Control Systems - II

Modeling and PID Control of a Cruise Control System for a Toyota Corolla Using MATLAB & Simulink

Submitted To:

Prof. Dr. Abdullah Sheeraz

Submitted By:

Muhammad Haider Ali (2022-MC-45)

Department of Mechatronics & Control Engineering, University of Engineering & Technology, Lahore

January 09, 2025

Introduction:

Cruise control systems have become a standard feature in modern vehicles, allowing the driver to maintain a constant speed without continuous throttle input. This project focuses on the modeling, simulation, and control design of a simple cruise control system using MATLAB Simulink, tailored specifically for a compact sedan — the **Toyota Corolla 1.8L**.

The system is modeled as a **first-order mass-damper system**, representing the vehicle's longitudinal dynamics. Key parameters for modeling — including vehicle mass, drag coefficient, rolling resistance, and maximum engine torque — are derived from the specifications of the Toyota Corolla 1.8L, ensuring realistic system behavior.

Both **continuous and discrete-time models** of the system have been developed in Simulink. To regulate the vehicle speed, a **PID controller** has been designed and implemented. PID gains were tuned using MATLAB's **PID Tuner tool** to achieve the desired performance. The control objective was to maintain a **target cruising speed of 100 km/h (27.78 m/s)**, representative of typical highway driving conditions.

To evaluate system performance, a disturbance input mimicking a transition from a flat road to an inclined terrain was introduced. The system's **stability and robustness** were then analyzed in the presence of this external disturbance. The controller was tuned such that the vehicle's **rise time** remained under **12 seconds**, aligned with the real-world capability of the Honda City, which accelerates from 0 to 100 km/h in approximately **10.8 seconds**.

This simulation serves as a foundation for understanding basic vehicle control systems and offers insight into control theory, real-world parameter tuning, and the practical application of PID controllers in automotive engineering.

Car Model: Toyota Corolla 1.8L (Example)

System Parameters for Toyota Corolla 1.8L:

- **Curb weight (m):** 1310 kg
- **Max torque (u):** 170 Nm
- **Drag coefficient (Cd):** 0.28
- **Rolling resistance coefficient (Cr):** 0.013
- **Wind speed:** 4.4 km/h
- **Wheel radius (r):** 0.305 m (approx.)
- **Inclination (θ):** 0 degrees

Mathematical Modeling:

The equation governing the dynamics of the vehicle can be written as:

$$m\{x\}''(t) = F - F_d - F_r - mgsin(\theta)$$

Where:

- m = mass of the vehicle
- F = nominal force applied by the engine
- $\mathbf{F_d}$ = aerodynamic drag force
- Fr = rolling resistance
- g = acceleration due to gravity (9.81 m/s²)
- θ = road inclination (0 for flat road)

Aerodynamic Drag Force (F_d):

The aerodynamic drag force is given by:

$$F_d = C_d \frac{1}{2} \rho \cdot v^2 \cdot A$$

Where:

- c_d = drag coefficient (for Toyota Corolla, c_d = 0.28)
- p = air density (1.2 kg/m³)
- v = velocity of the vehicle (in m/s)
- A = frontal area of the vehicle (2.24 m² for Toyota Corolla)

Substituting the values for Toyota Corolla, we get:

$$F_d = 365.1\ N$$

Rolling Resistance (F_r):

The rolling resistance is given by:

$$F_r = C_r mg$$

Where:

- C_r= rolling resistance coefficient (for Toyota Corolla, C_r= 0.013)
- m = mass of the vehicle (1310 kg)
- g = acceleration due to gravity (9.81 m/s²)

Substituting the values:

$$F_r = 168.9N$$

Nominal Force (u):

The nominal force is given by the maximum torque applied at the wheels, divided by the radius of the wheel:

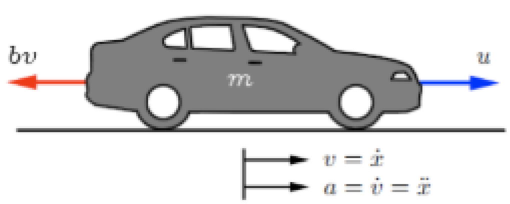
u = Max Torque / Wheel Radius

u = 557.38N

System Dynamics:

The **first-order mass-damper system** governing the motion of the vehicle can be written as:

$$m\frac{dv}{dt} + b\,v = u$$



Where v is the velocity, a is the acceleration and b is **damping coefficient** related to drag and rolling resistance forces.

From the drag and rolling resistance forces, the damping coefficient b can be calculated as:

$$b = \frac{F_d + F_r}{v}$$

For a reference speed v=27.78m/s, we get:

b = 19.2 Ns/m

State-Space Representation:

The state-space representation of the system is:

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

Where, x=[v] is the state vector.

The output equation is simply:

$$y = [1] * v$$

Design Criteria:

The **performance specifications** for the cruise control system are as follows:

- **Rise time** < 12 seconds
- **Overshoot** < 10%
- **Steady-state error** < 2%

These criteria are designed to closely match the real-world performance of a vehicle like the **Toyota Corolla**, which can accelerate from 0 to 100 km/h in approximately 9.5 seconds.

Closed Loop Simulink Model w/ PID Controller in Laplace Domain:

Non-linear model is designed in MATLAB Simulink. And MATLAB's in built 1 D.O.F PID Block is used to tune the model. And get the output. First, we made the model using Plant Transfer Function which is in s-domain (Laplace Domain). The PID is auto-tuned using MATLAB's PID Tuner App.

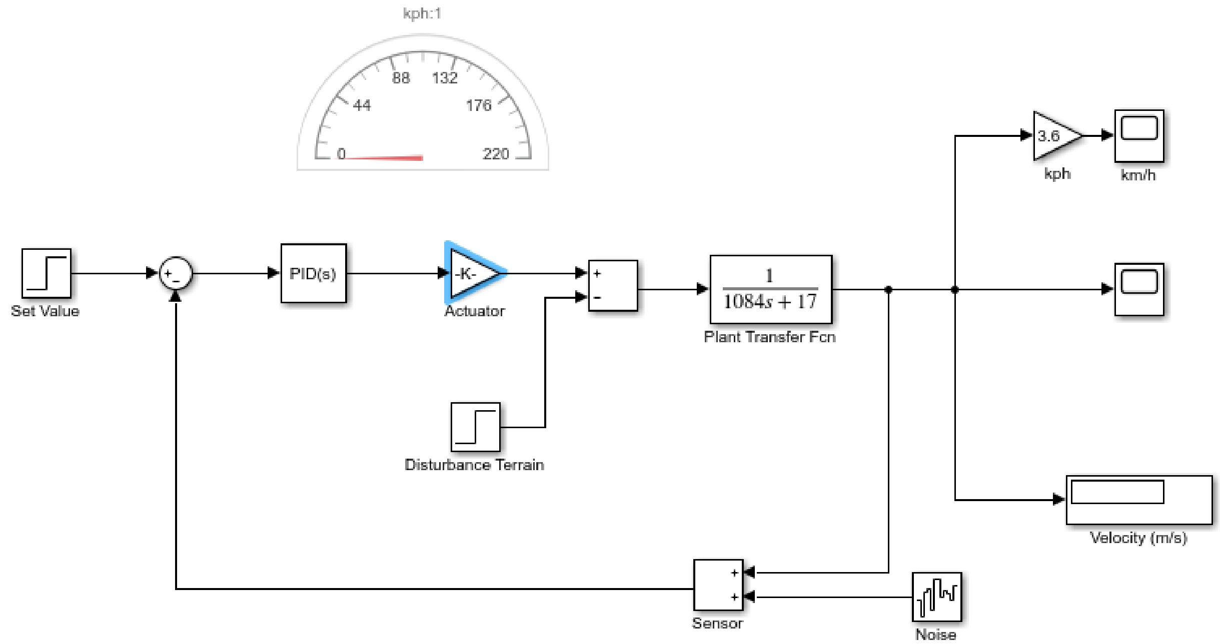


Fig1.2: Closed Loop Simulink Model w/ PID Controller in s-domain

The model is simulated at 700.0 Stop Time and Fig1.3 shows the velocity (m/s) of the cruise in s-domain.

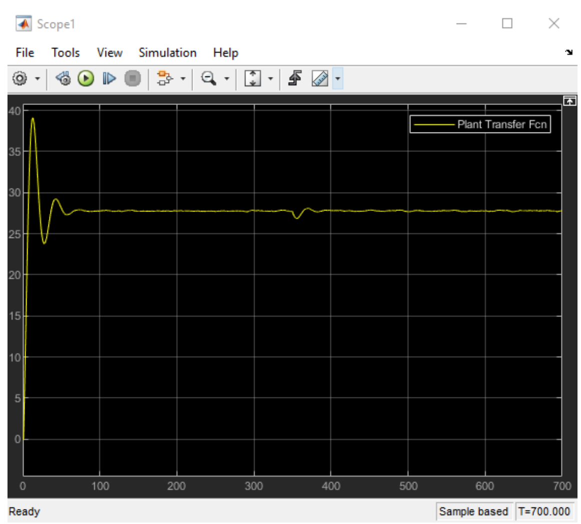


Fig1.3: Cruise's velocity (m/s) in s-domain

Closed Loop Simulink Model w/ PID Controller in Discrete-Time Domain:

Now the closed-loop model is modified to be in discrete-time domain. The PID controller in continuous time is replaced by discrete time and discrete transfer function is added instead of the plant model. The closed loop model in z-domain is shown in Fig1.4:

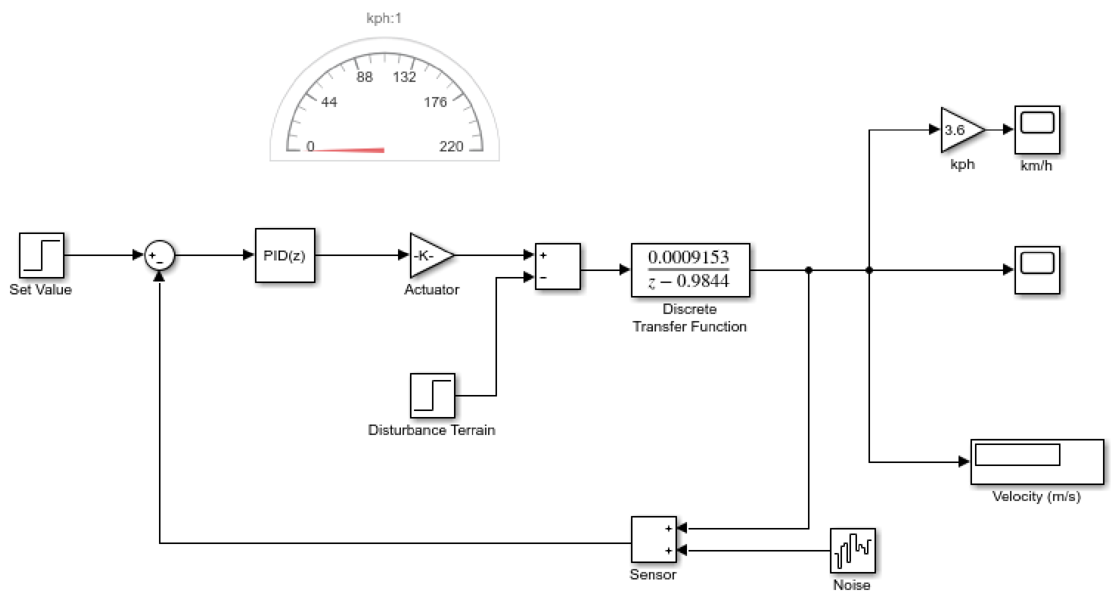


Fig1.4: Closed Loop Simulink Model w/ PID Controller in z-domain

Fig1.5 shows the velocity (m/s) of the cruise in z-domain.

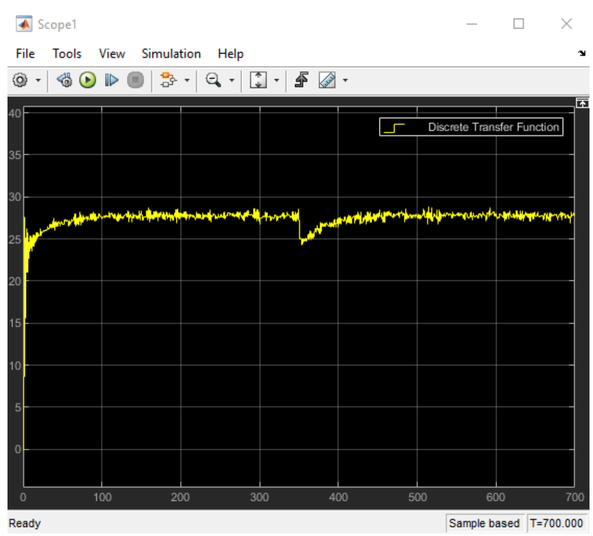


Fig1.5: Cruise's velocity (m/s) in z-domain

In both of the cases, the velocity is stabilized and remained constant at 28.16 m/s.

Analysis of Cruise Control System in MATLAB:

The cruise system is now analyzed with MATLAB. Following is the code for the cruise's control analysis.

```
% ===========================
%  Vehicle Speed Control System
%  Continuous to Discrete Conversion with Analysis
%  ===========================

clc;
clear;
close all;

% System Parameters
m = 1084;           % Mass of the vehicle (kg)
b = 17;             % Damping coefficient (N·s/m)
u = 475.72;         % Nominal force input (N)
r = 27.78;          % Desired speed (m/s)

% Continuous-Time Transfer Function (s-domain)
% G(s) = 1 / (ms + b)
numerator = 1;
denominator = [m, b];            % [mass, damping]
s = tf('s');
Gs = tf(numerator, denominator); % Continuous-time transfer function
Gs

% PID Controller Design
Kp = 1200;   % Proportional gain (tune as needed)
Ki = 10;     % Integral gain
Kd = 0;      % Derivative gain
Gc = pid(Kp, Ki, Kd);    % PID controller

% Closed-Loop System (Continuous-Time)
sys_cl = feedback(Gc * Gs, 1);  % Unity feedback

% Step Response of Continuous System
% Simulate the response to a step input of magnitude 'u'
figure;
step(u * Gs)
title('Step Response of Continuous-Time System');
xlabel('Time (s)');
ylabel('Velocity (m/s)');
grid on;

% Step Info
disp('Step Response Info (Continuous):');
stepinfo(u * sys_cl)
```

```
% Damping Characteristics
disp('Damping Characteristics (Continuous):');
damp(sys_cl)

% Convert to Discrete-Time System (z-domain)
Ts = 1;                      % Sampling time (seconds)
Gz = c2d(Gs, Ts);            % Discrete-time transfer function (default method:
ZOH)
Gz

% Pole-Zero Map of Discrete-Time System
figure;
pzmap(Gz)
title('Pole-Zero Map of Discrete-Time System');
grid on;
axis([-1.2 1.2 -1.2 1.2]);   % Focus around the unit circle

% Bode Plot of Discrete-Time System
figure;
bode(Gz)
title('Bode Plot of Discrete-Time System');
grid on;
```

The Step Response Info using stepinfo() in shown in Table1.1:

Rise Time	2.0260
Settling Time	3.8562
Settling Min	429.4958
Settling Max	472.7630
Overshoot	0
Undershoot	0
Peak	472.7630
Peak Time	9.4631

Table1.1: Cruise Control System Step Response Info

Whereas, the damping characteristics are shown in Table1.2:

Pole	**Damping**	**Frequency (rad/seconds)**	**Time Constant (seconds)**
-8.28e-03	1.00e+00	8.28e-03	1.21e+02
-1.11e+00	1.00e+00	1.11e+00	8.97e-01

Table1.2: Cruise Control System's Damping Characteristics

Fig1.6, Fig1.7, Fig1.8 shows the step response of the continuous-time system, bode-plot of discrete-time system and pole-zero map of discrete-time system respectively.

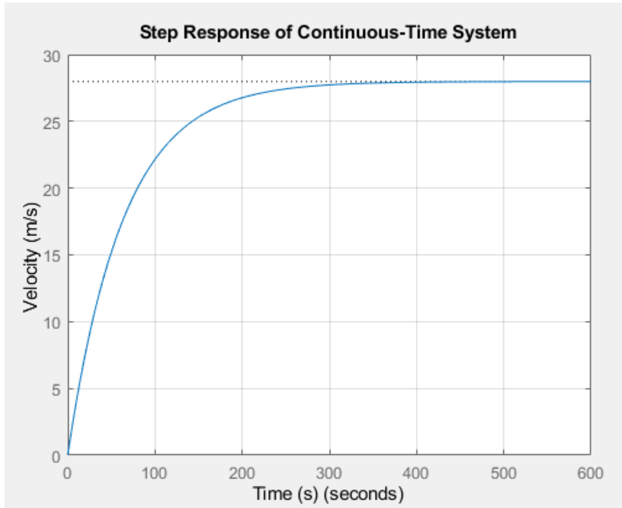


Fig1.6: Step Response of Continuous-Time System

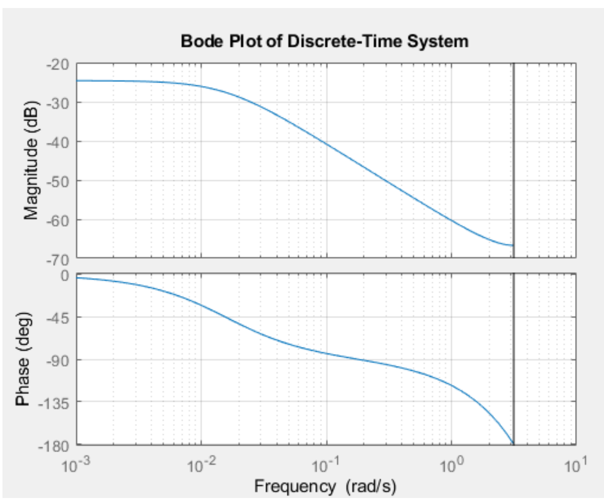


Fig1.7: Bode-Plot DT System

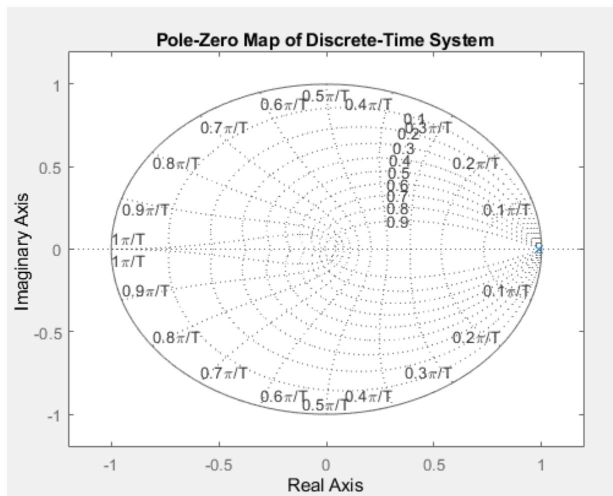


Fig1.8: Pole-Zero Map DT System

Whereas, in the following code. The transfer function in discrete-time domain is analyzed.

```
% ================================
% Discrete-Time System's TF Analysis in Z-Domain
% Bode, Pole-Zero Map, and Root Locus
% ================================

clc;
clear;
close all;

% Define z-domain transfer function
Ts = 1;                              % Sampling time (1 second)
z = tf('z', Ts);                     % Define 'z' as discrete-time variable

% Gz: Digital system transfer function
```

```
numerator = 0.000231*z^2 - 0.000299*z + 0.0000742;
denominator = z^3 - 1.9842*z^2 + 0.9842*z + 0.0000742;

Gz = numerator / denominator;

% Bode Plot (Frequency Response)
figure(1);
bode(Gz)
title('Bode Plot of G(z)');
grid on;

% Pole-Zero Map
figure(2);
pzmap(Gz)
title('Pole-Zero Map of G(z)');
zgrid();  % Show unit circle grid for z-domain
grid on;

% Root Locus
figure(3);
rlocus(Gz)
title('Root Locus of G(z)');
axis([-1 1 -1 1]);  % Focus on unit circle
zgrid();            % Add damping/frequency grid
```

Fig1.9, Fig2.0, Fig2.1 shows the Bode-Plot, Pole-Zero Map and Root Locus of the discrete-time Transfer Function respectively.

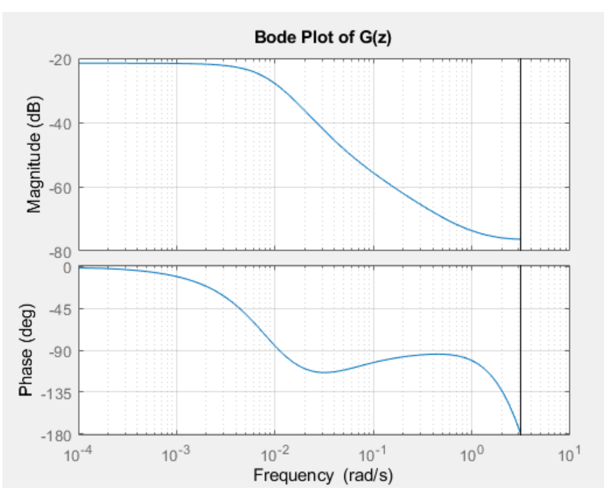


Fig1.9: Bode-Plot of G(z)

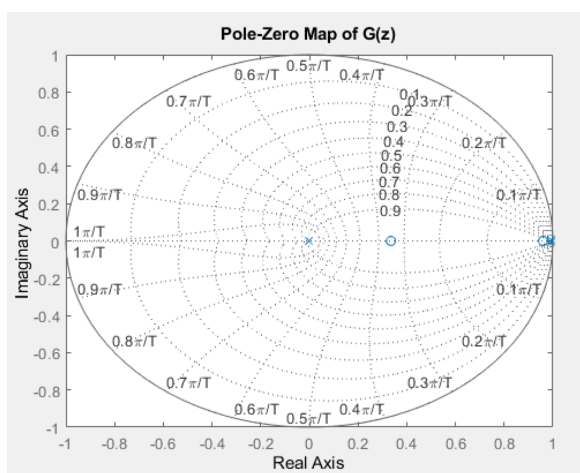


Fig1.9: Pole-Zero Map of G(z)

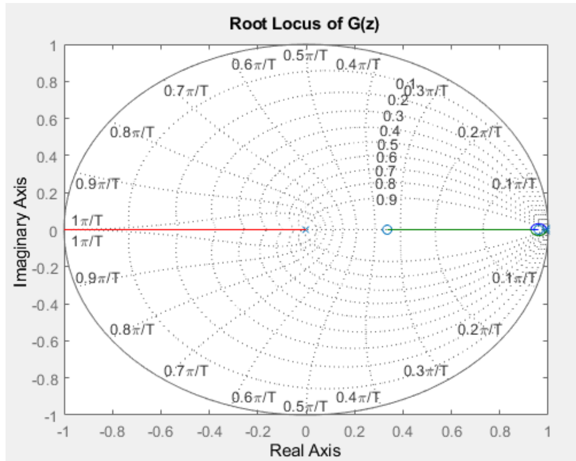


Fig1.9: Root Locus of G(z)