Simulation and Design of a Vehicle Cruise Control System

J.Sai Shanmukhnath
Department of Electrical And Computer Engineering
email: sjonnal3@gmu.edu

V.Shantan Rami Reddy
Department of Electrical And Computer Engineering
email: svenka20@gmu.edu

Abstract— A Cruise Control System automatically maintains a vehicle's speed without constant driver input, making it especially useful for highway travel by reducing the need for manual acceleration. While ideal for steady traffic, its effectiveness decreases in congested conditions due to frequent speed changes. The system uses feedback control to match the current speed with the desired value by adjusting the throttle. Modern versions also adapt to road conditions like slopes and tire variations. Effective cruise control relies on sensor integration and control algorithms to ensure accurate and reliable performance.

Keywords— Cruise Control System, feedback control, Throttle adjustment, Reliable performance, Slopes

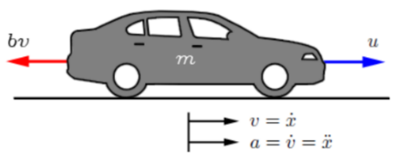
I. INTRODUCTION

A Cruise Control System (CCS) is a speed-regulating mechanism used in vehicles to automatically maintain a set speed without constant driver input. It is particularly useful during long highway journeys, reducing driver fatigue and promoting better fuel economy. The CCS operates through a closed-loop feedback system, where sensors monitor the vehicle's current speed and controllers—commonly PID (Proportional-Integral-Derivative)—compare it to the target speed set by the driver. Any difference (error) between the two triggers an adjustment in throttle position, ensuring the vehicle maintains the desired velocity. Vehicle dynamics, such as mass, engine force, aerodynamic drag, and road gradients, are incorporated into the system design. Engineers simulate these factors using software tools like MATLAB/Simulink to model and optimize system behaviour. The control system must respond smoothly to avoid overshoot or lag, providing a stable and comfortable driving experience. In addition to the basic functionality, modern implementations include features to adapt to external influences such as road slope, headwinds, or rolling resistance. Advanced control strategies—like fuzzy logic or adaptive gain tuning—may be used to improve accuracy and adaptability. The system typically includes several core components: speed sensors to monitor velocity, actuators to manipulate the throttle, a microcontroller to run the control logic, and a user interface (buttons or switches) for setting or modifying the cruise control settings. For safety, cruise control is designed to disengage when the driver applies the brake or clutch, or if a system malfunction occurs. As vehicle technology advances, cruise control systems are being enhanced into Adaptive Cruise Control (ACC), which uses radar or lidar sensors to maintain a safe distance from the vehicle ahead, enabling semi-autonomous driving capabilities. These systems require careful coordination of mechanical, electrical, and software components to ensure reliability under varied operating conditions. Sensor fusion, control algorithm tuning, and robust system design are critical in achieving a well-performing cruise control system.

II. METHODOLOGY

Automatic cruise control is an excellent example of a feedback control system found in many modern vehicles. The purpose of the cruise control system is to maintain constant vehicle speed despite external disturbances, such as changes in wind or road grade. This is accomplished by measuring the vehicle speed, comparing it to the desired or reference speed, and automatically adjusting the throttle according to a control law.

The mass of the car is represented by **'m'**, which is acted upon by the force, pull **'u'** that is generated at the interface between the tire and the surface of the road. It is also assumed of the model that **'u'** can be controlled while the dynamic of the power train is neglected. The oppositional pull, **'v'** is assumed to be varied linearly with velocity of the car, and it acts opposite to the motion of the car.



With these assumptions we are left with a first-order mass-damper system. Summing forces in the x-direction and applying Newton's 2nd law, we arrive at the following system equation:

$$\mathbf{m\dot{v} + bv = u} \quad \textbf{(1)}$$

Since we are interested in controlling the speed of the vehicle, the output equation is chosen as follows:

$$\mathbf{y = v} \quad \textbf{(2)}$$

First-order systems have only a single energy storage mode, in this case the kinetic energy of the car, and therefore only one state variable is needed, which is the velocity. The state-space representation is therefore given by:

$$\mathbf{\dot{v} = (-b/m)\, v + (1/m)\, u} \quad \textbf{(3)}$$

$$\mathbf{y = (1)(u)} \quad \textbf{(4)}$$

The transfer function of the system would be given as follows **equation (5):**

$$\boldsymbol{G(s) = C(SI - A)^{-1}B + D}$$

A = (-b/m), B = (1/m), C = 1, D = 0

Substitute all these values in equation (5):

$$\boldsymbol{G(s) = 1\left(SI - \left(-\frac{b}{m}\right)\right)^{-1}\left(\frac{1}{m}\right) + 0}$$

$$\boldsymbol{G(s) = 1\left(SI + \left(\frac{b}{m}\right)\right)^{-1}\left(\frac{1}{m}\right)}$$

$$\boldsymbol{G(s) = \left(SI + \left(\frac{b}{m}\right)\right)^{-1}\left(\frac{1}{m}\right)}$$

$$\boldsymbol{G(s) = \frac{\frac{1}{m}}{\left(S+\left(\frac{b}{m}\right)\right)}}$$

$$\boldsymbol{G(s) = \frac{1}{ms+b}} \quad \textbf{(6)}$$

III. CONTROLLER DESIGN

With its three-term functionality covering treatment to both transient and steady-state responses, proportional-integral-derivative (PID) control offers the simplest and yet most efficient solution to many real-world control problems. Since the invention of PID control in 1910 (largely owning to Elmer Sperry's ship autopilot), and the Ziegler–Nichols' (Z-N) straightforward tuning methods in 1942, the popularity of PID control has grown tremendously. With advances in digital technology, the science of automatic control now offers a wide spectrum of choices for control schemes. However, more than 90% of industrial controllers are still implemented based around PID algorithms, particularly at lowest levels, as no other controllers match the simplicity, clear functionality, applicability, and ease of use offered by the PID controller. Its wide application has stimulated and sustained the development of various PID tuning techniques, sophisticated software packages, and hardware modules.

A PID controller may be considered as an extreme form of a phase lead-lag compensator with one pole at the origin and the other at infinity. Similarly, its cousins, the PI and the PD controllers, can also be regarded as extreme forms of phase-lag and phase-lead compensators, respectively. A standard PID controller is also known as the "three-term" controller, whose transfer function is generally written in the "parallel form" or the "ideal form" given by:

$$\text{Parallel Form} = G(s) = Kp + Ki\left(\frac{1}{s}\right) + Kds$$

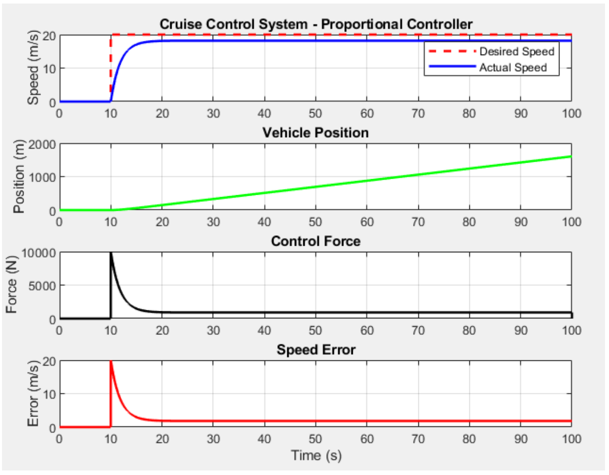
$$\text{Ideal Form} = G(s) = Kp(1 + \frac{1}{Tis} + Tds)$$

where **'Kp'** is the proportional gain, **'Ki'** the integral gain, **'Kd'** the derivative gain, **'Ti'** the integral time constant and, **'Td'** the derivative time constant. The "three-term" functionalities are highlighted by the following.

- The proportional term—providing an overall control action proportional to the error signal through the all-pass gain factor.
- The integral term—reducing steady-state errors through low-frequency compensation by an integrator.
- The derivative term—improving transient response through high-frequency compensation by a differentiator.

IV. ANALYSIS OF THE MODEL

Proportional Controller:



This image shows the simulation results of a cruise control system using only a Proportional (P) controller. The graphs depict the system's behaviour over time (100 seconds) with four key metrics:

1. Top graph (Cruise Control System - Proportional Controller): Shows the vehicle's speed response. The dashed red line represents the desired speed (20 m/s),

while the solid blue line shows the actual speed. The vehicle accelerates from 0 m/s starting around the 10-second mark and approaches but doesn't quite reach the target speed.

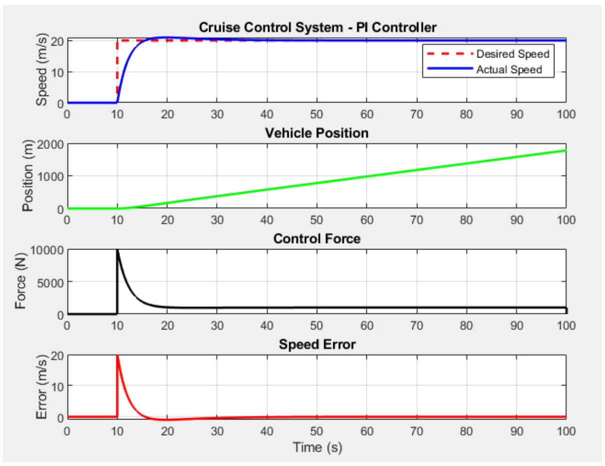
2. Second graph (Vehicle Position): Displays the cumulative distance travelled by the vehicle in meters. The green line shows a steadily increasing position as the vehicle moves forward.

3. Third graph (Control Force): Shows the force applied by the controller in Newtons. There's a large initial force spike (around 9,000 N) to accelerate the vehicle, which then decreases and settles at a constant level.

4. Bottom graph (Speed Error): Represents the difference between desired and actual speed. The red line shows a large initial error of 20 m/s (when the vehicle is stationary), which decreases but importantly never reaches zero, stabilizing at a constant steady-state error.

The key observation in this proportional-only controller is the persistent steady-state error shown in the bottom graph. The P controller cannot eliminate the steady-state error completely. This is a classic limitation of proportional-only control - it can respond to errors but cannot completely eliminate them in systems that require a non-zero control input to maintain the setpoint (like overcoming friction or wind resistance in a vehicle).

Proportional Integral Controller:



This image shows the simulation results of a cruise control system using a PI (Proportional-Integral) controller. The graphs depict the system's behaviour over time (100 seconds) with four key metrics:

1. Top graph (Cruise Control System - PI Controller): Shows the vehicle's speed response. The dashed blue line represents the desired speed (set point) of 20 m/s, while the solid blue line shows the actual speed. The vehicle starts at 0 m/s and accelerates, reaching the target speed around the 15-second mark with minimal overshoot.

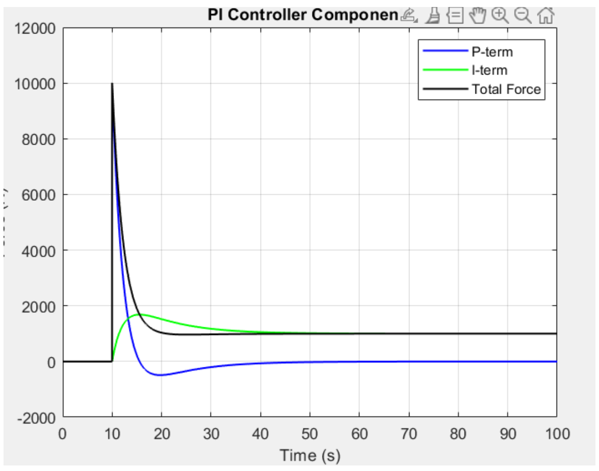
2. Second graph (Vehicle Position): Displays the cumulative distance travelled by the vehicle in meters. The green line shows a steadily increasing position as the vehicle moves forward, with the slope becoming steeper once the vehicle reaches its target speed.

3. Third graph (Control Force): Shows the force applied by the controller in Newtons. There's a large initial force (around 10,000 N) to accelerate the vehicle, which then decreases and stabilizes at approximately 2,000 N once the desired speed is maintained.

4. Bottom graph (Speed Error): Represents the difference between desired and actual speed. The red line shows a large initial error of 20 m/s (when the vehicle is stationary), which rapidly decreases as the vehicle accelerates, eventually approaching zero as the actual speed converges with the desired speed.

Overall, this visualization demonstrates a well-tuned PI controller for cruise control, showing how the system responds to reach and maintain a target speed while minimizing error and applying appropriate control forces.

Proportional Integral Controller Controller:



This graph shows the component breakdown of a PI (Proportional-Integral) controller for a cruise control system over a 100-second simulation period. The graph displays three key components:

1. P-term (Blue line): This shows the proportional component of the control force. It spikes sharply to about 8,000 N at around the 10-second mark when the error is largest (likely when the speed command is first applied). It then decreases rapidly and actually goes negative between approximately 15-40 seconds, indicating a brief period where the vehicle speed slightly overshoots the target. The P-term eventually stabilizes near zero as the error approaches zero.

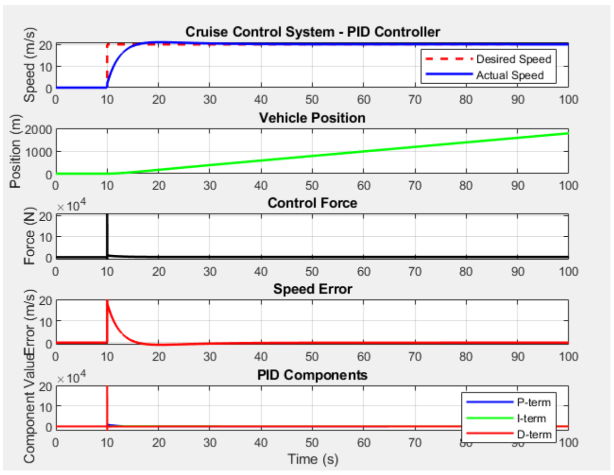
2. I-term (Green line): This represents the integral component of the control force. Starting from zero, it

gradually increases as the controller accumulates error over time. The I-term reaches approximately 1,600 N and maintains this level for the remainder of the simulation. This steady-state value is critical for maintaining the desired speed by overcoming constant disturbances like friction or wind resistance.

3. Total Force (Black line): This is the sum of the P-term and I-term, representing the total control force applied. It peaks at about 10,000 N during initial acceleration and then stabilizes at the same level as the I-term (around 1,000-1,200 N) once steady state is reached.

This graph illustrates the complementary roles of the two components in a PI controller: the P-term provides quick initial response to errors while the I-term eliminates steady-state error by accumulating over time. The negative P-term during part of the response shows how the proportional component can temporarily oppose the integral component to prevent excessive overshoot.

Proportional Integral Derivative Controller:



This image shows the simulation results of a cruise control system using a PID (Proportional-Integral-Derivative) controller. The graphs illustrate the system's performance over a 100-second time period:

1. Top graph (Cruise Control System - PID Controller): Shows the vehicle speed response. The dashed red line indicates the desired speed (20 m/s), while the solid blue line shows the actual speed. The vehicle starts from rest and reaches the target speed around the 15-second mark with minimal overshoot.

2. Second graph (Vehicle Position): Displays the cumulative distance travelled by the vehicle in meters. The green line shows a steadily increasing position as the vehicle accelerates and maintains its speed.

3. Third graph (Control Force): Shows the force applied by the controller in Newtons. There's a significant initial spike (around 20,000 N) to accelerate the vehicle, which then quickly decreases to a lower steady-state value needed to maintain speed.

4. Fourth graph (Speed Error): Represents the difference between desired and actual speed. The error starts at 20 m/s (when the vehicle is stationary) and rapidly decreases as the vehicle approaches the target speed.

5. Bottom graph (PID Components): Breaks down the contributions from each term of the PID controller. The blue line shows the proportional term (P-term), the green line shows the integral term (I-term), and the red line shows the derivative term (D-term). The P-term and D-term spike during the initial acceleration phase, while the I-term gradually increases to counteract steady-state errors.

The additional PID components graph (compared to the previous PI controller image) shows how each control component contributes to the overall system performance, with the derivative term helping to improve the transient response characteristics.

V. CONCLUSION

The cruise control system was successfully analyzed and simulated using different control strategies: Proportional (P), Proportional-Integral (PI), and Proportional-Integral-Derivative (PID) controllers. The Proportional controller provided a fast response but exhibited a steady-state error, failing to fully achieve the desired speed. Adding an Integral component (PI controller) eliminated the steady-state error, improving long-term accuracy and ensuring the vehicle maintained the target speed with minimal overshoot. However, the PI controller showed a slight overshoot and longer settling time during transients. The PID controller further enhanced performance by introducing a Derivative term, which significantly improved the transient response, minimized overshoot, and enabled quicker settling without compromising stability. The simulations demonstrated that the PID controller delivered the best overall performance for cruise control, providing both rapid error correction and steady-state accuracy. Therefore, for applications demanding both quick adaptations to changes and precise speed maintenance, a well-tuned PID controller is the most effective solution.
The study highlights the importance of selecting appropriate control strategies and fine-tuning controller parameters to achieve reliable, smooth, and efficient cruise control operation under varying driving conditions.

VI. REFERENCES

[1] https://www.researchgate.net/publication/353925596_CONTROL_DESIGN_AND_ANALYSIS_OF_CRUISE_CONTROL_SYSTEM

[2] https://ieeexplore.ieee.org/document/10009608

[3] https://ctms.engin.umich.edu/CTMS/index.php?example=CruiseControl&section=SystemModeling

[4] https://ieeexplore.ieee.org/document/1453566

[5] https://www.digikey.com/en/maker/projects/introduction-to-pid-controllers/763a6dca352b4f2ba00adde46445ddeb