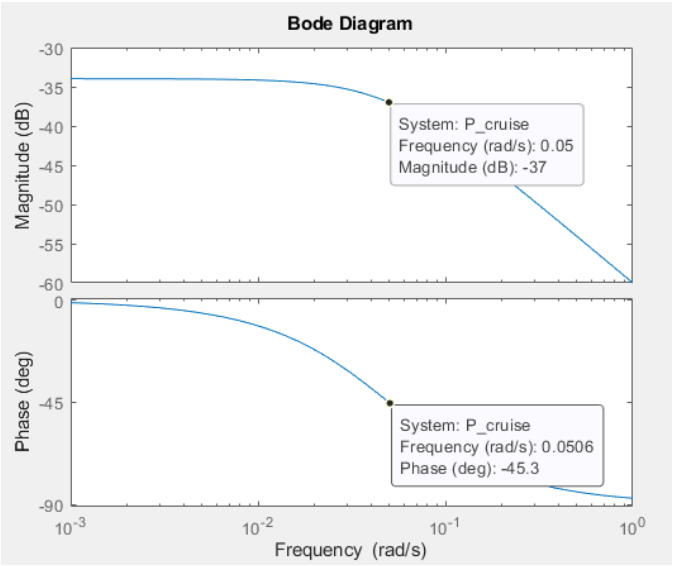

Fig. 7. Open-Loop Bode Plot Parameters

B. Proportional Control

To find the closed-loop transfer function with a proportional controller added, we reduce the unity feedback block diagram.

Given that a proportional controller, K_p, decreases the rise time, which is desirable in this case, we can use K_p, = 600 and a reference speed of 10 m/s for now.

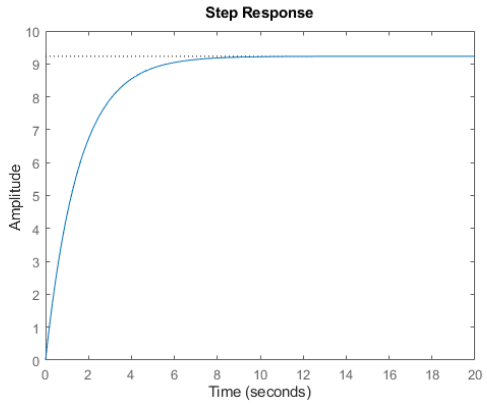

Fig. 8. P Controller Response

The steady-state error is now nearly zero and the rise time has decreased significantly

C. Proportional Integral Control

Introducing an integral controller to the system eliminates the steady-state error. Let's set $K_p = 600$ and $K_i = 1$ to observe the system's response.

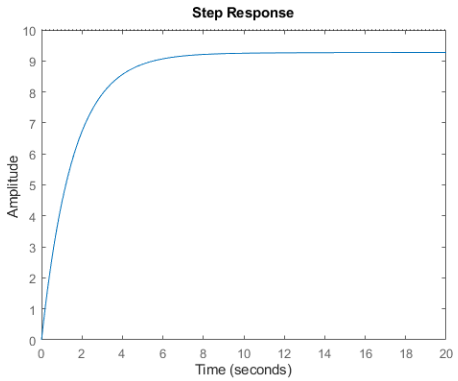

Fig. 9. PI Response at $K_p = 600$ and $K_i = 1$

Now we will set $K_p = 800$ and $K_i = 40$ to observe the system's response.

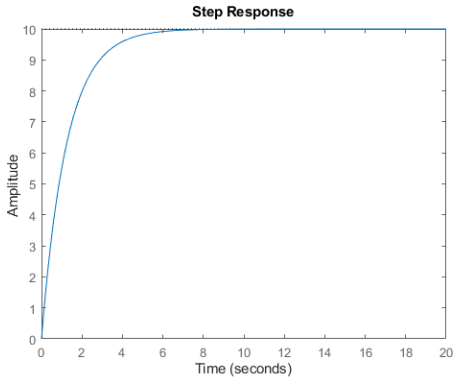

Fig. 10. PI Response at $K_p = 800$ and $K_i = 40$

D. Proportional Integral Derivative Control

In certain control system applications with relatively simple dynamics or low-order systems, the derivative (D) term in the PID controller may not be necessary to achieve the required output. The step response for K_p=600, K_d=20, and K_i=20 is given below

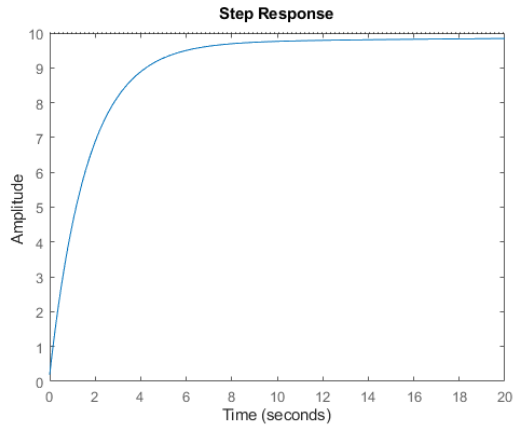

Fig. 11. PID Response at $K_p = 600$, $K_d = 20$ and $K_i = 20$

The step response for K_p=800, K_d=10, and K_i=40 is given below

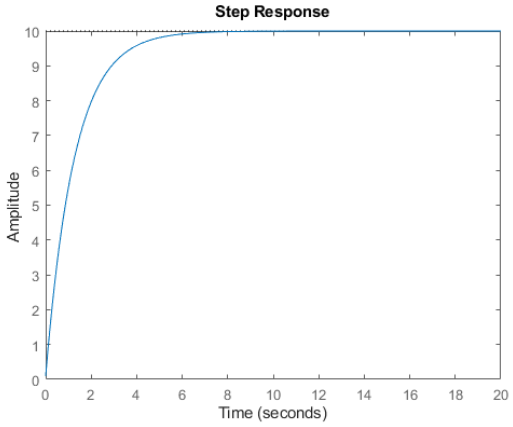

Fig. 12. PID Response at $K_p = 800$, $K_d = 10$ and $K_i = 40$

After several iterations of tuning, we can set the appropriate values of gain K_p, K_d and K_i to get the desired values. So now, we have designed a closed-loop system that exhibits less overshoot, achieves fast rise time and tries to eliminates steady-state error.

VI. Conclusion

We have explored the design and simulation of a Proportional-Integral-Derivative (PID) controller tailored for automobile cruise control systems. The objectives were to systematically tune PID controller parameters, analyze transient response parameters and evaluate the effectiveness of the proposed controller in regulating vehicle speed amid varying road conditions and other environmental disturbances. Through MATLAB simulation and analysis after that, the effectiveness of the proposed PID controller was able to be demonstrated. By adjusting controller parameters based on desired performance metrics such as settling time and maximum overshoot, the PID controller showcased improved stability and response compared to conventional methods.

VII. Future Perspective

A. Adaptive Cruise Control

Adaptive cruise control is an improved version of cruise control. In cruise control, vehicle adjust speed with respect to the feedback which is sent by the PID controller. But in adaptive cruise control, the

vehicle automatically adjusts the speed of your car to match the speed of the car in front of you. If the car ahead slows down, it can automatically match it. It can be implemented by including sensors and control or machine learning algorithms

B. Lane Assist Systems

Lane assist systems track the location of the vehicle on the road, identify when the driver accidentally leaves their lane and respond by actively guiding the vehicle back into its lane or issuing warnings.

When adaptive cruise control is combined with lane assist systems, we will be able to achieve Level 2 or Level 3 of autonomy. High end sensors like LiDAR, radar and cameras can be incorporated into those systems for better performance and efficiency.

REFERENCES

1 K. Osman, M. F. Rahmat and M. A. Ahmad, "Modelling and controller design for a cruise control system," 2009 5th International Colloquium on Signal Processing & Its Applications, Kuala Lumpur, Malaysia, 2009, pp. 254-258, doi: 10.1109/CSPA.2009.5069228.

2 Sailan, Khaled & Kuhnert, Klaus.Dieter. (2013). Modeling and Design of Cruise Control System with Feedforward for all Terrian Vehicles. Computer Science & Information Technology. 3. 339-349. 10.5121/csit.2013.3828.

3 M. K. Rout, D. Sain, S. K. Swain and S. K. Mishra, "PID controller design for cruise control system using genetic algorithm," 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Chennai, India, 2016, pp. 4170-4174, doi: 10.1109/ICEEOT.2016.7755502.

4 LibreTexts. "Modeling and PID Controller Example - Cruise Control for an Electric Vehicle", *Chemical Process Dynamics and Controls (Woolf)*, [Online]. Available: Reference Link