Hardware-in-the-Loop Plant Modeling for Autonomous Vehicle

First A. Author, *Fellow, IEEE*, Second B. Author, and Third C. Author, Jr., *Member, IEEE*

***Abstract*— This report presents the results of modeling vehicle systems and testing them using Hardware-in-the-Loop (HIL). First, data was collected for each system using a Lexus RX450H vehicle. After data collection, models were developed in order to accurately simulate the physical vehicle system. MATLAB's System Identification Toolbox was initially used to create these models. However, the transfer function models it provided weren't able to accurately represent these systems and their non-linear behaviors. After this was established, modeling was done using MATLAB's Neural Network Time Series application. Once a Neural Network model was created, it was then exported to Simulink and modified to be able to run without the use of the Deep Learning toolbox. These models gave much more accurate results when compared to the transfer function models developed using the System Identification Toolbox. (Add section about testing and results).**

***Index Terms*— Hardware-in-the-Loop, Neural Network, System Identification**

I. INTRODUCTION

AUTONOMOUS vehicles are being developed by many companies for commercial and personal use. These autonomous vehicles (see the AutonomouStuff vehicle fleet shown in Fig. 1(a)) would allow companies to continue crucial deliveries or transports of their products even if there is a shortage of drivers. In addition, autonomous vehicles have the ability to make roads safer for drivers and pedestrians alike. To develop a reliable and safe transportation system for modern world, a large body of research in the field of autonomous vehicles is being conducted in recent decades [1] [2]. Therefore, it is apparent that researchers of the autonomous vehicle community have been focusing on analysis, design, and development of different subsystems of self-driving/autonomous vehicles. Furthermore, modeling vehicle subsystems is a pre-requisite for the development of reliable controllers for these vehicles to be managed in the era of modern transportation systems in general.

Usually, there are six main subsystems of a self-driving vehicle:

1) Steering,
2) acceleration,
3) brake,
4) shift,
5) speed, and
6) speed control (cruise) subsystems.

These subsystems are to be modeled to get an accurate representation of how the vehicle should be controlled. Within the scope of this project and to expedite the modeling process, commercially available modeling tools in Mathworks' MATLAB, System Identification app in the System Identification Toolbox and Neural Network Time Series app in the Deep Learning toolbox, were used to model six subsystems for a Lexus vehicle platform, which is an experimental autonomous vehicle platform that belongs to AutonomouStuff Solutions [1]. The first subsystem we will model is the steering model, and then we will move onto the other subsystems. There are already controllers in place for the Lexus vehicle platform, but their reliability is lower than expected due to non-linear behaviors of the torque voltages required to control each subsystem. The scope of this project includes developing mathematical models for each subsystem so AutonomouStuff can implement control systems that improve the reliability of the autonomous vehicle platform. These models will be considered reliable if they can track actual vehicle subsystem behaviors with a minimum best fit percentage between 85 to 95 percent and fall within the error bounds defined for each subsystem. Once these models are developed, they will be tested using AutonomouStuff's Hardware-in-the-Loop (HIL) bench. Once there is confidence in each model, AutonomouStuff can use these models on their HIL bench to develop controllers that will remove the non-linear behaviors of each vehicle subsystem. There are two operating modes of the Lexus vehicle platform used in this project: Manual-drive and by-wire or autonomous. Fig. 1(b) depicts how the by-wire mode controls the vehicle. Each subsystem, like the steering system in this example, sends torque voltages to a motor that will control the subsystem. In Fig. 1(b), the motor would turn the pinion arms which would change the steering angle of the vehicle. This principle is applied to all other subsystems.

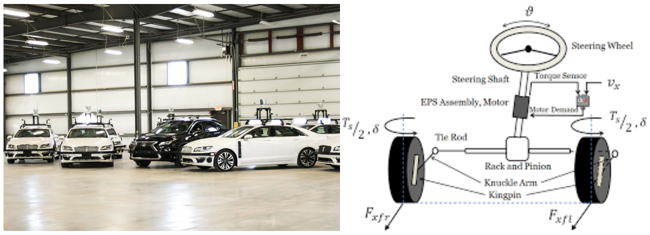
This paragraph of the first footnote will contain the date on which you submitted your paper for review. It will also contain support information, including sponsor and financial support acknowledgment. For example, "This work was supported in part by the U.S. Department of Commerce under Grant BS123456."

The next few paragraphs should contain the authors' current affiliations, including current address and e-mail. For example, F. A. Author is with the National Institute of Standards and Technology, Boulder, CO 80305 USA (e-mail: author@boulder.nist.gov).

S. B. Author, Jr., was with Rice University, Houston, TX 77005 USA. He is now with the Department of Physics, Colorado State University, Fort Collins, CO 80523 USA (e-mail: author@lamar.colostate.edu).

T. C. Author is with the Electrical Engineering Department, University of Colorado, Boulder, CO 80309 USA, on leave from the National Research Institute for Metals, Tsukuba, Japan (e-mail: author@nrim.go.jp).

[1] https://autonomoustuff.com/



(a) (b)

Fig. 1: (a) Autonomous vehicle fleet in AutonomouStuff Solutions and (b) steering model setup (courtesy of AutonomouStuff).

II. Literature Review

Authors in [3] illustrates identification of multiple-input single-output model for maximum power point tracking of photovoltaic system. A significant effort was conducted to model the photovoltaic system, where the two inputs were Solar irradiance and Cell temperature, and the output was DC current. To model this system, Matlab's System Identification Toolbox was used. In order to create different models, they collected and used data from an energy center in Malaysia. After generating different models, the authors ended up going with a fourth order ARX (also known as ARXQS) model because it was the most accurate, with a best fit percentage of 93.42%. The polynomial model equation for ARX is shown below.

$$y(t) + a_1 y(t-1) + ... + a_{na} y(t-n_a) = \\ b_1 u_1(t-n_k) + ... + b_{nb} u_{nb}(t-n_k-n_b+1) + e(t),$$

$y(t)$ is the system output at time t, while $u(t)$ is the input. The noise disturbance of the system is represented by $e(t)$. The variables na, nb, and nk are the system's number of poles, amount of b parameters, and the samples before the inputs begin to affect the system's output.

III. System Identification Preliminaries

System identification is the process of developing mathematical models for a dynamic system using the measurement of input and output signals of that system. There are many components that are used to accomplish a task that they are assigned without knowing the exact behavior of the system for given input signals. Without knowing the response of this black-box, there could be unexpected consequences from a system. The goal of the system identification methodology is to get an accurate estimation of the system response to any given input. Mathworks' MATLAB has the System Identification Toolbox, where a few existing examples demonstrate the working principle of this toolbox.

A. Example 1: Dealing with Multi-Variable Systems: Identification and Analysis

The example given in [4] shows how to create an iddata object from a dataset in order to get the inputs and the outputs. The next step was to look at the impulse and step responses in order to learn more about how the inputs and outputs act. From there the state space model was estimated using the first part of the given data. This model was compared to the step responses and with the second part of the data to see if it was a good fit. The model had a best fit percentage of 83.55% for the generated voltage data, and a best fit of 39.33%

for the speed data. The frequency response of the model was estimated with spectral analysis and bode plots were also created. The tutorial explained that if the data doesn't give nice models, then it is best to try out submodels for the different channels. Two single-input single-output (SISO) models were created and compared with the existing multiple-input multiple-output (MIMO) model and the actual data. The Nyquist plots were also compared. Both SISO models performed well during these comparisons. The next step in the tutorial was to create a multiple-input single-output (MISO) model in order to get a model that more accurately reflected the generated voltage data. By creating this model and comparing it to the validation data and previous models, we saw a best fit percentage of 90.18%. The last thing the example showed was how to merge the two SISO models we created earlier.

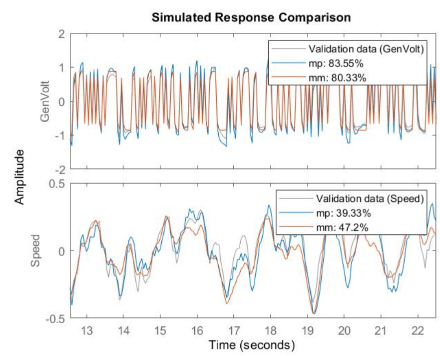
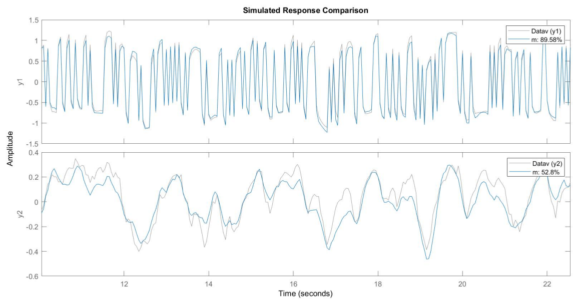


Fig. 2: Comparison of the state space model and merged SISO models with the validation data

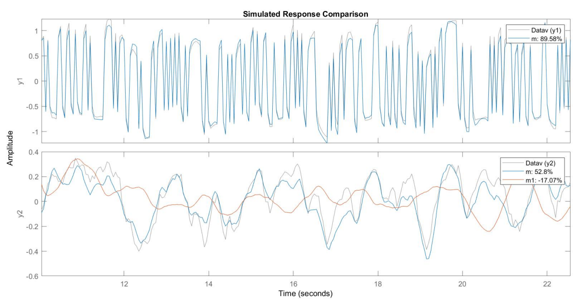
B. Example 2: Selecting Model Structures for Multivariable Systems

This example [5] discusses solutions for modeling both MISO and MIMO models using Mathworks' Matlab System Identification Toolbox. As the article discusses, MISO system models are easier to develop because all model structures used by the toolbox support models with a single output and multiple inputs. Therefore, the process for developing a model for a MISO system is importing the data as an iddata object, removing the mean from the data, and estimating the solution using any model structure available in the toolbox. The command line can be used by using the function associated with the model structure name and then using the compare function to get the best fit percentage. For MIMO systems, there are not model structures built into the System Identification Toolbox and they must be imported instead. Otherwise the process is very similar to that of a MISO system. For a MIMO system, using the compare function can be crucial. The compare function will tell which output channel is the most difficult to develop a model for, if it is possible at all.

With this information, the output channel that is hardest to model should first be modeled individually because there will be less freedom in what model structures are available. The other channels should be able to closely relate to the model for the output channel you selected.



(a)



(b)

Fig. 3: (a) State-space model of a MIMO system with a validation data comparison and (b) State-space model of a MIMO system and system-sized based state-space model with a validation data comparison

C. Example 3: Identify Linear Models Using the Command Line

This example in [6] shows how to create models for MISO systems using the command line. Before starting the model estimation process, the equilibrium values of the inputs and outputs had to be taken out. The data from each experiment also had to be separated into different iddata objects. The example showed how to estimate and compare non-parametric impulse response, transfer function, ARMA, state-space, and Box-Jenkins models with the measured experimental data. The state-space model with the five-step response prediction was the most accurate, with a best fit percentage of 85.83%.

IV. System Architecture

The overall system architecture of this project consists of six subsystems which are the steering, acceleration, brake, speed, shift, and speed control systems. Each of the six subsystems will be treated as a multiple-input single-output (MISO) system. For each subsystem, every input that is a torque voltage is actually two torque voltage signals, thus can not be treated as a single input. Also, each subsystem will be treated as a single output system. The brake subsystem is

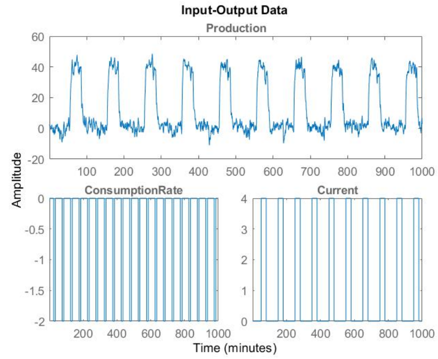


Fig. 4: Inputs and outputs of the given system

shown to have two outputs but the behaviors of one of the outputs is already known so it will be modeled based on the other output's behavior.

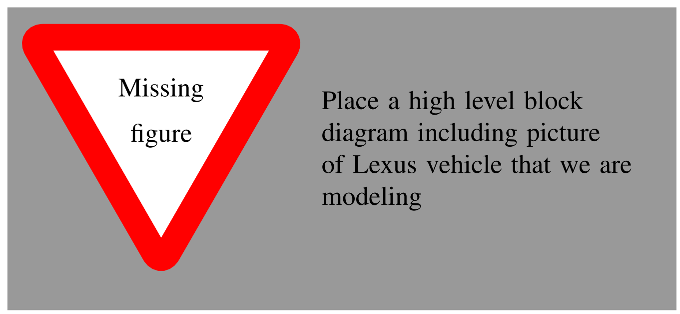


Fig. 5: Hexagon Lexus self-driving vehicle showing different subsystems

Fig. 6: AutonomouStuff Lexus RX450H vehicle.

A. Steering Subsystem

The steering subsystem consists of steering, the power steering motors, ... The ultimate goal of this subsystem is to control the steering angle for the vehicle to navigate in the desired heading. Therefore, the control system is designed to produce appropriate voltages to be applied to the power steering motors for the steering orient in the target direction. The block diagram of the control system designed for this subsystem is shown in Fig. 7

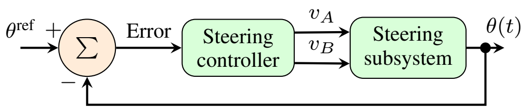


Fig. 7: Steering subsystem block diagram.

B. Brake Subsystem

The brake subsystem takes the Brake Pedal Pressure Voltages, Brake Pedal Stroke Voltages, and the Brake Pedal On/Off Switch values as inputs. Using these values, it generates a new Brake Pedal Position and a boolean value called Brake Pressed. This boolean value indicates to the user whether or not the brake pedal is being pressed. The brake subsystem is another example of a multiple-input multiple-output system.

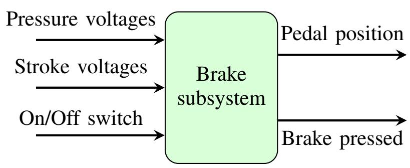


Fig. 8: Brake subsystem block diagram

C. Acceleration Subsystem

The acceleration subsystem is a multiple-input multiple-output system. Acceleration pedal voltages are sent to the subsystem. A new acceleration pedal position value is generated to better match the real-time pedal position. This is then the output of the acceleration subsystem.

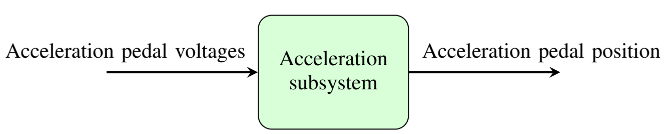


Fig. 9: Acceleration subsystem block diagram

D. Shift Subsystem

This subsystem can be classified as a single-input single-output system. The subsystem takes the desired shifter gear value from the user. Within the subsystem, the actual shifter gear changes to better reflect the desired gear. This actual shifter gear value is then the output of the shift subsystem.

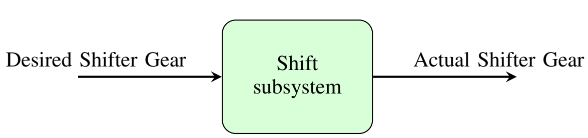


Fig. 10: Shift subsystem block diagram

E. Speed Subsystem

This speed subsystem would fall under the multiple-input single-output system. There are four inputs, the position of the acceleration pedal and the brake pedal, along with the shifter actual gear. Taking these inputs, the speed subsystem finds the vehicle speed. This vehicle speed is the output of the system.

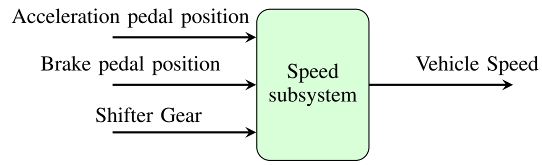


Fig. 11: Speed subsystem block diagram

F. Speed Control Subsystem

The speed control subsystem is a straightforward single-input single-output system. The desired vehicle speed is set by the user and sent to the speed control subsystem. Taking this input, the subsystem calculates the new vehicle speed. This value is then sent out to the rest of the vehicle system.

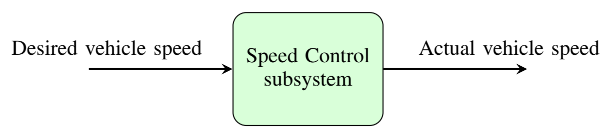
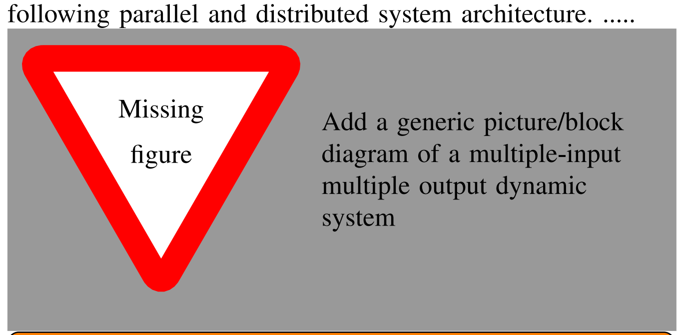


Fig. 12: Speed Control subsystem block diagram

V. Modeling Dynamic System using Neural Network

In this section, we illustrate how a neural network can be used to model a dynamic system in general. Neural network is a system approximation technique that .... The main structure of a neural network consists neurons, input layer of neurons, one or more hidden layers, and the output layer of neurons. The neurons of the hidden layers and the output layer have processing capabilities. The actual network of a nural network following parallel and distributed system architecture. .....



Add a generic picture of a neural network

Weights of a neural network

Training of a neural network

Testing a neural network

VI. Modeling Vehicle Subsystems

To start this project, we first read documentation outlining the uses of MATLAB's System Identification Toolbox. From there we worked on MATLAB tutorials on how to model systems from data using System Identification and then find the most accurate model. We specifically tried to find examples with Multiple-Input Multiple-Output (MIMO) and Multiple-Input Single-Output (MISO) systems, since most of the vehicle subsystems we will model fall into one of these categories. A literature review was also conducted to see how systems with small non-linearities like our steering subsystem were modeled using System Identification. On October 7th we traveled to AutonomouStuff in order to collect data from the steering, acceleration, and braking subsystems in an autonomous vehicle. The data we collected will be used to generate and then verify our models.

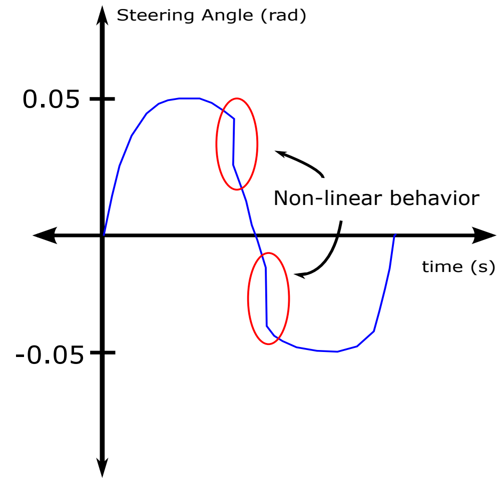


Fig. 13: Steering Non-linear Behavior

A. System Requirements

Based on the system requirements listed above, the plant model will meet the following specifications:

- The steering subsystem will be modeled first due to its non-linearities, depicted in Fig. 13, and because it is important to the operation of the vehicle
- The steering subsystem will be able to handle small steering angles
- Each vehicle subsystem will be modeled separately

The vehicle plant model will fulfill the requirements listed below:

- The resulting plant model will consist of accurate subsystem models, as defined above
- The subsystems can be used to create a HIL testbench
- The subsystems can handle very small changes in values accurately

All of the subsystems have nonlinear behaviors when there are small changes in the output of the subsystem. The steering subsystem, for example, should behave in a smooth, continuous manner. However, AutonomouStuff observed that when trying to implement features such as lane tracking for the Lexus vehicle platform, that when small changes in the steering angle (less than five degrees) occurred, the torque voltages would momentarily stall and then suddenly change causing a more drastic change. This behavior, depicted in Fig. 13, made it very challenging to complete features like lane tracking. These models will aid in the development of controllers that will remove the nonlinearities allowing features like lane tracking to be implemented in a safer and smoother manner.

Add subsections for each subsystem on how NNs are used to model each subsystem

For each subsystem you will need to draw different NN structure

VII. Validation and Testing

To start this project, we first read documentation outlining the uses of MATLAB's System Identification Toolbox. From there we worked on MATLAB tutorials on how to model systems from data using System Identification and then find the most accurate model. We specifically tried to find examples with Multiple-Input Multiple-Output (MIMO) and Multiple-Input Single-Output (MISO) systems, since most of the vehicle subsystems we will model fall into one of these categories. A literature review was also conducted to see how systems with small non-linearities like our steering subsystem were modeled using System Identification.

A. Experimental Setup

On October 7th we traveled to AutonomouStuff in order to collect data from the steering, acceleration, and braking subsystems in an autonomous vehicle. The data we collected will be used to generate and then verify our models. We collected data on the Lexus RX450H vehicle platform shown in Fig. 6.

The following is the hardware required for this project:

- Laptop
- PACMod ECU
- CANCase
- CAN bus

The laptop is used to pass commands or log data, such as steering angle or acceleration or brake pedal position. This data is sent or received using the CANCase and CAN bus. These are connected to the AutonomouStuff designed PACMod ECU, which sends torque voltages to the desired vehicle subsystem allowing the laptop to either control the desired vehicle system or log data.

The following is the software required for this project:

- MATLAB's System Identification Toolbox
- Vector CANAlyzer

The Vector CANAlyzer software is installed on the laptop and is used to parse the collected data that is sent from the CANCase. This is how we were able to collect logs of data that we would use to develop models of the autonomous vehicle subsystems. MATLAB's System Identification Toolbox is the software that gave us the capabilities to develop these models of the vehicle subsystems. Using this toolbox, we are able to use the logs of data we collected to create sets of estimation and validation data that will be used for training the model.

Each subsystem that we are modeling is set up in a similar manner. In manual mode, the torque voltages that control each subsystem are sent by the vehicle's electronic control unit (ECU). In order to control the vehicle autonomously, the vehicle subsystem switches to by-wire mode. In by-wire mode, the torque voltages from the vehicle's ECU are discarded by open-circuiting the motors that control each subsystem. Instead, the PACMod ECU built by AutonomouStuff sends the torque voltages to the motor using relays. In Fig. 14, the experimental setup for data collection is shown. The laptop is used to collect the data from the desired vehicle subsystem by the use of Vector's CANAlyzer software. The CAN Case collects data from the PAC Mod and ECU and sends the data using a CAN bus to the laptop which is then parsed and displayed through the use of CANAlyzer. The ECU and PAC Mod are not shown in Fig. 14 as they are fixed behind panels of the vehicle.

Fig. 14: Autonomous Vehicle Data Collection Setup

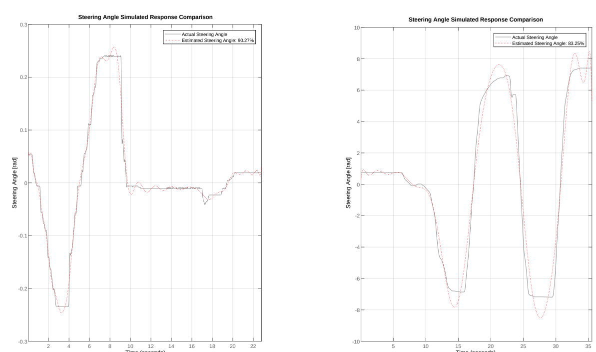
Using the data we collected for the steering, acceleration, and braking subsystems, we initially separated the data so we could analyze the by-wire and manual modes individually. This effort was made to identify if the models we developed could be used interchangeably regardless of what mode the autonomous vehicle subsystems were operating in. After analyzing the models we developed, we determined that they are not interchangeable as there were differences in the models. When the vehicle subsystem is in by-wire mode, the controller AutonomouStuff developed is generating the torque voltages that are applied to each subsystem motor. As a result, we decided the most accurate model of each subsystem would be developed using the data when the subsystems are in manual mode. For reference, the differences in the models created for by-wire and manual mode are depicted in VII for the steering and acceleration subsystems. For all other models we develop, we will only use manual mode data and will not create models for by-wire mode.

Another decision we made was to start using the Neural Network to create models. We decided to do this instead of continuing with the System Identification Toolbox for multiple reasons. The main one is because while transfer function models are able to accurately model linear systems, they struggle to model highly non-linear systems. Neural Networks work well with highly non-linear systems and are able to accurately model them. Since our systems, and the steering system in particular, exhibit non-linear behavior, they are not able to be accurately modeled by transfer functions. Another reason is that once the System Identification models were created in Simulink, there was confusion over how they would connect to produce the output. Some of the system models, such as the brake system, had four transfer functions due to having four inputs. These models were easily generated, but it wasn't clear on how they should be connected in order to reflect the physical system. By using Neural Network models this problem can be avoided.

B. Transfer Function Modeling

Discuss results using the setup you explained in the previous section. See some IEEE papers

1) Steering Subsystem: In Fig. 15(a), the output steering angle is plotted versus time when the vehicle is in by-wire mode. The figure shows the output steering angle from some data collected to depict the steering system behavior plotted with the behavior of the estimated model. The best fit percentage for this model is 85.54%. Likewise, Fig. 15(b) depicts the output steering angle plotted versus time when the vehicle is in manual mode. The best fit percentage for this model is 90.27%.



(a) Output of Estimated By-Wire System Model (b) Output of Estimated Manual System Model

Fig. 15: Steering System Estimated Steering Angle Comparison

The model of the by-wire steering system is a twentieth order transfer function. The twentieth order transfer function is the best proposed estimation considering the system costs associated with a higher order transfer function while also obtaining an acceptable best fit percentage. Table **??** shows the coefficients of the twentieth order transfer function for the output steering angle with respect to the input torque voltage A signal.

TABLE I: By-Wire Mode Steering Transfer Function Torque Voltage A Coefficient Table

a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}	a_{18}	a_{19}	a_{20}
1.949E-16	1.455E-14	8.797E-13	2.304E-11	9.041E-10	9.01E-9	2.714E-7	9.241E-7	1.946E-5	3.889E-5	0.0005721	0.0008035	0.008695	0.008922	0.07499	0.05422	0.3631	0.1693	0.9394	0.2118	1

b_0	b_1	b_2	b_3	b_4	b_5
3.44E-16	-2.203E-15	1.24E-13	5.975E-13	3.001E-12	5.23E-12

Table **??** shows the coefficients of the twentieth order transfer function for the output steering angle with respect to the input torque voltage B signal.

TABLE II: By-Wire Mode Steering Transfer Function Torque Voltage B Coefficient Table

a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}	a_{18}	a_{19}	a_{20}
2.92E-15	2.758E-13	1.761E-11	4.228E-10	1.422E-8	1.561E-7	1.501E-6	8.426E-6	5.981E-5	0.0002016	0.001226	0.002649	0.01456	0.02044	0.104	0.09228	0.4417	0.2257	1.025	0.2305	1

b_0	b_1	b_2	b_3	b_4	b_5
-6.645E-15	2.161E-14	-2.605E-14	-4.744E-13	-1.813E-13	-4.858E-12

The manual steering system is modeled by a twentieth order transfer function. After considering the system costs that come with a higher order transfer function and the need to achieve a sufficient best fit percentage, it is clear that a twentieth order transfer function is the best estimation of the system. Table III shows the coefficients of the twentieth order transfer function for the output steering angle with respect to the input torque voltage A signal.

TABLE III: Manual mode steering transfer function torque voltage A coefficient table.

a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}	a_{18}	a_{19}	a_{20}
1.224E5	4.227E5	2.202E6	3.54E6	7.98E6	7.62E5	1.071E7	6.745E6	6.85E6	3.006E6	2.365E6	7.37E5	4.669E5	1.021E5	5.319E4	7752	3353	287.5	102.4	3.653	1

b_0	b_1	b_2	b_3	b_4	b_5
6.516E4	-2.944E5	2.328E5	-2.487E5	8.25E4	-3.038E4

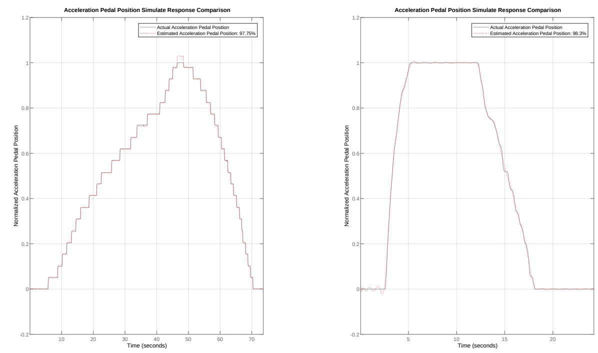
Table IV shows the coefficients of the twentieth order transfer function for the output steering angle with respect to the input torque voltage B signal.

TABLE IV: Manual Mode Steering Transfer Function Torque Voltage B Coefficient Table

a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}	a_{18}	a_{19}	a_{20}
4.687E4	7.037E5	2.624E6	6.133E6	1.197E7	1.487E7	1.887E7	1.446E7	1.314E7	7.01E6	4.79E6	1.886E6	9.836E5	2.937E5	1.149E5	2.617E4	7205	1232	200	23.54	1

b_0	b_1	b_2	b_3	b_4	b_5
1.209E4	1.855E5	-2.801E5	4.72E5	-3.976E4	7.693E4

2) Acceleration System: In Figure 16(a), the output acceleration pedal position is plotted versus time when the vehicle is in by-wire mode. The figure shows the output acceleration pedal position from some data collected to depict the acceleration system behavior plotted with the behavior of the estimated model. The best fit percentage for this model is 97.69%. Likewise, Figure 16(b) depicts the output acceleration pedal position plotted versus time when the vehicle is in manual mode. The best fit percentage for this model is 98.3%.



(a) Output of Estimated By-Wire System Model (b) Output of Estimated Manual System Model

Fig. 16: Acceleration System Estimated Pedal Position Comparison

The model of the by-wire acceleration system is a fourth order ARX model. The fourth order ARX model defined below represents the output acceleration pedal position, A(z), based on input torque voltage A, $B_1(z)$, and input torque voltage B, $B_2(z)$.

$$A(z)y(t) = B_1(z)u_1(t) + B_2(z)u_2(t) + e(t),$$

where na = 4, nb = 4, nk = 0, and,

$$A(z) = 1 - 1.018z^{-1} - 0.002901z^{-2} + 0.4631z^{-3} - 0.2038z^{-4}$$
$$B_1(z) = -0.0207 - 0.01912z^{-1} - 0.02159z^{-2} - 0.03307z^{-3}$$
$$B_2(z) = 0.02017 + 0.01833z^{-1} + 0.09461z^{-2} + 0.05683z^{-3}$$

The model of the manual acceleration system is a twenty-fourth order transfer function. The twenty-fourth order transfer function is the best proposed estimation considering the system costs associated with a higher order transfer function while also obtaining an acceptable best fit percentage. Table V shows the coefficients of the twenty-fourth order transfer function for the output acceleration pedal position with respect to the input torque voltage A signal.

TABLE V: Manual Mode Acceleration Transfer Function Torque Voltage A Coefficient Table

a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}	a_{18}	a_{19}	a_{20}	a_{21}	a_{22}	a_{23}	a_{24}
2.307E7	8.369E7	9.137E8	1.992E9	3.753E9	5.454E9	5.548E9	5.869E9	3.453E9	3.161E9	1.54E9	9.654E8	3.556E8	1.789E8	5.082E7	2.071E7	4.569E6	1.498E6	2.553E5	6.532E4	8443	1559	147.3	15.5	1

b_0	b_1	b_2	b_3	b_4	b_5	b_6
-5.741E6	-2.644E7	-2.797E6	-1.304E7	-3.203E6	-1.159E6	-5.519E5

Table VI shows the coefficients of the twenty-fourth order transfer function for the output acceleration pedal position with respect to the input torque voltage B signal.

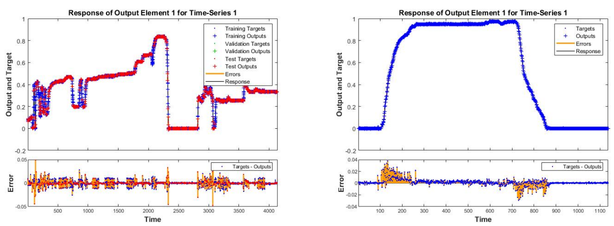
3) Brake System: The brake system was modeled using only the manual log data. It was trained using the data from the first brake log, and the result and error plots can be seen in

TABLE VI: Manual Mode Acceleration Transfer Function Torque Voltage B Coefficient Table

a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}	a_{18}	a_{19}	a_{20}	a_{21}	a_{22}	a_{23}	a_{24}
5.224E6	4.205E7	1.429E8	3.233E8	5.444E8	7.326E8	8.247E8	7.402E8	6.301E8	3.99E8	2.731E8	1.264E8	7.223E7	2.48E7	1.216E7	3.084E6	1.324E6	2.422E5	9.264E4	1.159E4	4003	307.4	96.87	3.451	1

b_0	b_1	b_2	b_3	b_4	b_5	b_6
-4.159E6	1.851E6	-4.457E6	9.043E5	-7.45E5	1.865E5	-1.545E4

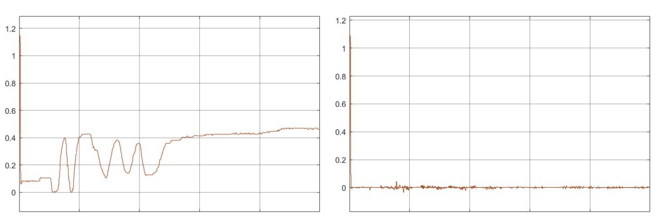
Figure 17(a). The plots show that the model is able to predict the outputs relatively accurately, along with keeping the error below 5%. In order to further verify the accuracy of the model, it was tested using data from the second brake log. The results from that test, shown in Figure 17(b), also show that the model can accurately predict the output values and do this within the accepted error bound of 5%.



(a) Model Output Tracking and Error Plots (b) Model Output Tracking and Error Plots for Log Two Data

Fig. 17: Brake System Training Plots

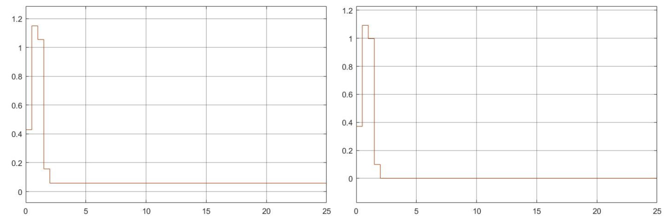
Once the model was created, it was exported to Simulink and tested again. One test involved using To Workspace blocks to access the pressure voltages, position voltages, and desired pedal position values in the first brake log. These values were used as inputs for the model, and the actual pedal position along with the error between the desired and actual pedal positions were then plotted. The actual pedal position plot can be seen in Figure 18(a). The resulting error plot between the desired pedal position and the model generated position can be found in Figure 18(b). Despite an initial spike within the first few seconds of the graph, the error plot shows that the difference between the pedal positions generated by the system and the desired pedal positions is never more than 5%.



(a) Actual Pedal Position (b) Error Between the Actual and Desired Pedal Position

Fig. 18: Actual Pedal Position and Error Plots Using the From Workspace Blocks

The second test used constant blocks to represent the voltages and desired pedal position at a specific time instant in the first brake log. The error and the actual pedal position were then plotted again. The pedal position plot is shown in Figure 19(a), and the error plot is represented by Figure 19(b). As the plots show, the model is able to produce the desired pedal position and the error falls to 0% after a spike at the start.



(a) Actual Pedal Position (b) Error Between the Actual and Desired Pedal Position

Fig. 19: Actual Pedal Position and Error Plots Using the Constant Blocks

C. Neural Network Modeling

Discuss results NN modeling using the setup you explained in the previous section. See some IEEE papers

D. Discussion

Discuss pros and cons of the results you presented in the previous sections on each of the techniques you incorporated

VIII. Conclusion and Future Work

One paragraph on conclusion, make sure you to recommend what modeling technique for the company

One paragraph on future work

References

[1] M. Liu, G. Li, and B. Liu, "Real-time knowledge discovery from public data of internet to improve decision-making of autonomous vehicles," in *2017 Second International Conference on Reliability Systems Engineering (ICRSE)*, 2017, pp. 1–3.

[2] B. Prasad, Q. Y. Huang, and J.-J. Tang, "Development of a prototype ev autonomous vehicle for systematic research," in *2020 International Computer Symposium (ICS)*, 2020, pp. 459–461.

[3] M. Hussain, A. Omar, and A. Samat, "Identification of multiple input-single output (miso) model for mppt of photovoltaic system," in *2011 IEEE International Conference on Control System, Computing and Engineering*, 2011, pp. 49–53.

[4] "Dealing with multi-variable systems: Identification and analysis," https://www.mathworks.com/help/ident/ug/dealing-with-multi-variable-systems-identification-and-analysis.html.

[5] L. Ljung, "Selecting model structures for multivariable systems," http://www.ece.northwestern.edu/local-apps/matlabhelp/toolbox/ident/ch3tut44.html, 2021.

[6] "Identify linear models using the command line," https://www.mathworks.com/help/ident/gs/identify-linear-models-using-the-command-line.html.