Analysis and Design of Controllers for Automotive Industry for a Better Vehicle Performance

Oscar Lara *olara@students.tamuk.edu*
EECS Department, MSC 192
Texas A&M University-Kingsville
Kingsville, Texas, 78363-8202, U.S.A.

Rajab Challoo *rajab.challoo@tamuk.edu*
EECS Department, MSC 192
Texas A&M University-Kingsville
Kingsville, Texas, 78363-8202, U.S.A.

Abstract

In our world, technology is adapting to the needs that society strives for in advancements, and control systems play an enormous role in improving technology. Different techniques, ranging from classical to intelligent controllers, have been proposed and applied to enhance and optimize general system performance. For example, in the automotive industry, multiple control systems are implemented into the automotive system to create successful and better-performing vehicles. As part of this research, adaptive cruise control in cars as applied by researchers in the automotive industry is investigated, and re-designed systems for better performance are recommended. As part of the research work, a Proportional-Integral-Derivative controller, Fuzzy Logic controller, Model Predictive Controller, Linear Quadratic Tracking, Neural Network, Neuro-Fuzzy, Neuro-Fuzzy-LQT, and LQT-Neural Network controllers are reviewed and discussed regarding the advantages and disadvantages of each technique. These controllers are applied in various cases where the objectives are to keep a desired speed and avoid a collision. The system performances are compared, analyzed, and discussed based on the simulation results. As a result, the Linear Quadratic Tracking combined with a Neuro-Fuzzy controller outperforms the other controllers in the objectives assigned.

Keywords: Control Systems, Cruise Control, Linear Quadratic Tracking, Model Predictive Control, Neural Networks, Fuzzy Logic Control.

1. INTRODUCTION

Control systems have advanced beyond expectations from the past, with controllers ranging from classical to intelligent. Control systems allow us to manage, direct and command behavior that enables a user to achieve a goal. Two types of loops can be utilized within the controllers: open and closed loops. In an open loop, the controller is independent of the process output, while in a closed loop is dependent on the output (Steffano et al., 1967). With the many different applications for control systems, we look towards the automotive industry. This industry had come a long way since 1886 when the first vehicle was manufactured (Wright, 2021). The sole purpose before was transportation; however, so many factors go into the system.

1.1 Cruise Control Systems

Cruise control was created to reduce fatigue in drivers, especially on longer-duration drives. The previous cruise control systems were connected to the accelerator by cable (KIA, 2023). This connection allowed maintaining the gas pedal in a particular position to keep the pre-set speed (KIA, 2023).The system in newer cars electronically manages the speed through a program without a cable. Instead, a computer connected with various sensors and throttle controls operates the feature through a wireless system (KIA, 2023). This newer technology can

automatically adjust the speed based on how fast the vehicle ahead is going while maintaining a safe distance. In addition, it dramatically reduces drivers' fatigue since drivers do not have to press and release the accelerator pedal repeatedly (KIA, 2023). Drivers will find the system particularly useful on highways or roads with traffic jams during rush hour.

1.2 Innovations of Cruise Control

There have been numerous additions to the cruise control systems in vehicles. Adaptive Cruise Control is an advanced version of cruise control that uses radar or other sensors to detect the distance between the car and the vehicle in front of it. ACC can automatically adjust the vehicle's speed to maintain a safe following distance, bringing the car to a complete stop if necessary (Hearst Auto Research, 2023). Stop-and-go cruise control is commonly found in ACC systems, allowing the vehicle to automatically stop and start in heavy traffic. This feature helps to reduce driver fatigue and can improve fuel efficiency by reducing unnecessary acceleration and braking. Finally, predictive cruise control is a feature that uses GPS and map data to anticipate upcoming changes in the road ahead, such as hills, curves, and speed limit changes. This allows the system to adjust the vehicle's speed in advance, reducing the need for sudden braking or acceleration (Chu et al., 2022).

Lane-keeping assist is a feature that uses cameras or other sensors to detect the road's lane markings and can help keep the vehicle centered in its lane. This feature can work with cruise control to provide a more comfortable and stress-free driving experience (MyCarDoesWhat, 2016). Traffic sign recognition is a feature that uses cameras or other sensors to detect and interpret road signs, such as speed limit signs and no-passing signs. The cruise control system can use this information to adjust the vehicle's speed automatically. Finally, connected vehicle technology allows vehicles to communicate with each other and the infrastructure around them, such as traffic lights and road sensors. This technology can improve the accuracy and responsiveness of cruise control systems. It can also help to improve safety by providing real-time information about road conditions and potential hazards.

1.3 Research Objectives

Within the remaining sections, we will learn the details of the selected cruise control system. This will be followed by research about each controller and the implementation of the controller into the cruise control design via MATLAB/Simulink. Finally, cases will be simulated to determine which controller is optimal for the design.

2. CRUISE CONTROL MODEL DESIGN

The vehicle considered in work is a passenger car powered by a 1.6-L gasoline engine, a torque converter (TC), a five-seat automatic transmission, two driving and two drive wheels, and a hydraulic braking system (Li et al., 2016). The control variables are the brake pedal and throttle defluxion, ranging from 0 to 1, being 0 not pressed and one fully pressed (Li et al., 2016). The value for these defluxions is computed by the cruise control system, which uses as information the speed of the ego-car χ, the position of the ego-car χ, both measured using a GPS and inertial sensors, the distance from the ego car to the front car ∆ χ, and the minimal safe distance χ safe the cars should maintain (Li et al., 2016).

Regarding vehicle dynamics, the following assumptions are adopted. First, the engine operates under 2000-6000 RPM (Li S et al., 2016). Second, power train dynamics are simplified to a first-order transfer function with time constant τe (Li et al., 2016). Third, the vehicle runs on a dry road with road-tire friction and no longitudinal slip. Therefore, the vehicle moves only in a longitudinal direction. Finally, the braking system is simplified to a first-order time delay with a time constant τb (Li et al., 2016).

The gear-switching logic of an automated transmission with shift-up and shift-down rules determines when to shift to a higher or lower gear based on certain conditions. For example, the shift-up rule is used when the vehicle speed and engine speed are at appropriate levels to switch

to a higher gear, while the shift-down rule is used when the vehicle speed and engine speed are at proper levels to switch to a lower gear (Li et al., 2016).

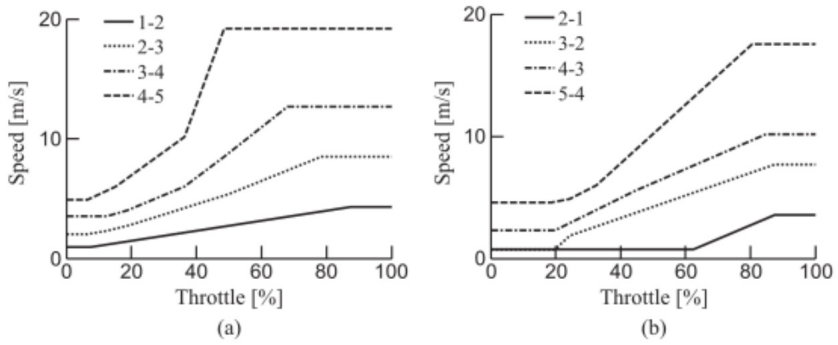

FIGURE 1: Gear Switching Logic of Automated Transmission (a) Shift-Up Rule (b) Shift-Down Rule [7].

The throttle-torque relationship is a fundamental relationship in automotive engineering that describes the relationship between the position of the throttle pedal (input) and the torque produced by the engine (output) (Li et al., 2016). In Figure 2, the throttle-torque relationship can be plotted with the throttle position as the x-axis and the torque output as the y-axis. The output torque starts at zero when the throttle is closed and increases as the throttle position is increased. At a certain point, the torque output reaches a maximum value, which decreases with further increases in throttle position (Li et al., 2016).As a result, the engine has reached its maximum torque capacity and cannot produce more torque (Li et al., 2016).

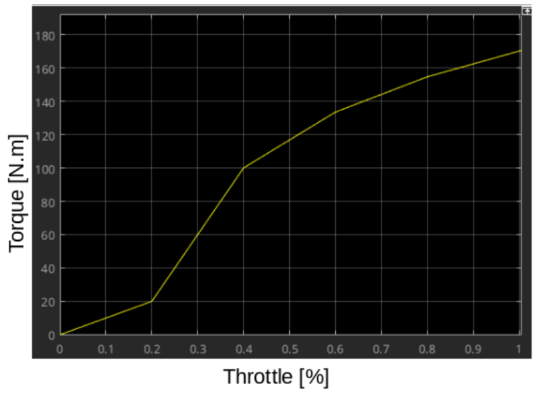

FIGURE 2: Throttle-Torque Relationship [7].

Figure 3 represents the block diagram with the ego car, front car, and controller. The system is responsible for maintaining the constant speed of the vehicle. It should communicate with the transmission system to ensure that the car stays in the appropriate gear and with the engine dynamics to ensure the engine operates at the optimal speed (Abdulnabi, 2017). The front car refers to the vehicle in front of the ego car. For example, if the front vehicle slows down or stops, the system should signal the ego-car's brake system also to slow down or stop. Engine dynamics refers to the operation of the engine. The system should monitor the engine speed and output torque to ensure they remain within safe and efficient limits. The system should also signal the transmission system to shift gears to maintain optimal engine performance. The transmission system transmits power from the engine to the wheels (Abdulnabi, 2017).

The system should work closely with the engine dynamics to ensure the vehicle stays in the appropriate gear. The system should also communicate with gear-switching logic to ensure that gear changes occur smoothly and efficiently. The brake system is the reason for slowing down or

stopping the vehicle. The system should work closely with the cruise control and front car monitoring systems to ensure that the vehicle maintains a safe distance from the front car and slows down or stops as necessary. The gear-switching logic determines when to shift gears based on various factors, such as vehicle and engine speed. The system should work closely with the transmission system to ensure that gear changes occur smoothly and efficiently. The ego-car body dynamics refer to the vehicle's behavior, such as acceleration, deceleration, and turning. The system should monitor these dynamics to ensure they remain within safe and efficient limits (Abdulnabi, 2017).

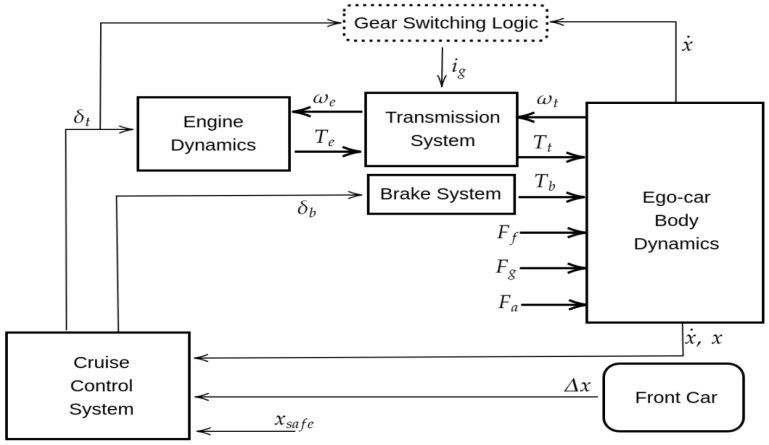

FIGURE 3: System Block Diagram.

Overall, the interactions between these components should be coordinated and optimized to create a safe and efficient driving experience for the driver. The following figure shows how the system was modeled in Simulink.

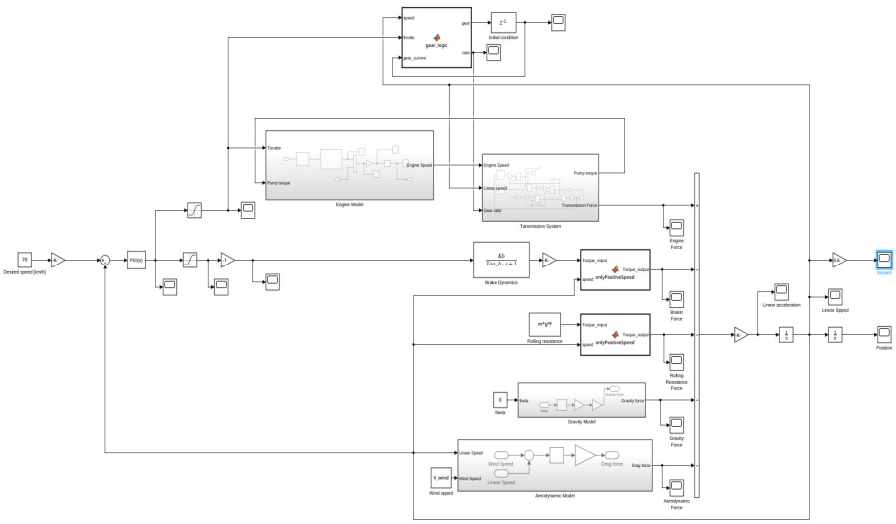

FIGURE 4: System Block Diagram Modeled in Simulink.

A transmission system, gravity model, and aerodynamic model were added to the design to model a real-life car scenario. A transmission system in an automotive system is responsible for transmitting power from the engine to the vehicle's wheels. The Gravity Model represents the effect of gravity and can also be applied to an automotive system to estimate the flow of vehicles between two locations based on various factors such as distance, population, and road network.

Finally, the aerodynamic model in MATLAB for an automotive system is a mathematical model that simulates the flow of air around the vehicle and calculates the forces and moments acting on the vehicle due to this flow.

This main code defines and executes a simulation of a cruise control system using a particle swarm optimization (PSO) algorithm to optimize the controller gains (Abdulnabi, 2017). The first three lines clear the workspace, close all figures, and reset the random number generator to its default settings. The next section of the code sets up the options for the PSO algorithm using the optim options function. It specifies that the swarm size should be 50 and that the hybrid function (i.e., the optimization function used after the PSO algorithm) should be the fmincon function. The following lines define the lower and upper bounds of the four controller gains KI, KP, KD, and KPB (Abdulnabi, 2017). The nvars variable is set to 4, which tells the PSO algorithm that there are four variables to optimize. The following line sets the initial values of the controller gains K. It is commented out that the PSO algorithm is not executed, and the code uses these initial values to run a simulation. The myCruise function is then defined, which represents the simulation of the cruise control system (Abdulnabi, 2017).

The function takes the controller gains K as inputs and returns the cost of the system. The cost is the final value of the out.cost_function. The data variable is obtained from simulating the cruise control system using the sim function (Abdulnabi, 2017). The simulation time is set to 200 seconds. The rest of the code defines the parameters and initial conditions of the cruise control system, such as the vehicle properties, wheels properties, driveline, brakes, and initial conditions of the plan (Abdulnabi, 2017).Finally, the controller gains KI, KP, KD, and KPB are extracted from K and used to calculate the control input for the cruise control system [8]m. Overall, this code sets up a simulation of a cruise control system and defines the parameters and initial conditions, and later optimizes the controller gains and improves the performance of the cruise control system (Abdulnabi, 2017).

```
% Setting up the parameters for the PSO algorithm
options = optimoptions('particleswarm','SwarmSize',50,'HybridFcn',@fmincon);

%     KI      KP   KD     KDB  KPB
lb = [0.1 0.1 0.1 0.1];
ub = [2 5 2 5];
nvars = 4;
myCruise()
function cost = myCruise()
    %Simumation time
    Simulation_Time = 500;
    model_name = 'Cruise_control_NN.slx'
    % Enviroment
    g = 9.81;       %gravity acceleration [m/s²]
    theta = 0;      %Angle of inclination of the road [rad]
    V_wind = 0;     %Wind speed [m/s]
    rho = 1.225;    %Air density [kg/m³]
    f = 0.01;       %Rolling resistance coeficient [adm]
    %Vehicle properties
    m = 1573;       %Vehicle mass [kg]
    Af = 1.6 + 0.00056*(m - 765);   %Frontal area of  the vehicle [m²]
    Cd = 0.324;     %Aerodynamic drag coefficient [adm]
    %Wheels properties
    Je = 0.21;      %Inertia of fly wheel [kg.m²]
    rw = 0.28;      %Rolling radius of wheels [m]
    %Driveline
    Nu = 0.89;      %Mechanical efficiency of driveline [adm]
    Tau_e = 0.3;    %Time constant of engine dynamics [s]
    io = 1;      %Ratio of final gear [adm]
    ig = 0.74;         %Gear ratio of transmission [adm]
    Tau_p = 1;
    %Brakes
    Tau_b = 0.15;   %Time constant of the braking system [s]
    Kb = 1185;      %Braking gain of four wheels[N.m/MPa]
    %Inital condition
    initial_gear = 1;
    x = 0;
    xd = 80/3.6;
    simopt = simset('solver','ode45','SrcWorkspace','Current','DstWorkspace','Current');  % Initialize sim options
    out = sim(model_name, [0 Simulation_Time], simopt);
    cost = out.cost_function.Data(end);
    disp(cost)
end
```

FIGURE 5: MATLAB Main Code for Simulink Design (Abdulnabi, 2017).

3. CONTROLLER RESEARCH AND IMPLEMENTATION

3.1 PID Controller

The Proportional Integral Derivative controller is one of the most common controllers utilized in control systems. The P Controller stabilizes the gain but produces a constant steady-state error. The I Controller reduces or eliminates the steady-state error. Finally, the D Controller minimizes the rate of change of error (Vandoren, 2016).

The PID formula weights the proportional term by a factor of P, the integral term by a factor of P/TI, and the derivative term by a factor of P.TD where P is the controller gain, TI is the integral time, and TD is the derivative time (Vandoren, 2016). That terminology bears some explanation. "Gain" refers to the percentage by which the error signal will gain or lose strength as it passes through the controller en route to becoming part of its output. A PID controller with a high gain will tend to generate particularly aggressive corrective efforts (Vandoren, 2016).

A controller with a long integral time is more heavily weighted toward proportional action than integral action. A PID controller with a long derivative time is more heavily weighted toward derivative than proportional action (Vandoren, 2016). The following figure shows how the PID controller is implemented into the Simulink design.

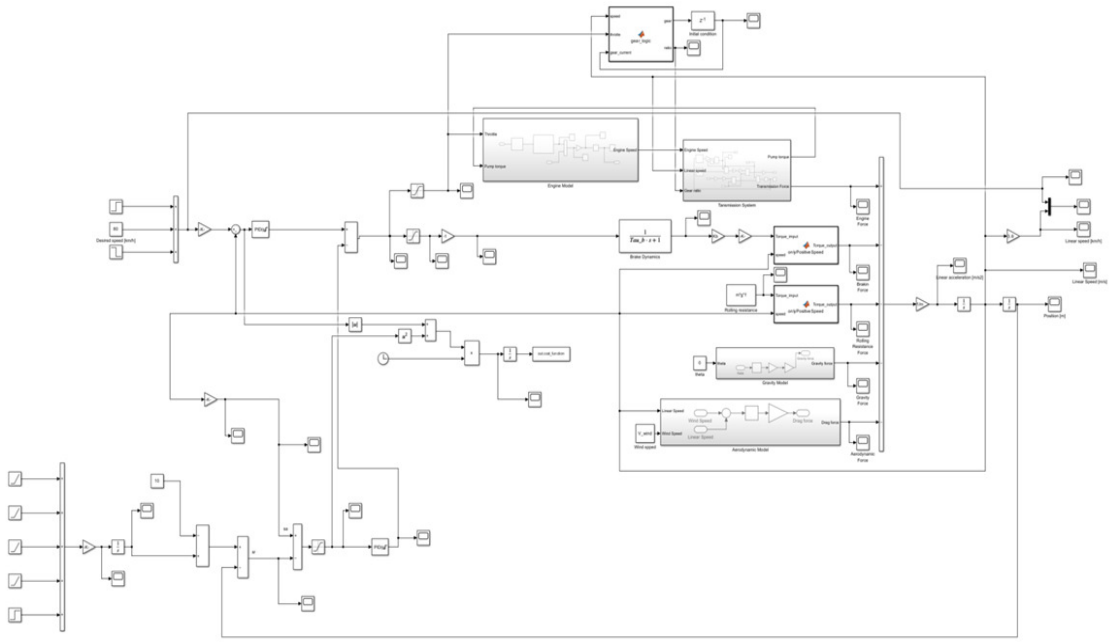

FIGURE 6: PID Controller Simulink Design.

The PID controller in a cruise control system is modeled with several components. First, a PID block takes the error between the desired and actual speeds and calculates the corresponding P, I, and D components. The outputs of these components are then summed together to produce the control signal that adjusts the throttle or fuel injection to maintain the desired speed.

The output of the PID controller is then passed through a saturation block that limits the range of the control signal to prevent overshooting or undershooting the desired speed. The saturation block ensures the control signal stays within a safe and stable range (Vandoren, 2016). Next, the control signal is passed to a switch that determines whether to apply the powertrain or the brake. If the control signal is positive, the powertrain accelerates the vehicle. If the control signal is negative, the brake slows down the car. The switch prevents both the powertrain and the brake from being applied simultaneously.

The powertrain block models the dynamics of the engine and drivetrain and translates the control signal into a torque that drives the wheels. The brake block models the dynamics of the braking system and decodes the control signal into a braking force that slows down the vehicle. Finally, the vehicle dynamics block models the vehicle's motion, considering factors such as air resistance, rolling resistance, and gravitational force. The vehicle dynamics block calculates the speed and feeds it back to the PID controller, completing the feedback loop.

3.2 Fuzzy Logic Controller

Fuzzy logic is an approach to computing based on "degrees of truth" rather than the usual "true or false" (1 or 0) Boolean logic on which the modern computer is based. It is employed to handle the concept of partial truth, where the truth value may range between entirely true and false. The architecture of an FLC generally consists of three main components: the fuzzifier, the inference engine, and the defuzzifier (Chen et al., 2019).

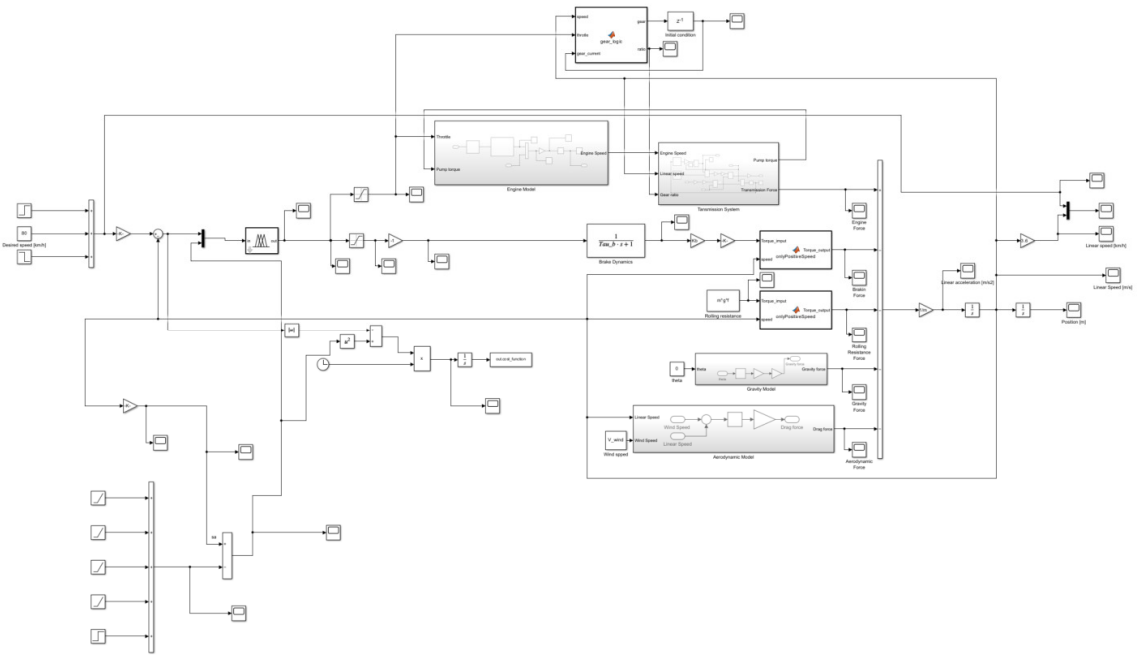

FIGURE 7: Fuzzy Logic Controller Simulink Design.

The FLC controller is designed to take in the inputs such as the speed of the ego car, the distance to the front car, and the relative speed between the two vehicles and generate the optimal control signals that minimize a cost function, which penalizes deviations from the desired speed and acceleration of the car and collisions with the front vehicle (Chen et al., 2019).

The inputs of the FLC controller are first fuzzified into fuzzy sets using membership functions. The fuzzy sets are then used to determine the degree of membership of each input to each fuzzy set, which is then used to determine the degree of activation of each rule in the rule base (Chen et al., 2019).The rule base contains a set of fuzzy if-then rules that relate the inputs to the output control signals. Each rule has a corresponding output membership function that defines the degree of membership of the output to a particular fuzzy set.

The output of the FLC controller is then defuzzified to obtain crisp control signals that adjust the speed and acceleration of the ego car. The defuzzification process involves calculating the weighted average of the output membership functions for each fuzzy set, where the weights are the degree of activation of each rule (Chen et al., 2019). To avoid collisions with the front car, the FLC controller constantly monitors the distance between the front vehicle and the ego car and adjusts the control signals accordingly to maintain a safe distance. The controller also maintains

cruise control by keeping the car's speed and acceleration close to the desired values (Chen et al., 2019).

The two saturations limit the control signals to prevent the car from exceeding safe or comfortable limits. The first saturation determines the control signal that adjusts the car's speed, and the second saturation limits the control signal that changes the car's acceleration. The cost function is defined as a function of the error between the desired and actual values of speed and acceleration. Also, it incorporates a penalty term for collisions with the front car (Chen et al., 2019). Overall, the FLC controller allows the cruise control system to maintain a safe and comfortable following distance while avoiding collisions with the front car.

3.3 Model Predictive Controller

Model predictive control (MPC) is an optimal control technique in which the calculated control actions minimize a cost function for a constrained dynamical system over a finite, receding horizon (Saleh & Farmanbordar, n.d). It can be used to design advanced control strategies for complex dynamic systems. The control loop is based on a system model that predicts the system's future behavior based on the current state and the applied control action. The expected behavior is then optimized to achieve the desired control objectives, such as tracking a setpoint or minimizing a cost function, subject to various constraints on the system variables (VisibleBreadcrumbs, 2023). The following figure is the design created in Simulink.

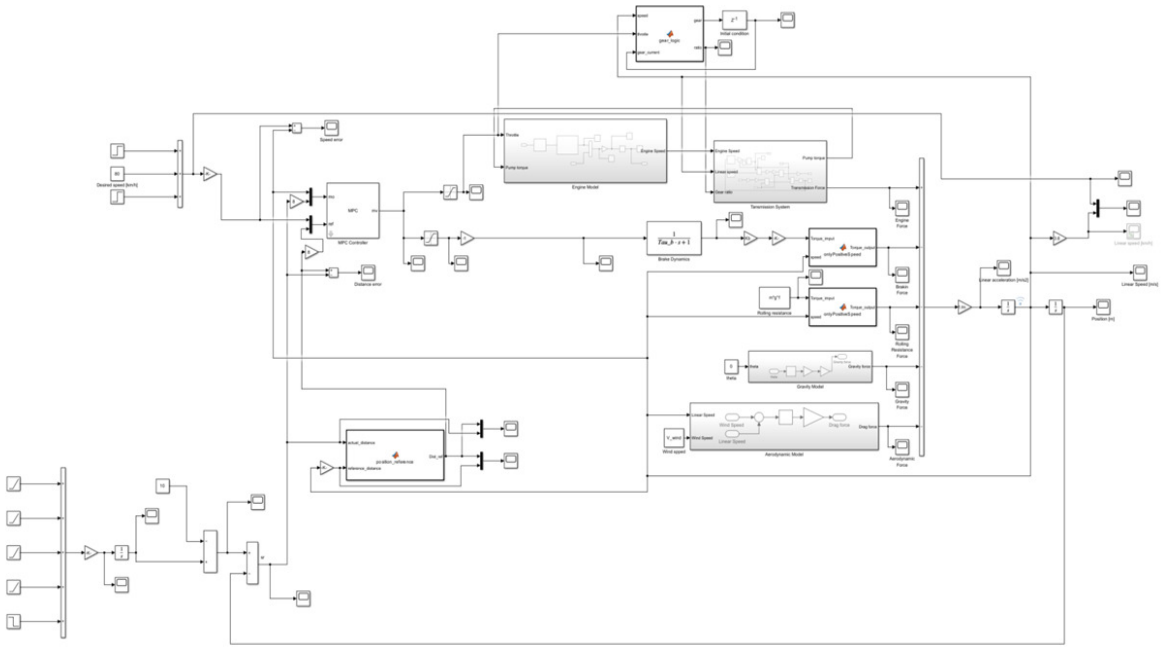

FIGURE 8: Model Predictive Controller Simulink Design.

The MPC controller is designed to predict the future behavior of the car and the front vehicle and generate optimal control signals that minimize a cost function, penalizing deviations from the vehicle's desired speed and acceleration and collisions with the front car (VisibleBreadcrumbs, 2023).

The front car's speed and position are measured by sensors and used as input to the MPC controller, along with the speed and acceleration of the ego car. The MPC controller uses a predictive model to estimate the future behavior of the vehicle and the front car based on the input variables and then generates the optimal control signals that minimize the cost function.

The MPC controller also incorporates two saturations that limit the control signals to prevent the car from exceeding safe or comfortable limits (VisibleBreadcrumbs, 2023). The MPC controller consists of two main blocks: the MPC block and the optimizer block. First, the MPC block uses the predictive model to estimate the future behavior of the car and the front vehicle. Then, it generates the predicted states and predicted outputs for the next step. The optimizer block then uses the expected conditions and results to generate the optimal control signals that minimize the cost function (Saleh & Farmanbordar, n.d).

To avoid collisions with the front car, the MPC controller monitors the distance between the front vehicle and the ego car and adjusts the control signals accordingly to maintain a safe distance. The controller also maintains cruise control by keeping the car's speed and acceleration close to the desired values.

3.4 Linear Quadratic Tracking Controller

The Linear Quadratic Tracking or Linear Quadratic Regulator is an optimal control theory, a branch of mathematical optimization that deals with finding a control for a dynamic system over a period such that an objective function is optimized (Saleh & Farmanbordar, n.d). Optimal control is an extension of the calculus of variations and is a mathematical optimization method for deriving control policies.

The LQT algorithm reduces the work done by the control systems engineer to optimize the controller. However, the engineer must still specify the cost function parameters and compare the results with the design goals (Saleh & Farmanbordar, n.d). Often this means that controller construction will be an iterative process in which the engineer judges the "optimal" controllers produced through simulation and then adjusts the parameters to create a controller more consistent with design goals.

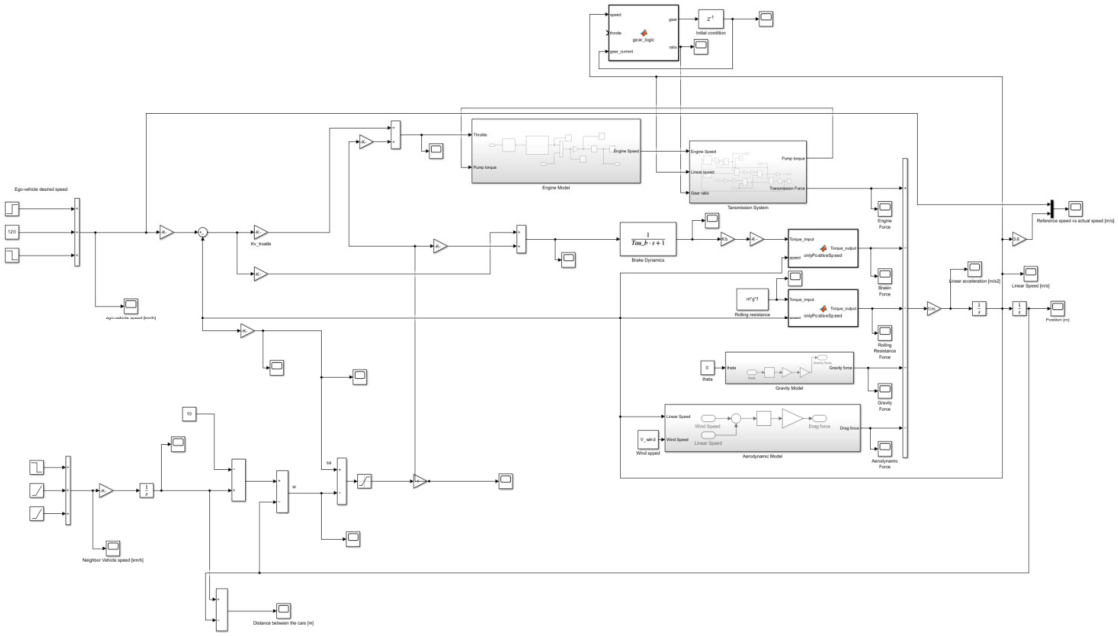

FIGURE 9: Linear Quadratic Tracking Controller Simulink Design.

A Linear Quadratic Tracking (LQT) controller can be used in a cruise control system with a front car and an ego car to maintain a safe and comfortable following distance while avoiding collisions with the front vehicle (Saleh & Farmanbordar, n.d). The LQT controller is designed to minimize a cost function that penalizes deviations from the desired speed and acceleration of the car while also penalizing collisions with the front car.

The front car's speed and position are measured by sensors and used as input to the LQT controller, along with the speed and acceleration of the ego car (Saleh & Farmanbordar, n.d). The LQT controller uses the input variables to generate the optimal control signals for the vehicle that minimizes the cost function.

The LQT controller is designed to maintain cruise control by keeping the car's speed and acceleration as close as possible to the desired values while maintaining a safe following distance from the front car. Using a linear quadratic optimization algorithm, the controller calculates the optimal control signals based on the input variables and the desired objectives (Saleh & Farmanbordar, n.d).

To avoid collisions with the front car, the LQT controller also incorporates a collision avoidance mechanism that adjusts the control signals to ensure the vehicle maintains a safe following distance. The controller constantly monitors the distance between the front and ego cars and adjusts the control signals accordingly to maintain a safe distance.

3.5 Neural Network Controller

A neural network (also called an artificial neural network) is an adaptive system that learns using interconnected nodes or neurons in a layered structure resembling a human brain (Al-Aubidy, 2023). It can learn from data and be trained to recognize patterns, classify data, and forecast future events. A neural network breaks down the input into layers of abstraction. It can be trained using many examples to recognize patterns in speech or images, just as the human brain does. Its behavior is defined by how its elements are connected and the strength, or weights, of those connections (Al-Aubidy, 2023). These weights are automatically adjusted during training to a specific learning rule until the artificial neural network performs the desired task correctly.

A primary neural network has interconnected artificial neurons in three layers: The input Layer, which has information from the outside world, enters the artificial neural network from the input layer. Input nodes process the data, analyze or categorize it, and pass it on to the next layer. The hidden layers take their input from the input layer or other hidden layers (Rawat, 2022). Artificial neural networks can have many hidden layers. Each hidden layer analyzes the output from the previous layer, processes it further, and passes it on to the next layer.

The output layer gives the result of all the data processing by the artificial neural network. It can have single or multiple nodes (Rawat, 2022). For instance, if we have a binary (yes/no) classification problem, the output layer will have one output node, giving the result 1 or 0. However, if we have a multi-class classification problem, the output layer might have more than one output node.

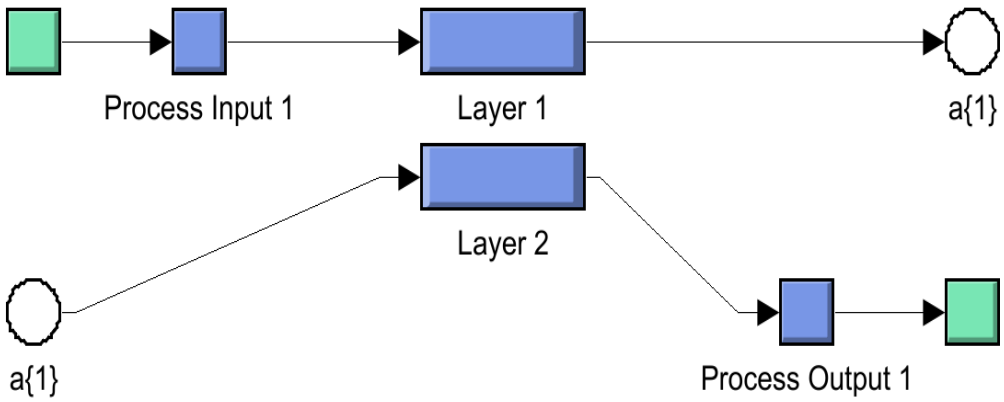

FIGURE 10: Neural Network Controller Design.

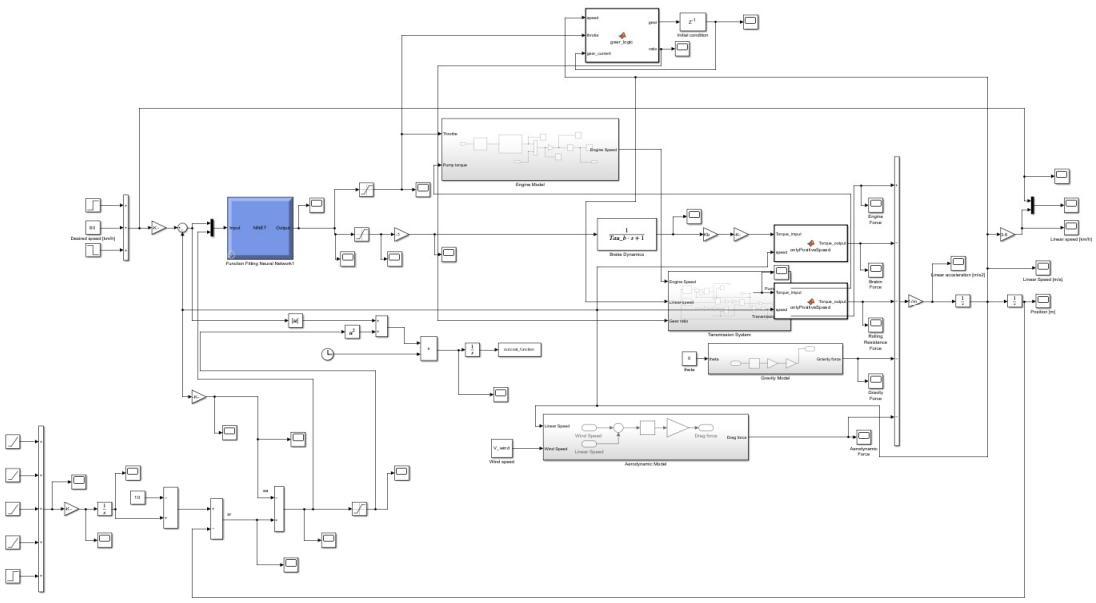

FIGURE 11: Neural Network Controller Implementation Simulink Design.

The front car's speed and position are measured by sensors and used as input to the neural network, along with the speed and acceleration of the ego car. Then, the neural network is trained to map the input variables to the optimal control signals for the vehicle using a supervised learning algorithm, such as back propagation. The neural network has two layers: the input layer, which takes in the input variables, and the output layer, which generates the control signal for the car. The input layer has 35 neurons, one for each weight, and the output layer has one neuron, which produces the control signal.

The neural network is designed with saturation, which limits the network output to a maximum value, preventing the control signal from exceeding safe or comfortable limits (Al-Aubidy, 2023). Once the neural network is trained, it can generate control signals in realtime. The input variables are fed into the neural network's input layer, and the output layer generates the corresponding control signal for the car (Rawat, 2022). The following is the code that allowed the controller to be trained.

```
load('Input_training_data.csv','inputtrain')
load('Input_training_data.csv','outputtrain1')
x = inputtrain';
t = outputtrain1';


trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
hiddenLayerSize = 20;
net = fitnet(hiddenLayerSize,trainFcn);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train the Network
[net,tr] = train(net,x,t);

% Test the Network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)

% View the Network
view(net)

% Plots
%
figure, plotperform(tr)
figure, plottrainstate(tr)
figure, ploterrhist(e)
figure, plotregression(t,y)
figure, plotfit(net,x,t)
```

FIGURE 12: Neural Network MATLAB Training Code

3.6 Neuro-Fuzzy Controller

A Neuro-Fuzzy Controller is an artificial intelligence system that combines fuzzy logic and artificial neural networks to provide a more accurate and adaptable control system.

Fuzzy logic is a computing method based on degrees of truth instead of binary logic (true/false). It is used to deal with uncertainty and imprecision in decision-making processes. Fuzzy logic works by assigning degrees of membership to a particular input value based on how well it fits into a specific category. On the other hand, artificial neural networks (ANNs) are machine learning algorithms inspired by the biological neural networks that make up the human brain. ANNs are trained to recognize patterns in data by adjusting the strength of connections between neurons in the network (Rawat, 2022).

A Neuro-Fuzzy Controller combines these two approaches using fuzzy logic to classify input values and ANNs to learn from data and make decisions based on those classifications. The system can learn from past experiences and adjust its behavior accordingly.

The Neuro-Fuzzy Controller is helpful in situations with a lot of uncertainty or where traditional control systems are ineffective. It is used in robotics, automation, and process control applications.

A Neuro-Fuzzy controller can be used in a cruise control system with a front and ego car to maintain a safe and comfortable following distance (Al-Aubidy, 2023). The system can be designed with a Fuzzy Logic Controller (FLC) and a neural network, which controls the car's speed. The front car's speed and position are measured by sensors and used as input to the FLC, along with the speed and acceleration of the ego car (Chen et al., 2019). The FLC uses fuzzy logic to determine the optimal control signals for the vehicle based on the input variables.

The output of the FLC is then passed to the neural network, which considers the state of the car, the desired speed, and the control signals generated by the FLC to determine the optimal control signals for the vehicle (Chen et al., 2019). The neural network also incorporates a cost function penalizes deviations from the desired speed and accelerations (VisibleBreadcrumbs, 2023). This allows the LQT controller to optimize the control signals to minimize the cost function and maintain a safe following distance (Saleh & Farmanbordar, n.d). The figure below is the controller implemented in Simulink.

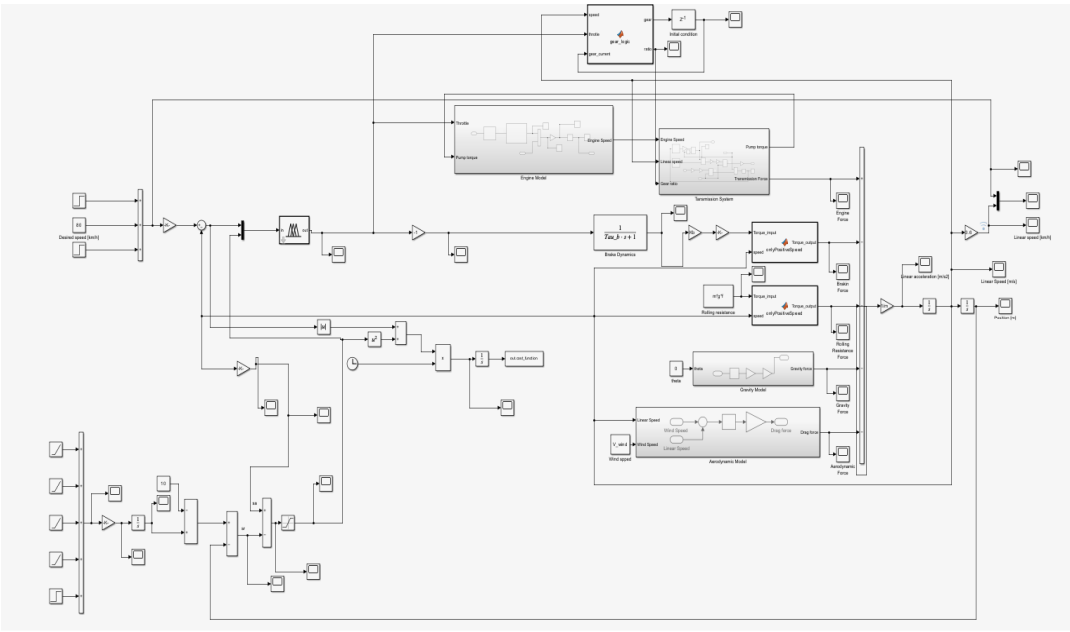

FIGURE 13: Neuro-Fuzzy Controller Simulink Design.

The following figure is the ANFIS structure. The input component of an ANFIS structure is the data fed into the system. The inputmf feature defines each input variable's membership functions (MFs). Membership functions describe the degree to which a given input value belongs to a particular category or set. The rule component of an ANFIS structure defines the rules that govern how the input data is processed.

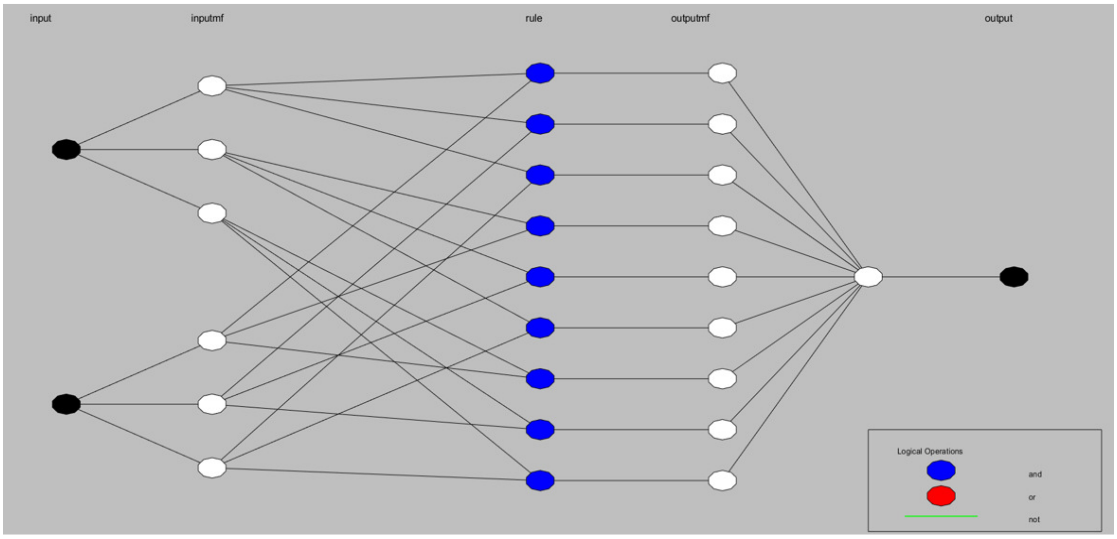

FIGURE 14: ANFIS Structure for Neuro-Fuzzy Controller.

These rules are usually represented as "if-then" statements, which specify how the input variables relate to the output variable. Finally, the outputmf component defines the membership functions for the output variable. These membership functions describe the degree to which the output value belongs to a particular category or set (Chen et al., 2019).

The output component is the final stage of the ANFIS structure, where the input data is processed according to the rules defined in the rule component, and an absolute output value is generated. This output value is then mapped to the membership functions described in the outputmf detail, which produces a crisp output value.

3.7 Neuro-Fuzzy Linear Quadratic Tracking Controller

A Neuro-Fuzzy Linear Quadratic Tracking controller (NFLQT) is a type of control system that combines the benefits of fuzzy logic and neural networks with linear quadratic control theory to track and control the behavior of a system. The NF-LQT controller is designed to adjust the system's input signals to achieve a desired output response. A feedback control system uses measurements of the system's current state to compute control signals that steer the system toward a desired state.

The controller consists of three main components. The FLC component is responsible for converting the system's input signals into fuzzy sets, which are then used to create rules for the system's behavior. The fuzzy logic system handles the uncertainty and imprecision inherent in many control applications. The neural network component learns the relationship between the system's input signals and the desired output response. The neural network is trained using historical data to optimize the controller's performance. This LQT component uses a mathematical optimization technique called linear quadratic control to compute the optimal control signals based on the system's current state and the desired output response. Finally, the linear quadratic regulator component ensures the system's response is stable and robust.

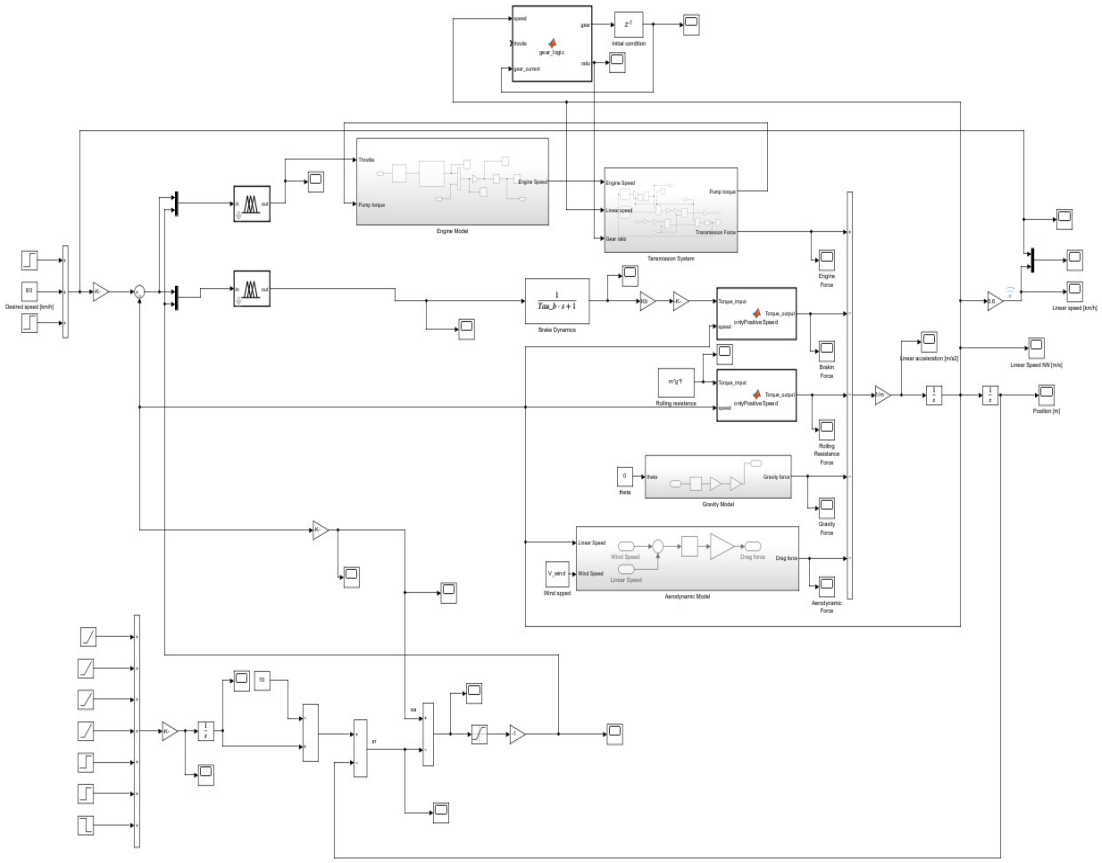

FIGURE 15: Neuro-Fuzzy Linear Quadratic Tracking Controller Design.

A Neuro-Fuzzy Linear Quadratic Tracking (LQT) controller can be used in a cruise control system with a front and ego car to maintain a safe and comfortable following distance. The system can be designed with two Fuzzy Logic Controllers (FLCs), one for the throttle and one for the torque, which work together to control the car's speed. The front car's speed and position are measured by sensors and used as input to the FLCs, along with the speed and acceleration of the ego car. The FLCs use fuzzy logic to determine the car's optimal torque and throttle settings based on the input variables. The output of the FLCs is then passed to the Neuro-Fuzzy LQT controller, which uses a neural network to learn the optimal control policy for the car. The LQT controller considers the state of the car, the desired speed, and the control signals generated by the FLCs to determine the optimal control signals for the car (Saleh & Farmanbordar, n.d).

The LQT controller also incorporates a cost function that penalizes deviations from the desired speed and acceleration. This allows the controller to optimize the control signals to minimize the cost function and maintain a safe following distance (Saleh & Farmanbordar, n.d). Overall, the Neuro-Fuzzy LQT controller allows the cruise control system to adapt to changes in the environment, such as the presence of the front car, and maintain a safe and comfortable following distance while also optimizing the control signals to minimize the cost function.

3.8 Linear Quadratic Tracking Neural Network Controller

The LQT-NN controller combines the LQT and neural network controller to take advantage of the strengths of both techniques. For example, the LQT provides an excellent initial control law that can be improved upon by the neural network controller. In addition, the neural network controller compensates for the uncertainties and nonlinearities of the system.

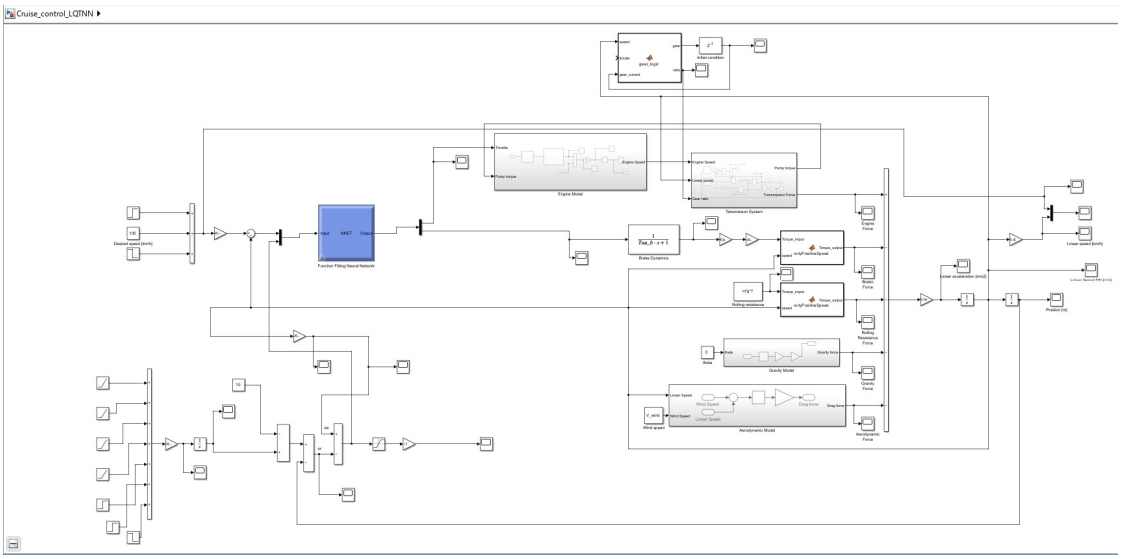

FIGURE 16: Neural Network Linear Quadratic Tracking Controller Design.

A Neural Network Linear Quadratic Tracking (LQT) controller can be used in a cruise control system with a front and ego car to maintain a safe and comfortable following distance. The system can be designed with a function-fitting neural network and a linear quadratic tracker, which work together to control the car's speed.

The front car's speed and position are measured by sensors and used as input to the neural network, along with the speed and acceleration of the ego car. Then, the neural network learns to map the input variables to the optimal control signals for the car using a supervised learning algorithm, such as back propagation.

The output of the neural network is then passed to the linear quadratic tracker, which considers the state of the car, the desired speed, and the control signals generated by the neural network to determine the optimal control signals for the car (VisibleBreadcrumbs, 2023).

The linear quadratic tracker also incorporates a cost function that penalizes deviations from the desired speed and acceleration. This allows the controller to optimize the control signals to minimize the cost function and maintain a safe following distance (Saleh & Farmanbordar, n.d).Overall, the Neural Network LQT controller allows the cruise control system to adapt to changes in the environment, such as the presence of the front car, and maintain a safe and comfortable following distance while optimizing the control signals to minimize the cost function.

4. SIMULATION CASES OF CONTROLLER IMPLEMENTATION

4.1 Case 1: Cruise Control

In the first case, to test the controllers, it was necessary to see if they could maintain a speed which is the purpose of cruise control. So the path to be set was for the ego car to travel at 120 km/hr to begin. Then, at 70 seconds, the car would increase its speed to 130 km/hr.

This would prove the system functions appropriately to accelerate and increase speed. Next, we would test whether the system could decelerate and decrease speed. For example, at 125 seconds, the car would reduce to 70 km/hr, simulating hitting the highway and keeping a constant speed.

When we see the outputs for all controllers listed in the figures, each controller could maintain a set speed. They all traveled from 120 km/hr to 13 km/hr to 70 km/hr at the time specified for the case. The control design for each controller achieved one of the system's goals to ensure that it could keep a constant speed as cruise control systems do.

The simulations proved that the controllers could all perform for a cruise control system as anticipated. In the following cases, we will witness how more complex scenarios have been created to see each controller's performance. Based on case 1, the controllers are of equal performance, as shown in the figure below.

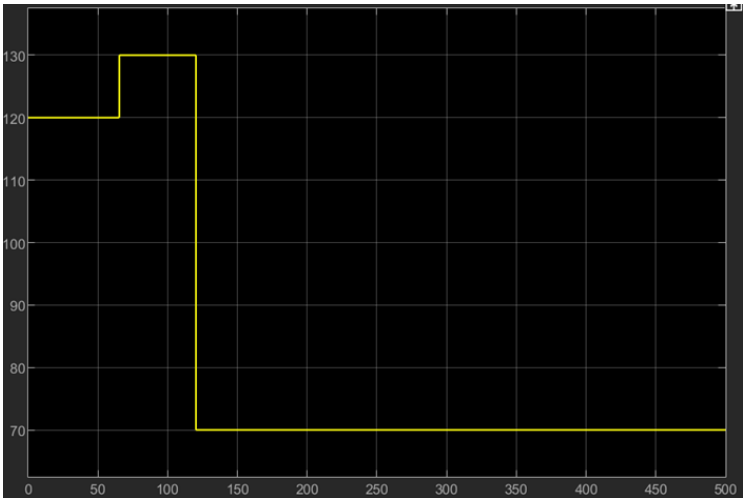

FIGURE 17: Case 1 Simulation Results of Controllers.

4.2 Case 2: Collision Avoidance

In the second case, we will observe the ability to avoid a collision with another vehicle. As cruise control systems are becoming more adaptive today, this seemed like a clever idea to test.

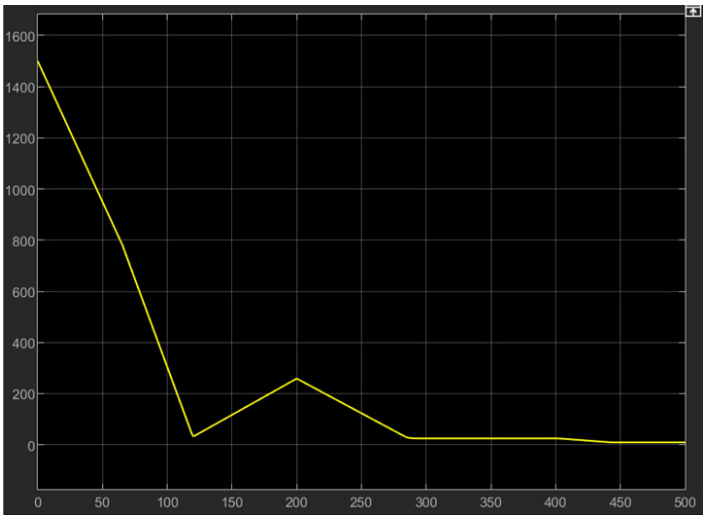

FIGURE 18: Case 2 Simulation Results of Controllers.

The following above output responses show the distance between the vehicles by showing the ego car distance from the front car. Here you will notice that it starts at a high distance and approaches the front vehicle because it has a higher speed. Although you then see the distance increase as the front car speeds up. However, the distance decreases as the front vehicle slows down again to where the ego car follows at the same speed because it is under its target speed. The distance is never negative or below 0, which means they never collided with any of the controllers, proving they successfully achieved the second objective.

When we see the outputs for all controllers, each controller could avoid a car crash or collision. Each controller's control design achieved one of the system's goals to ensure it prevented a collision. Once again, this proved that all controllers could perform in the system as anticipated. We determined that no crash occurred because none of the simulations went below zero into the negatives in all the output responses. If the distance between the cars became negative, this would indicate that a collision had happened. The controllers can get quite close to the front car but do not collide.

4.3 Case 3: Collision Avoidance & Maintaining Cruise Control

During this next case, the controller must respond accordingly if the front car changes speed and could cause a collision while maintaining cruise control. The goal figure shows the simulation for the front car at which it will be traveling. The yellow line in the rest of the figures represents the cruise control at which the ego car is set. It will begin at 80 km/hr, then increase to 105 km/hr at 65 seconds, and once again grow to 225 km/hr at 250 seconds, remaining for the rest of the simulation.

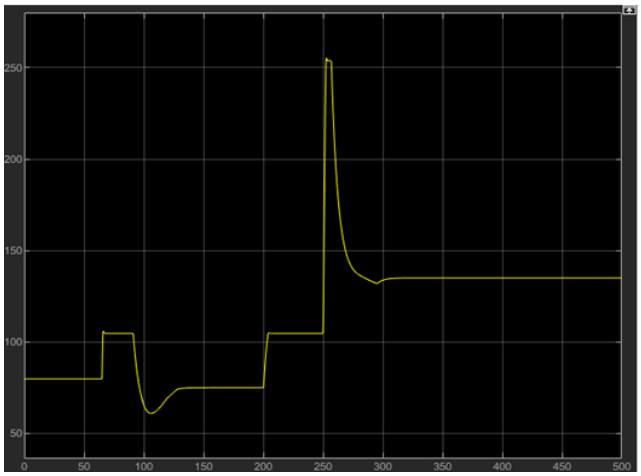

FIGURE 19: Front Car Path.

The PID gave an appropriate transient response to the front car's desired output. However, it begins to get off track when the response severely undershoots the change of speed of the front car at 250 seconds. Finally, the system attempts to settle at the desired cruise control. The controller prioritized the cruise control over doing both objectives as precisely as possible.

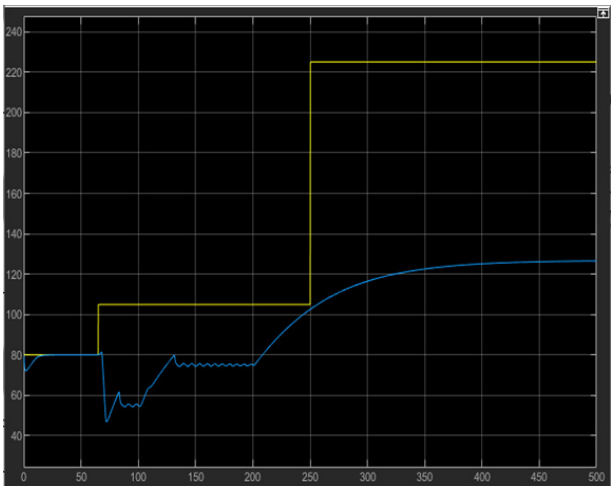

FIGURE20: Case 3 PID Controller Output Response.

The FLC and MPC performed similarly. The FLC seems to have performed a little more efficiently as the MPC overshoots in the beginning. However, within these two models, the controllers seem to prioritize following the set cruise control more. Near the end of the simulation, you can see they settle with the ending response of the front car, where avoiding collision begins to take priority.

The LQT controller outperformed the previous three controllers, although it again began with an overshoot. Still, it then performs the objective of following cruise control that is set perfectly. However, this meant it was disregarding the avoidance of the collision with the path of the front car. Therefore, in the 2nd half of the simulation, it takes a bit longer for the system to settle with the front car's desired response, where it then changes prioritization in the objectives.

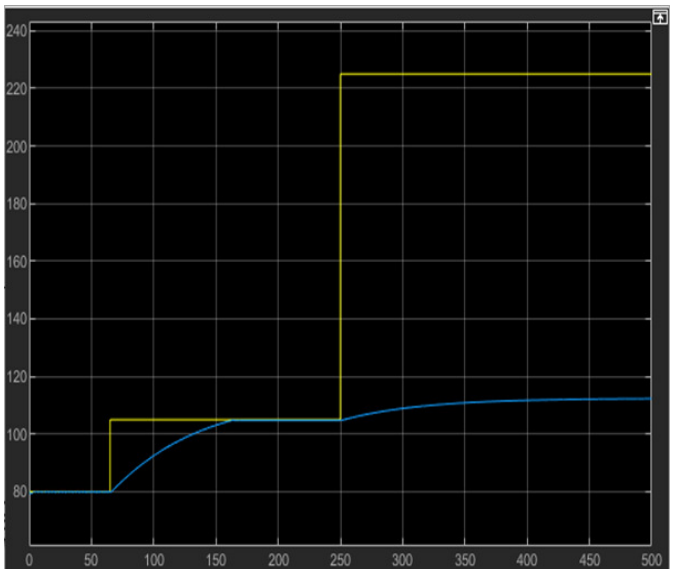

FIGURE 21: Case 3 Fuzzy Logic Controller Output.

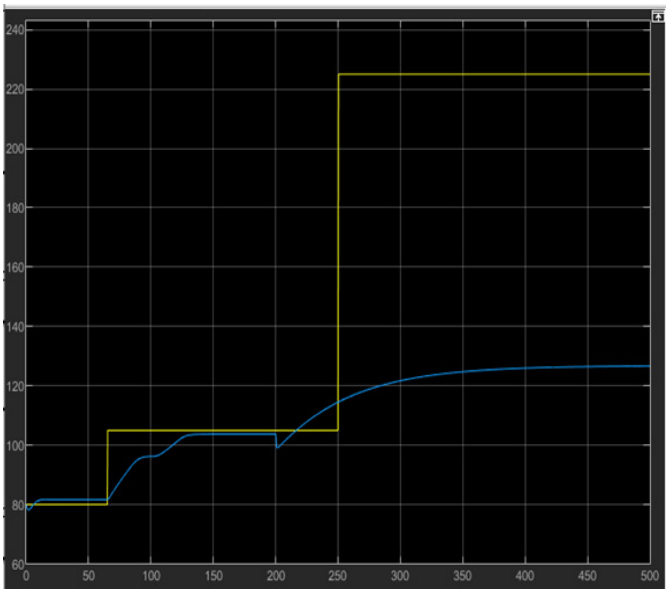

FIGURE 22: Case 3 Model Predictive Controller Output.

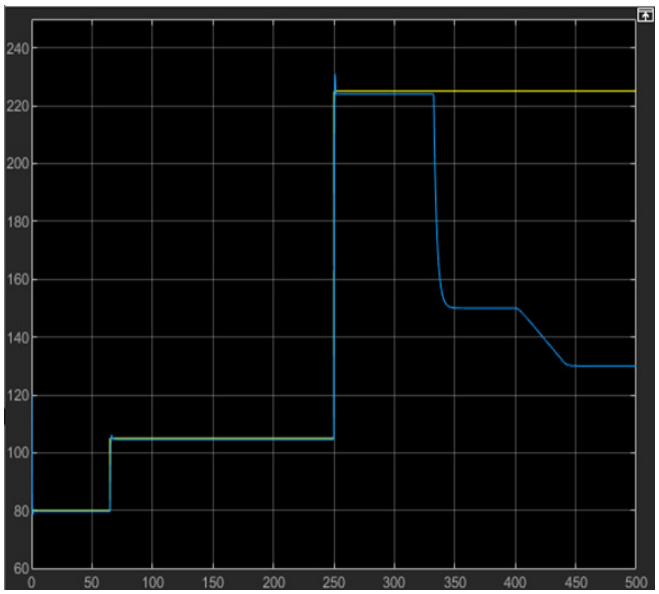

FIGURE 23: Case 3 Linear Quadratic Tracking Controller Output.

The NN controller does a more efficient job of following the desired output response of the front car. At points where we can see the overshoots and undershoots between 200 and 300 seconds, the controller prioritized avoiding a collision. The performance was good as it always keeps a safe distance of 50-100 m from the car in front and adapts well to the changes in speed and distance. The only problem is that the model always tries to get to the same speed as the front car, even if the desired rate is below that value.

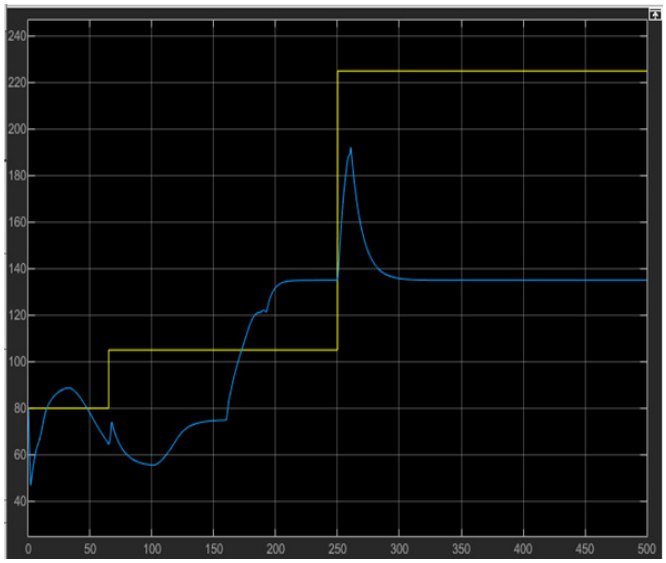

FIGURE 24: Case 3 Neural Network Controller Output.

The ANFIS controller, in this case, performs exceptionally well. The controller can take into consideration both objectives and perform them quite well. Mostly, it respects the avoidance of collisions and cruise control that was set. In addition, it keeps adequate room to avoid a collision, although we will see in the following two models that a greater undershoot occurred during this case.

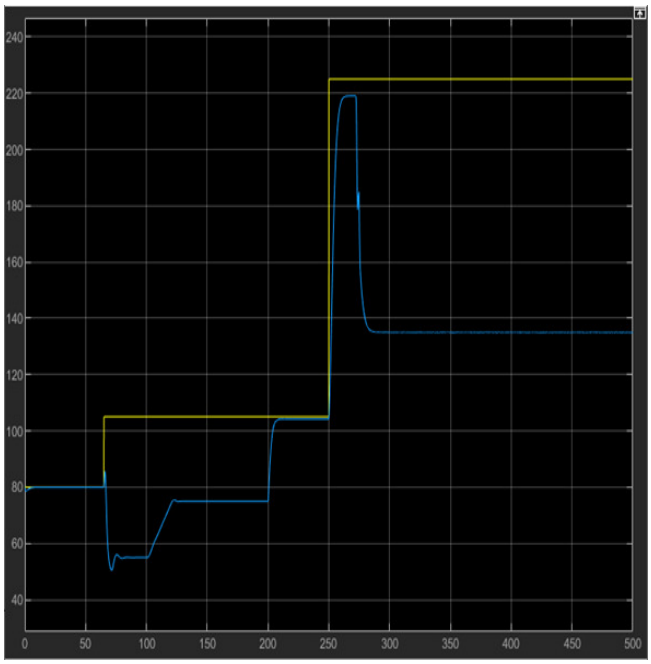

FIGURE 25: Case 3 Neuro-Fuzzy Controller Output.

The ANFIS-LQT performed most optimally during this case, too, as the output response tracks the desired output response of the front car while following the set cruise control. It settles the fastest, and the steady-state response is better. At the 300-second mark, it was the only controller to follow the front car exactly how it was.

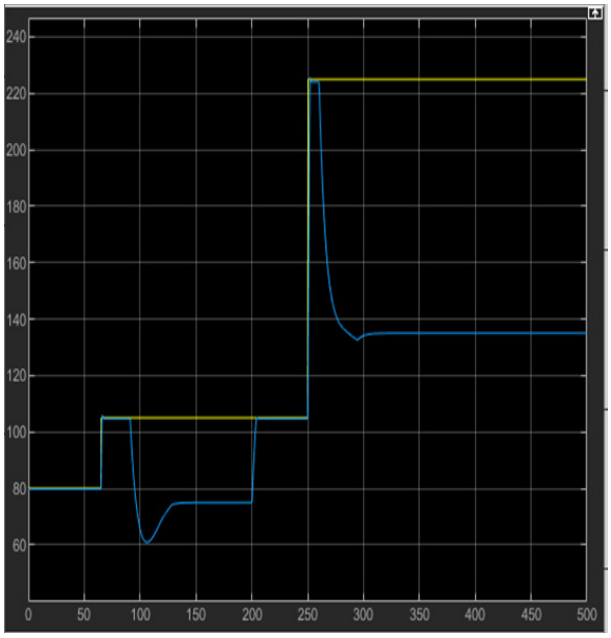

FIGURE 26: Case 3 Neuro-Fuzzy Linear Quadratic Tracking Controller Output.

The LQT-NN controller came up just short as it performed similarly to the ANFIS-LQT; however, the simulation had overshoots at 75 seconds and 200 seconds. Nevertheless, the ability to react and return to the desired response is excellent.

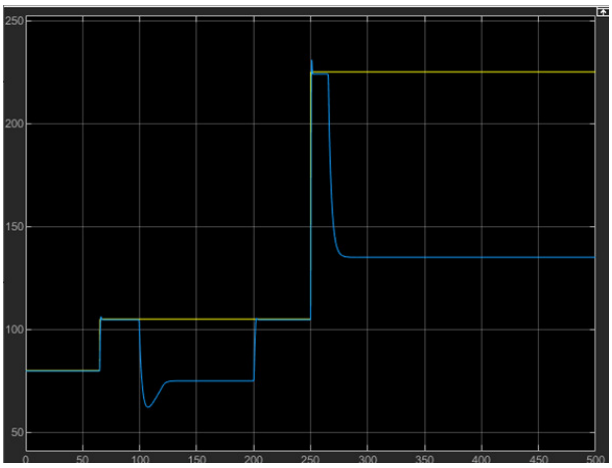

FIGURE 27: Case 3 Linear Quadratic Tracking Neural Network Controller Output.

5. CONCLUSION

This research aimed to identify the optimal controller for the cruise control system created based on recent research in automotive control systems. Based on the research conducted on controllers and the implementation of design and analysis of the cruise control systems, it can be concluded that the Neuro-Fuzzy Linear Quadratic Tracking (ANFIS-LQT) controller is optimal out of all controllers for this system. In each case created where the goal was to maintain a set speed and avoid a collision, the ANFIS-LQT controller outperformed the other seven controllers. The LQT-NN would have to be the controller that performed the closest to the top model design controller. When it comes to respecting the safe distance margin, such as the NN and NF models, while others show a better response to speed change and appreciate the desired speed, the ANFIS-LQT ability to provide an optimally controlled feedback gain to enable the closed-loop stable allowed for a high-performance design of the system. Even more exciting is that this controller can be further improved by tuning the data to its best and implementing it with these conditions.

Could the other controllers outperform the ANFIS-LQT controller under different conditions? It is quite possible. For example, if the MPC controller had been created as a MIMO controller, it would have had the chance to. However, it suffered due to the linear model using the spared prediction. Due to the control action not being symmetric, it would over-compute the brake action and under-compute the throttle action. Implementing a nonlinear model for the MPC could have been another possibility for improvement. The PID controller could have implemented gain scheduling, undoubtedly enhancing the performance. The fuzzy logic system could have created more domains, as only nine parts were used. If there were 20-plus domains, the performance would have been more optimal. Finally, the neural network could have been trained with more layers or data. Based on the designs created and controllers implemented, the Neuro-Fuzzy Linear Quadratic Tracking controller is the optimal controller until proven wrong.

6. REFERENCES

Abdulnabi, A. (2017). PID Controller Design for Cruise Control System using Particle Swarm Optimization. Iraqi Journal for Computers and Informatics, vol. 43, no. 2, pp. 30–35, doi:10.25195/ijci. v43i2.61.

Al-Aubidy, K. (2023) Neural Networks in Control Systems. Intelligent Control Systems. Philadelphia. University-Jordan.[Online]. Available: https://www.philadelphia.edu.jo/academics/ka-ubaidy/uploads/IntCon-lec5.pdf.

Chen, X., et al., (2019). A synergetic strategy of automobile intelligent cruise system based on fuzzy control adopting hierarchical structure. International Journal of Advanced Robotic Systems, 16(5), p.172988141987775.

Chu, H., Dong, S., Hong, J., Chen, H., and Gao, B. (2022). Predictive Cruise Control of Full Electric Vehicles: A Comparison of Different Methods. ScienceDirect,[Online], https://www.sciencedirect.com/science/article/pii/S2405896321015536#section-cited-by/.

Hearst Auto Research. (2023). What is Adaptive Cruise Control?. Car And Driver. [Online]. Available:https://www.caranddriver.com/research/a32813983/adaptive-cruise-control/.

KIA Motors. (2023). How does Cruise Control Work?. KIA. [Online]. Available: https://www.kia.com/dm/discover-kia/ask/how-does-cruise-control-work.html n.d.

Li, S et al. (2016). Multiple Model Switching Control of Vehicle Longitudinal Dynamics for Platoon Level Automation. IEEE Transactions on Vehicular Technology. 65. 1-1. 10.1109/TVT.2016.2541219.

MyCarDoesWhat.org, My Car Does What, (2016). https://mycardoeswhat.org/safetyfeatures/lane-keeping-assist/.

Rawat, S. (2022) Advantages and Disadvantages of Neural Networks. Analytics Steps. [Online]. Available: https://www.analyticssteps.com/blogs/advantages-anddisadvantages-neural-networks.

Saleh, A., & Farmanbordar, A. (n.d.). Modeling Linear Quadratic Regulator LQR/LQT/LQGT for Inverted Pendulum System. [Online]. Available: http://textroad.com/pdf/JAEBS/J.%20Appl.%20Environ.%20Biol.%20Sci.,%205(4S)1-11,%2020-15.pdf.

Steffano, I., Stubberud, A., & Williams, I. (1967). Feedback and Control Systems. Schaum OutlineSeries. McGraw-Hill.

Vandoren, V. (2016). Understanding PID control and loop tuning fundamentals. Control Engineering; Control Engineering. https://www.controleng.com/articles/understanding-pid-control-and-loop-tuning-fundamentals/.

VisibleBreadcrumbs.(2023). What is Model Predictive Control. Mathworks.com. https://www.mathworks.com/help/mpc/gs/what-ismpc.html.

Wright, I. (2021). Evolution of Modern Car Technology. CarBuzz. [Online]. Available: https://carbuzz.com/features/evolution-of-car-technology-and-features.