UNIVERSITY OF CALIFORNIA,

IRVINE

PHEV Power Management Optimization Using Trajectory Forecasting Based Machine Learning

DISSERTATION

submitted in partial satisfaction of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in Mechanical and Aerospace Engineering

by

Joseph Garcia

Dissertation Committee:

Doctor Gregory Washington, Chair

Professor Faryar Jabbari

Professor Fadi Kurdahi

2021

TABLE OF CONTENTS

List of Figures

List of Tables

Acknowledgements

I would like to first thank Dr. Washington for all the help and guidance he has given me over the years. He has pushed and guided me throughout my academic career into becoming the student I am today. I owe a large portion of who I am as a researcher and engineer to him. I would also like to thank Professor Jabbari and Professor Kurdahi for serving on my defense committee.

I would next like thank the many members of the ISSL lab. Joseph Bell and Vatche Donikian provided me with the resources to obtain the ADVISOR simulation software required for the basis of my research. Most notably, I would like to thank my lab mate and good friend Theron Smith. He helped me formulate a lot of my research's main ideas, as well advise me in the development of my research's control strategy.

Lastly, I would like to thank my family and many loved ones. Most important are my mother, Mirna Guardado, and sister, Karla Peña. They have sacrificed much throughout their lives to ensure my success and education. Next is Robin Jeffers, whose help in my undergraduate studies played a large role in leading me to my current position. For always being by my side, I would like to thank my remaining loved ones and friends who have supported me throughout my academic career. Thank you to all of you for always supporting me and motivating me to continue forward.

Abstract of Dissertation

Real-Time Route Optimization of PHEVs using Trajectory Forecasting Based Machine Learning

By

Joseph Augusto Garcia

Doctor of Philosophy in Mechanical and Aerospace Engineering

University of California, Irvine, 2021

Professor Gregory Washington, Chair

In hopes of lessening the reliance on fossil fuels, Plug-in Hybrid Electric Vehicles (PHEVs) have become an attractive option as an alternative fuel vehicle due to their larger electric motors and energy storage systems (ESS). PHEVs can propel themself relying solely on their internal combustion engine (ICE), electric motor (EM), and or a combination of both. To improve their fuel efficiency, many studies have been done to investigate the use of a priori route information to optimize the use of a PHEV's ICE and EM. This study introduces a real-time machine learning application of a supervisory control strategy known as Trajectory Forecasting (TF). TF takes a priori knowledge of a PHEV's pre-planned route to determine when the vehicle will use its different forms of propulsion in the form of propulsion mode scheduling. However, it assumes constant route data such as traffic and resulting driving speed for its scheduling to be applicable. To automatically account for changing traffic as well as choose better alternative routes, this study looks at the use of a Convolutional Neural Network (CNN) to simulate a PHEV's operation along available routes beforehand according to the rules of TF to choose a route that best satisfies a driver's want, better fuel efficiency and possibly lower emissions. This novel real-time TF based machine learning control strategy is evaluated and compared to common PHEV control strategies such as Charge Sustaining (CS) and Charge Depletion (CD) using National Renewable Energy

Laboratory’s vehicle simulator ADVISOR. Results show possible increases in MPGGE from 1.72%-130%, decreases in emitted hydrocarbons (HC), carbon monoxide (CO), and nitrous oxides (NOx) from 0.05%-70%, and 0.05%-90.35% reduction in gasoline consumption depending on overall route length and PHEV configuration.

1. Introduction

Daily transportation has become extremely dependent upon fossil fuels. Currently, the United States has less than 5% of the world's population, but approximately one-fifth of the world's automobiles [1]. Serving as the primary fuel source for this vast amount of transportation is petroleum. In 2017, the United States led the world in petroleum consumption at a rate of 19.88 million barrels per day and had a net petroleum import of 3.8 million barrels per day [2]. In 2018, the consumption of petroleum in the United States increased to a rate of 20.5 million barrels per day [3] while the net petroleum import decreased to 2.34 million barrels per day [4]. In 2019, the United States had a petroleum consumption of 20.46 million barrels per day [5] and a net petroleum import of 670 thousand barrels per day [4]. Even with a decreased net petroleum import, the current use of such fossil fuels forces the United States to heavily rely on them with the combustion of petroleum distillates from use as fuel in transportation leading to serious environmental issues from pollution.

When fossil fuels are burned, nitrogen oxides, hydrocarbons, particulate matter, and other pollutants are released into the atmosphere. These pollutants lead to harmful respiratory issues in people, the formation of smog, and function as a heat trapping greenhouse gas that worsen the effects of climate change [1]. In response to the problems from burning fossil fuels, many states have begun to implement legislation to promote the search of alternatives and solutions. California established its Renewables Portfolio Standard (RPS) in 2002 under Senate Bill 1078, requiring that electrical corporations increase their procurement of eligible renewable resources by 1% per year until 20% of its total retail sales are procured from renewable resources [6]. In 2015, the passing of Senate Bill 350 then required electrical retail sellers and publicly owned utilities in California to procure 50% of their electricity from renewable resources by 2030 [7]. California's latest RPS goal, according to Senate Bill 100 passed in 2018, is now 60% by 2030 with all state’s electricity being required to come from carbon-free resources by 2045 [8].

To add to the list of practical solutions for lessening the effects of fossil fuels, alternative fuel vehicles have been identified as viable options. Currently, drivers can choose to drive vehicles that use alternative fuels such as biodiesel, electricity, ethanol, hydrogen, natural gas, or propane [9]. Increased interest in fuel efficiency over the past few decades has made them, specifically hybrid electric and plug-in hybrid electric vehicles (PHEVs), more accepted with increased attention leading to numerous technological improvements and cost reduction.

2. PHEV Technology

2.1. PHEV Vehicle Design

On the market today, two main types of hybrid electric vehicles (HEV) are sold; plug-ins and non-plug-ins. A non-plugin HEV, as shown in Figure 1, is propelled by a combination of both an internal combustion engine (ICE) and an electric motor (EM). The electricity that powers the EM can be generated by the vehicle’s own regenerative braking system and/or on-board generator. The use of regenerative braking is a process where the electric motor helps to slow the vehicle and converts the resulting kinetic energy into usable electricity [10]. Its on-board generator is powered by the ICE, producing electricity to power the EM and recharge the vehicle’s battery. A PHEV, as shown in Figure 2, is similar but has the added ability to have its battery re-energized by being plugged into an external electrical charging source. This allows PHEVs to carry larger electric motors and batteries, giving PHEVs’ electric vehicle (EV) mode an increased driving range with the ability to produce zero emissions. As a result, PHEVs can have higher overall miles per gallon (MPG) compared to vehicle that depend solely on an ICE. The added features and capabilities of PHEVs have made them a great area of focus, with companies and consumers looking for ways to further increase their electric driving range in between charges.

PHEVs can have a series, parallel, or a series-parallel drivetrain configuration. In a series configuration, a PHEV runs solely on the EM with an ICE only being used to power a generator and recharge the vehicle’s battery. In a parallel configuration, a PHEV can run on an ICE and electric motor individually or in a blended mode. The series-parallel configuration allows the PHEV to behave with a series or parallel configuration depending on which is more efficient at the given speed and torque request. A PHEV with the ability to run solely powered by its ICE, EM, or a combination of both is categorized as a full hybrid, or strong hybrid. In comparison, non-plugin hybrids are typically categorized as mild hybrids, which comprise vehicles with limited hybrid technology, as they generally cannot run as a full EV due to the limited size of their batteries and electric motors.

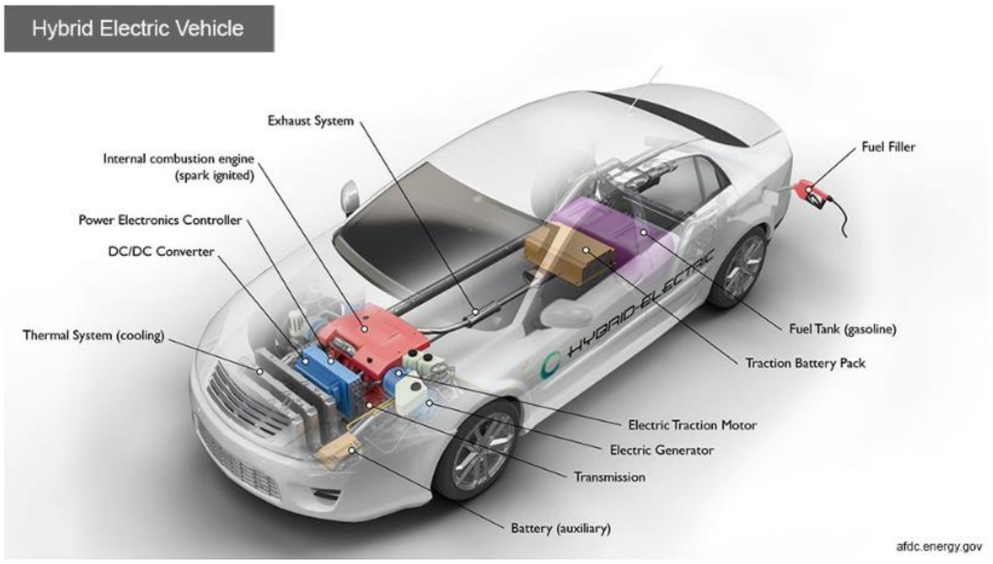


Figure 1: Hybrid Electric Vehicle Design [11]

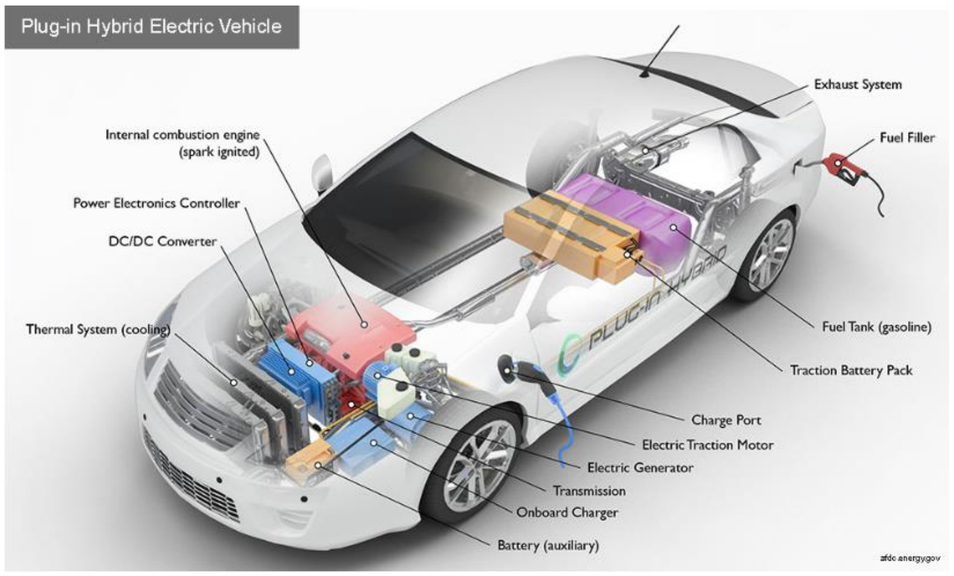


Figure 2: Plug-in Hybrid Electric Vehicle Design [12]

A larger battery will increase the all-electric range of a PHEV compared to an HEV, decreasing the fuel consumption over a given distance, thus leading to improved tank to wheel fuel economy for the vehicle and decreased harmful emissions [13]. This has resulted in sales rising over the years with 72,885 sold in 2016 as reported by the Office of Energy Efficiency and Renewable Energy. However, it is important to keep in consideration that additional battery weight decreases the attainable efficiency in miles per kWh and miles per gallon for a PHEV [14]. Therefore, an optimum in tank to wheel fuel economy can be reached when larger battery capacity and addition battery weight are both properly taken into consideration.

From data taken from 2015 modeled light duty passenger vehicles, on-road fuel economy was averaged at about 31 mpg [15]. Compared to a traditional HEV, a PHEV has the capability of alternating between using an electric motor, the internal combustion engine, and a mixture of both depending upon which driving mode is chosen. The modes include electric vehicle (EV) mode, charge depletion (CD) mode, charge sustaining (CS) mode, and internal combustion engine (ICE) mode. EV mode is when the PHEV runs solely on the battery and electric motor until it completes a predefined cycle or reaches a predefined minimum state of charge (SOC) in the vehicle's storage system [16]. CD mode is when the PHEV runs primarily using the electric motor with a net decrease in SOC, with the ICE turning on when the power demand is too high for the electric motor to handle or if the SOC drops too low [16]. CS mode is when the PHEV is propelled by the electric motor, ICE, or a combination of both, with the constraint of maintaining a constant SOC in the battery [16]. ICE mode is where the PHEV runs solely on the ICE to propel itself. These modes allow maximum flexibility in reducing overall fuel consumption when coupled to a supervisory control that optimally chooses the driving mode that is most efficient for vehicle operation from those available. For example, in stop-and-go urban driving EV mode will be more efficient as internal combustion engines are less efficient at low engine speed, usually characterized by revolutions per minute (RPM).

2.2. PHEV Energy Management Control Strategy Research

To further optimize and expand the all-electric range that a PHEV can travel, an onboard computer can use specific energy management strategies to determine which of its various modes to run in. With a proper control strategy, a PHEV can autonomously decide when and to what extent to use its two energy sources (battery energy or fuel) to increase its overall efficiency over a given drive cycle. This feature allows power management systems to strongly influence and increase PHEV fuel efficiency [17]–[19]. Giving some insight into the energy management strategies, switching between a PHEV's various modes can be controlled automatically as a function of battery SOC, vehicle speed, engine speed, engine torque, environment temperature, battery temperature, and air conditioning need [20]. It is important to note that these decision factors all deal with the state of the PHEV itself in real-time. Therefore, having accurate readings of these factors proves to be very important. A battery's SOC has proven to be the most difficult to measure with extensive research into improving measurement accuracy through open circuit voltage and coulomb counting (current integration) measurement techniques. Fortunately, research has shown that with proper evaluation of a battery's state of health (SOH) at recharging and discharging, estimation error for SOC can be reduced to 1% at the operating cycle [21].

In many actual cases a PHEV's driving mode can also be manually selected by the driver based upon which mode they desire to use at a given moment. It is this manual control option that has led to the consideration of a new energy management strategy that considers factors beyond the confines of the PHEV. Drivers of PHEVs have been known to manually switch between operating modes based on road conditions, such as traffic, that they anticipate facing on their commutes. Choosing to run in EV mode as often as possible in instances of medium traffic or forcing the vehicle to run in CS mode in small instances of stop-and-go traffic, are strategies that consider a driver's knowledge of their route to further improve their vehicle's fuel economy. Given that modern vehicles have on-board global positioning systems (GPS) with included traffic, weather, road grade and hazard information, supervisory controllers like the one outlined in this research can optimize which driving mode will be utilized at what time to optimize fuel economy over the whole route.

The most examined strategies used to enhance PHEV fuel efficiency over a particular route use optimal control and optimization [22]. From the perspective of optimal control, many have considered the use of rule-based control strategies, such as fuzzy logic controls (FLC), driving mode classification, and dynamic feedback control [22]. Studies in [22]–[24] have shown that FLC type techniques make controllers easier to implement as operation merely requires matching immediate driving conditions to different prearranged scenarios for which to adjust the power contributions from the electric motor and ICE. In a similar fashion, driving mode classification techniques rely upon different parameters obtained from past and current driving conditions to characterize real-time driving patterns and adjust driving control strategies accordingly [25]. Studies investigating the benefits of using a combination of both FLC and driving mode classification techniques have been presented in the past by Langari and Won [26], [27]. As discussed in [28] and [29], dynamic feedback control approaches solve for the control strategies based on current and previous operations, which are easier for real-time implementation. Unfortunately, these algorithms are not able to reach global optimality in terms of power distribution over an entire route [22], thus investigations have then been made into dynamic programming (DP).

DP is a common optimization technique that has been used to obtain global optimality [30]–[34]. However, it uses a distinct few standardized dive cycles from the U.S. Department of Transportation to optimize power management for various other routes, serving more as a reference than an exact solution [22]. Fortunately, with the development and accessibility of trip prediction and modeling such as intelligent transportation systems, geographical information systems (GIS), and global positioning systems (GPS), models for individual trips can be accessed a priori [22]. These models can include information such as speed limits, traffic flow, and road grade from one location to another. In combination with a priori drive cycle knowledge, DP has been studied as a near globally optimized power management approach reinforcing the charge-depletion approach [22], [35]. While proven to efficiently optimize PHEV power management, especially with the integration of advanced route modeling, DP is a very computationally expensive technique that would require the optimization be performed offline [22]. Studies have been conducted to reduce the algorithm's computational load to implement it in real-time. A two-scale dynamic programming approach solves for a globally optimized state of charge (SOC) model offline on a macro-level and then adapts the model on a micro-level in real-time onboard the vehicle [36]. However, all these methods still require significant computing power prior and during vehicle operation, especially when trying to adjust to changes in a vehicle's route in real-time.

Another, easier implementable approach is that of a driver-centric approach. In many real-life situations, a PHEV's driving mode is manually selected by the driver based upon which mode they desire to use at a given moment. Drivers of PHEVs have been known to manually switch between operating modes based on road conditions, such as traffic, that they anticipate facing on their commutes. It is this manual control option that has led to the consideration of a new energy management strategy that considers factors beyond the confines of the PHEV. Given that modern vehicles have on-board global positioning systems (GPS) with included traffic, weather, road grade and hazard information, supervisory control strategies can optimize driving mode choice to optimize fuel economy over a whole route. Most notably, works by Chau [37], [38], Murakami [39], and Nejad [40] have begun to focus on this driver-centric type approach to improve fuel efficiency. Using extracted road network data from Southeast Michigan, Nejad creates multigraphs of plausible routes broken up into segments according to road characteristics [40]. Calculating the gasoline consumption according to fuel economy vs speed plots from public domains and charge consumption according to battery consumption rates posted by actual hybrid users along road sections, Nejad uses DP to iteratively find a charge sustaining (CS) and charge depleting (CD) drive mode schedule that minimizes gasoline consumption [40]. Murakami then extends the model created by Nejad to consider time constraints and the availability of using recharging stations on a route [39]. Similarly, Chau develops a DP based strategy to optimize fuel efficiency using path planning to consider both filling and charging stations [37], [38]. These works use constant road characteristics for path planning in their driver-centric approach to implement less computationally intensive dynamic programming strategies. However, they are not designed for considering changing route conditions such as traffic, a characteristic capable of indirectly accounting for things like weather and construction, not to mention the increasing complexity and computational cost from larger scale implementation.

In the area of fuel efficiency approximation, the automobile miles per gallon (MPG) prediction problem is classified as a typical nonlinear regression problem with several car attributes as inputs [41]. Machine learning techniques such as neural networks can solve complex problems by imitating animal brain processes in a simplified manner [42], performing well with non-linear problems due to their ability to learn complex feature relations. Studies by Jamala [42], Aliyu [43], and Meng [41] have been done to better approximate the fuel efficiency of common automobiles using back-propagation neural network computing algorithms. They utilize input features such as number of cylinders, displacement horsepower, weight, acceleration, model year, and origin to estimate a vehicle's MPG with high accuracy. Expanding on the area of fuel efficiency, Topić introduces the use of a convolutional neural network (CNN) to accurately estimate the fuel consumption and remaining state of charge (SOC) of hybrid vehicles at the end of varied routes using only on preprocessed velocity and acceleration data taken from a vehicle's duty cycle [44]. As hybrids switch from CD to CS mode along a route depending on their SOC, Topić shows the capability of neural networks to learn and simulate the inner non-linear dynamics of a hybrid vehicle's powertrain.

This research will focus on implementing machine learning to capture the inner workings of a PHEV using a more fuel-efficient control algorithm known as Trajectory Forecasting (TF) to create an adaptable driver-centric control strategy. TF uses a combination of classification and FLC type rule-based controls to plan and execute an efficient SOC distribution for any given route based upon *a priori* knowledge (of the route and initial SOC) to improve fuel economy. Studies on similar algorithms [45], [17] show the possibility of using *a priori* route information to optimize fuel usage in PHEVs. Creating a machine learning algorithm to estimate the fuel economy and emissions of a PHEV using TF rules will produce a control strategy that will continuously look at updated route options, then choose and optimize the best choice according to fuel economy and then fuel consumption or emissions for the driver while being scalable, computationally fast, and adaptable to changing route data in real-time.

3. Control Strategy

The Trajectory Forecasting based machine learning control strategy proposed in this study is formed using a combination of the newly developed control algorithm known as Trajectory Forecasting and a Convolution Neural Network machine learning algorithm.

3.1. Trajectory Forecasting

A PHEV's driving mode is chosen based on internal factors, such as engine power output, needed acceleration, and SOC, usually not considering the impacts that outside factors also have on the fuel economy. Driving routes can have numerous factors, such as traffic conditions and physical road conditions, that affect a PHEV's entire drive cycle and required power output. Many PHEV drivers manually control the modes of operation of their PHEVs based on information they have regarding their commute. In this study, the term TF will be used to enhance the range of operation of a PHEV by using information of a predetermined route to decide the best time to run in either all-electric (EV) mode, hybrid mode, or ICE mode. The first part of the TF control algorithm will develop an optimal SOC distribution that uses both charge depletion (CD) and charge sustaining (CS) operation, while the second part of the algorithm will implement the distribution.

3.1.1. Setup

The setup of the TF algorithm (corresponding MATLAB code is presented in the appendix section as Trajectory Forecasting Setup Code) initializes by taking the speed limits and traffic flow along the entire distance of a desired route as inputs. This is information that is easily accessible because of the widely available trip prediction and modeling services mentioned earlier. Examples of such a service, the service used as a reference for the route data used in this study, is Google

Maps. Here we present a full traffic flow breakdown of a 64.4-mile route from Irvine, CA in Orange County to Panorama City, CA in Los Angeles County, shown in Figure 3. Due to the long distance of the route, the battery capacity of a PHEV would not be able to sustain the use of EV and hybrid mode through the entirety of it. Thus, a decision must be made for when and how to use the battery so that an improved fuel efficiency for the overall route rather than instantaneous efficiency can be achieved. Using the traffic information, a driver can formulate a general plan for which PHEV propulsion mode to use during different flows of traffic. Shown in Figure 4, the following example explains the reasoning for assigning driving modes along the first quarter of the route.

The planned route takes a PHEV from an apartment on the campus of UC Irvine to an uncongested freeway known as the 73 freeway for a few miles, merging onto the well-known congested 405 freeway and continuing for a long distance. Uncongested traffic, symbolized by the blue colored route portions, from the starting location in Irvine to the merging point between the two freeways indicates that the PHEV can run in ICE mode, hybrid mode, or EV mode with high efficiency. Large amounts of traffic congestion, shown by the red, in following section of the route resulting from the merging of cars between the freeways can then cause extremely reduced speeds and stops. Such stop-and-go traffic is handled more efficiently by a PHEV in EV and/or hybrid mode when charge from the battery is available for use. Experiencing medium traffic, as shown in orange a slight distance after the merging, hybrid mode and EV mode can both efficiently handle any remaining speed variations and reductions when charge is available. In consideration of overall route efficiency, conserving battery charge in sections of little to no traffic by using ICE mode would enable the use of EV and/or hybrid mode in later sections that contain higher traffic congestion when the amount of charge in the vehicle’s battery is not enough to support their consistent use.

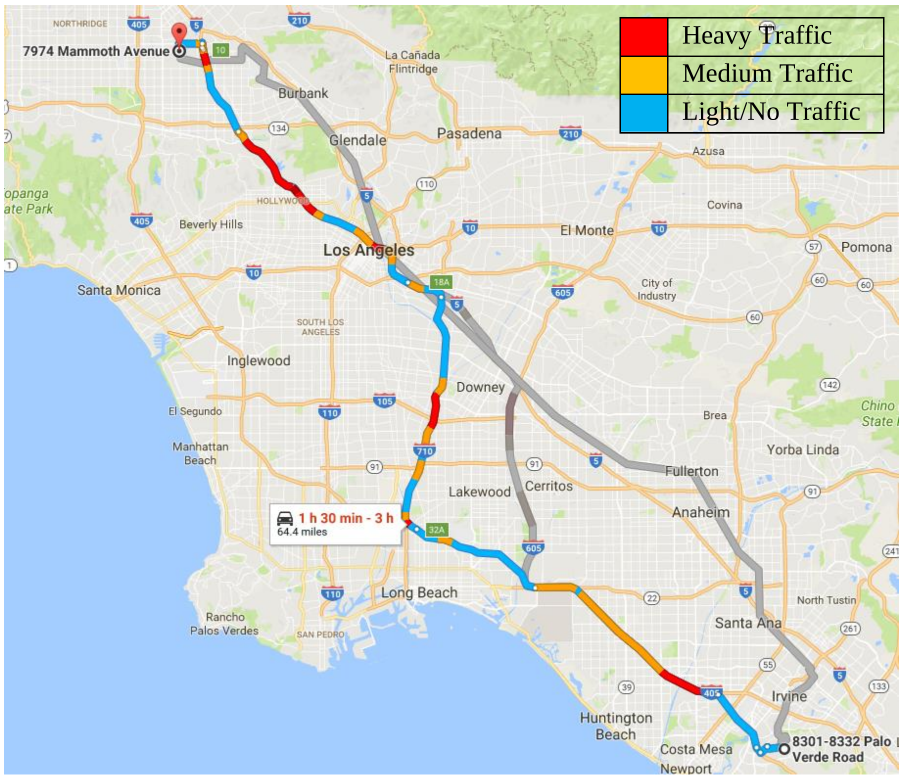


Figure 3: Typical Google Maps Route [46]

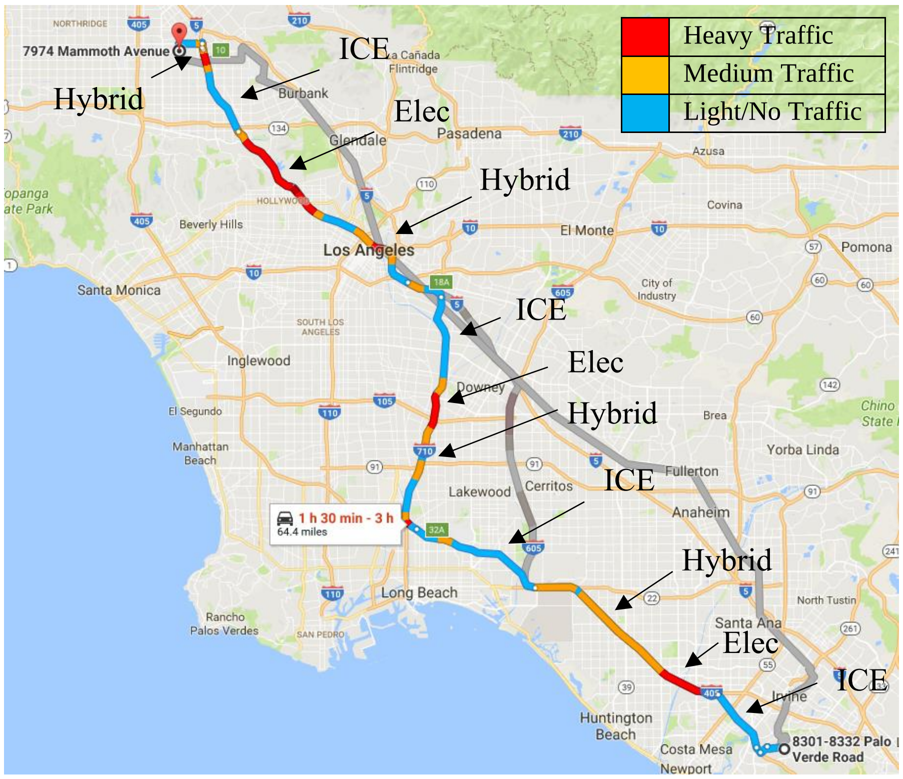


Figure 4: Google Maps Route [46]; modes assigned according to only traffic conditions

Solely relying on traffic flow information, however, is not enough to optimize mode uses in PHEVs. Speed limits and traffic flow must both be used together to estimate the speed a vehicle will most likely drive at each second of the route. This is done using a rule-based classification strategy that assigns the estimated speed using the combination of speed limits and traffic flow. Along each second of the route, a suggested priority driving mode is also assigned using a similar rule-based classification strategy according to the combination of speed limits and traffic flow. Both rule-based classification strategies can be seen in Section 3.1.3 Rule-based Classification. Consecutive seconds with the same suggested priority driving mode classification are then grouped together into sections. The classifications are set as priority 3, priority 2, and priority 1. A section classified as priority 3 is suggested to be run in EV mode. A section classified as priority 2 is suggested to be run in hybrid mode. A section classified as priority 1 is suggested to be run in ICE mode. The resulting plan for the example route is shown in Figure 5.

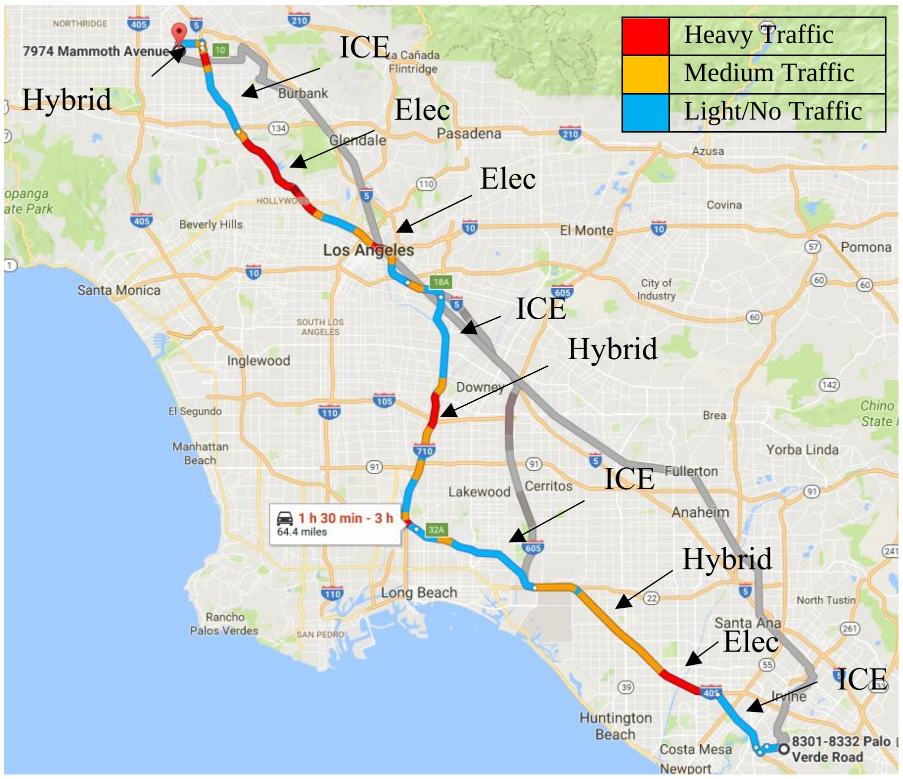
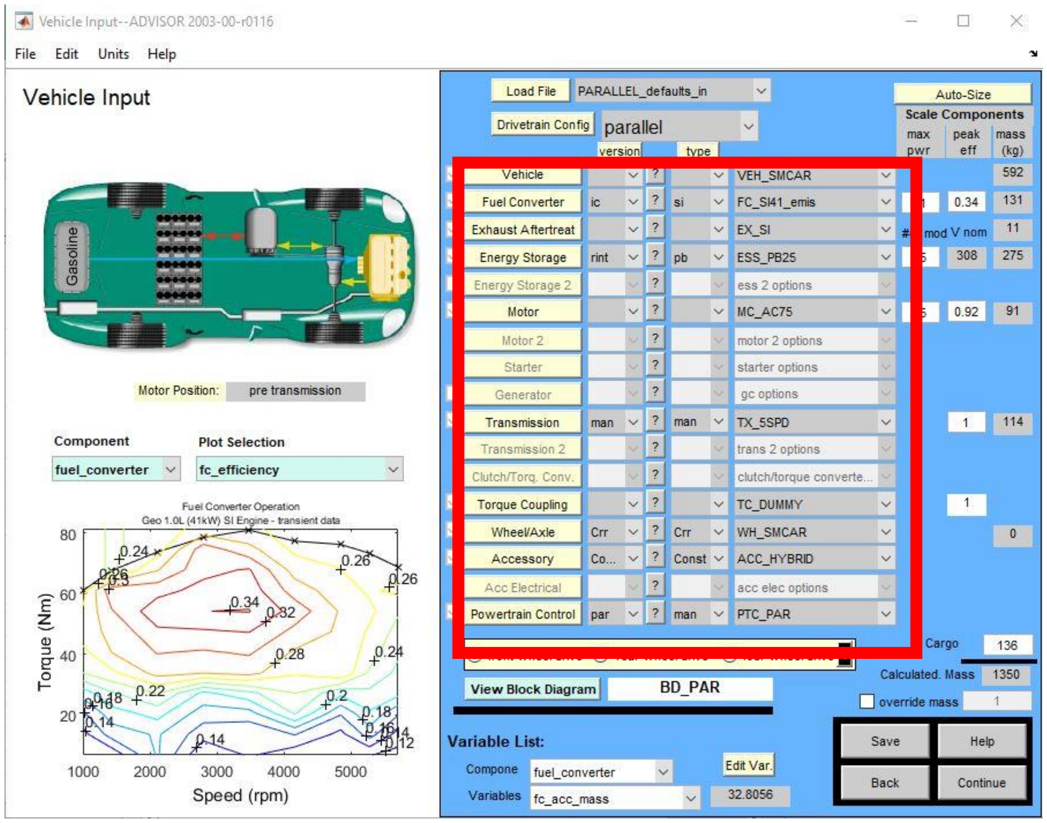


Figure 5: Google Maps Route [46]; modes assigned according to traffic and speed limits

The algorithm is then completed by approximating how much SOC is allocated to the different sections based upon their assigned priority classification, battery type, initial SOC level, max SOC level allowed, and minimum SOC level allowed. In this study, the vehicle simulator ADVISOR, developed by the National Renewable Energy Laboratory, is used perform advanced vehicle simulations. ADVISOR is a simulation program developed to perform rapid analyses of the performance and fuel economy of conventional, electric, and hybrid vehicles, providing support for detailed simulations and studies of user defined vehicle components [47]. Given a required/desired speed input, ADVISOR determines the drivetrain torques, speeds, and power requirements needed to meet the required/desired speed input [47]. This flow of information back through the drivetrain, from the tire to the axle to the gearbox and so on, makes it a backward-facing vehicle simulation type program [47]. For all simulations, a common test parallel hybrid vehicle with the configuration shown in Figure 6 and described in Table 1 was used as the PHEV.



Motor Position: pre transmission

Component: fuel_converter

Plot Selection: fc_efficiency

Load File: PARALLEL_defaults_in

Drivetrain Config: parallel

	version	type		max pwr	peak eff	mass (kg)
Vehicle			VEH_SMCAR			592
Fuel Converter	ic	si	FC_SI41_emis	[illegible]	0.34	131
Exhaust Aftertreat			EX_SI	# mod	V nom	11
Energy Storage	rint	pb	ESS_PB25	[illegible]	308	275
Energy Storage 2			ess 2 options			
Motor			MC_AC75	[illegible]	0.92	91
Motor 2			motor 2 options			
Starter			starter options			
Generator			gc options			
Transmission	man	man	TX_5SPD		1	114
Transmission 2			trans 2 options			
Clutch/Torq. Conv.			clutch/torque converte...			
Torque Coupling			TC_DUMMY		1	
Wheel/Axle	Crr	Crr	WH_SMCAR			0
Accessory	Co...	Const	ACC_HYBRID			
Acc Electrical			acc elec options			
Powertrain Control	par	man	PTC_PAR			

Cargo: 136

Calculated. Mass: 1350

View Block Diagram: BD_PAR

override mass: 1

Variable List:

Compone: fuel_converter

Variables: fc_acc_mass 32.8056

Figure 6: ADVISOR PHEV Configuration [47]

Parameter	Component	Description
Vehicle	VEH_SMCAR	Defines road load parameters for a hypothetical small car, roughly based on a 1994 Saturn SL1 vehicle
Fuel Converter	FC_SIPrime_emis	2017 Prius Prime 1.8L 4 cylinder engine with maximum power of 71 kW @ 5200 rpm and peak torque of 142 Nm @ 3600 rpm
Exhaust After Treatment	EX_SI	Defines exhaust aftertreatment catalyst parameters for hypothetical vehicle equipped with a gasoline-powered SI engine (Masses, areas, etc. are scaled based on engine peak power)
Energy Storage	ESS_PB25	Parameters describe the Hawker Genesis 12V 25Ah 10EP sealed valve-regulated lead-acid (VRLA) battery
Motor	MC_AC75	Westinghouse, 75 kW, AC Induction motor with efficiency/loss data appropriate for a 320 V system
Transmission	TX_5SPD	Defines a 5-speed gearbox by defining gear ratios and gear number, and calling TX_VW to define loss characteristics
Torque Coupling	TC_DUMMY	Defines lossless belt drive with a motor-to-engine speed ratio that ensures the motor is at top speed when the engine is at top speed
Wheel Axle	WH_SMCAR	Defines tire, wheel, and axle assembly parameters for use of a hypothetical small car
Accessory	ACC_HYBRID	Defines standard accessory load data for use with a hybrid in ADVISOR
Powertrain Control	PTC_PAR	Defines all powertrain control parameters, including gearbox, clutch, hybrid and engine controls, for a parallel hybrid using a multi-speed gearbox

Table 1: ADVISOR PHEV Configuration Component Descriptions [47]

For approximating the SOC profile along the path, the program's configured PHEV simulation was performed at different constant speeds in EV mode, 0 mph, 20 mph, 40 mph, 55 mph, and 60 mph, and the average current drawn by the selected motor for the test PHEV at each constant speed was recorded to form an equation for Amps per mph, shown in Figure 7 and Equation 1. To obtain an equation to understand the number of Amp-hours used per mile driven at a specific speed, Equation 2 is formed by taking the derivative of Equation 1. This

$$y = 0.0099x^2 2 + 0.1689x + 0.7278 \quad (1)$$

$$y = 0.0198x + 0.1689 \quad (2)$$

approximation is for determining the PHEV's battery usage in Amp-hours over the sections' distance, as shown in Figure 8. The number of Amp-hours needed to run in EV mode for each segment are calculated and then used to determine the total needed for larger sections. Taking these values and dividing them by the total Amp-hour capacity of the chosen battery, we use the known initial SOC and determine the preferred total initial SOC profile for the route sections. This application of SOC profile approximation using only current from and to the battery and corresponding vehicle speed aims at using a linear version of the relation between SOC estimation and battery current draw found in coulomb counting. Coulomb counting is an efficient SOC estimation method currently used and researched in many battery applications that, with the pre-known capacity of the battery, calculates SOC by integrating the charging and discharging currents over the operating periods of the vehicle [48].

To address any issues of over or under approximating the total charge needed, an adjustment coefficient (AC) in terms of a percentage is added to the equation to increase or decrease the approximation and make it more fail-safe, giving Equation 3. To determine the value

$$y = AC(0.0198x + 0.1689) \qquad (3)$$

of AC, the 3 shortest routes of the 8 used in this thesis, routes 5, 7, and 8 whose characteristics are found in the route data section, are run with the test PHEV as a zero emissions vehicle and the SOC profile for every section along the routes is then recorded. The complete SOC profile recorded for each section priority type of the 3 routes is then compared to the amount approximated using Equation 3, shown in Table 2, Table 3, and Table 4. In comparing the actual SOC profile to the approximated SOC profile, it becomes noticeable that using different AC values for the different speed ranges of the three types of section priorities is more efficient. An AC value is chosen for each of the three section priorities based upon use of the value that results in the expected SOC profile that best matches the actual SOC profile across the multiple sections of the same priority. For this study, the SOC profile is estimated using this method to simplify the non-linear dynamic nature of the vehicle's EM and electronics. These AC values are only valid for the specific PHEV configuration shown earlier that will be consistently used throughout the study. To approximate the SOC profile for any other PHEV EM and electronics setup, another equation fit will have to be done.

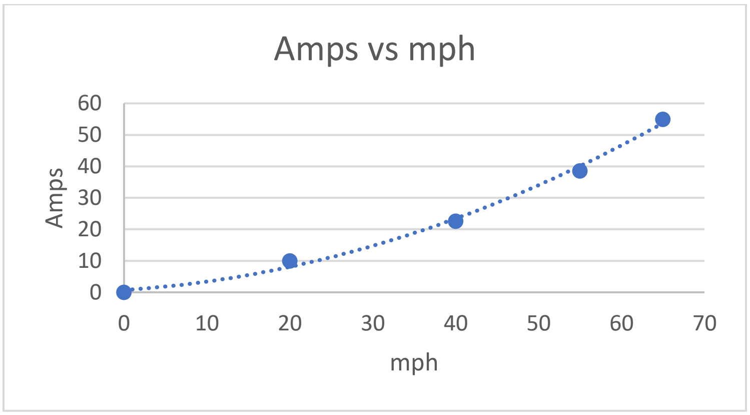


Figure 7: Plot for Amps/mph

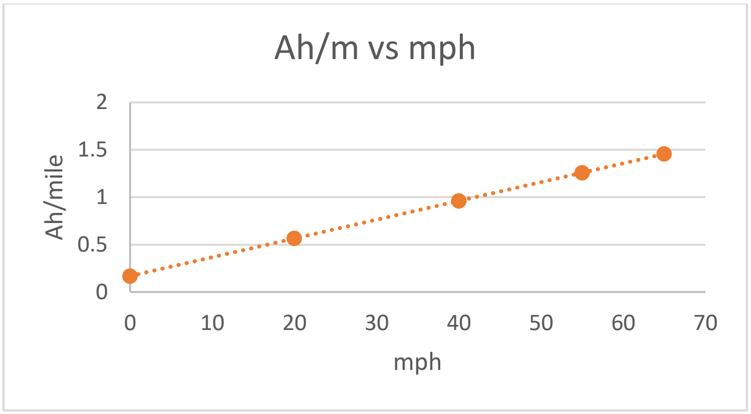


Figure 8: Plot for Amp-hours/mile/mph

Using AC = 0.87 for priority 3 sections, AC = 0.42 for priority 2 sections, and AC = 0.34 for priority 1 sections, the preferred SOC profile for each section and possible SOC profile allowed by the available charge in the battery are used to assess the viability of the suggested modes of each section. The comparison of the SOC profile using the AC values for the three routes can be seen in Table 2, Table 3, and Table 4. Depending upon what the estimated SOC profile along the entire route and the current battery SOC is, four different proposed high efficiency driving scenarios for how to switch between the PHEV's driving modes along the route can occur.

Section Priority Type	Estimated SOC Use	Actual SOC Use	% Error
Priority 1	0.2415	0.2259	6.906
Priority 2	0.2053	0.2041	0.5879
Priority 3	0.2027	0.1991	1.808

Table 2: Route 5 (20.1 miles) SOC Estimation Comparison

Section Priority Type	Estimated SOC Use	Actual SOC Use	% Error
Priority 1	0.2361	0.2216	6.543
Priority 2	0.1882	0.1753	7.359
Priority 3	0.1376	0.1431	-3.843

Table 3: Route 7 (17.1 miles) SOC Estimation Comparison

Section Priority Type	Estimated SOC Use	Actual SOC Use	% Error
Priority 1	0.0897	0.091	-1.429
Priority 2	0.0672	0.0765	-12.16
Priority 3	0.0535	0.062	-13.71

Table 4: Route 8 (6.5 miles) SOC Estimation Comparison

A newly proposed construct for achieving high efficiency is broken down with the following four driving scenarios:

1) within +0.125 SOC of the PHEV's all-electric range, EV mode will be used to drive all sections of the route

2) enough SOC to drive suggested EV and hybrid mode sections in EV mode, and drive suggested ICE mode sections in hybrid mode using the leftover SOC

3) enough SOC to drive suggested EV sections in EV mode with the leftover SOC being used to drive suggested hybrid mode sections in hybrid mode, and remaining ICE suggested sections driven in ICE mode

4) enough SOC to drive some suggested EV mode sections in EV mode and all remaining sections of the route in ICE mode

According to the resulting driving scenario, the current SOC of the battery is distributed among all the sections according to highest priority, attempting to fulfill the sections' SOC needs. Among multiple sections with the same priority, a sublevel priority is given based on length of sections where longer sections have a higher sublevel priority. In driving scenarios where there are sections driven in hybrid mode, the remaining SOC is taken and distributed between all hybrid mode driven sections according to the percentage of the total hybrid mode section distance each one contains, Equation 4. Finalizing the SOC distribution, the created plan dictates how much of the PHEV's total SOC can be used by each section, setting a profile of minimum SOC levels the battery can drain to during each categorized section of the route. This part of the control strategy then outputs the SOC minimum profile, the priority levels of the sections, and preferred high efficiency driving scenario.

$$Hybrid\ Section\ SOC = Remaining\ SOC * \frac{Hybrid\ Section\ Distance}{\sum_{i=1}^{n} Hybrid\ Section\ Distacnce_i} \quad (4)$$

3.1.2. Implementation

For part two of the control algorithm, the outputs from part one are sent as variables to the ADVISOR Simulink PHEV model and TF control strategy block, shown in Figure 9 and Figure 10, respectively. The PHEV simulation is then initialized with ADVISOR's default battery parameters and the set route from part 1 in Section 3.1, both shown in Figure 11. The starting SOC value will be set at 0.8 (80%). Detailed descriptions of ADVISOR's default battery parameters can be found within its documentation [47]. Once the simulation begins, the PHEV's drivetrain experiences a desired torque load request that tries to be met every iteration of the simulation to properly drive at the requested speed along the route. The first step in the TF control strategy block is setting the current SOC minimum, done within the SOC min assignment function block boxed in blue within Figure 10. It takes the vehicle's current distance traveled as an input and uses it to output what SOC minimum from the SOC distribution plan it should follow. The TF control strategy block, boxed in red within Figure 10, then takes the ICE's torque load request as an input and gives a percentage of the ICE's available torque load as an output to satisfy the request. The amount of desired torque that is left to be satisfied is left for the electric motor (EM) to fulfill, represented by Equation 5. The code used in the control strategy function block, boxed in red within Figure 10, and the SOC min assignment function block can be found in the appendix section of this study. A block-by-block walk-through of the ADVISOR Simulink model with corresponding descriptions can be found in the online ADVISOR documentation [47] and the study of Intelligent Control of Parallel Hybrid Electric Vehicles performed by Glenn [49].

$$T_{EM} = T_{LOAD} - T_{ICE} \qquad (5)$$

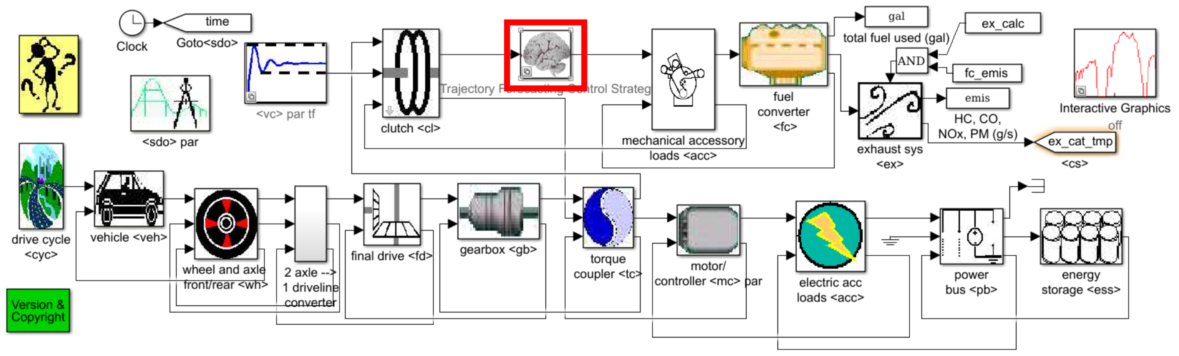


Figure 9: ADVISOR PHEV Simulink Model; control strategy block in red [47]

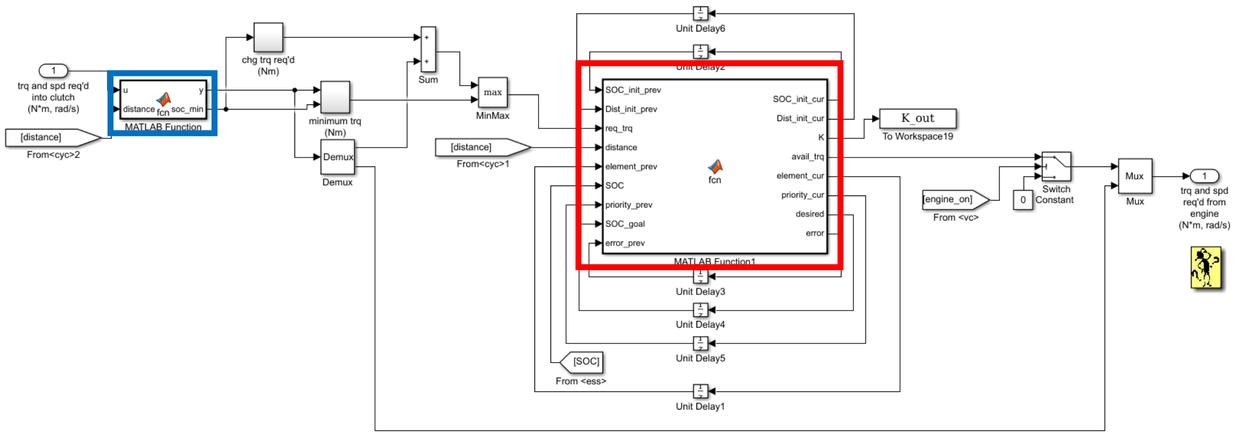
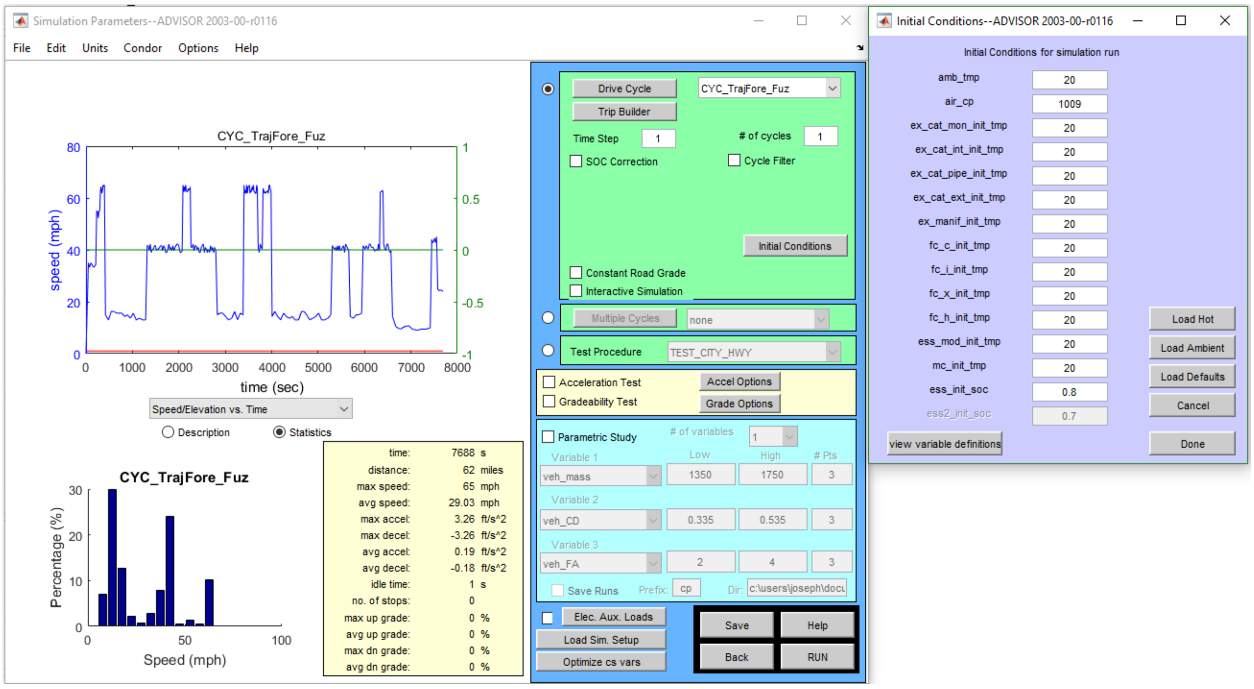


Figure 10: Control Strategy Block Interior [47]; TF control strategy function block additive in red; SOC min assignment function block additive in blue



Drive Cycle: CYC_TrajFore_Fuz

Time Step: 1 — # of cycles: 1

☐ SOC Correction ☐ Cycle Filter

☐ Constant Road Grade

☐ Interactive Simulation

Multiple Cycles: none

Test Procedure: TEST_CITY_HWY

☐ Acceleration Test

☐ Gradeability Test

☐ Parametric Study

# of variables	Low	High	# Pts
1			
Variable 1: veh_mass	1350	1750	3
Variable 2: veh_CD	0.335	0.535	3
Variable 3: veh_FA	2	4	3

☐ Save Runs Prefix: cp Dir: c:\users\joseph\docu

time:	7688	s
distance:	62	miles
max speed:	65	mph
avg speed:	29.03	mph
max accel:	3.26	ft/s^2
max decel:	-3.26	ft/s^2
avg accel:	0.19	ft/s^2
avg decel:	-0.18	ft/s^2
idle time:	1	s
no. of stops:	0	
max up grade:	0	%
avg up grade:	0	%
max dn grade:	0	%
avg dn grade:	0	%

Initial Conditions for simulation run

amb_tmp	20
air_cp	1009
ex_cat_mon_init_tmp	20
ex_cat_int_init_tmp	20
ex_cat_pipe_init_tmp	20
ex_cat_ext_init_tmp	20
ex_manif_init_tmp	20
fc_c_init_tmp	20
fc_i_init_tmp	20
fc_x_init_tmp	20
fc_h_init_tmp	20
ess_mod_init_tmp	20
mc_init_tmp	20
ess_init_soc	0.8
ess2_init_soc	0.7

Figure 11: ADVISOR Route and PHEV Battery Parameters [47]

Considering the propulsion modes in a PHEV, the control strategy can attempt to satisfy the desired torque load with the ICE and no contribution from the EM, with the EM and no contribution from the ICE, or with contributions from both. EV and ICE modes are implemented by setting the output of the TF control strategy block to 100% or 0% of the ICE's available torque output, respectively. In an EV mode driven route section, the PHEV will run only using the battery until the assigned SOC minimum is reached or it reaches the end of the route section. Hybrid mode is implemented by using the heuristic control approach, FLC, because of its easy implementation, low computational cost, and ability to work on complex non-linear models. The use of FLC will work to ensure that the specific amount of SOC distributed to each of the hybrid sections will be used evenly along each of their lengths by determining what percentage of the ICE's available torque output will be used. The exact breakdown of the FLC method specific to this study's control strategy will be explained in further detail in a later section. For driving scenarios 3 and 4, the sections along a route that require only the use of the ICE may require safety conditions that ensure any SOC recovered through regenerative braking is accounted for and used. This is done by having the torque contribution from the ICE be only 85% or 90% of its available load if the amount of SOC regenerated puts the overall battery SOC at 0.01 or 0.005, respectively, above the section's assigned SOC minimum. The remaining 10% or 15% of the requested torque load will then be fulfilled by the EM. During the use of EV mode, if the available power and torque output from the EM is slightly insufficient to allow the PHEV to achieve a desired speed and the vehicle's SOC is above the current minimum, the control strategy will prioritize efficiency by keeping the ICE off and not permit it to assist in providing the remaining needed torque. This will cause a slight throttling of the PHEV's requested speed in return for less fuel burning, where the requested speeds and resulting accelerations are not fully met by the vehicle. The methodology behind this algorithm allows the PHEV's fuel efficiency to be improved from a global standpoint.

3.1.3. Rule-based Classification

The rule-based classification in the TF algorithm mimics the use of if-then statements for its logic, where explicit outcomes are triggered from different combinations of input parameters. When implementing, we consider the traffic flow and speed limits along a route as the input parameters to the logic. Traffic flow can be broken down into three categories, zero-to-light traffic, medium traffic, and heavy traffic, similarly to how GPS services define traffic flow. What these categories represent is how close to the speed limit a driver can drive their vehicles, setting zero-to-light traffic as driving at 100% of the speed limit, medium traffic as driving at 65% of the speed limit, and heavy traffic as driving at 25% of the speed limit. For speed limits, we consider a total of 7 different possibilities based on different speed limits a driver can encounter, from parking lots to school areas, to residential streets, to rural streets, to business districts, to different highways.

The proposed combinations of the different traffic flow and speed limits, along with their triggered outcomes created in this study for use as desired input parameters for the control strategy can be expressed in Table 5 and Table 6. Table 5 shows the resulting priority the different input parameter combinations trigger, while Table 6 shows an approximated speed a car would drive at the given traffic and speed limit combination. The values from Table 6 are used as approximated speed inputs into Equation 3 for approximating the SOC for the PHEV during different sections of the route.

		Traffic Flow		
		Zero/Light Traffic	Medium Traffic	Heavy Traffic
Speed Limit	10	3	3	3
	15	3	3	3
	25	3	3	3
	35	2	3	3
	45	2	2	3
	55	1	2	3
	65	1	2	3

Table 5: Priority Value Table

		Traffic Flow		
		Zero/Light Traffic	Medium Traffic	Heavy Traffic
Speed Limit	10	10	6.75	2.5
	15	15	9.75	3.75
	25	25	16.25	6.25
	35	35	22.75	8.75
	45	45	29.25	11.25
	55	55	35.75	13.75
	65	65	42.25	16.25

Table 6: Approximated Speed Value Table

3.1.4. Fuzzy Logic Control

FLC is a heuristic form of control logic that focuses on using a practical method to produce a solution not guaranteed to be the optimal solution, but rather a solution that is enough for the immediate goal. It relies on the concept of partial truth, or the degree of truth, to determine which user defined rules are active in calculation of the desired solution [50]. The basics of fuzzy logic can be explained by breaking it down into four main parts: fuzzification, rule-base, inference mechanism, and defuzzification. A more comprehensive breakdown of fuzzy logic control can be found in Fuzzy Control [51].

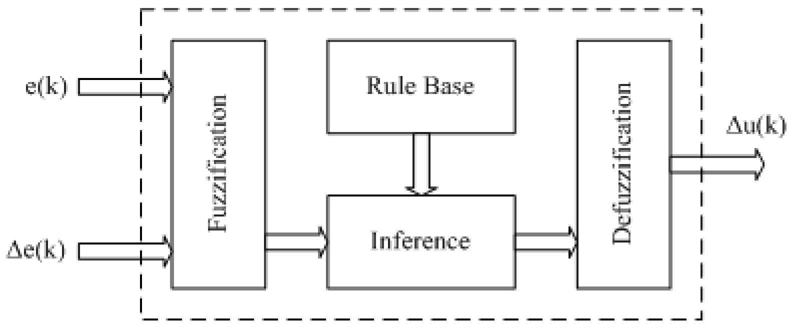


Figure 12: Fuzzy Logic Controller [52]

Before the step of fuzzification, we take the input and output control variables, referred to as linguistic variables, u_i, and categorize them into areas called membership functions. For each hybrid mode section in the route, their given amount of SOC is used up along their distance according to a rate, SOC per mile, found initially at the beginning of the section. This desired rate is found using Equation 6, taking the initial SOC at the start of the hybrid section, subtracting the minimum SOC value it can drain the battery to in the section, and dividing it by the hybrid section's length. The current rate is calculated every iteration along the section in a similar way using Equation 7, but instead uses the difference in SOC between the previous simulation iteration and

$$Desired\ Rate = \frac{Initial\ Section\ SOC - Section\ SOC\ Min}{Section\ Distance} \quad (6)$$

$$Current\ Rate = \frac{Previous\ SOC - Current\ SOC}{Traveled\ Distance} \quad (7)$$

the current simulation iteration, dividing it by the distance traveled between them. Its value changes according to how much of the desired torque load is met by the EM each iteration. How much higher or lower the current rate is compared to the desired rate is taken as a percentage and referred to as the error, Equation 8. From one iteration to the next, we determine the change in error to understand the speed at which the error increases or decreases, Equation 9.

$$e(t) = \frac{Desired\ Rate - Current\ Rate}{Desired\ Rate} \qquad (8)$$

$$\Delta e(t) = Current\ Error - Previous\ Error \qquad (9)$$

The error and change in error are the input linguistic variables that will be categorized into membership functions with corresponding linguistic values, shown in Figure 13. Membership functions are chosen to be a set of shapes, overlapping triangles for the purpose of this study, that break up the domain of the linguistic variables into smaller sections. For the input linguistic variables, each triangular membership function corresponds to a linguistic value, ranging from 1 to 7 for the purpose of this study. These values are descriptors for how a human would describe the size of the input linguistic variables. The linguistic values represent the size of the variables as follows:

1 to represent “large negative”
2 to represent “medium negative”
3 to represent “small negative”
4 to represent “zero”
5 to represent “small positive”
6 to represent “medium positive”
7 to represent “large positive”

The output linguistic variable, defined as K(t), will be a coefficient that regulates the percentage of ICE torque used from what is available at the current engine speed. The possible linguistic values it can take, and corresponding membership functions, will be formed the similarly to those of the input linguistic variables. A difference will be that for the K(t) output, the positive linguistic values will refer to numerical values greater 0.4, the negative linguistic values will refer to values less than 0.4, and the zero linguistic value will refer to values of about 0.4, as shown in Figure 14. Cases showing the general effects of the input variables' values on the output variable value can be seen in Table 7.

Case	$e(t)$	$\Delta e(t)$	K(t)
1	<0	<0	>0.4
2	<0	>0	>0.4
3	>0	<0	<0.4
4	>0	>0	<0.4

Table 7: Qualitative Input/Output Summary

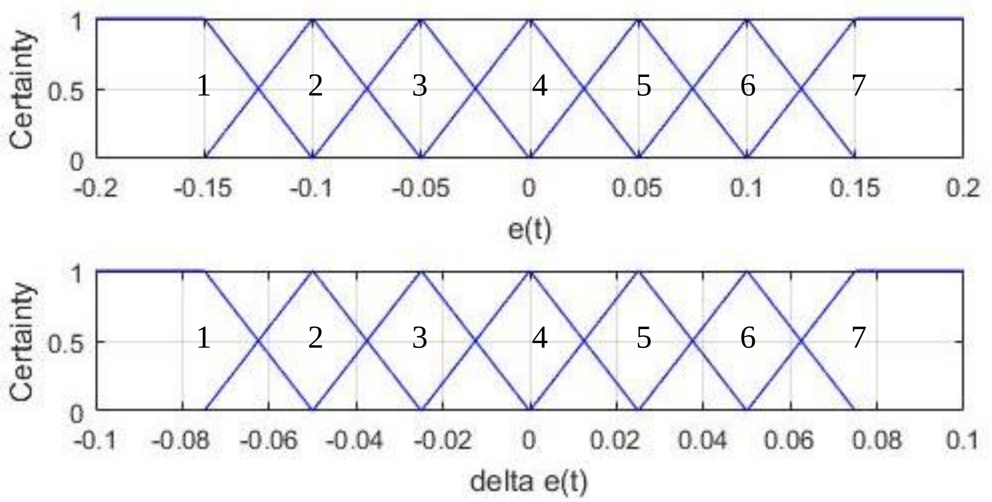


Figure 13: Error and Change in Error Membership Functions

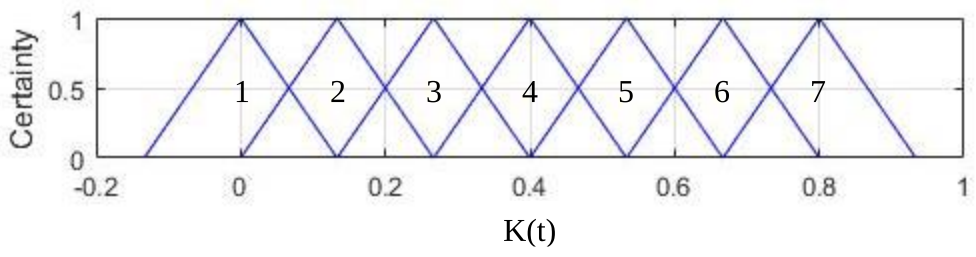


Figure 14: K Output Membership Functions

3.1.4.1.Fuzzification

It is possible for a linguistic variable to be categorized as more than one linguistic value due to the overlapping of membership functions. The plots within Figures 13 and 14 are of a function μ_i versus the numerical values of the linguistic variables. The function μ_i quantifies the certainty, or degree of truth, that a linguistic variable can be classified as a specific linguistic value and can range from 0 to 1. With the numerical value of a linguistic variable possibly falling into one or two membership functions, there will be a certainty value, μ_i, given to each membership function's corresponding linguistic value. A set of numerical values that can be described by μ_i being a distinct linguistic value is called a fuzzy set and is denoted by A_i. Membership functions can define a fuzzy set of A_i for a linguistic variable of u_i in the form of Equation 10 [51].

$$\mu_{A_i}(u_i) = \ certainty \qquad (10)$$

For example, Figure 15 shows an instant in time during the drive cycle when the error input is 0.025 and the change in error is 0.039. For the error input, the line crosses the 4 membership function at 0.5 and crosses the 5 membership function at 0.5. The fuzzification of the input variable says that there is a 50% certainty that the error input is a 4, meaning about "zero" error, and a 50% certainty that error input is a 5, meaning a "small positive" error. For the change in error input, the line crosses the 5 membership function at 0.44 and crosses the 6 membership function at 0.56. The fuzzification of the input variable says that there is a 44% certainty that the change in error input is a 5, meaning a "small positive" change in error, and a 56% certainty that change in error input is a 6, meaning a "medium positive" change in error.

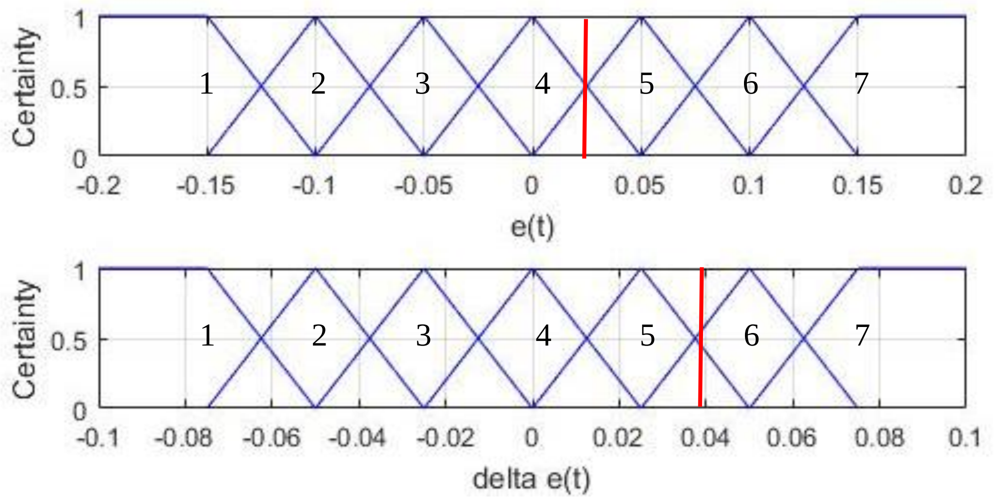


Figure 15: Fuzzification Example

$$\mu_{zero}\big(e(t)\big) = \mu_{small\_pos}\big(e(t)\big) = 0.5$$

$$\mu_{small\_pos}\big(\Delta e(t)\big) = 0.44, \; \mu_{med\_pos}\big(\Delta e(t)\big) = 0.56$$

3.1.4.2. Rule-Bases

Linguistic values can be used to specify a set of rules that capture an expert's knowledge about how to control a system and its dynamics, a rule-base [51]. The rules take the following general form,

If premise, **Then** consequent. (11)

where the premise is associated with the fuzzy linguistic inputs and the consequent is associated with the resulting linguistic output. A generic rule form for two inputs and one output that will be used for the purpose of this study is

If the error is “ ” and the change in error is “ ”,

Then the percentage of ICE torque, K, used is “ ”. (12)

With two inputs and seven linguistic values for each of them, there are at most $7^2 = 49$ possible rules within the fuzzy logic of this study. A tabular representation referred to as a rule table, is used to properly list all possible rules in a convenient way, as shown in Table 8.

		Change in Error						
		1	**2**	**3**	**4**	**5**	**6**	**7**
Error	**1**	7	7	7	7	6	5	4
	2	7	7	7	6	5	4	3
	3	7	7	6	5	4	3	2
	4	7	6	5	4	3	2	1
	5	6	5	4	3	2	1	1
	6	5	4	3	2	1	1	1
	7	4	3	2	1	1	1	1

Table 8: Rule Table

3.1.4.3. Inference Mechanism

The inference mechanism, consisting of two steps, represents the decision-making process of an expert. The first step is referred to as matching, where it is determined which rules apply to the current situation or are considered on. Continuing with the example from the fuzzification step, the following rules are determined to be on:

1. **If** the error is “zero” and the change in error is “small positive”, **Then** the percentage of ICE torque, K, used is “small negative”.

2. **If** the error is “zero” and the change in error is “medium positive”, **Then** the percentage of ICE torque, K, used is “medium negative”.

3. **If** the error is “small positive” and the change in error is “small positive”, **Then** the percentage of ICE torque, K, used is “medium negative”.
4. **If** the error is “small positive” and the change in error is “medium positive”, **Then** the percentage of ICE torque, K, used is “large negative”.

Before the second step, a method for quantifying the overall linguistic premise of each active rule must be agreed upon. The main idea behind this is agreeing on how to quantify the logical “and” operation that combines the individual linguistic input variables into the premise [51]. This value represents the certainty that the rule to which the premise belongs to applies to the current situation and is denoted by $\mu_{premise(i)}$. Common methods for doing this are known as the minimum and the product. The minimum method uses the minimum of the two membership function certainties, while the product method multiplies them together [49]. The minimum method is used for the purpose of this study, as shown in Equation 13.

$$\mu_{premis(i)} = \min\{\mu_{A_i}(e(t)), \mu_{A_i}(\Delta e(t))\} \qquad (13)$$

$$\mu_{premis(1)} = \min\{0.5,0.44\} = 0.44$$
$$\mu_{premis(2)} = \min\{0.5, 0.56\} = 0.5$$
$$\mu_{premis(3)} = \min\{0.5, 0.44\} = 0.44$$
$$\mu_{premis(4)} = \min\{0.5, 0.56\} = 0.5$$

The second step then establishes the conclusion for each active rule. The membership function for the consequent reached by each rule quantifies how certain the rule is that the output variable should take on a certain linguistic value, Equation 14. These membership functions now define the implied fuzzy sets. The justification for using the minimum operator in Equation 14 to quantify the certainty is that we can be no more certain about our consequent than our premise [51].

$$\mu_i\big(K(t)\big) = \min\{\mu_{premis(i)}, \mu_{A_i}\big(K(t)\big)\} \qquad (14)$$

$$\mu_1\big(K(t)\big) = \min\left\{0.44, \mu_{small_{neg}}\big(K(t)\big)\right\} = 0.44$$
$$\mu_2\big(K(t)\big) = \min\left\{0.5, \mu_{med_{neg}}\big(K(t)\big)\right\} = 0.5$$
$$\mu_3\big(K(t)\big) = \min\left\{0.44, \mu_{med_{neg}}\big(K(t)\big)\right\} = 0.44$$
$$\mu_4\big(K(t)\big) = \min\left\{0.5, \mu_{large_{neg}}\big(K(t)\big)\right\} = 0.5$$

3.1.4.4. Defuzzification

The defuzzification process operates on the implied fuzzy sets produced by the inference mechanism and combines their effects to provide the most certain control output μ^{crisp} [51]. For the scope of this study, this output will be achieved using the most popular method, the "center of gravity" (COG) method. The method is defined by Equation 15 as:

$$K = \mu^{crisp} = \frac{\sum_{i=1}^{n} b_i \int \mu_i}{\sum_{i=1}^{n} \int \mu_i} \qquad (15)$$

where b_i is the center of the output membership for the consequent of rule i and $\int \mu_i$ is the area underneath the output membership function "chopped off" at a height of $\mu_i\big(K(t)\big)$ for the consequent of rule i. The calculation gives the K coefficient that will best regulate the ICE torque output so that the desired charge consumption rate and resulting preferred SOC profile can be achieved in the hybrid section.

$$K = \frac{(0.2667 * 0.0915) + (0.1333 * 0.1000) + (0.1333 * 0.0915) + (0 * 0.1000)}{0.0915 + 0.1000 + 0.0915 + 0.1000} = 0.1303$$

3.2. Trajectory Forecasting Testing

3.2.1. Route Data

The data used to test TF is a set of 8 different routes planned out using Google Maps services. Distances are varied between every route, from 6.5 miles to 70 miles. The traffic flow breakup from Google Maps, e.g., route 1 shown in Figure 16, is based on historical and real-time data gathered from traffic sensors and cellphone users. Each Google Maps route is used to portion out the changing traffic sections of the route data for the purpose of testing. Traffic is a rough representation of the speed relative to the speed limit a vehicle is likely to experience when on the road. This usage of traffic information and different speed limits according to road type along the route allows for the use of Table 6, shown earlier, to approximate the speed a vehicle would have at different points along a route. However, using the constant values that each combination in Table 6 provides would result in the vehicle unrealistically traveling at constant speeds during every portioned-off section of the route. To produce more realistic route data, randomized variations are added to the approximated speed values. The variations reduce the speeds presented in Table 6 an additional 0%-5% for each type of traffic. This results in new drive cycles, e.g., shown in Figure 17 derived from Figure 16, that appears to be more realistic as compared to the EPA developed Urban Dynamometer Driving Schedule (UDDS) used for light duty vehicle emissions and fuel economy testing, as shown in Figure 18 [53].

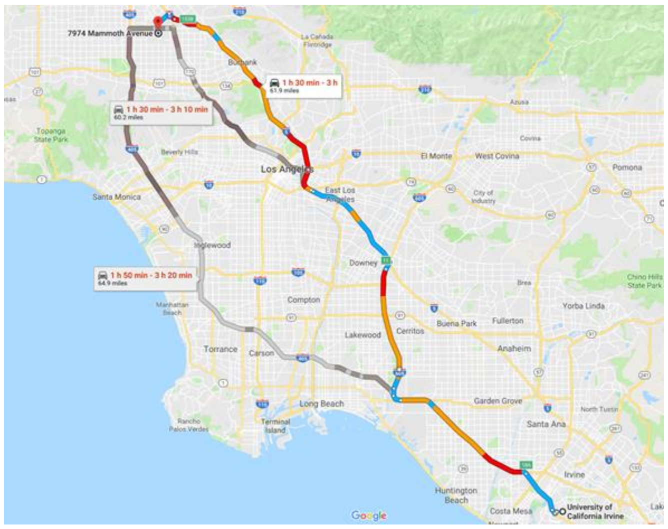


Figure 16: Google Maps Route 1 [46]

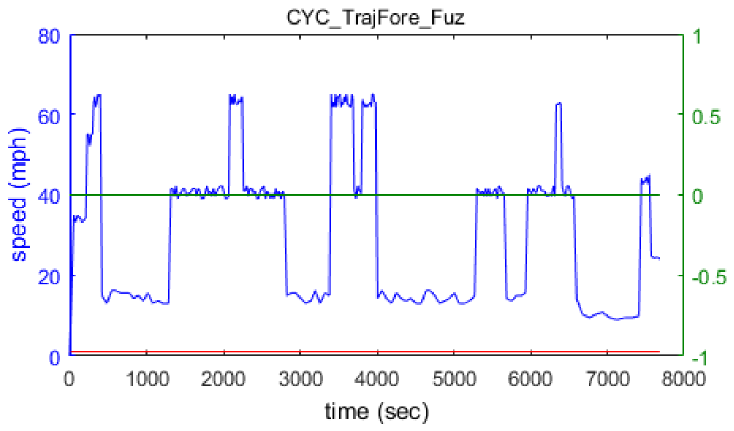


Figure 17: Route 1 Drive Cycle

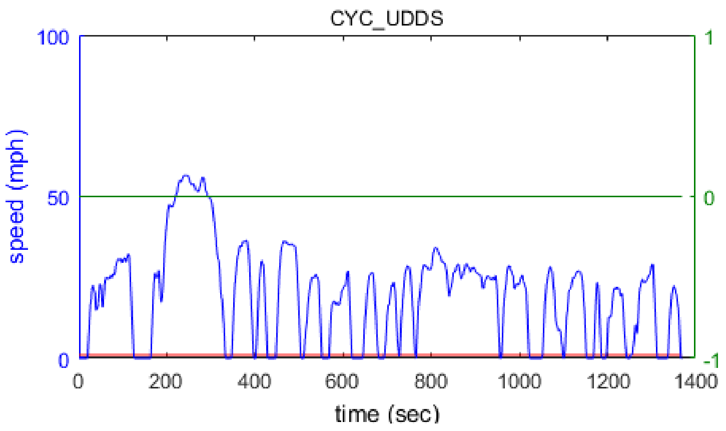


Figure 18: EPA Created UDDS Drive Cycle [53]

3.2.2. Simulation Parameters

The test results for showing the functionality of TF will be for the parallel hybrid PHEV control strategies in ADVISOR, charge depletion (CD) and charge sustaining (CS), and the trajectory forecasting (TF) control algorithm. The numerical values usable for comparing the performance of the three control algorithms are the "Fuel Economy" (FE) value in miles per gallon (mpg) and the "Gas Equivalent" (MPGGE) value in miles per gallon of gasoline equivalent (MPGGE) given in the results window of the ADVISOR simulations. The FE value is calculated using Equation 16 and the MPGGE value is calculated using Equation 17. The MPGGE value gives a better understanding of the PHEV's fuel efficiency because it considers the consumption of both the liquid fuel and electric power source in the vehicle. In instances where the vehicle's route can be driven nearly all in EV mode, with only a small portion left to be driven in ICE or Hybrid mode, the amount of liquid fuel that would be used would significantly smaller numerically than the length of route. These scenarios would then result in very skewed FE values that would seem unrealistic. Thus, the FE value will be ignored, and the MPGGE value will serve as the key value for quantifying and comparing control strategy performance. The parameters of the parallel hybrid control strategies in ADVISOR are defined as shown in Table 9.

$$Fuel\ Economy = \frac{Total\ Route\ Distance}{Total\ Gasoline\ Consumption} \qquad (16)$$

$$Gas\ Equivalent\ = \frac{Total\ Route\ Distance}{Total\ Gasoline\ Consumption + \frac{kWh\ Needed\ to\ Recharge\ to\ Initial\ SOC}{kWh\ per\ Gallon\ of\ Gasoline}} \qquad (17)$$

Parameter	Definition
cs_hi_soc	highest desired SOC, used as initial SOC for every case
cs_lo_soc	lowest desired battery SOC
cs_electric_launch_spd_lo	speed below which vehicle operates as ZEV at low SOC
cs_electric_launch_spd_hi	speed below which vehicle operates as ZEV at low SOC
cs_off_trq_frac	required fraction of max torque when SOC < cs_lo_soc below which engine shuts off
cs_min_trq_frac	torque as a fraction of max torque engine puts out when required is below this value, when SOC < cs_lo_soc
cs_charge_trq	accessory-like torque load on engine that goes to recharging the batteries whenever the engine is on
cs_charge_deplete_bool	charge depleting hybrid strategy flag, 1=> use charge deplete strategy, 0=> use charge sustaining strategy
cs_electric_decel_spd	speed above which no engine shut down occurs due to low torque requests

Table 9: ADVISOR Parallel Hybrid Control Strategy Parameters

The cs_charge_deplete_bool parameter is set to 0 to enable the CS strategy and set to 1 to enable the CD strategy. The parameter of interest for running these simulations is cs_electric_launch_spd_hi. For the CS strategy, the parameter is set to 11.18 m/s (≈ 25.0 mph), shown in Table 10, to allow EV mode to be used when the vehicle is at or below 25 mph and has adequate SOC. This allows the normal parallel hybrid PHEV control strategy to save electrical energy, represented by the level of SOC, at higher speeds where the ICE and hybrid modes are more efficient, leaving more usable SOC for EV mode at low speeds. The CS strategy will result in the PHEV using its battery sparingly, making it available for use during large amounts of the route. Using the CD strategy, the parameter is set to 33.53 m/s (≈ 75.0 mph), shown in Table 10, to allow EV mode to be used when the vehicle is at or below 75 mph and has adequate SOC. The max speed achievable in any of the routes is 75 mph, allowing the PHEV to use the EV mode regardless of speed provided there is adequate SOC. The PHEV will start the beginning a route in EV mode and continue until the vehicle's usable SOC is depleted, sustaining its SOC at the minimum allowable level until the end of the route. The CD strategy will have the PHEV always attempt to use up all its SOC first regardless of current and future route conditions.

For testing the TF control strategy, its ADVISOR parameters, shown in Table 10, will match the parameters used for the CS control strategy in all except the cs_electric_launch_spd_hi and cs_electric_decel_spd. These parameter values will change according to one of the four high efficiency driving scenarios explained earlier in the methodology. The value for cs_charge_deplete_bool will not have any effect on the strategy's performance due to its Simulink additives, therefore it will be left as 0.

Parameter	Values		
	CS	CD	TF
cs_hi_soc	0.8	0.8	0.8
cs_lo_soc	0.25	0.25	0.25
cs_electric_launch_spd_lo	0 m/s	0 m/s	0 m/s
cs_electric_launch_spd_hi	11.18 m/s	33.53 m/s	11.18 m/s (driving scenarios 3 & 4)
			24.59 m/s (driving scenario 2)
			33.53 m/s (driving scenario 1)
cs_off_trq_frac	0.35	0.35	0.35
cs_min_trq_frac	0.48	0.48	0.48
cs_charge_trq	0.25*min(fc_max_trq)	0.25*min(fc_max_trq)	0.25*min(fc_max_trq)
cs_charge_deplete_bool	0	1	0
cs_electric_decel_spd	11 m/s	33 m/s	11 m/s (driving scenarios 3 & 4)
			24 m/s (driving scenario 2)
			33 m/s (driving scenario 1)

Table 10: ADVISOR Parallel Hybrid Control Strategy Parameter Values

3.2.3. TF Performance

The route 1 drive cycle, shown in Figure 19, is 62 miles long with the TF control strategy resulting in the best MPGGE value, 53.3 MPGGE. Its MPGGE value is 4.7% greater than that produced by the CS control strategy, 50.9 MPGGE, and 3.5% greater than that produced by the CD control strategy, 51.5 MPGGE.

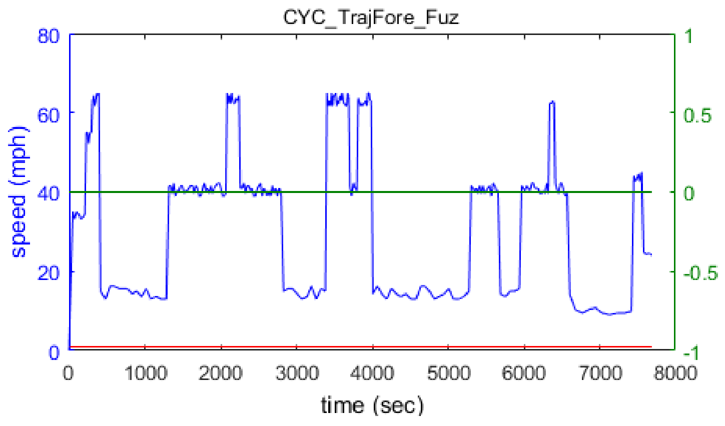


Figure 19: Route 1 Drive Cycle

The route 2 drive cycle, shown in Figure 20, is 65 miles long with the TF control strategy resulting in the best MPGGE value, 52.8 MPGGE. Its MPGGE value is 10.2% greater than that produced by the CS control strategy, 47.9 MPGGE, and 10.2% greater than that produced by the CD control strategy, 47.9 MPGGE.

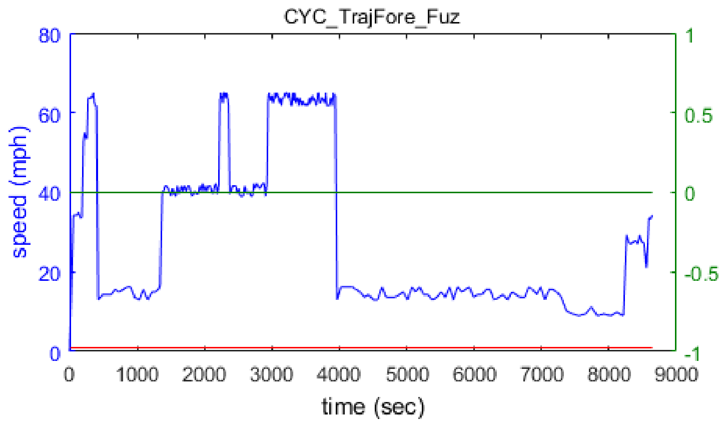


Figure 20: Route 2 Drive Cycle

The route 3 drive cycle, shown in Figure 21, is 70.7 miles long with the TF control strategy resulting in the best MPGGE value, 51.1 MPGGE. Its MPGGE value is 6.2% greater than that produced by the CS control strategy, 48.1 MPGGE, and 8.3% greater than that produced by the CD control strategy, 47.2 MPGGE.

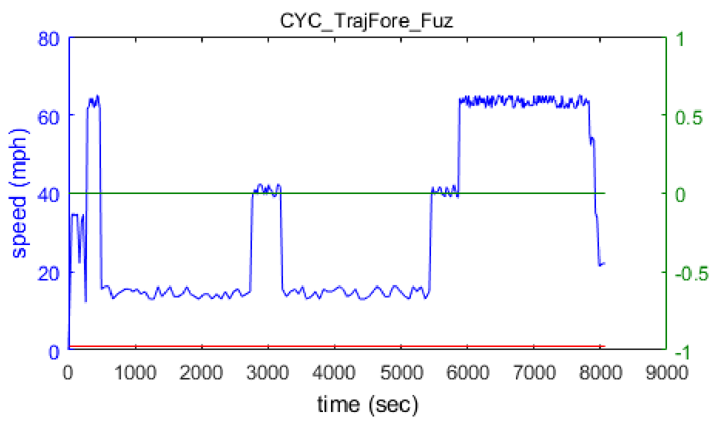


Figure 21: Route 3 Drive Cycle

The route 4 drive cycle, shown in Figure 22, is 52.2 miles long with the TF control strategy resulting in the best MPGGE value, 50.5 MPGGE. Its MPGGE value is 0.60% greater than that produced by the CS control strategy, 50.2 MPGGE, and 2.4% greater than that produced by the CD control strategy, 49.3 MPGGE.

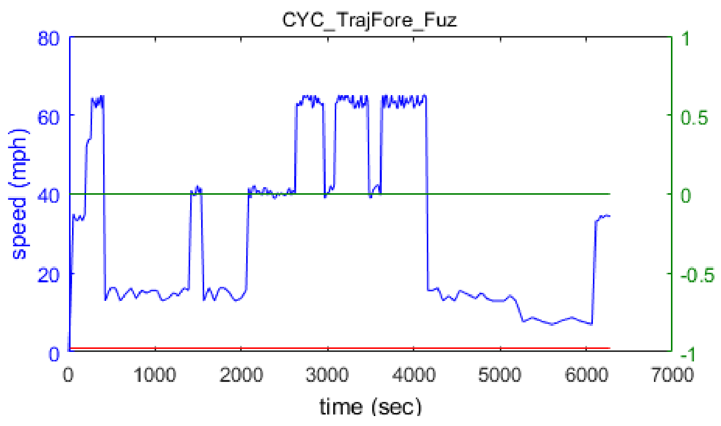


Figure 22: Route 4 Drive Cycle

The route 5 drive cycle, shown in Figure 23, is 20.1 miles long with the TF control strategy resulting in the best MPGGE value, 79.1 MPGGE. Its MPGGE value is 49.81% greater than that produced by the CS control strategy, 52.8 MPGGE, and 10.47% greater than that produced by the CD control strategy, 71.6 MPGGE.

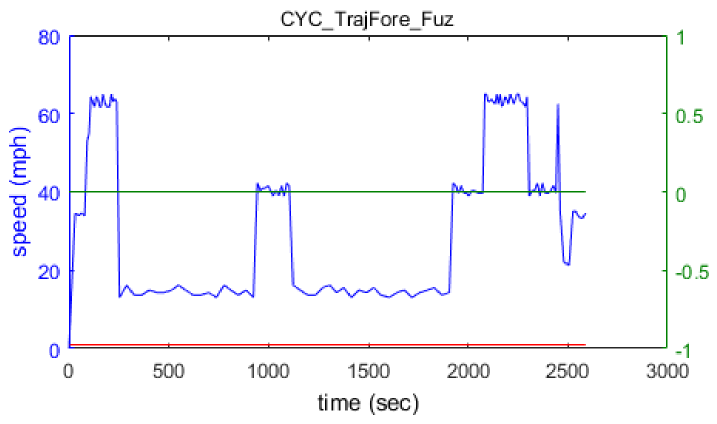


Figure 23: Route 5 Drive Cycle

The route 6 drive cycle, shown in Figure 24, is 40.7 miles long with the TF control strategy resulting in the best MPGGE value, 53.1 MPGGE. Its MPGGE value is 0.19% greater than that produced by the CS control strategy, 53 MPGGE, and 2.7% greater than that produced by the CD control strategy, 51.7 MPGGE.

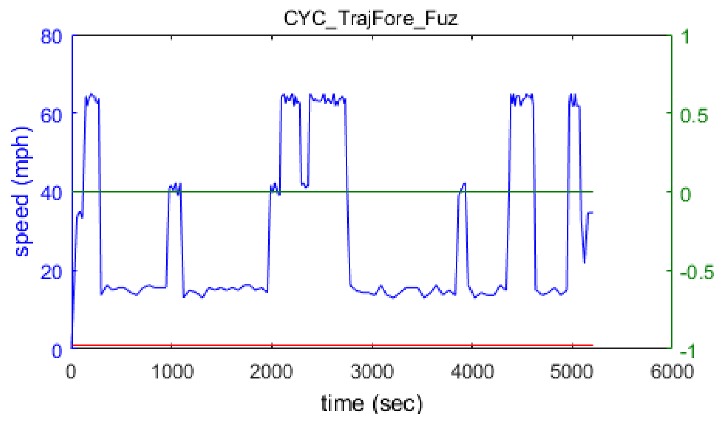


Figure 24: Route 6 Drive Cycle

The route 7 drive cycle, shown in Figure 25, is 17.1 miles long with the TF control strategy resulting in the best MPGGE value, 93.7 MPGGE. Its MPGGE value is 84.45% greater than that produced by the CS control strategy, 50.8 MPGGE, and 41.98% greater than that produced by the CD control strategy, 66 MPGGE.

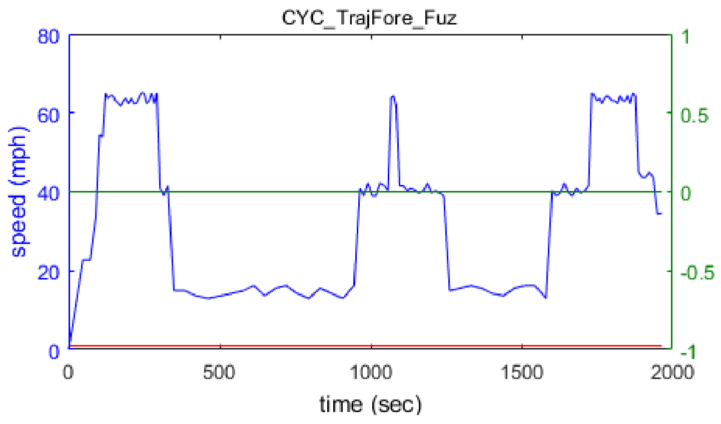


Figure 25: Route 7 Drive Cycle

The route 8 drive cycle, shown in Figure 26, is 6.5 miles long with the TF control strategy resulting in the best MPGGE value, 88.3 MPGGE. Its MPGGE value is 105.3% greater than that produced by the CS control strategy, 43 MPGGE, and 87.9% greater than that produced by the CD control strategy, 47 MPGGE.

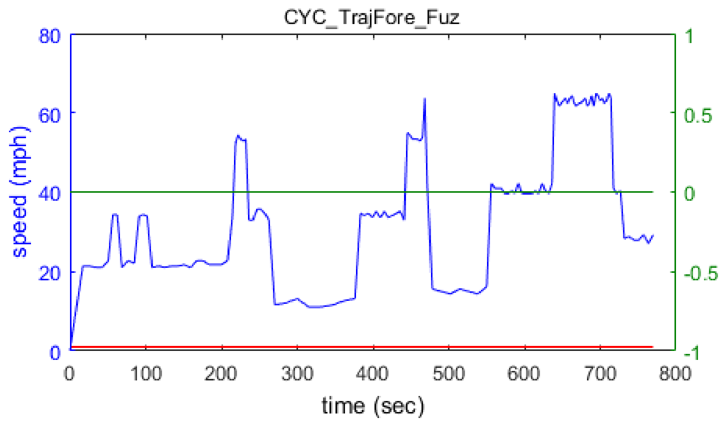


Figure 26: Route 8 Drive Cycle

Route	Control Strategy	Gas Equivalent (MPGGE)	Distance (miles)
1	CS	50.9	62
	CD	51.5	
	TF	53.3	
2	CS	47.9	65
	CD	47.9	
	TF	52.8	
3	CS	48.1	70.7
	CD	47.2	
	TF	51.1	
4	CS	50.2	52.2
	CD	49.3	
	TF	50.5	
5	CS	52.8	20.1
	CD	71.6	
	TF	79.1	
6	CS	53	40.7
	CD	51.7	
	TF	53.1	
7	CS	50.8	17.1
	CD	66	
	TF	93.7	

Table 11: ADVISOR MPGGE Results

Figures showing the ADVISOR results window for each route simulation for each of the three control strategies can be found in the appendix section of this study. These figures will show graphs of the vehicle's drive cycle, SOC profile, emissions output, and actual torque output of the engine and transmission. In all the routes tested, the TF-based control algorithm achieved similar or greater fuel economy. This is important because PHEV fuel economy enhancements can lead to a tremendous reduction in fuel consumption for the nation and possibly shorter payback time for customers in terms of vehicle investment [36]. The TF-based control strategy proved to be the most beneficial in routes 2, 3, 5, 7, and 8. In routes 1, 4, and 6, the TF-based control strategy gave results that were only slightly better than the other control strategies.

3.3. Convolution Neural Network

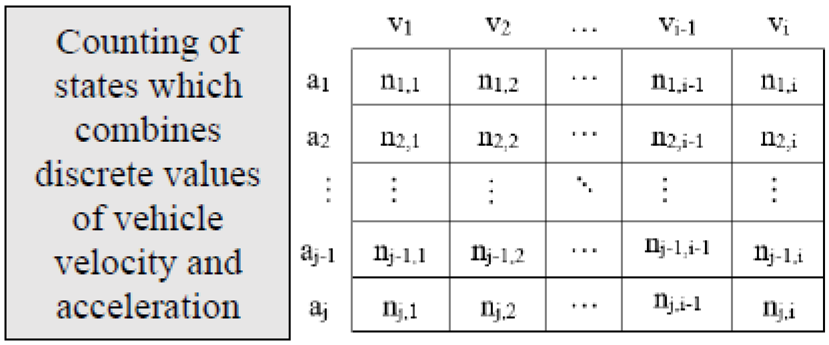
Furthering advancing the application of the TF algorithm, a TF Convolutional Neural Network (TFCNN) machine learning algorithm is created to complete this research's control strategy. A CNN is a type of deep learning machine learning algorithm that can assign importance to different aspects within an image and learn to differentiate them from one another [54]. This type of machine learning algorithm is especially useful for the purpose of this research since route data can treated and analyzed as images when processed correctly. This study will extend the work of Topić [44] by using his developed CNN architecture and input feature preprocessing strategy as a starting point for preprocessing the drive cycle and TF setup variables to estimate to greenhouse gas emissions (GGE) and MPGGE of the PHEV. Once properly trained, this machine learning algorithm can allow for computationally fast and inexpensive calculations that can be used for any size route due to its powerful generalization capabilities. This makes it possible for the vehicle to automatically assess and switch to new more efficient routes in real-time during travel.

3.3.1. CNN Input Data Preprocessing

To train, validate, and test the CNN, 8750 drive cycles containing both velocity and acceleration data collected by the NREL using in-vehicle GPS devices in its 2010-2012 California Household Travel Survey [55] are processed and used as inputs to the TFCNN. The drive cycles are made up of an equal 2750 routes from three different distance ranges, less than 10 miles, between 10 and 30 miles, and greater than 30 miles. This equal distribution of route distance ranges is done to remove possible bias in performance due to route distance, ensuring the CNN can perform consistently across all routes. Each drive cycle will be driven using the TF algorithm within ADVISOR, and have their resulting SOC schedule plot, mpg gas equivalent (MPGGE),

GGE (HC, CO, and NOx) in g/mile, and used gal. of gasoline recorded. Designed as a tool for rapid analysis of performance and fuel economy of conventional, electric, and hybrid vehicles that many companies such as Ford Motor Company and General Motors Corp. use [47], we can confidently use its MPGGE and GGE outputs as truth values to use for training the CNN.

To ensure the use of a single CNN that can take a route of varying length of time and distance as an input, there is a need to preprocess route data into a statically sized input. Thus, the preprocessing method for the TFCNN will be based off of the method developed by Topić [44]. Topić takes the expected speed and acceleration a vehicle will face and transforms it into a statically sized 2D matrix, as shown in Figure 27, with columns representing the range of speed as discrete values and the rows representing the range of acceleration as discrete values. Once the



Counting of states which combines discrete values of vehicle velocity and acceleration

$$\begin{array}{c|ccccc} & v_1 & v_2 & \cdots & v_{i-1} & v_i \\ \hline a_1 & n_{1,1} & n_{1,2} & \cdots & n_{1,i-1} & n_{1,i} \\ a_2 & n_{2,1} & n_{2,2} & \cdots & n_{2,i-1} & n_{2,i} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{j-1} & n_{j-1,1} & n_{j-1,2} & \cdots & n_{j-1,i-1} & n_{j-1,i} \\ a_j & n_{j,1} & n_{j,2} & \cdots & n_{j,i-1} & n_{j,i} \end{array}$$

Figure 27: Topić’s Preprocessing 2D Matrix

matrix is formed, he then incorporates the initial SOC value of the PHEV by adding it to the elements of the matrix. Building off this, the TFCNN input will be of the same format for its rows, columns, and incorporation of initial SOC, but with 3 layers. This will make the input a 3D matrix with each layer representing the following:

- Layer 1 – count of the number of times the PHEV uses ICE mode at a specific combination of discrete speed and acceleration values

- Layer 2 – sum of percentage of lacking SOC needed to run each hybrid mode section, *i*, in EV mode

$$\sum_{i=1}^{n} 1 - \frac{Allotted\_SOC(i)}{Required\_SOC(i)} \qquad (18)$$

- Layer 3 – count of the number of times the PHEV uses EV mod at a specific combination of discrete speed and acceleration values

However, it is important to note that when attempting to determine any one of the vehicle's well-to-wheel emissions as an output from the TFCNN, layer 3 can be omitted from the 3D input matrix. The reason is that it is the consumption of gasoline during the vehicle's use of ICE and hybrid mode that contributes to the creation of emissions. All three layers will still be needed for the 3D input matrix when attempting to determine the vehicle's MPGGE, as it depends on the consumption of both gasoline and battery charge.

The range of speed and acceleration used for setting the row and column dimensions of the input matrix are determined by first extracting the maximum and minimum values the vehicle sees throughout the 8250 drive cycles used in the training data. These lower and upper boundaries are then rounded to a desired resolution, with the columns of the matrix corresponding to the values from the minimum speed to maximum speed in intervals of the desired speed resolution, and the rows of the matrix corresponding to the values from the minimum acceleration to the maximum acceleration in intervals of the desired acceleration resolution. The resulting range of speed and acceleration, desired resolutions, and input matrix dimensions are shown in Table 12 and Table 13. If the speed and acceleration value the vehicle experiences exceed the limits extracted from the training data, they will be counted as the limit values.

Dimension	Speed (mph)		Acceleration (m/s^2)		Drive Modes
	Range	Resolution	Range	Resolution	
107 x 201 x 3	[0,100]	0.5	[-5.3,5.3]	0.1	3

Table 12: Input Matrix Parameters for MPGGE Output

Dimension	Speed (mph)		Acceleration (m/s^2)		Drive Modes
	Range	Resolution	Range	Resolution	
107 x 201 x 2	[0,100]	0.5	[-5.3,5.3]	0.1	2

Table 13: Input Matrix Parameters for Emission Outputs

3.3.2. CNN Architecture

Looking at the preprocessed matrix input, they can be considered informational images containing important features of a PHEV’s EV, hybrid, and ICE mode usage along a route. Thus, a CNN type architecture can give the most accurate learning results when paired with these inputs, as they have been proven to be very successful in image classification due to its effectiveness in automatic feature extraction [44]. Using the CNN architecture developed by Topić as a starting point, extensive testing was used to develop the working architecture shown in Figure 28. The CNN is made up of a mixture of two-dimensional convolutional layers characterized by (# of filters)@(filter size), max pooling layers characterized by (filter dimension)@(filter size), a flattening layer, fully connected layers characterized by (# of nodes), and activation layers.

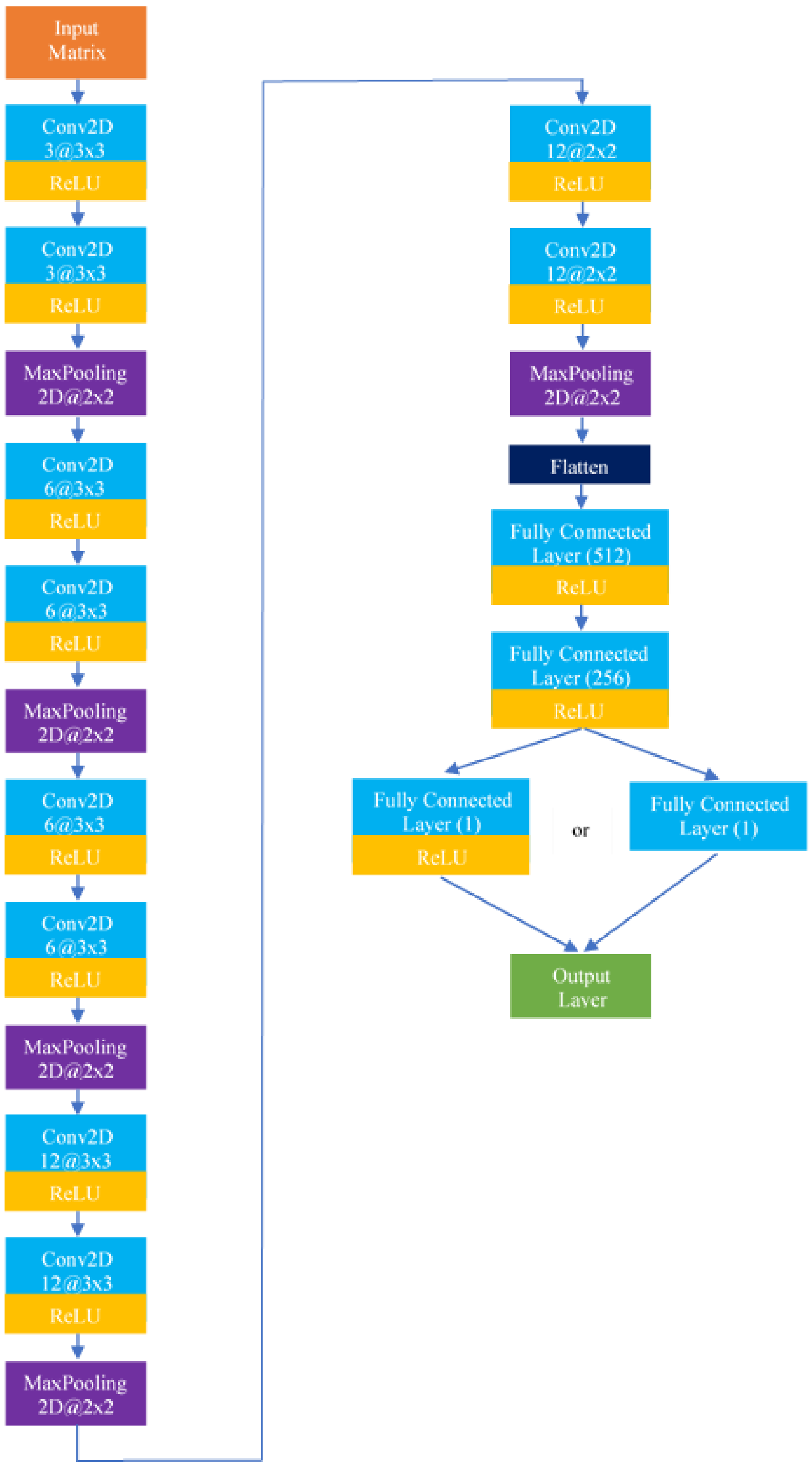


Figure 28: CNN Architecture

The convolutional layers work to extract features from images by using a set of weights known as a convolutional filter, K_i, to slide over inputted data, X_i, and generate a filtered version, Z_i, for a given layer, n. This filtered version of the image data is a feature map. Different convolutional filters extract different features, and it is the combination of the resulting feature maps that helps power the CNN's predictions [56]. An example of the convolutional layer process is shown in Figure 29, with Equation 19 showing the process being akin to simple matrix

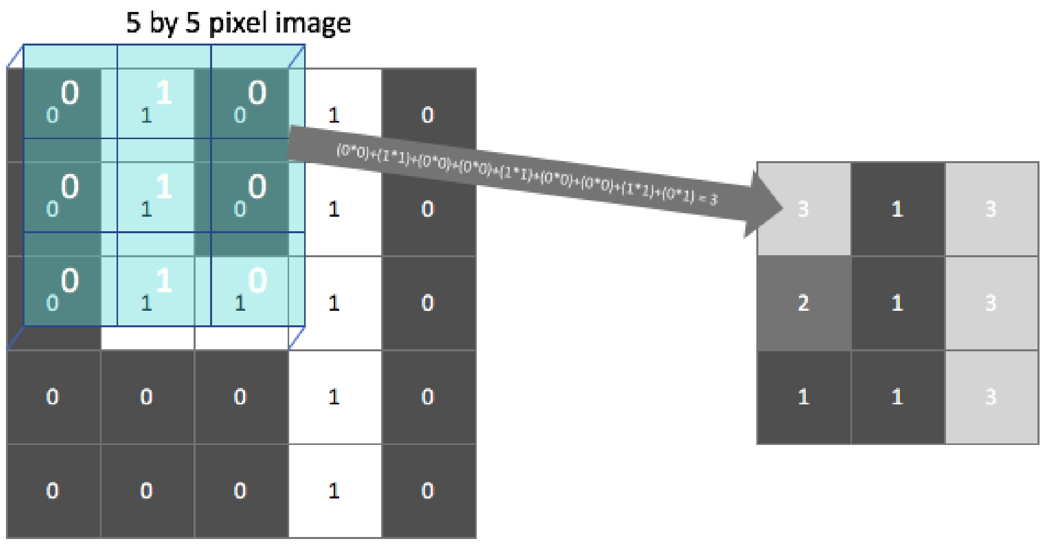


Figure 29: Convolutional Layer Calculation Example [57]

$$Z_i = X_i * K_i \quad (19)$$

multiplication. Enhancing the feature maps obtained from the convolutional layers, activation functions, g, are used to determine what node signals are sent forward through the network and to what extent [56]. The activation function chosen for this research's CNN is a non-linear activation function known as ReLU, rectifier linear unit. It prevents negative numbers from being passed forward by following Equation 20, x being an individual element, resulting in a layer's final processed output and the following layer's input, Equation 21.

$$\mathrm{ReLU}(\mathrm{x}) = \max(0, x) \qquad (20)$$

$$X_{i+1} = g_i(Z_i) \qquad (21)$$

With inputs to CNNs being so large and the number of filters chosen for convolutional layers possibly increasing the number of parameters through feature extraction, the issue of compression and increasing processing power come into play. To help remedy this, pooling layers are used to help reduce feature map size without loss of information [56]. This is especially helpful as it reduces the amount of processing time and power needed during the training stage of CNN development. The pooling method chosen for this research's CNN is the max pooling method, shown in Figure 30 and Equation 22. Max pooling is commonly chosen because of how efficient it is at maintaining features [56].

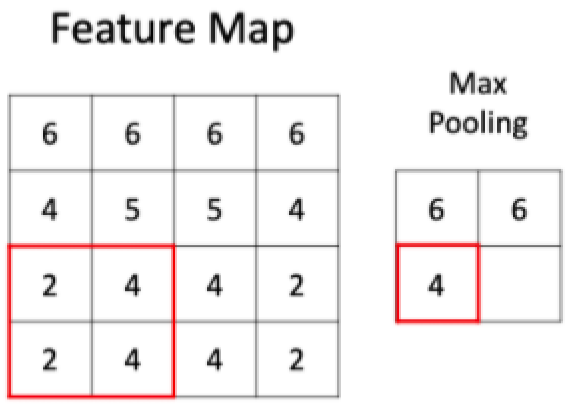


Figure 30: Max Pooling, 2D with 2x2 Filter

$$X_{i+1} = maxpool(X_i) \qquad (22)$$

Once processed through a series of alternating convolutional layers, activation functions, and pooling layers, the resulting data is flattened into a vector and fed through a series of fully connected layers to be able to produce the regression outputs needed for the control strategy. ReLU activation functions are also used to enhance the data outputted by each fully connected layer. Each fully connected layer and accompanying activation can be described by Equation 23, where X_i is that layer's input data and θ_i is its set of weights by which the input data is multiplied. Depending on the output being solved for, however, a ReLU activation function may or may not be needed for the output of the last fully connected layer of the CNN architecture, Y_{net}. This is determined through trial and error.

$$X_{i+1} = g_i({\theta_i}^T X_i) \qquad (23)$$

$$Y_{net} = g({\theta_l}^T X_l) \qquad (24)$$

The cost function, J, being optimized throughout the training of the CNN is the mean square error between the network's output Y_{net} (MPGGE, g/mile HC, g/mile CO, g/mile NOx, or gal. of gasoline used) and the desired outputs Y_{truth} for all n sets of training date as shown:

$$J = \frac{1}{2n}\sum_{i=1}^{n}(Y_{net,i} - Y_{truth,i})^2 \qquad (25)$$

To train the CNN, the adaptive moment estimation optimization algorithm for stochastic gradient descent known as Adam is used during the process of backpropagation [58]. In this process, Adam optimizes and updates the CNN's learnable parameters, θ, the convolutional layers' and fully connected layers' weights, in a backwards fashion from the last layer to the first layer using adaptive learning rates and moment estimation. This is shown in Equation 26, where a current time step's weights, θ_{t+1}, is calculated from the previous time step, θ_t, and an update vector made up of a bias-corrected exponentially decaying average of past gradients and past squared gradients, $\hat{m}_t$ and $\hat{v}_t$, taken of the cost function in respect to the learnable parameters [59].

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \qquad (26)$$

Suggestions for the learning rate, η, and smoothing term, ϵ, can be found in [58]. The 8250 drive cycles used to develop the TFCNN are split using a 68/17/15 ratio, where 68% of the drive cycles are allocated for training, 17% for validation, and 15% for testing purposes. For more information regarding the intricacies and math behind neural networks, a detailed explanation can be found in [60].

3.3.3. CNN Performance

A TFCNN for each of the 5 possible outputs (MPGGE, HC, CO, NOx, or gal. of gasoline) for four different vehicles is created and trained. Each of the four vehicles uses the same ADVISOR vehicle configuration previously described in Table 1, but with a specific combination of fuel converter (FC) and energy storage system (ESS) as shown in Table 14. This results in a total of 20 TFCNNs, all trained according to the machine learning training options within MATLAB shown in Table 15. To show their performance, the average percent accuracy across each TFCNN's test data set is taken between the obtained truth output values from ADVISOR and the corresponding network output, as shown in Table 16.

Fuel Converter	FC_SI41_emis	1991 Geo Metro 1.0L SI engine with maximum power of 41 kW @ 5700 rpm and peak torque of 81 Nm @ 3477 rpm
	FC_SIPrime_emis	2017 Prius Prime 1.8L 4 cylinder engine with maximum power of 71 kW @ 5200 rpm and peak torque of 142 Nm @ 3600 rpm
Energy Storage System	ESS_PB25	Hawker Genesis 12V 25Ah 10EP sealed valve-regulated lead-acid (VRLA) battery
	ESS_PB28	Johnson Controls 12-95 lead-acid battery

Table 14: ADVISOR PHEV Fuel Converter and Energy Storage Options [47]

Optimization Solver	Adam	Solver for training network
Mini Batch Size	50	Size of the mini batch to use for each training iteration
Max Epochs	1000	Maximum number of epochs
Initial Learning Rate	0.0001	Initial learning rate for training
Shuffle	Every-Epoch	Frequency of training data shuffling
Validation Frequency	25	Frequency of network validation in # of iterations
Validation Patience	30	Number of times the loss on the validation set can be larger than or equal to the previously smallest loss

Table 15: MATLAB Machine Learning Training Options [61]

FC/ESS Combination	MPGGE	HC	CO	NOx	Gal Gas Used
FC_SI41_emis/ ESS_PB25	96.22%	91.94%	73.62%	87.57%	81.73%
FC_SI41_emis/ ESS_PB28	95.64%	89.89%	73.05%	84.66%	80.34%
FC_SIPrime_emis/ ESS_PB25	95.63%	90.49%	73.29%	88.05%	84.15%
FC_SIPrime_emis/ ESS_PB28	94.93%	87.77%	68.20%	84.23%	80.64%

Table 16: TFCNN’s Test Data Average Accuracy

When looking at the data, it becomes apparent that for each type of fuel converter, a higher average accuracy is produced when paired with the larger energy storage system. CO also consistently produces the lowest average accuracy across all four vehicles. To increase this performance, more research can be conducted relative to fine-tuning the preprocessing methodology and improving the efficiency of CO's network with different optimization strategies. Regardless, the current levels of average accuracy the TFCNNs can achieve for the various outputs show that a machine learning algorithm has the capability to learn the complex dynamics of a PHEV and its control algorithms. It can perform this feat as an input/output calculation that is as simple as a linear equation. Thus, the overall goal of this research to develop a working globally oriented control strategy that automatically chooses and optimizes the best choice according to MPGGE and/or emissions for the driver while being scalable, computationally fast, and adaptable to changing route data in real-time becomes plausible.

3.4. Implementation

With the Trajectory Forecasting and machine learning control algorithms properly performing, the implementation of the complete Trajectory Forecasting based machine learning control strategy can be broken down into the following:

1. Take in a priori drive cycle data (speed and acceleration) of multiple available routes going from the driver's starting point to their desired end point
2. Create a drive-mode and SOC discharge schedule for each possible route using the TF control algorithm
3. Preprocess the drive cycle (speed and acceleration) and drive-mode schedule of each available route along with the current initial SOC into the input matrix for the chosen TFCNN

4. Plug-in input matrix for each available route into the TFCNN to approximate the value of interest (MPGGE, g/mile HC, g/mile CO, g/mile NOx, or gal. of gasoline used)
5. Choose the route that best fits the driver's request, e.g. higher MPGGE, lower emissions, lower gas usage, as ideal choice
6. Implement drive-mode and SOC discharge schedule derived for a chosen route from TF control algorithm during travel in real-time
7. Repeatedly check updated route data, rerunning control strategy from step 1 if changes are detected, and changing routes if the current one is no longer the ideal choice

4. Simulation

To evaluate the performance of the complete Trajectory Forecasting based machine learning control strategy, ADVISOR will be used to simulate its usage with the four different PHEV vehicle configurations used in Section 3.3.3. For the simulation, there are two important details to consider, route updates experienced while driving and a PHEV's starting SOC. Route updates can happen at random times along a route in real life, they will instead be preset to happen after driving specific amounts of distance in the simulations. Routes where the distance is 30 miles or less will encounter a single update after traveling a distance equal to 20% of the original route distance, and routes greater than 30 miles will encounter two updates, one after traveling a distance equal to 20% of the original route distance and another at 40%. This is done to properly extract the vehicle's SOC at the time of updates and set it as the new initial SOC value to use in the input matrix when using the control strategy. The updates of traffic at the preset distances are also chosen randomly at the time of the simulations' execution, with traffic having the ability to improve, worsen, or stay relatively the same at each update. For the starting SOC of the PHEV's, the value will vary according to Table 17.

Distance (miles)	Initial SOC
$x < 5$	0.4
$5 \leq x < 10$	0.5
$10 \leq x < 25$	0.6
$25 \leq x < 40$	0.7
$x \geq 40$	0.8

Table 17: Distance Based Starting Initial SOC

4.1. Route Data

Similar to the eight routes developed to test the performance of the TF control algorithm in Section 3.2.1, six different routes of distances from 10-60 miles in increments of 10 miles are created to test the performance of the research's Trajectory Forecasting based machine learning control strategy. They will serve as the original routes each vehicle will drive along before any updates are seen. The routes use the same speed limit and traffic conditions listed in Table 6 to derive the approximated speeds the PHEVs will experience. As mentioned in Section 3.2.1, assuming the speeds stay constant wherever the speed limit and traffic conditions are the same would result in the vehicle unrealistically traveling at constant speeds during various sections of the routes. To counteract this, additional random drops in speed ranging from 0%-5% are added to all approximated speed values. The largest of the original routes, route 6 of 60 miles, can be seen in Figure 31 below, with the others available in Section 7.3.1 of the Appendix. An example of how a route looks after a traffic update is applied, simulating changing traffic conditions in the original route and alternative route choices, is shown in Figure 32. To assure a more consistent and fair analysis across all control strategies used in performance comparisons, every route driven by each control strategy will always result in a distance equal to that of the original route it is derived from.

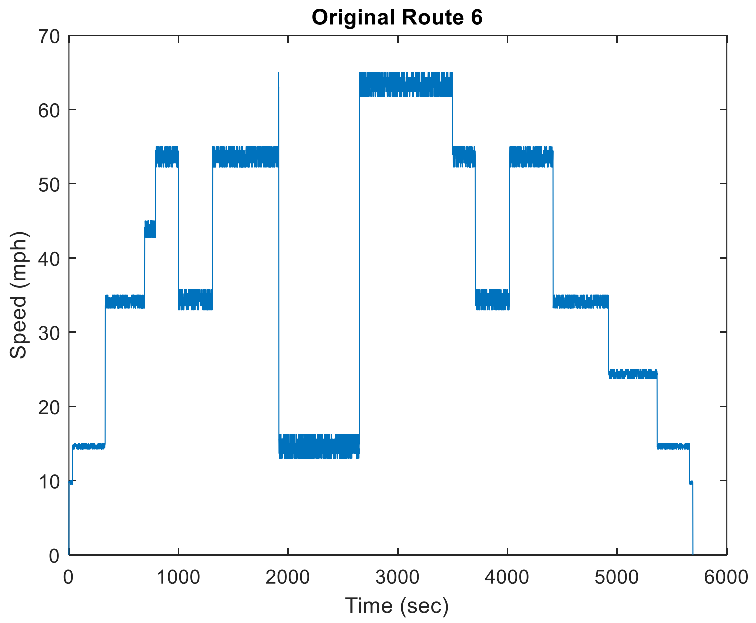


Figure 31: Original Route 6, 60 miles

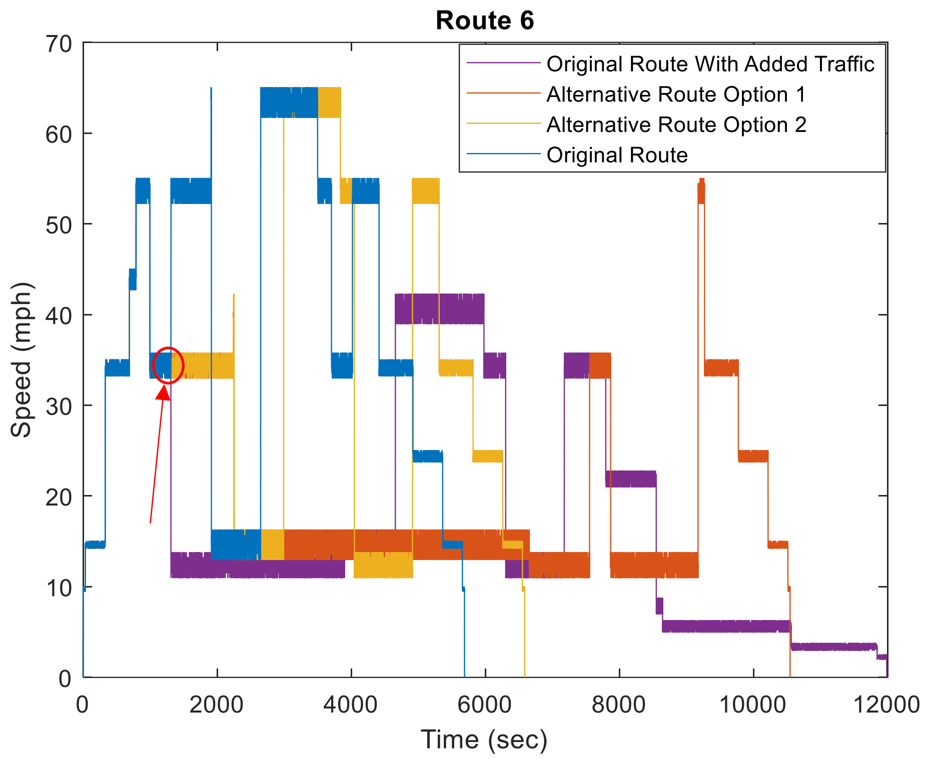


Figure 32: Original route 6 compared to route 6 with added traffic and alternate route choices after first update at identified point circled in red

4.2. Simulation Parameters

The results shown in this study show the performance of the TF based machine learning control strategy in comparison to the default PHEV control strategies, charge depletion (CD) and charge sustaining (CS), included within ADVISOR, and commonly used is PHEV now. The parameters used to compare the strategies are the MPGGE, g/mile HC, g/mile CO, g/mile NOx, and total gal. of gasoline used. For the CD and CS strategies, they will be used in ADVISOR on the original routes with the added traffic from the route updates. The comparison values are then simply taken from the ADVISOR simulation results. To obtain the comparison values for the TF based machine learning control strategy, however, each of the ideal route options chosen by the strategy in going from point A to B are individually run in ADVISOR using the TF control algorithm. The raw ADVISOR data for each chosen route option, grams of HC, grams of CO, grams of NOx, and gal. of gasoline used, is then taken, and used in equations 27-29 to obtain the desired comparison values. In the equations, n is the number of chosen route options, E is the grams of an emission emitted during a chosen route option, and G is the gallons of gasoline consumed during a chosen route option. For a detailed explanation of the parameters within ADVISOR that enable to the use of CD, CS, and TF within a PHEV, refer to Section 3.2.2.

$$Total\ Emmisions = \frac{\sum_{i=1}^{n} E_i}{Total\ Route\ Distance} \qquad (27)$$

$$Total\ Gasoline\ Consumption = \sum_{i=1}^{n} G_i \qquad (28)$$

$$Mpgge = \frac{Total\ Route\ Distance}{Total\ Gasoline\ Consumption + \frac{kWh\ Needed\ to\ Recharge\ to\ Initial\ SOC}{kWh\ per\ Gallon\ of\ Gasoline}} \qquad (29)$$

4.3. Results

For the results displayed below for each PHEV, a strategy proves to be better regarding optimizing MPGGE when its value is greater than that of the other strategies. This means we desire a positive percent difference between the TF based machine learning control strategy and the other strategies. Regarding optimizing emissions and gasoline consumption, a strategy is better if its value is less than that of the other strategies. For these outputs, we desire a negative percent difference between the TF based machine learning control strategy and the other strategies. The best raw output between the three strategies for each output type for each route is highlighted in yellow within the raw output result tables.

4.3.1. PHEV #1, FC_SI41_emis/ ESS_PB25 Combination

	Strategy	MPGGE	HC (g/mile)	CO (g/mile)	NOx (g/mile)	Gas (gal)
Route 1	CD	56.687	0.495	3.780	0.390	0.102
	CS	50.877	0.476	4.542	0.380	0.101
	TF CNN	60.431	0.413	1.388	0.244	0.095
Route 2	CD	33.433	0.355	2.772	0.251	0.465
	CS	31.840	0.385	2.793	0.247	0.460
	TF CNN	39.173	0.286	1.861	0.228	0.338
Route 3	CD	27.654	0.321	2.383	0.310	0.892
	CS	27.732	0.319	2.979	0.288	0.908
	TF CNN	43.965	0.298	2.632	0.205	0.612
Route 4	CD	32.245	0.266	1.357	0.304	1.126
	CS	32.592	0.281	2.514	0.318	1.127
	TF CNN	40.062	0.332	3.759	0.224	0.805
Route 5	CD	28.854	0.280	1.505	0.341	1.375
	CS	29.363	0.282	1.712	0.328	1.358
	TF CNN	32.640	0.314	1.956	0.259	1.170
Route 6	CD	34.728	0.246	1.825	0.245	1.760
	CS	34.173	0.249	2.031	0.243	1.726
	TF CNN	36.215	0.282	3.064	0.233	1.565

Table 18: PHEV #1, Raw Output Results for all Routes and Strategies

	Comparison	MPGGE % Diff	HC % Diff	CO % Diff	NOx % Diff	Gas % Diff
Route 1	TF CNN vs CD	6.60	-16.58	-63.27	-37.36	-6.88
	TF CNN vs CS	18.78	-13.21	-69.44	-35.74	-6.57
Route 2	TF CNN vs CD	17.17	-19.68	-32.87	-9.21	-27.27
	TF CNN vs CS	23.03	-25.75	-33.36	-7.79	-26.54
Route 3	TF CNN vs CD	58.98	-7.34	10.48	-33.82	-31.37
	TF CNN vs CS	58.53	-6.62	-11.65	-28.95	-32.54
Route 4	TF CNN vs CD	24.24	24.57	177.04	-26.29	-28.51
	TF CNN vs CS	22.92	18.12	49.51	-29.68	-28.56
Route 5	TF CNN vs CD	13.12	12.21	29.95	-24.14	-14.92
	TF CNN vs CS	11.16	11.51	14.24	-21.11	-13.81
Route 6	TF CNN vs CD	4.28	14.65	67.86	-5.21	-11.10
	TF CNN vs CS	5.97	13.27	50.88	-4.43	-9.33

Table 19: PHEV #1, Percent Difference Between Strategies

Comparison Strategy	Performance	MPGGE % Diff	HC % Diff	CO % Diff	NOx % Diff	Gas % Diff
CD	Best	58.98	-19.68	-63.27	-37.36	-31.37
	Worst	4.28	24.57	177.04	-5.21	-6.88
CS	Best	58.53	-25.75	-69.44	-35.74	-32.54
	Worst	5.97	18.12	50.88	-4.43	-6.57

Table 20: PHEV #1, Best and Worst Performance

4.3.2. PHEV #2, FC_SI41_emis/ ESS_PB28 Combination

	Strategy	MPGGE	HC (g/mile)	CO (g/mile)	NOx (g/mile)	Gas (gal)
Route 1	CD	47.038	0.459	1.260	0.294	0.027
	CS	46.813	0.504	3.713	0.323	0.110
	TF CNN	75.344	0.459	1.251	0.302	0.025
Route 2	CD	31.430	0.339	2.585	0.355	0.369
	CS	32.064	0.381	2.718	0.382	0.362
	TF CNN	38.370	0.260	1.690	0.178	0.329
Route 3	CD	39.160	0.317	2.115	0.339	0.866
	CS	38.538	0.335	2.168	0.371	0.823
	TF CNN	40.489	0.274	2.079	0.190	0.575
Route 4	CD	31.195	0.253	2.233	0.277	1.058
	CS	31.098	0.266	2.363	0.315	1.082
	TF CNN	41.796	0.324	2.208	0.182	0.840
Route 5	CD	27.468	0.259	1.180	0.314	1.310
	CS	27.850	0.263	1.912	0.307	1.335
	TF CNN	38.529	0.308	1.791	0.284	1.210
Route 6	CD	32.423	0.241	2.223	0.270	1.750
	CS	33.268	0.247	1.725	0.278	1.748
	TF CNN	37.127	0.276	3.628	0.224	1.533

Table 21: PHEV #2, Raw Output Results for all Routes and Strategies

	Comparison	MPGGE % Diff	HC % Diff	CO % Diff	NOx % Diff	Gas % Diff
Route 1	TF CNN vs CD	60.18	-0.05	-0.71	2.67	-7.44
	TF CNN vs CS	60.95	-8.97	-66.31	-6.53	-77.72
Route 2	TF CNN vs CD	22.08	-23.36	-34.64	-49.78	-10.72
	TF CNN vs CS	19.67	-31.92	-37.83	-53.35	-8.94
Route 3	TF CNN vs CD	3.40	-13.58	-1.68	-43.94	-33.60
	TF CNN vs CS	5.06	-18.38	-4.09	-48.75	-30.10
Route 4	TF CNN vs CD	33.98	28.13	-1.08	-34.32	-20.62
	TF CNN vs CS	34.40	21.79	-6.56	-42.43	-22.40
Route 5	TF CNN vs CD	40.27	18.97	51.80	-9.35	-7.70
	TF CNN vs CS	38.34	17.31	-6.36	-7.32	-9.42
Route 6	TF CNN vs CD	14.51	14.53	63.19	-17.09	-12.38
	TF CNN vs CS	11.60	11.88	110.36	-19.51	-12.30

Table 22: PHEV #2, Percent Difference Between Strategies

Comparison Strategy	Performance	MPGGE % Diff	HC % Diff	CO % Diff	NOx % Diff	Gas % Diff
CD	Best	60.18	-23.36	-34.64	-49.78	-33.60
	Worst	3.40	28.13	63.19	2.67	-7.44
CS	Best	60.95	-31.92	-66.31	-53.35	-77.72
	Worst	5.06	21.79	110.36	-6.53	-8.94

Table 23: PHEV #2, Best and Worst Performance

4.3.3. PHEV #3, FC_SIPrime_emis/ ESS_PB25 Combination

	Strategy	MPGGE	HC (g/mile)	CO (g/mile)	NOx (g/mile)	Gas (gal)
Route 1	CD	42.733	0.683	3.277	0.420	0.157
	CS	40.600	0.678	3.864	0.417	0.149
	TF CNN	63.700	0.524	1.399	0.318	0.015
Route 2	CD	36.242	0.401	2.373	0.323	0.439
	CS	36.905	0.408	2.558	0.298	0.431
	TF CNN	39.672	0.326	1.846	0.227	0.393
Route 3	CD	28.108	0.403	2.064	0.405	0.967
	CS	27.740	0.402	2.078	0.408	0.966
	TF CNN	44.963	0.399	2.610	0.224	0.760
Route 4	CD	31.190	0.307	2.049	0.319	1.154
	CS	30.560	0.321	2.248	0.334	1.168
	TF CNN	38.412	0.377	2.596	0.315	0.898
Route 5	CD	32.475	0.317	1.868	0.280	1.518
	CS	32.374	0.314	2.011	0.278	1.554
	TF CNN	35.203	0.458	2.907	0.250	1.380
Route 6	CD	35.189	0.268	1.832	0.324	1.646
	CS	33.913	0.278	1.968	0.320	1.705
	TF CNN	35.794	0.353	3.265	0.236	1.622

Table 24: PHEV #3, Raw Output Results for all Routes and Strategies

	Comparison	MPGGE % Diff	HC % Diff	CO % Diff	NOx % Diff	Gas % Diff
Route 1	TF CNN vs CD	49.06	-23.37	-57.33	-24.38	-90.35
	TF CNN vs CS	56.90	-22.73	-63.81	-23.75	-89.84
Route 2	TF CNN vs CD	9.46	-18.80	-22.21	-29.76	-10.47
	TF CNN vs CS	7.50	-20.14	-27.83	-23.96	-8.79
Route 3	TF CNN vs CD	59.96	-1.05	26.45	-44.57	-21.41
	TF CNN vs CS	62.09	-0.74	25.61	-44.99	-21.38
Route 4	TF CNN vs CD	23.15	22.81	26.69	-1.32	-22.17
	TF CNN vs CS	25.69	17.57	15.44	-5.65	-23.11
Route 5	TF CNN vs CD	8.40	44.54	55.61	-10.57	-9.08
	TF CNN vs CS	8.74	45.60	44.57	-10.00	-11.19
Route 6	TF CNN vs CD	1.72	31.65	78.27	-26.96	-1.45
	TF CNN vs CS	5.55	27.20	65.94	-26.06	-4.85

Table 25: PHEV #3, Percent Difference Between Strategies

Comparison Strategy	Performance	MPGGE % Diff	HC % Diff	CO % Diff	NOx % Diff	Gas % Diff
CD	Best	59.96	-23.37	-57.33	-44.57	-90.35
	Worst	1.72	44.54	78.27	-1.32	-1.45
CS	Best	62.09	-22.73	-63.81	-44.99	-89.84
	Worst	5.55	45.60	65.94	-5.65	-4.85

Table 26: PHEV #3, Best and Worst Performance

4.3.4. PHEV #4, FC_SIPrime_emis/ ESS_PB28 Combination

	Strategy	MPGGE	HC (g/mile)	CO (g/mile)	NOx (g/mile)	Gas (gal)
Route 1	CD	39.444	0.643	1.394	0.298	0.081
	CS	41.660	0.667	4.643	0.295	0.122
	TF CNN	91.954	0.642	1.381	0.299	0.080
Route 2	CD	40.661	0.400	2.187	0.320	0.495
	CS	40.442	0.451	2.688	0.347	0.518
	TF CNN	44.958	0.395	1.848	0.242	0.411
Route 3	CD	29.089	0.387	2.317	0.327	0.619
	CS	30.095	0.401	2.396	0.338	0.583
	TF CNN	43.322	0.384	2.268	0.272	0.557
Route 4	CD	32.419	0.293	2.414	0.281	1.036
	CS	32.536	0.312	2.284	0.314	1.015
	TF CNN	37.524	0.414	3.327	0.228	0.881
Route 5	CD	35.779	0.312	1.183	0.305	1.524
	CS	36.278	0.303	1.925	0.300	1.521
	TF CNN	39.445	0.472	3.169	0.249	1.287
Route 6	CD	32.626	0.254	2.124	0.278	1.772
	CS	32.773	0.257	1.806	0.281	1.766
	TF CNN	34.152	0.345	4.879	0.224	1.765

Table 27: PHEV #4, Raw Output Results for all Routes and Strategies

	Comparison	MPGGE % Diff	HC % Diff	CO % Diff	NOx % Diff	Gas % Diff
Route 1	TF CNN vs CD	133.13	-0.05	-0.94	0.22	-1.90
	TF CNN vs CS	120.72	-3.67	-70.25	1.21	-34.61
Route 2	TF CNN vs CD	10.57	-1.41	-15.49	-24.41	-17.02
	TF CNN vs CS	11.17	-12.48	-31.24	-30.27	-20.72
Route 3	TF CNN vs CD	48.93	-0.75	-2.13	-16.86	-10.06
	TF CNN vs CS	43.95	-4.39	-5.35	-19.52	-4.45
Route 4	TF CNN vs CD	15.75	41.17	37.81	-18.70	-14.93
	TF CNN vs CS	15.33	32.71	45.62	-27.20	-13.19
Route 5	TF CNN vs CD	10.25	51.41	167.84	-18.35	-15.54
	TF CNN vs CS	8.73	55.90	64.68	-17.03	-15.38
Route 6	TF CNN vs CD	4.68	36.07	129.68	-19.57	-0.43
	TF CNN vs CS	4.21	34.32	170.08	-20.36	-0.05

Table 28: PHEV #4, Percent Difference Between Strategies

Comparison Strategy	Performance	MPGGE % Diff	HC % Diff	CO % Diff	NOx % Diff	Gas % Diff
CD	Best	133.13	-1.41	-15.49	-24.41	-17.02
	Worst	4.68	51.41	167.84	0.22	-0.43
CS	Best	120.72	-12.48	-70.25	-30.27	-34.61
	Worst	4.21	55.90	170.08	1.21	-0.05

Table 29: PHEV #4, Best and Worst Performance

4.3.5. Results Summary

Looking over the results, the TF based machine learning control strategy proves to be a viable and efficient option when compared to the CD and CS control strategies in terms of better MPGGE, gasoline consumption, and NOx emissions. It outperforms the CD and CS strategies in every PHEV across all routes in terms of MPGGE and gasoline consumption. This type of performance is extremely beneficial to buyers of PHEVs, as it can lead to shorter payback time in terms of vehicle investment [36]. Similarly, it produces the least amount of NOx emissions across all routes for PHEV #1 and PHEV #3. For PHEV #2 and PHEV #4, it produces the least amount of NOx emissions in all except Route 1, producing only 2.67% and 1.21% more g/mile of NOx, respectively, than the leading control strategy. With such a small negative difference in performance occurring in only one route for two of the PHEVs, the TF based machine learning control strategy still proves to be a better and more efficient control strategy overall in improving MPGGE, gasoline consumption, and NOx emissions.

In attempting to improve HC emissions, the control strategy’s performance becomes less competitive compared to the CD and CS strategies after a certain distance. For all the PHEVs, the TF based machine learning strategy is only able to outperform the other two strategies in HC emissions for Routes 1 (10 miles), Route 2 (20 miles), and Route 3 (30 miles). For these routes, the control strategy produces 0.05%-23.37% less g/mile of HC than the CD strategy and 0.74%-31.92% less g/mile of HC than the CS strategy. In Route 4 (40 miles), Route 5 (50 miles), and Route 6 (60 miles), the TF based machine learning strategy underperforms, producing 12.21%-51.41% more g/mile of HC than the CD strategy and 11.51%-55.90% more g/mile of HC than the CS strategy. For these three routes, the performance worsened going from the 1.0L engine to the 1.8L engine and from the 25 Ah lead-acid battery to the 28 Ah lead-acid battery, possibly because of the increase in mass. Vehicle mass is one of the main factors influencing a vehicle’s fuel consumption, as increases in the operating mass increases fuel consumption, as more power is needed to accelerate the vehicle during acceleration phases and rolling resistance is also increased proportionally [62]. Regardless of engine and battery choice, however, past 30 miles, the TF based machine learning control strategy is not effective for reducing the HC emissions compared to the CD and CS strategies.

Similarly, the TF based machine learning strategy is only more effective than the CD and CS strategies at reducing g/mile of CO emissions at certain distances depending on the configuration. In PHEV #1, 1.0L engine and 25 Ah battery, and PHEV #3, 1.8L engine and 25 Ah battery, the strategy outperforms both the CD and CS strategy in Route 1 and 2, outputting 22.21%-63.27% less g/mile of CO than the CD strategy and 27.83%-69.44% less g/mile of CO than the CS strategy. Pairing the engines with a larger battery in PHEV #2, 1.0L engine and 28 Ah battery, and PHEV #4, 1.8L engine and 28 Ah battery, further increases the number of routes the strategy outperforms the CD and CS strategies. In PHEV #2, it outperforms both strategies in Routes 1, 2, 3, and 4, outputting 0.71%-34.64% less g/mile of CO than the CD strategy and 4.09%-66.31% less g/mile than the CS strategy. In PHEV #4, it outperforms both strategies in Routes 1, 2, and 3, outputting 0.94%-15.49% less g/mile of CO than the CD strategy and 5.35%-70.25% less g/mile of CO than the CS strategy. With the increase in battery size for each type of engine, the range in distance that the strategy can outperform the CD and CS strategies increases while the overall amount of improvement decreases. This is most likely due to the increase in operating mass as well. Inconsistent performance like this makes the strategy ineffective for reducing the CO emissions compared to the CD and CS strategies for distances past 20-30 miles.

5. Conclusions and Future Work

Two of the parameters that the TF based machine learning strategy consistently outperforms the other strategies in, NOx and gasoline consumption, received lower average accuracies in the testing data when training the CNNs than one of the parameters it falls short in, HC. In addition, the strategy performed comparably in terms of HC and CO overall, even though the HC parameter received the second-best set of average accuracies in CNN training across all PHEVs and the CO parameter received the worst. With these observations, it can be reasoned that the strategy's lack of performance in HC and CO emissions has little to do with a lack of accuracy in the trained CNNs. This shows a possible benefit, perfect accuracy in the TFCNN is not a requirement for the strategy's good performance. We then look at the laws used in TF for mode scheduling and SOC distribution in the PHEVs for why the strategy proves less effective for HC and CO emissions. The TF control algorithm the strategy is based on was made with laws meant for improving fuel economy, MPGGE, by intelligently choosing modes of propulsion and SOC usage along a route based on expected speeds. It does this even at the cost of forcing the PHEV to rely solely on the ICE at very inefficient torque and speed combinations if deemed needed. Unfortunately, the improved fuel economy and fuel consumption that the TF control algorithm can produce in PHEVs using the TF based machine learning strategy is not enough to always overcome the production of HC and CO emissions from fuel at inefficient torque and speed combinations.

While further investigation can be done on improving the strategy's performance for HC and CO emissions at longer distances, at shorter distances it proves to be a viable option for decreasing emissions. Its remarkable performance in fuel economy, fuel consumption, and NOx emissions may make it a worthwhile strategy to use for all lengths of distance and PHEV configurations. These are all due to the effectiveness of the TF control algorithm and the TFCNN's main benefit, its ability to simplify the inner workings of a PHEV and its supervisory control strategy into a fast input/output relation. Furthermore, the TFCNN can perform its task while only needing a minimum amount of already available information, expected speed on a route and its own SOC.

The main contributions of this work are the TF based machine learning strategy and the methodology used to create it. The strategy itself is scalable, computationally fast, and adaptable to changing route data in real-time, with potential for more. It is effective at improving fuel economy, fuel consumption, and NOx emissions, so much so that real world testing and application is a plausible close next step. The methodology outlined in this work to develop the TF based machine learning strategy can serve as a guide for other researchers to implement their own supervisory control algorithms in a less computationally heavy way in real-time. The TF control algorithm may even be fine-tuned to prioritize optimizing emission output rather than fuel economy, and then be implemented with the same methodology. These contributions confirm the significance of this work in the field of PHEV power management.

Future work for this research would consists of two main steps. The first step would be a deeper investigation into the conditions in which the strategy's performance begins to fall below that of other strategies for HC and CO emissions. The reason for this would be to modify the strategy so that its performance in HC and CO emission reduction can be improved, possibly without sacrificing its performance in other outputs. The second step would be further testing of the strategy with more PHEV models and routes. This would show whether application of the strategy is limited by PHEV configuration and/or route, key information to have before going into real-life testing.

6. **References**

[1] The National Academies, “How We Use Energy, Transportation — The National Academies,” *What You Need To Know About Energy*. http://needtoknow.nas.edu/energy/energy-use/transportation/ (accessed Feb. 05, 2020).

[2] “Fossil Fuels | EESI.” https://www.eesi.org/topics/fossil-fuels/description (accessed Jan. 29, 2021).

[3] “In 2018, the United States consumed more energy than ever before - Today in Energy - U.S. Energy Information Administration (EIA).” https://www.eia.gov/todayinenergy/detail.php?id=39092 (accessed Jan. 29, 2021).

[4] “U.S. Net Imports of Crude Oil and Petroleum Products (Thousand Barrels per Day).” https://www.eia.gov/dnav/pet/hist/LeafHandler.ashx?n=pet&s=mttntus2&f=a (accessed Jan. 29, 2021).

[5] “Frequently Asked Questions (FAQs) - U.S. Energy Information Administration (EIA).” https://www.eia.gov/tools/faqs/faq.php (accessed Jan. 29, 2021).

[6] “Bill Text - SB-1078 Renewable energy: California Renewables Portfolio Standard Program.” https://leginfo.legislature.ca.gov/faces/billNavClient.xhtml?bill_id=200120020SB1078 (accessed Jan. 29, 2021).

[7] C. E. Commission, “Clean Energy and Pollution Reduction Act - SB 350,” *California Energy Commission*, current-date. https://www.energy.ca.gov/rules-and-regulations/energy-suppliers-reporting/clean-energy-and-pollution-reduction-act-sb-350 (accessed Jan. 29, 2021).

[8] “RPS.” https://www.cpuc.ca.gov/rps/ (accessed Jan. 29, 2021).

[9] “Alternative Fuels Data Center: Alternative Fuels and Advanced Vehicles.” https://afdc.energy.gov/fuels/ (accessed Jan. 29, 2021).

[10] E. Energy, “Types Of Electric Cars,” *Ergon Energy*, Aug. 25, 2015. https://www.ergon.com.au/network/smarter-energy/electric-vehicles/types-of-electric-vehicles (accessed Jan. 29, 2021).

[11] “Alternative Fuels Data Center: How Do Hybrid Electric Cars Work?” https://afdc.energy.gov/vehicles/how-do-hybrid-electric-cars-work (accessed Jan. 29, 2021).

[12] “Alternative Fuels Data Center: How Do Plug-In Hybrid Electric Cars Work?” https://afdc.energy.gov/vehicles/how-do-plug-in-hybrid-electric-cars-work (accessed Jan. 29, 2021).

[13] H. Banvait, S. Anwar, and Y. Chen, “A rule-based energy management strategy for Plug-in Hybrid Electric Vehicle (PHEV),” in *2009 American Control Conference*, Jun. 2009, pp. 3938–3943, doi: 10.1109/ACC.2009.5160242.

[14] C.-S. N. Shiau, C. Samaras, R. Hauffe, and J. J. Michalek, “Impact of battery weight and charging patterns on the economic and environmental benefits of plug-in hybrid vehicles,” *Energy Policy*, vol. 37, no. 7, pp. 2653–2663, Jul. 2009, doi: 10.1016/j.enpol.2009.02.040.

[15] D. Stone and M. Hamilton, “Fuel economy improvements are projected to reduce future gasoline use,” *U.S. Energy Information Administration - Today In Energy*, May 23, 2017. https://www.eia.gov/todayinenergy/detail.php?id=31332 (accessed Jan. 29, 2021).

[16] S. G. Wirasingha and A. Emadi, “Classification and Review of Control Strategies for Plug-In Hybrid Electric Vehicles,” *IEEE Trans. Veh. Technol.*, vol. 60, no. 1, pp. 111–122, Jan. 2011, doi: 10.1109/TVT.2010.2090178.

[17] S. Cordiner, M. Galeotti, V. Mulone, M. Nobile, and V. Rocco, “Trip-based SOC management for a plugin hybrid electric vehicle,” *Appl. Energy*, vol. 164, pp. 891–905, Feb. 2016, doi: 10.1016/j.apenergy.2015.06.009.

[18] A. Sciarretta and L. Guzzella, “Control of hybrid electric vehicles,” *IEEE Control Syst. Mag.*, vol. 27, no. 2, pp. 60–70, Apr. 2007, doi: 10.1109/MCS.2007.338280.

[19] X. Hu, N. Murgovski, L. Johannesson, and B. Egardt, “Energy efficiency analysis of a series plug-in hybrid electric bus with different energy management strategies and battery sizes,” *Appl. Energy*, vol. 111, pp. 1001–1009, Nov. 2013, doi: 10.1016/j.apenergy.2013.06.056.

[20] T. H. Bradley and A. A. Frank, “Design, demonstrations and sustainability impact assessments for plug-in hybrid electric vehicles,” *Renew. Sustain. Energy Rev.*, vol. 13, no. 1, pp. 115–128, Jan. 2009, doi: 10.1016/j.rser.2007.05.003.

[21] M. Murnane and A. Ghazel, “A Closer Look at State Of Charge (SOC) and State Of Health (SOH) Estimation Techniques for Batteries,” p. 8, 2017.

[22] Q. Gong, Y. Li, and Z.-R. Peng, “Trip-Based Optimal Power Management of Plug-in Hybrid Electric Vehicles,” *IEEE Trans. Veh. Technol.*, vol. 57, no. 6, pp. 3393–3401, Nov. 2008, doi: 10.1109/TVT.2008.921622.

[23] B. M. Baumann, G. Washington, B. C. Glenn, and G. Rizzoni, “Mechatronic design and control of hybrid electric vehicles,” *IEEEASME Trans. Mechatron.*, vol. 5, no. 1, pp. 58–72, Mar. 2000, doi: 10.1109/3516.828590.

[24] N. J. Schouten, M. A. Salman, and N. A. Kheir, “Fuzzy logic control for parallel hybrid vehicles,” *IEEE Trans. Control Syst. Technol.*, vol. 10, no. 3, pp. 460–468, May 2002, doi: 10.1109/87.998036.

[25] S. Jeon, S. Jo, Y. Park, and J. Lee, “Multi-Mode Driving Control of a Parallel Hybrid Electric Vehicle Using Driving Pattern Recognition,” *J. Dyn. Syst. Meas. Control*, vol. 124, no. 1, pp. 141–149, Mar. 2002, doi: 10.1115/1.1434264.

[26] R. Langari and Jong-Seob Won, “Intelligent energy management agent for a parallel hybrid vehicle-part I: system architecture and design of the driving situation identification process,” *IEEE Trans. Veh. Technol.*, vol. 54, no. 3, pp. 925–934, May 2005, doi: 10.1109/TVT.2005.844685.

[27] Jong-Seob Won and R. Langari, “Intelligent energy management agent for a parallel hybrid vehicle-part II: torque distribution, charge sustenance strategies, and performance results,” *IEEE Trans. Veh. Technol.*, vol. 54, no. 3, pp. 935–953, May 2005, doi: 10.1109/TVT.2005.844683.

[28] S. Delprat, J. Lauber, T.-M. Guerra, and J. Rimaux, “Control of a Parallel Hybrid Powertrain: Optimal Control,” *IEEE Trans. Veh. Technol.*, vol. 53, pp. 872–881, Jun. 2004, doi: 10.1109/TVT.2004.827161.

[29] A. Sciarretta, M. Back, and L. Guzzella, “Optimal control of parallel hybrid electric vehicles,” *IEEE Trans. Control Syst. Technol.*, vol. 12, no. 3, pp. 352–363, May 2004, doi: 10.1109/TCST.2004.824312.

[30] U. Zoelch and D. Schroeder, “Dynamic optimization method for design and rating of the components of a hybrid vehicle,” *Int. J. Veh. Des.*, vol. 19, no. 1, pp. 1–13, Jan. 1998, doi: 10.1504/IJVD.1998.062090.

[31] A. Brahma, Y. Guezennec, and G. Rizzoni, “Optimal energy management in series hybrid electric vehicles,” in *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334)*, Jun. 2000, vol. 1, pp. 60–64 vol.1, doi: 10.1109/ACC.2000.878772.

[32] Chan-Chiao Lin, Huei Peng, J. W. Grizzle, and Jun-Mo Kang, “Power management strategy for a parallel hybrid electric truck,” *IEEE Trans. Control Syst. Technol.*, vol. 11, no. 6, pp. 839–849, Nov. 2003, doi: 10.1109/TCST.2003.815606.

[33] L. V. Pérez, G. R. Bossio, D. Moitre, and G. O. García, “Optimization of power management in an hybrid electric vehicle using dynamic programming,” *Math. Comput. Simul.*, vol. 73, no. 1, pp. 244–254, Nov. 2006, doi: 10.1016/j.matcom.2006.06.016.

[34] M. Koot, J. T. B. A. Kessels, B. de Jager, W. P. M. H. Heemels, P. P. J. van den Bosch, and M. Steinbuch, “Energy management strategies for vehicular electric power systems,” *IEEE Trans. Veh. Technol.*, vol. 54, no. 3, pp. 771–782, May 2005, doi: 10.1109/TVT.2005.847211.

[35] Yang Bin, Y. Li, Q. Gong, and Z.-R. Peng, “Multi-information integrated trip specific optimal power management for plug-in hybrid electric vehicles,” in *2009 American Control Conference*, Jun. 2009, pp. 4607–4612, doi: 10.1109/ACC.2009.5160626.

[36] Q. Gong, Y. Li, and Z.-R. Peng, “Trip Based Power Management of Plug-in Hybrid Electric Vehicle with Two-Scale Dynamic Programming,” in *2007 IEEE Vehicle Power and Propulsion Conference*, Sep. 2007, pp. 12–19, doi: 10.1109/VPPC.2007.4544089.

[37] C.-K. Chau, K. Elbassioni, and C.-M. Tseng, “Drive Mode Optimization and Path Planning for Plug-In Hybrid Electric Vehicles,” *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 12, pp. 3421–3432, Dec. 2017, doi: 10.1109/TITS.2017.2691606.

[38] C.-K. Chau, K. Elbassioni, and C.-M. Tseng, “Fuel minimization of plug-in hybrid electric vehicles by optimizing drive mode selection,” in *Proceedings of the Seventh International Conference on Future Energy Systems*, Waterloo, Ontario, Canada, Jun. 2016, pp. 1–11, doi: 10.1145/2934328.2934341.

[39] K. Murakami, “Formulation and algorithms for route planning problem of plug-in hybrid electric vehicles,” *Oper. Res.*, vol. 18, no. 2, pp. 497–519, Jul. 2018, doi: 10.1007/s12351-016-0274-5.

[40] M. M. Nejad, L. Mashayekhy, D. Grosu, and R. B. Chinnam, “Optimal Routing for Plug-In Hybrid Electric Vehicles,” *Transp. Sci.*, vol. 51, no. 4, pp. 1304–1325, Mar. 2017, doi: 10.1287/trsc.2016.0706.

[41] J. Meng and X. Liu, “MPG Prediction based on BP Neural Network,” in *2006 1ST IEEE Conference on Industrial Electronics and Applications*, May 2006, pp. 1–3, doi: 10.1109/ICIEA.2006.257357.

[42] M. Jamala and S. Abu-Naser, “Predicting MPG for Automobile Using Artificial Neural Network Analysis,” *Inf. Syst. Res.*, vol. 2, pp. 5–21, Oct. 2018.

[43] A. Aliyu and S. A. Adeshina, “Classifying auto-MPG data set using neural network,” *2014 11th Int. Conf. Electron. Comput. Comput. ICECCO*, pp. 1–4, 2014, doi: 10.1109/ICECCO.2014.6997582.

[44] J. Topić, B. Škugor, and J. Deur, “Neural Network-Based Modeling of Electric Vehicle Energy Demand and All Electric Range,” *Energies*, vol. 12, no. 7, p. 1396, Apr. 2019, doi: 10.3390/en12071396.

[45] H. Alipour, B. Asaei, and G. Farivar, “Fuzzy Logic Based Power Management Strategy for Plug-in Hybrid Electric Vehicles with Parallel Configuration,” Mar. 2012.

[46] “Google Maps,” *Google Maps*. https://www.google.com/maps (accessed Jan. 30, 2020).

[47] A. Brooker *et al.*, “ADVISOR Documentation,” *ADVISOR Advanced Vehicle Simulator*, Mar. 26, 2013. http://adv-vehicle-sim.sourceforge.net/ (accessed Jan. 30, 2020).

[48] K. S. Ng, C.-S. Moo, Y.-P. Chen, and Y.-C. Hsieh, “Enhanced coulomb counting method for estimating state-of-charge and state-of-health of lithium-ion batteries,” *Appl. Energy*, vol. 86, no. 9, pp. 1506–1511, Sep. 2009, doi: 10.1016/j.apenergy.2008.11.021.

[49] B. C. Glenn, “Intelligent Control of Parallel Hybrid Electric Vehicles,” The Ohio State University, 1999.

[50] R. Belohlavek and G. J. Klir, Eds., *Concepts and Fuzzy Logic*. The MIT Press, 2011.

[51] K. M. Passino and S. Yurkovich, *Fuzzy Control*. Menlo Park, Calif: Addison-Wesley Longman, Inc., 1998.

[52] O. Akyazi, M. A. Usta, and A. S. Akpinar, “A Self-Tuning Fuzzy Logic Controller for Aircraft Roll Control System,” *Int. J. Control Sci. Eng.*, vol. 2, no. 6, pp. 181–188, 2012.

[53] “Vehicle and Fuel Emissions Testing - Dynamometer Drive Schedules,” *US EPA*, Sep. 16, 2015. https://www.epa.gov/vehicle-and-fuel-emissions-testing/dynamometer-drive-schedules.

[54] S. Saha, “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way,” *Medium*, Dec. 15, 2018. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (accessed Mar. 01, 2021).

[55] “Transportation Secure Data Center,” *National Renewable Energy Laboratory*, 2017. www.nrel.gov/tsdc (accessed Feb. 04, 2020).

[56] M. West, “Convolutional Neural Networks : The Theory,” *Bouvet Norge*. https://www.bouvet.no/bouvet-deler/understanding-convolutional-neural-networks-part-1 (accessed Feb. 04, 2020).

[57] “Convolutional neural networks.,” *Jeremy Jordan*, Jul. 26, 2017. https://www.jeremyjordan.me/convolutional-neural-networks/ (accessed Feb. 12, 2021).

[58] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *ArXiv14126980 Cs*, Jan. 2017, Accessed: Feb. 04, 2020. [Online]. Available: http://arxiv.org/abs/1412.6980.

[59] “An overview of gradient descent optimization algorithms,” *Sebastian Ruder*, Jan. 19, 2016. https://ruder.io/optimizing-gradient-descent/ (accessed Mar. 02, 2021).

[60] A. Ng, K. Katanforoosh, and A. Avati, “Deep Learning.” Accessed: Feb. 12, 2021. [Online]. Available: http://cs229.stanford.edu/notes2019fall/cs229-notes-deep_learning.pdf.

[61] “Options for training deep learning neural network - MATLAB trainingOptions.” https://www.mathworks.com/help/deeplearning/ref/trainingoptions.html (accessed Feb. 12, 2021).

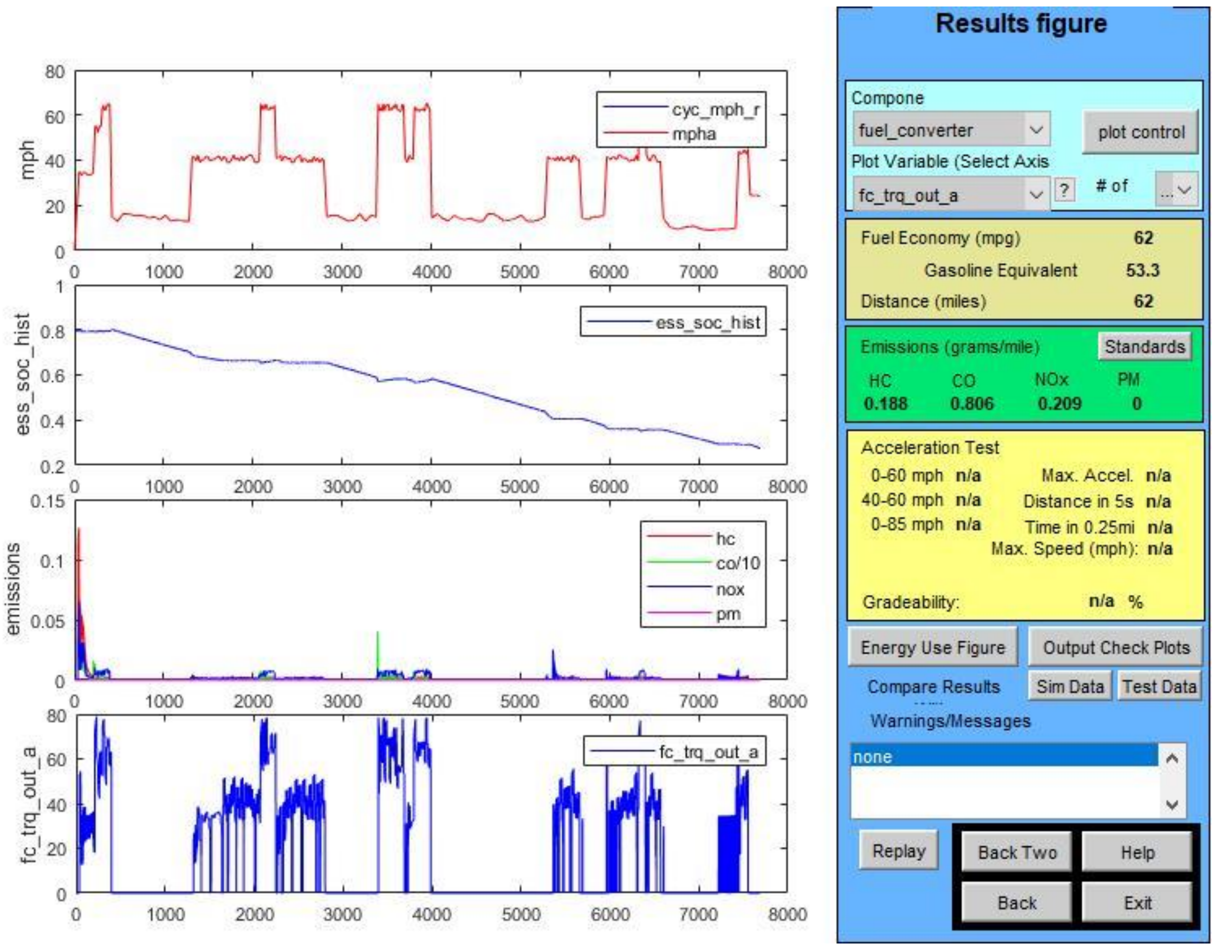
[62] G. Fontaras, N.-G. Zacharof, and B. Ciuffo, “Fuel consumption and CO2 emissions from passenger cars in Europe – Laboratory versus real-world emissions,” *Prog. Energy Combust. Sci.*, vol. 60, pp. 97–131, May 2017, doi: 10.1016/j.pecs.2016.12.004.

7. Appendix

7.1. Original Trajectory Forecasting Development

7.1.1. Performance Results Figures

7.1.1.1. Route 1



Compone: fuel_converter

Plot Variable (Select Axis: fc_trq_out_a

Fuel Economy (mpg)	62
Gasoline Equivalent	53.3
Distance (miles)	62

Emissions (grams/mile)

HC	CO	NOx	PM
0.188	0.806	0.209	0

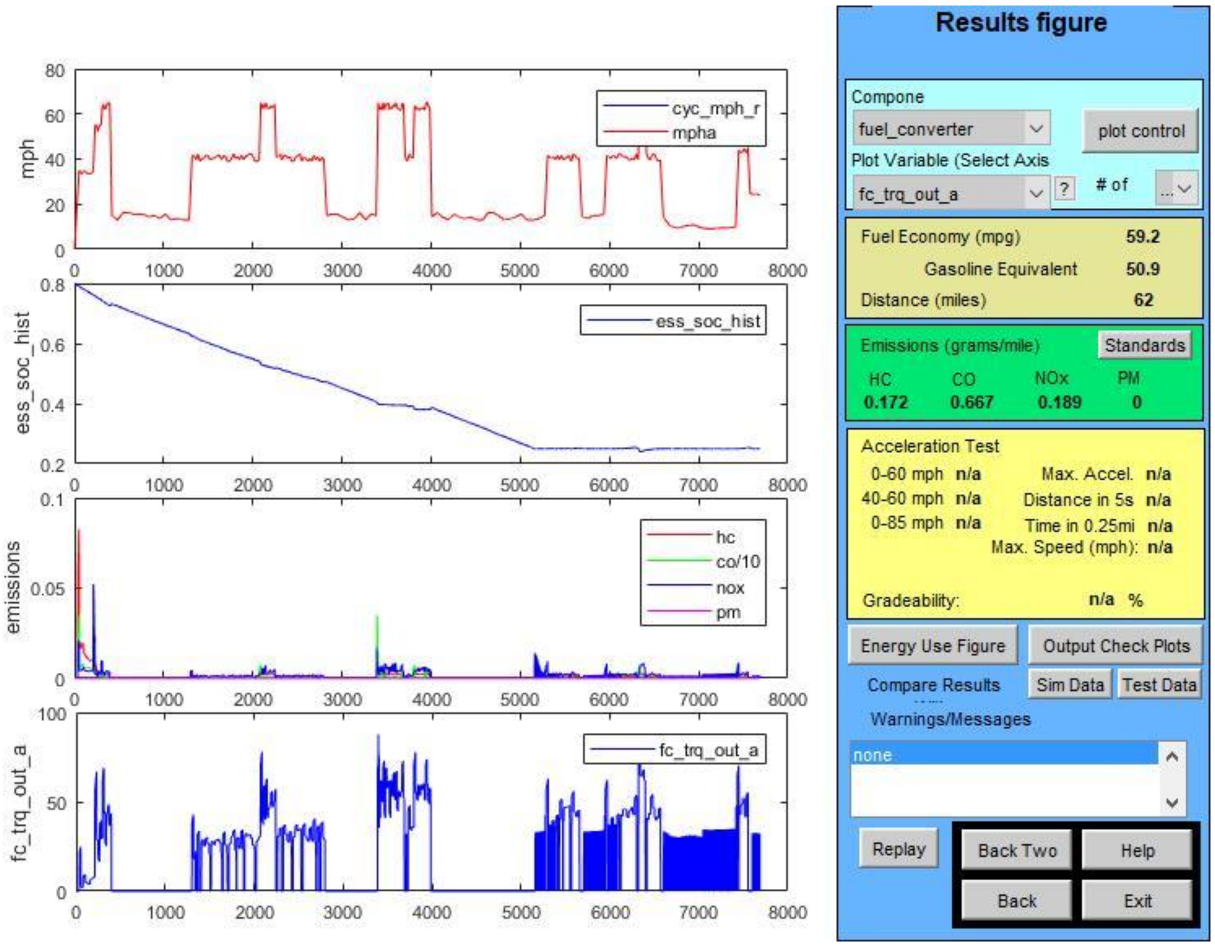
Acceleration Test

0-60 mph	n/a	Max. Accel.	n/a
40-60 mph	n/a	Distance in 5s	n/a
0-85 mph	n/a	Time in 0.25mi	n/a
		Max. Speed (mph):	n/a

Gradeability: n/a %

Warnings/Messages: none

Figure 33: Route 1 TF control strategy ADVISOR simulation results



Compone: fuel_converter

Plot Variable (Select Axis: fc_trq_out_a

Fuel Economy (mpg)	59.2
Gasoline Equivalent	50.9
Distance (miles)	62

Emissions (grams/mile)

HC	CO	NOx	PM
0.172	0.667	0.189	0

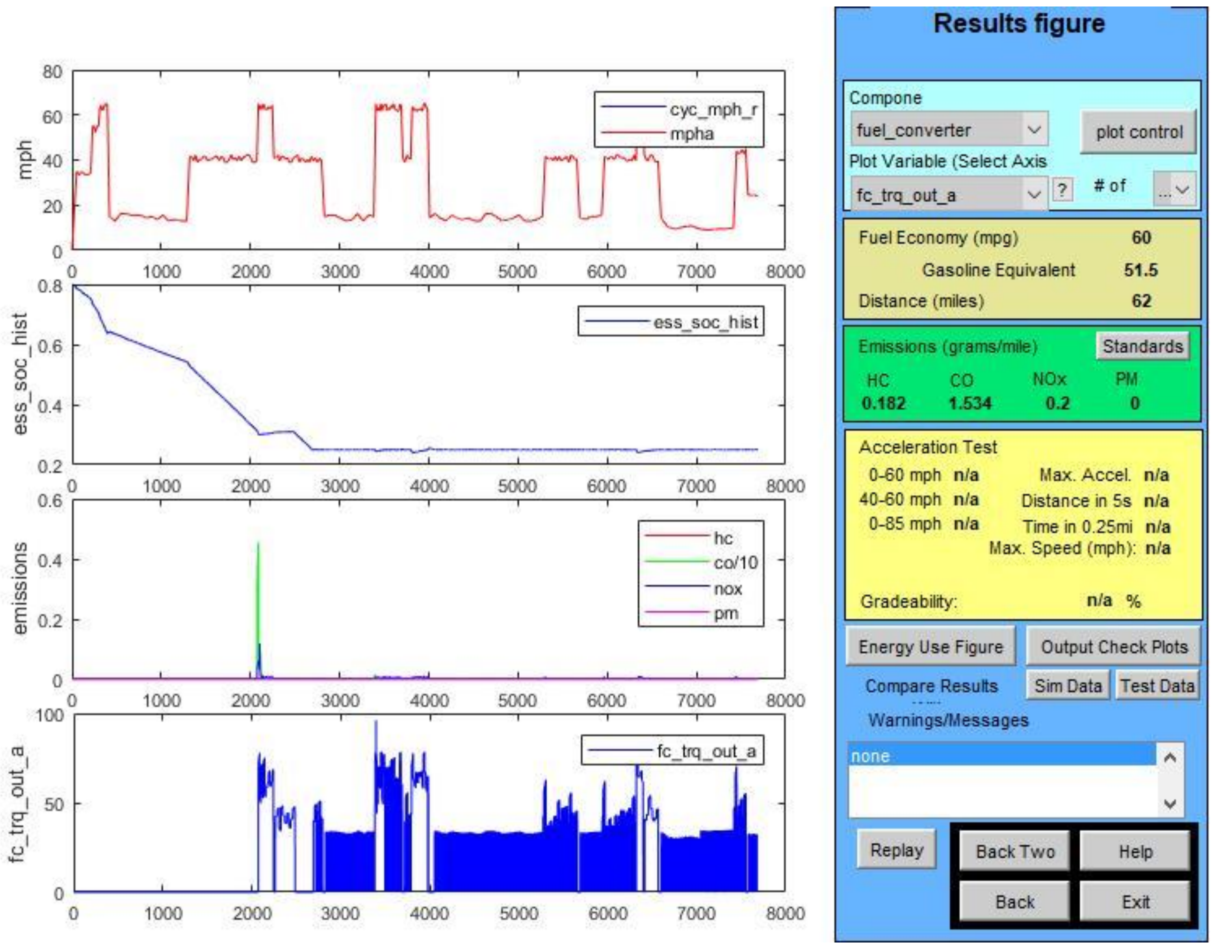
Acceleration Test

0-60 mph	n/a	Max. Accel.	n/a
40-60 mph	n/a	Distance in 5s	n/a
0-85 mph	n/a	Time in 0.25mi	n/a
		Max. Speed (mph):	n/a

Gradeability: n/a %

Warnings/Messages: none

Figure 34: Route 1 CS control strategy ADVISOR simulation results



Compone: fuel_converter

Plot Variable (Select Axis: fc_trq_out_a

of

Fuel Economy (mpg)	60
Gasoline Equivalent	51.5
Distance (miles)	62

Emissions (grams/mile)

HC	CO	NOx	PM
0.182	1.534	0.2	0

Acceleration Test

0-60 mph	n/a	Max. Accel.	n/a
40-60 mph	n/a	Distance in 5s	n/a
0-85 mph	n/a	Time in 0.25mi	n/a
		Max. Speed (mph):	n/a

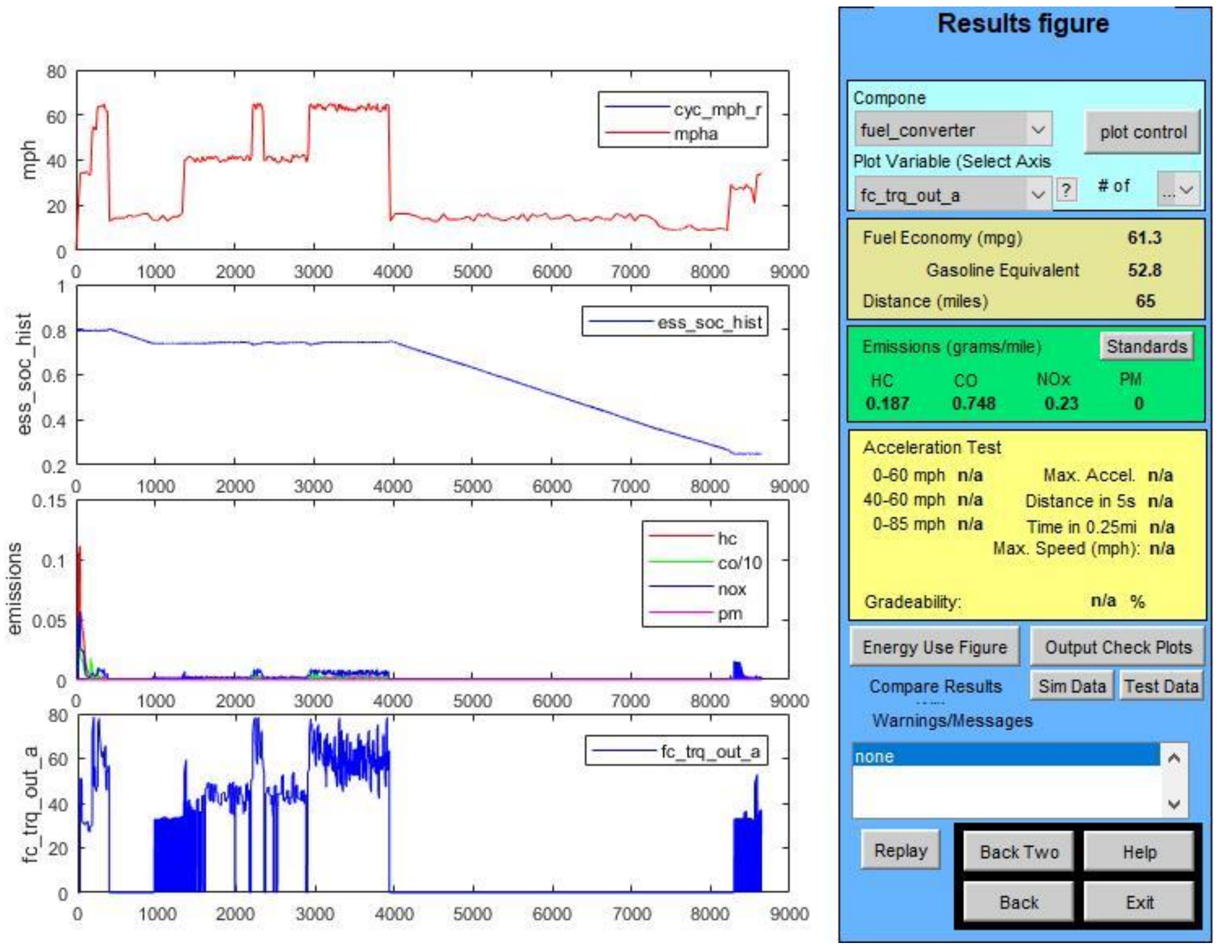
Gradeability: n/a %

Warnings/Messages

none

Figure 35: Route 1 CD control strategy ADVISOR simulation results

7.1.1.2. Route 2



Results figure

Compone: fuel_converter
Plot Variable (Select Axis: fc_trq_out_a

Fuel Economy (mpg)	61.3
Gasoline Equivalent	52.8
Distance (miles)	65

Emissions (grams/mile)

HC	CO	NOx	PM
0.187	0.748	0.23	0

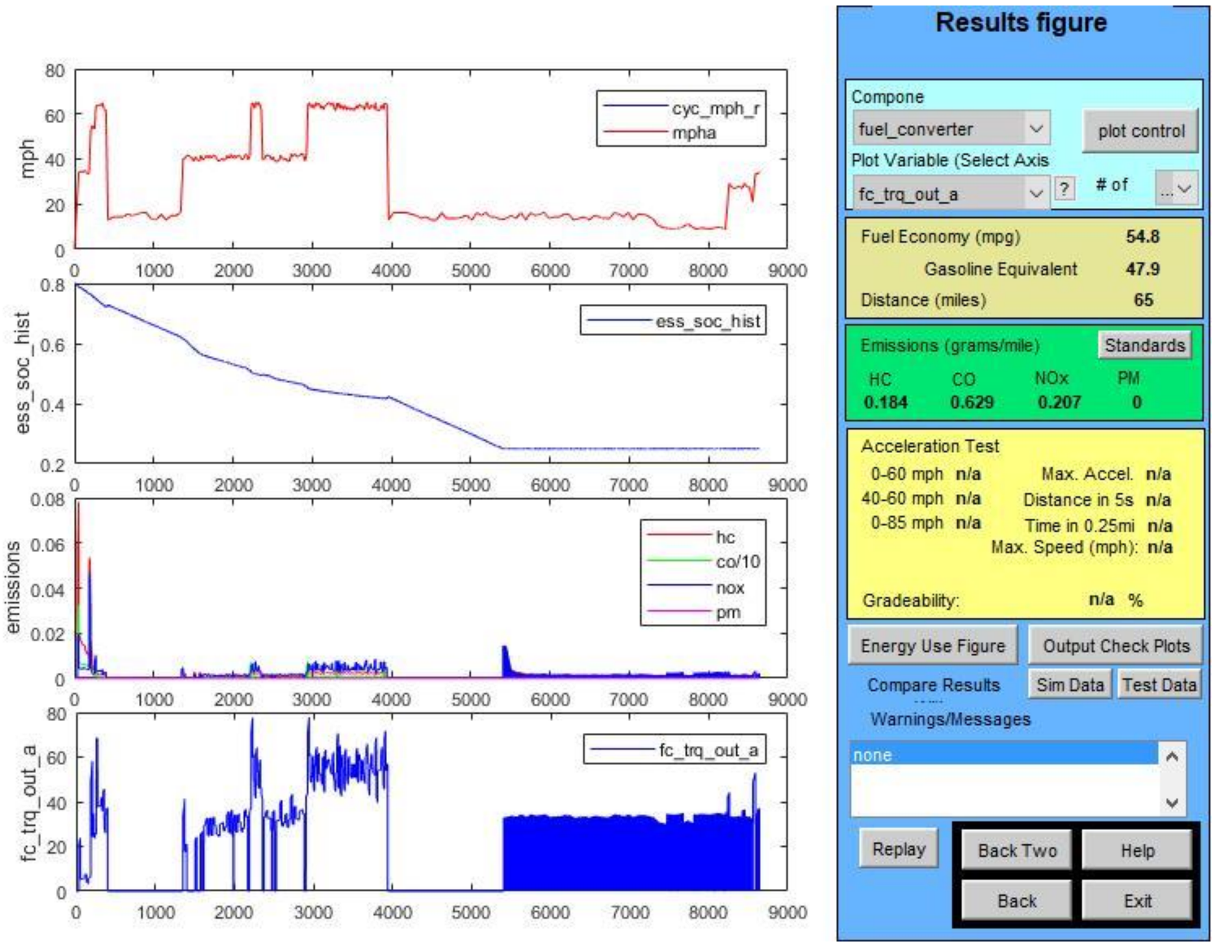
Acceleration Test

0-60 mph	n/a	Max. Accel.	n/a
40-60 mph	n/a	Distance in 5s	n/a
0-85 mph	n/a	Time in 0.25mi	n/a
		Max. Speed (mph):	n/a

Gradeability: n/a %

Warnings/Messages: none

Figure 36: Route 2 TF control strategy ADVISOR simulation results



Fuel Economy (mpg) 54.8
Gasoline Equivalent 47.9
Distance (miles) 65

Emissions (grams/mile)

HC	CO	NOx	PM
0.184	0.629	0.207	0

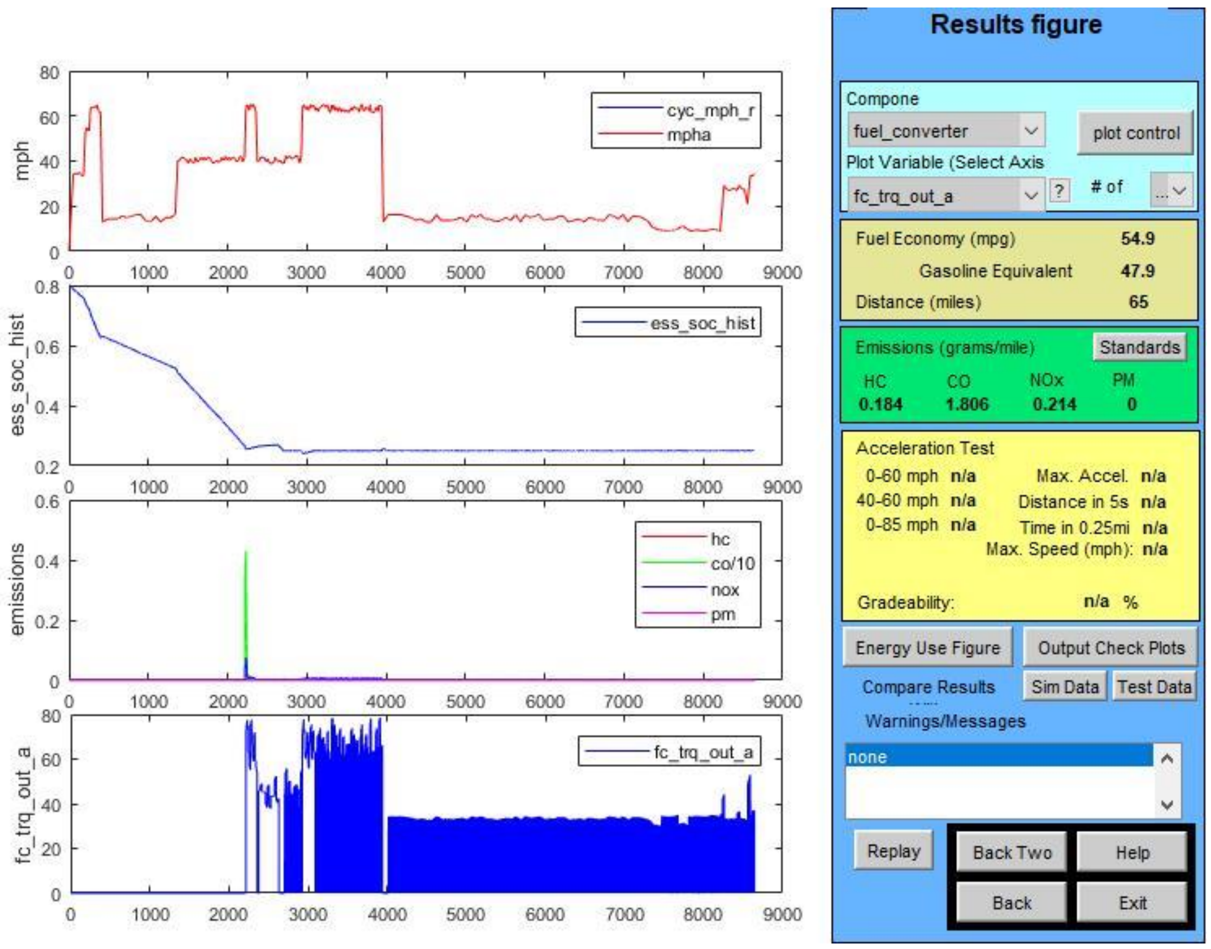
Acceleration Test

0-60 mph n/a
40-60 mph n/a
0-85 mph n/a
Max. Accel. n/a
Distance in 5s n/a
Time in 0.25mi n/a
Max. Speed (mph): n/a

Gradeability: n/a %

none

Figure 37: Route 2 CS control strategy ADVISOR simulation results



fuel_converter

fc_trq_out_a

Fuel Economy (mpg)	54.9
Gasoline Equivalent	47.9
Distance (miles)	65

Emissions (grams/mile)

HC	CO	NOx	PM
0.184	1.806	0.214	0

Acceleration Test

0-60 mph	n/a	Max. Accel.	n/a
40-60 mph	n/a	Distance in 5s	n/a
0-85 mph	n/a	Time in 0.25mi	n/a
		Max. Speed (mph):	n/a

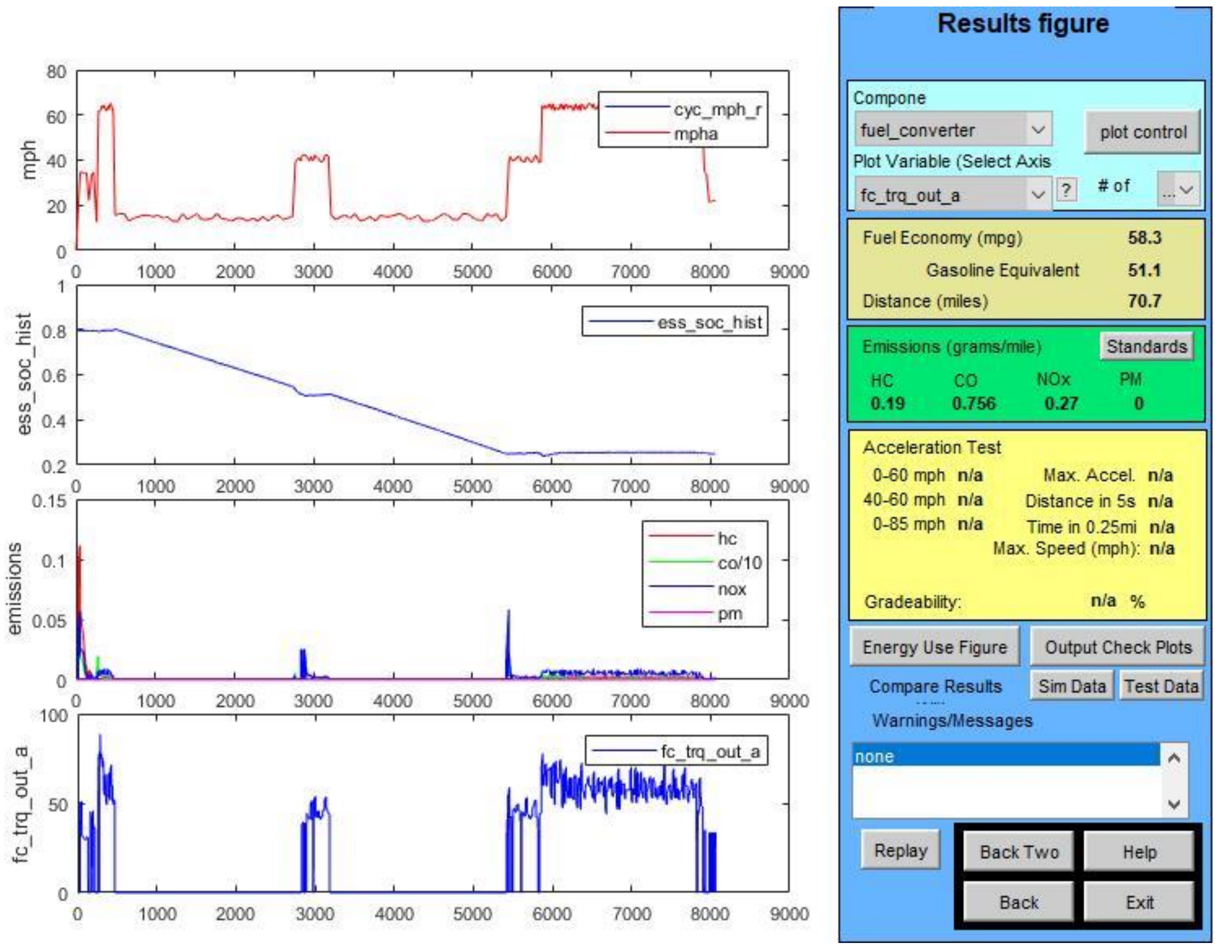
Gradeability: n/a %

Warnings/Messages

none

Figure 38: Route 2 CD control strategy ADVISOR simulation results

7.1.1.3.Route 3



Fuel Economy (mpg) 58.3
Gasoline Equivalent 51.1
Distance (miles) 70.7

Emissions (grams/mile)

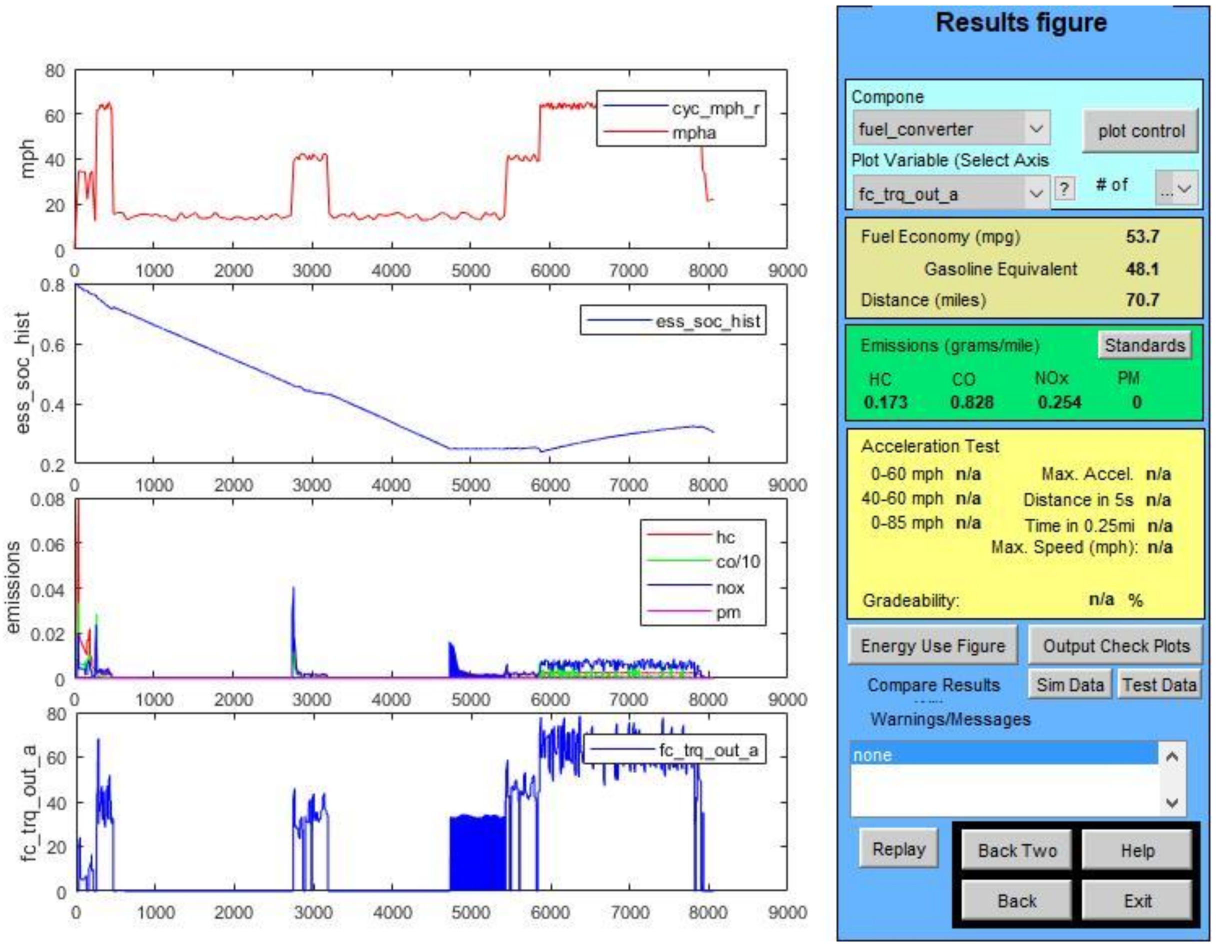
HC	CO	NOx	PM
0.19	0.756	0.27	0

Acceleration Test
0-60 mph n/a
40-60 mph n/a
0-85 mph n/a
Max. Accel. n/a
Distance in 5s n/a
Time in 0.25mi n/a
Max. Speed (mph): n/a

Gradeability: n/a %

none

Figure 39: Route 3 TF control strategy ADVISOR simulation results



Fuel Economy (mpg) 53.7
Gasoline Equivalent 48.1
Distance (miles) 70.7

Emissions (grams/mile)

HC	CO	NOx	PM
0.173	0.828	0.254	0

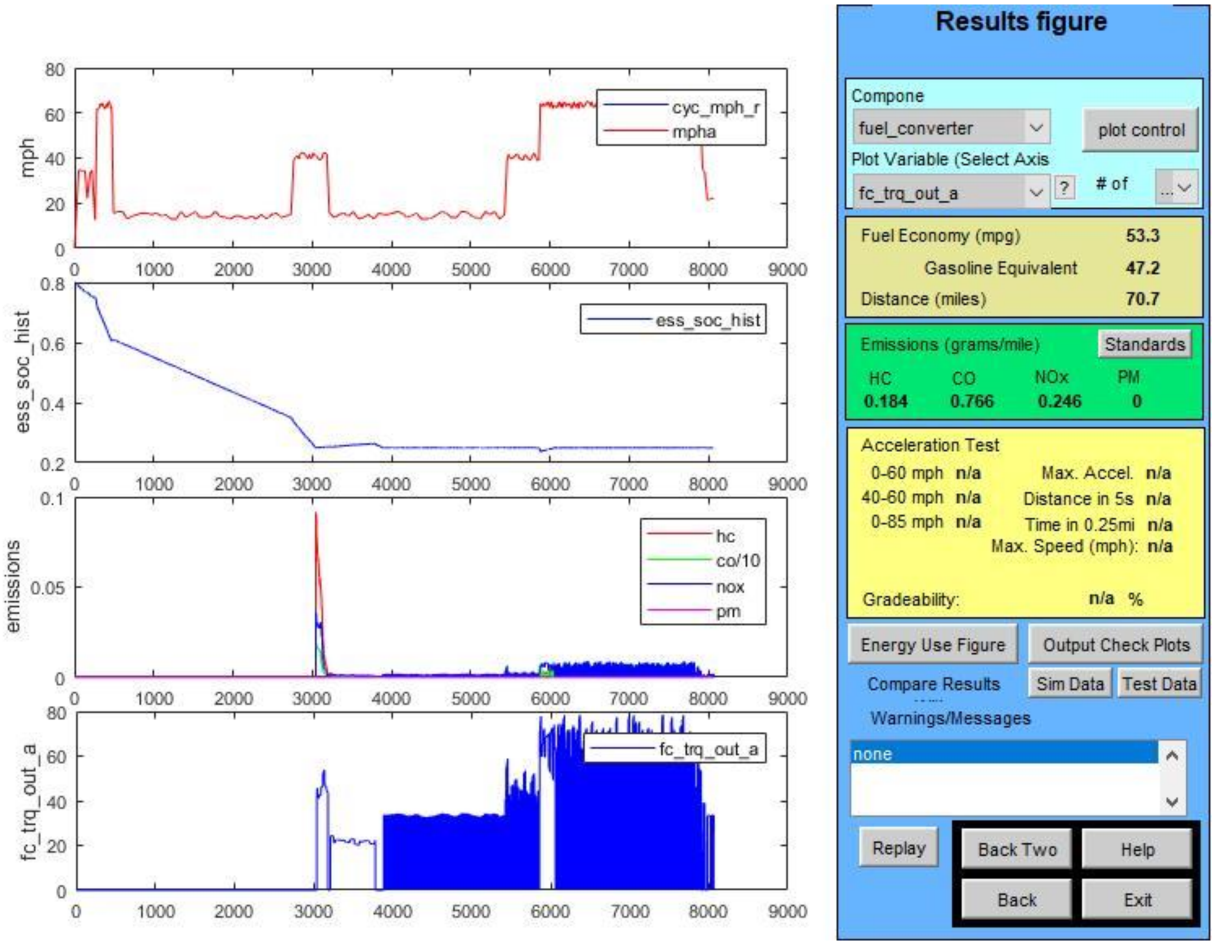
Acceleration Test

0-60 mph	n/a	Max. Accel.	n/a
40-60 mph	n/a	Distance in 5s	n/a
0-85 mph	n/a	Time in 0.25mi	n/a
		Max. Speed (mph):	n/a

Gradeability: n/a %

none

Figure 40: Route 3 CS control strategy ADVISOR simulation results



Fuel Economy (mpg) 53.3
Gasoline Equivalent 47.2
Distance (miles) 70.7

Emissions (grams/mile)

HC	CO	NOx	PM
0.184	0.766	0.246	0

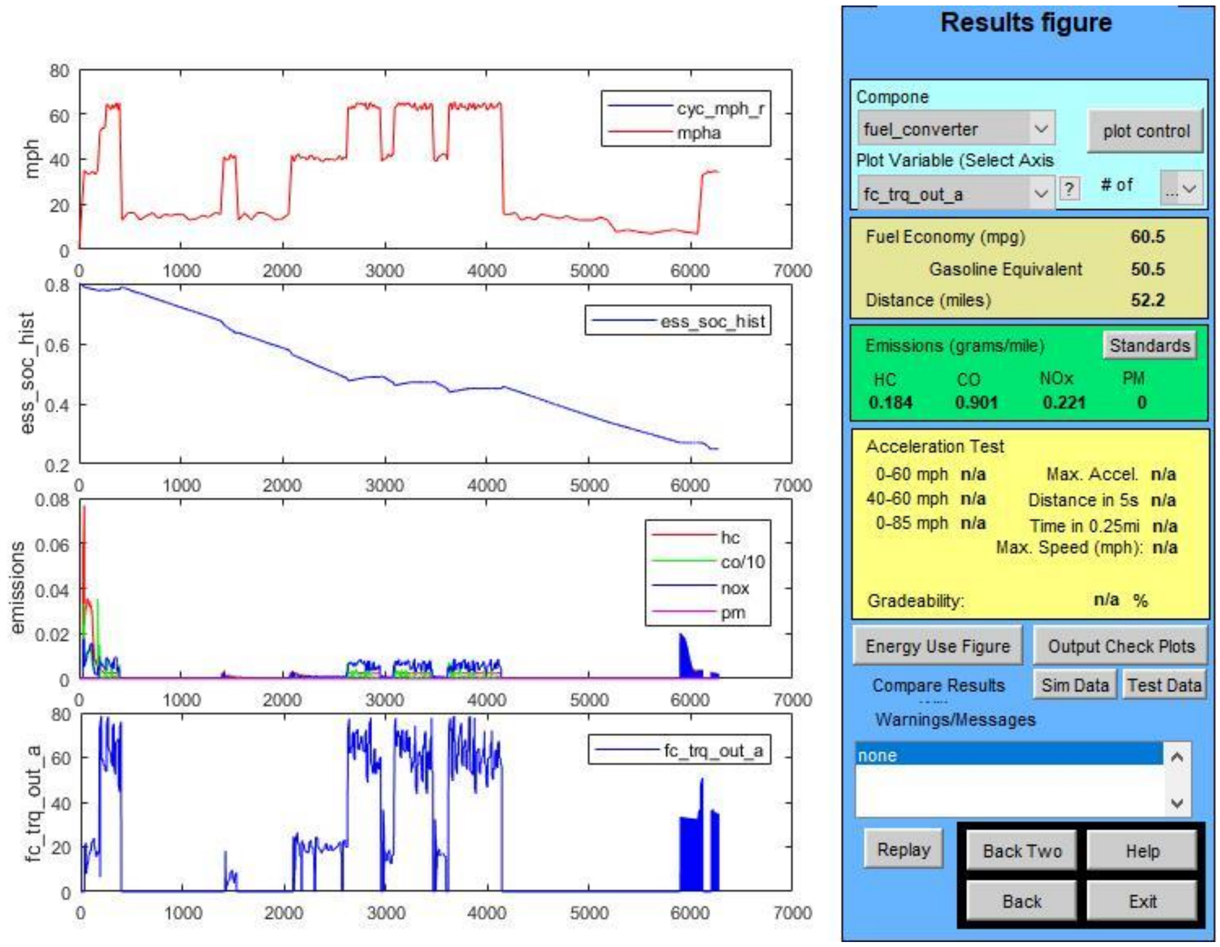
Acceleration Test
0-60 mph n/a
40-60 mph n/a
0-85 mph n/a
Max. Accel. n/a
Distance in 5s n/a
Time in 0.25mi n/a
Max. Speed (mph): n/a

Gradeability: n/a %

none

Figure 41: Route 3 CD control strategy ADVISOR simulation results

7.1.1.4. Route 4



Compone: fuel_converter
Plot Variable (Select Axis: fc_trq_out_a

Fuel Economy (mpg)	60.5
Gasoline Equivalent	50.5
Distance (miles)	52.2

Emissions (grams/mile)

HC	CO	NOx	PM
0.184	0.901	0.221	0

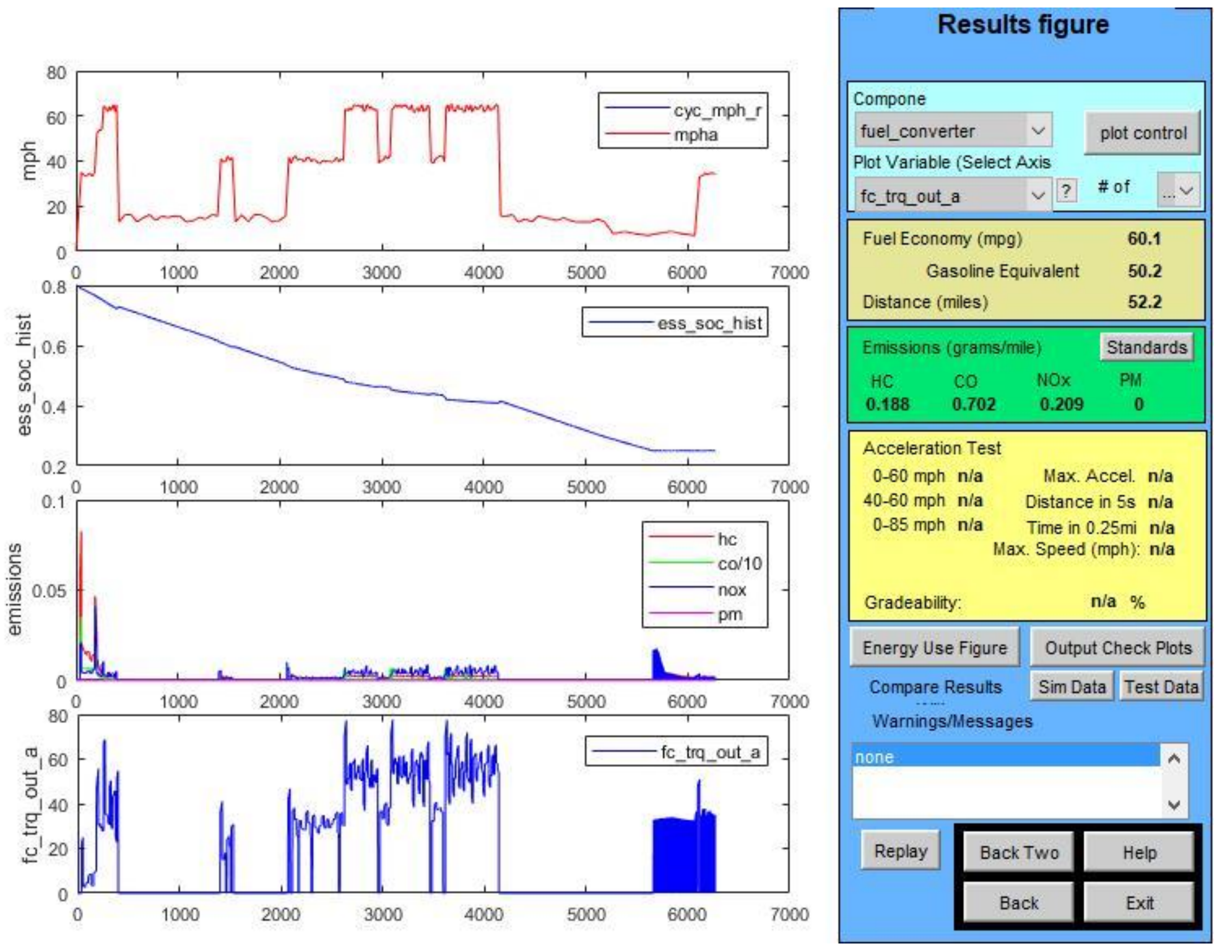
Acceleration Test

0-60 mph	n/a	Max. Accel.	n/a
40-60 mph	n/a	Distance in 5s	n/a
0-85 mph	n/a	Time in 0.25mi	n/a
		Max. Speed (mph):	n/a

Gradeability: n/a %

Warnings/Messages: none

Figure 42: Route 4 TF control strategy ADVISOR simulation results



Fuel Economy (mpg) 60.1
Gasoline Equivalent 50.2
Distance (miles) 52.2

Emissions (grams/mile)

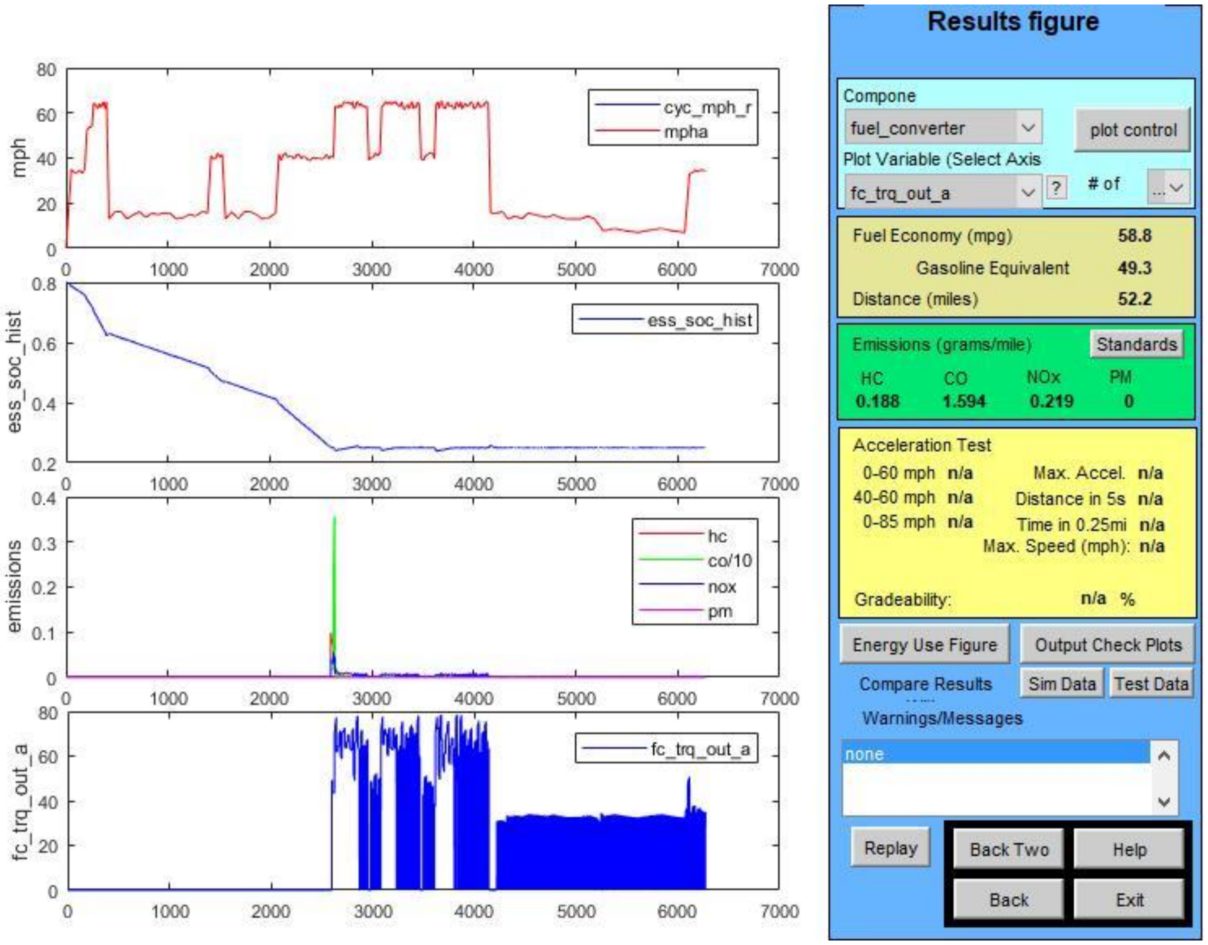
HC	CO	NOx	PM
0.188	0.702	0.209	0

Acceleration Test
0-60 mph n/a
40-60 mph n/a
0-85 mph n/a
Max. Accel. n/a
Distance in 5s n/a
Time in 0.25mi n/a
Max. Speed (mph): n/a

Gradeability: n/a %

Warnings/Messages
none

Figure 43: Route 4 CS control strategy ADVISOR simulation results



Compone: fuel_converter
Plot Variable (Select Axis: fc_trq_out_a
# of

Fuel Economy (mpg) 58.8
Gasoline Equivalent 49.3
Distance (miles) 52.2

Emissions (grams/mile)

HC	CO	NOx	PM
0.188	1.594	0.219	0

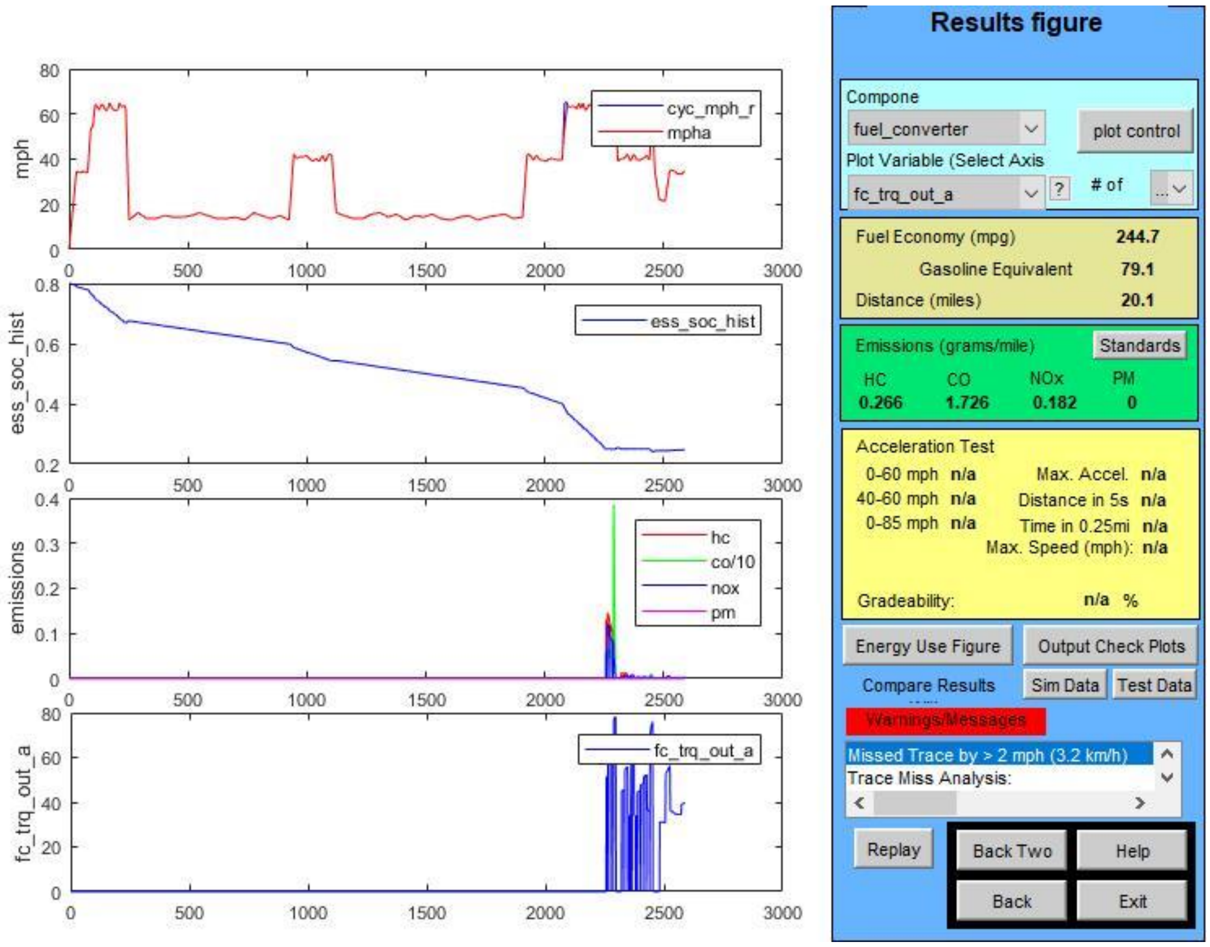
Acceleration Test
0-60 mph n/a
40-60 mph n/a
0-85 mph n/a
Max. Accel. n/a
Distance in 5s n/a
Time in 0.25mi n/a
Max. Speed (mph): n/a

Gradeability: n/a %

Warnings/Messages
none

Figure 44: Route 4 CD control strategy ADVISOR simulation results

7.1.1.5. Route 5



Results figure

Fuel Economy (mpg)	244.7
Gasoline Equivalent	79.1
Distance (miles)	20.1

Emissions (grams/mile)

HC	CO	NOx	PM
0.266	1.726	0.182	0

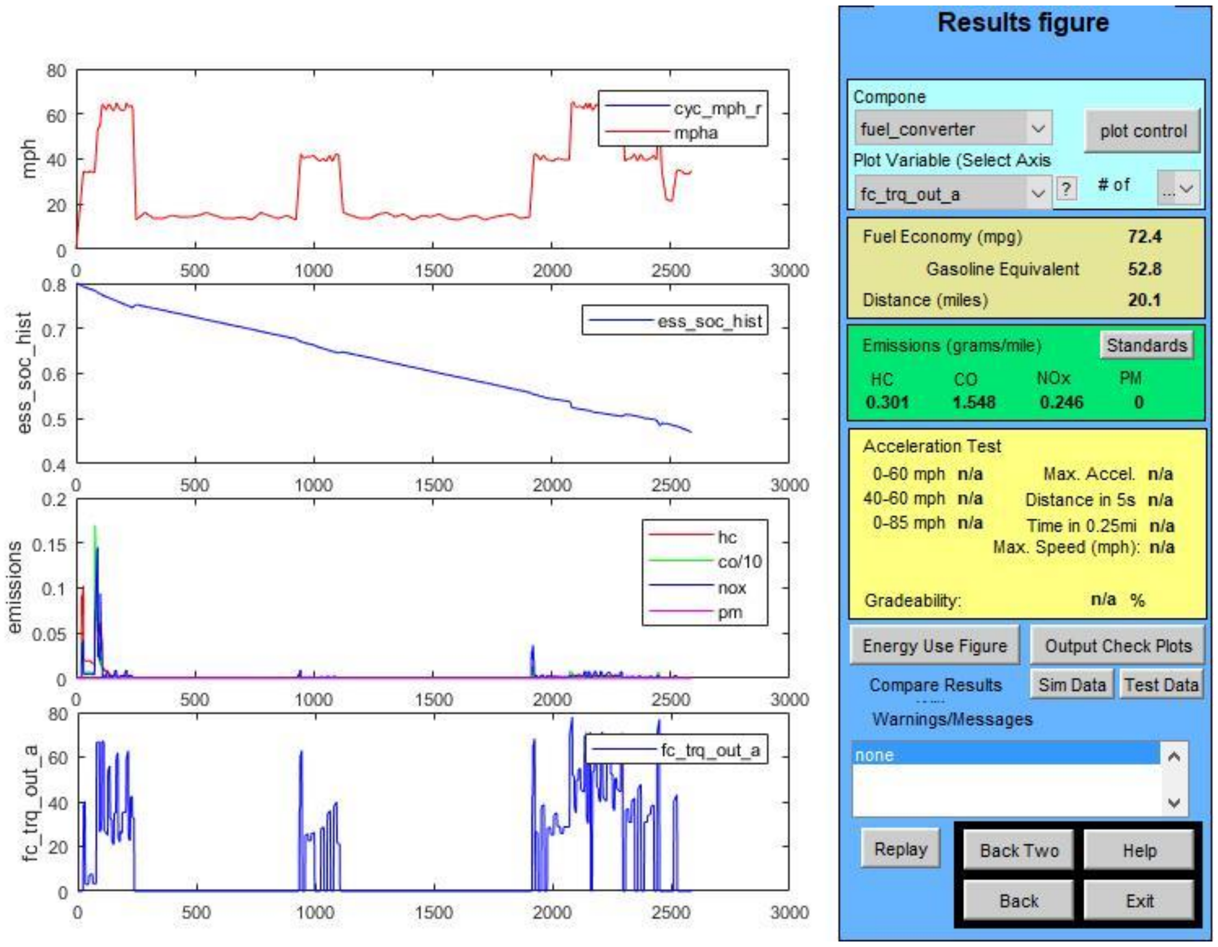
Acceleration Test

0-60 mph	n/a	Max. Accel.	n/a
40-60 mph	n/a	Distance in 5s	n/a
0-85 mph	n/a	Time in 0.25mi	n/a
		Max. Speed (mph):	n/a

Gradeability: n/a %

```
Missed Trace by > 2 mph (3.2 km/h)
Trace Miss Analysis:
```

Figure 45: Route 5 TF control strategy ADVISOR simulation results



Fuel Economy (mpg) 72.4
Gasoline Equivalent 52.8
Distance (miles) 20.1

Emissions (grams/mile)

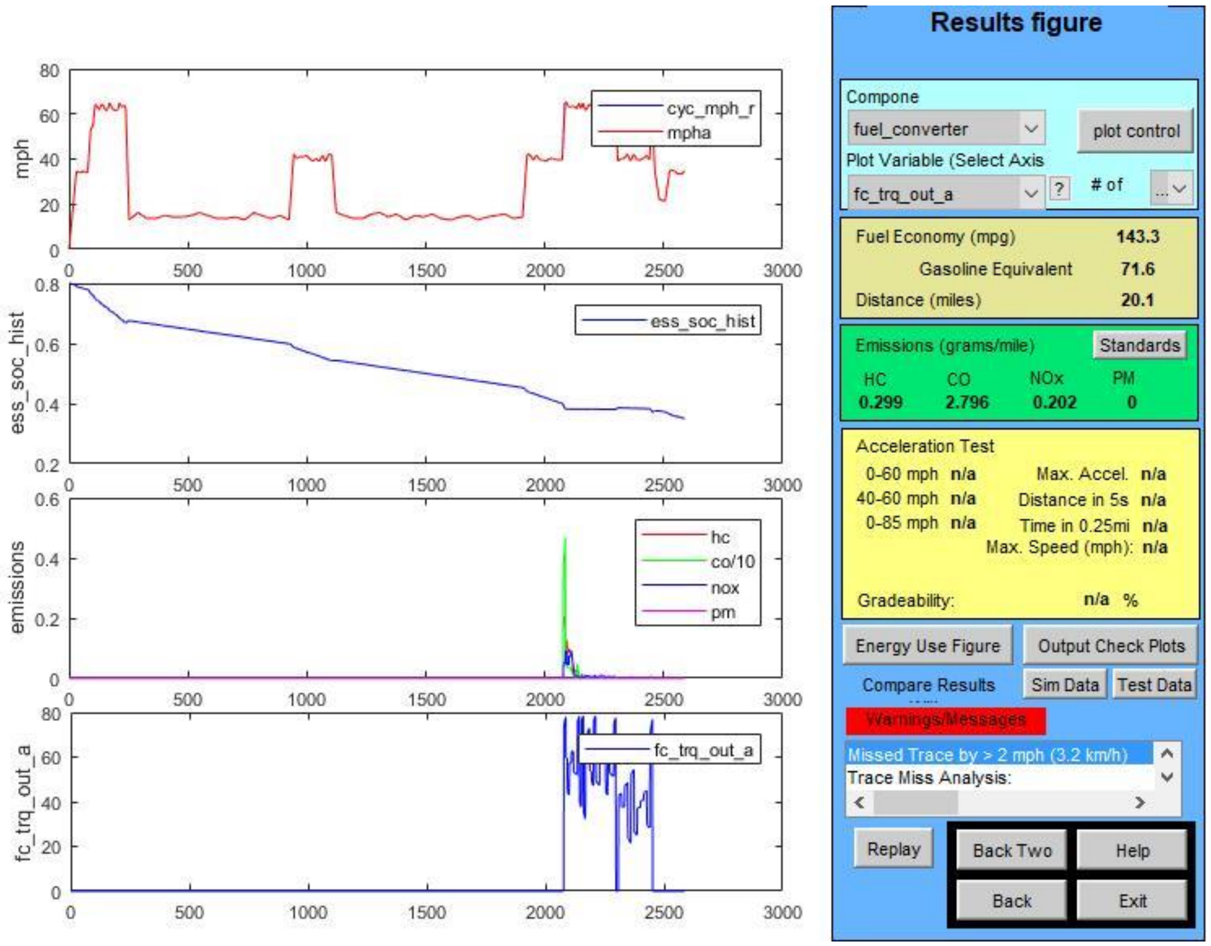
HC	CO	NOx	PM
0.301	1.548	0.246	0

Acceleration Test

0-60 mph	n/a	Max. Accel.	n/a
40-60 mph	n/a	Distance in 5s	n/a
0-85 mph	n/a	Time in 0.25mi	n/a
		Max. Speed (mph):	n/a

Gradeability: n/a %

Figure 46: Route 5 CS control strategy ADVISOR simulation results



Compone: fuel_converter

Plot Variable (Select Axis: fc_trq_out_a

of

Fuel Economy (mpg)	143.3
Gasoline Equivalent	71.6
Distance (miles)	20.1

Emissions (grams/mile)

HC	CO	NOx	PM
0.299	2.796	0.202	0

Acceleration Test

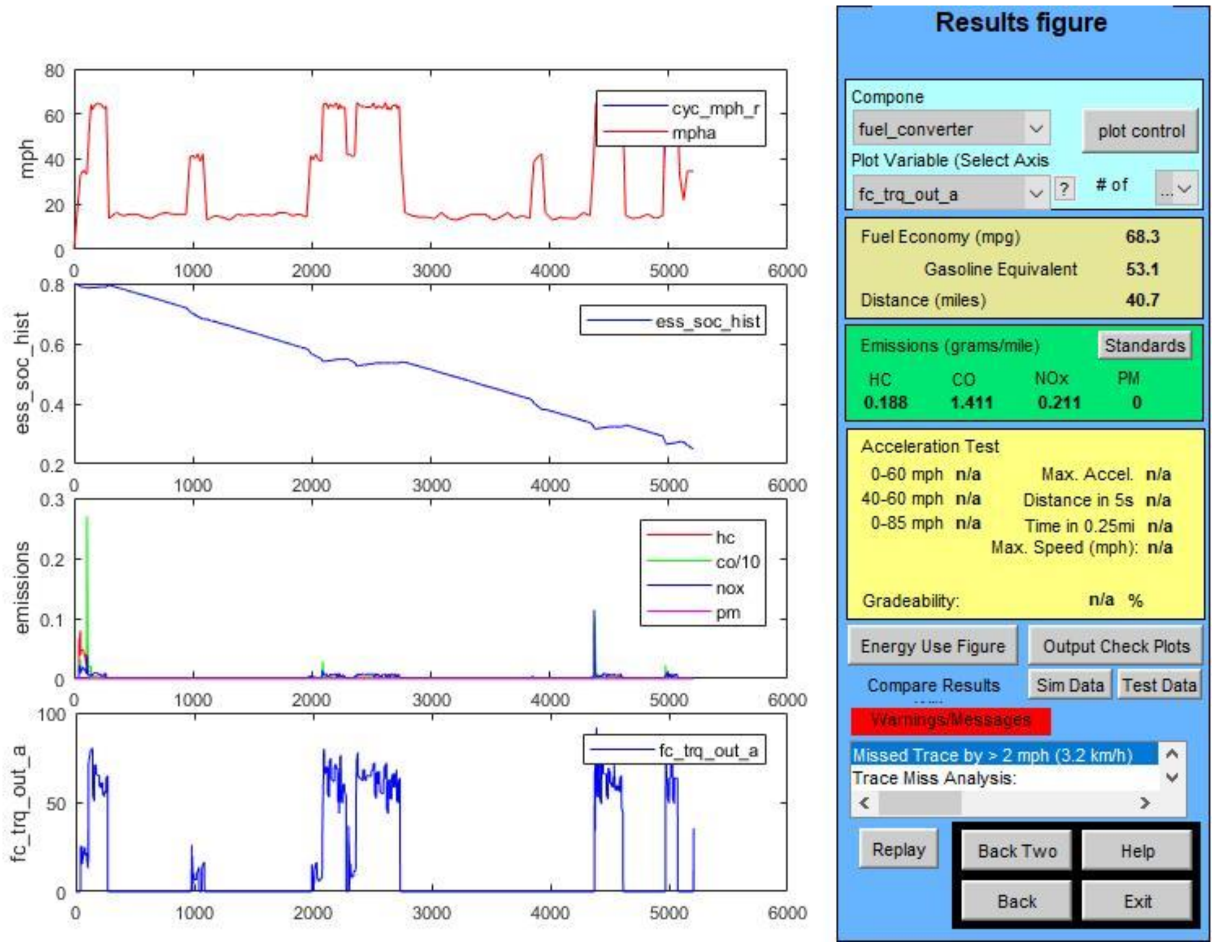
0-60 mph	n/a	Max. Accel.	n/a
40-60 mph	n/a	Distance in 5s	n/a
0-85 mph	n/a	Time in 0.25mi	n/a
		Max. Speed (mph):	n/a

Gradeability: n/a %

Missed Trace by > 2 mph (3.2 km/h)
Trace Miss Analysis:

Figure 47: Route 5 CD control strategy ADVISOR simulation results

7.1.1.6. Route 6



Fuel Economy (mpg) 68.3
Gasoline Equivalent 53.1
Distance (miles) 40.7

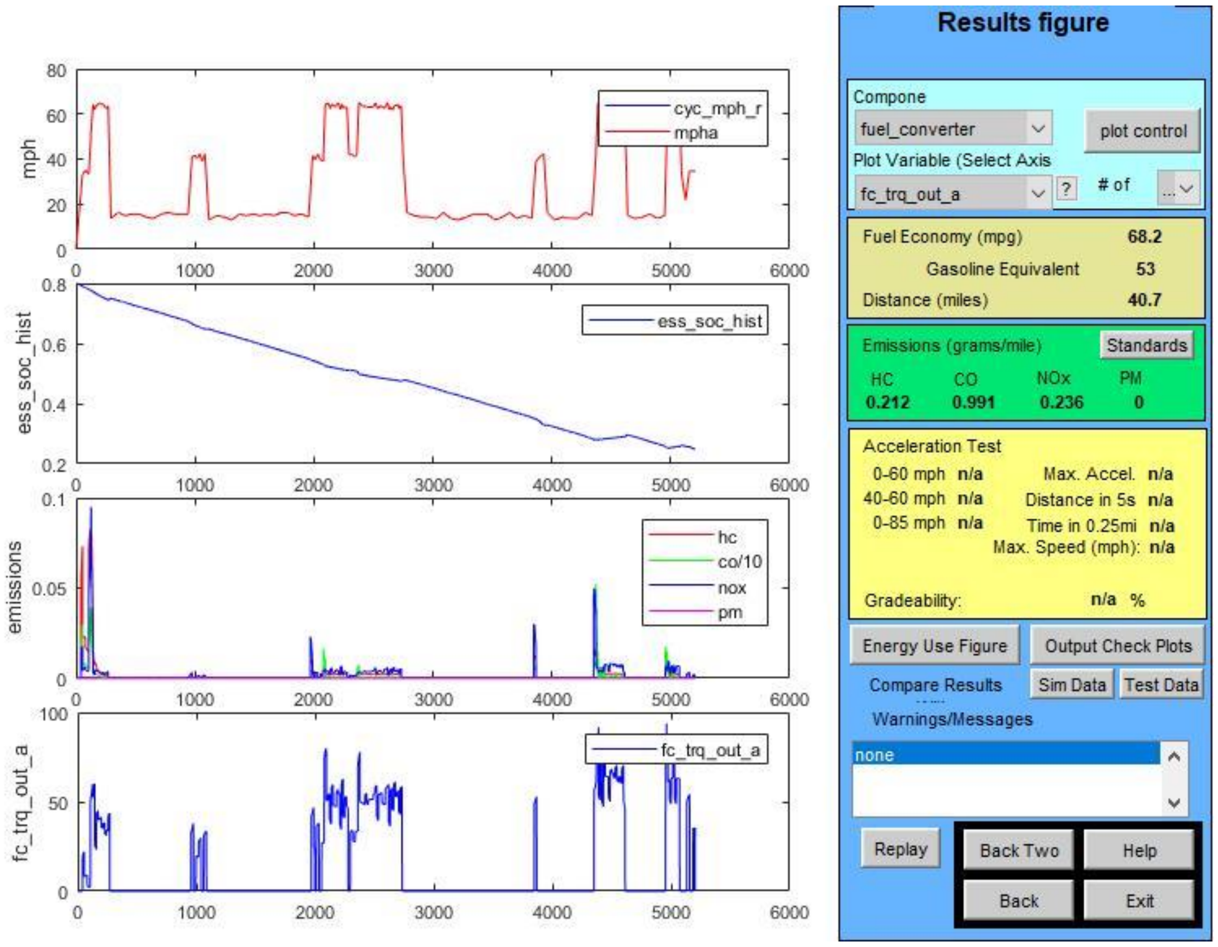
Emissions (grams/mile)

HC	CO	NOx	PM
0.188	1.411	0.211	0

Acceleration Test
0-60 mph n/a
40-60 mph n/a
0-85 mph n/a
Max. Accel. n/a
Distance in 5s n/a
Time in 0.25mi n/a
Max. Speed (mph): n/a
Gradeability: n/a %

Missed Trace by > 2 mph (3.2 km/h)
Trace Miss Analysis:

Figure 48: Route 6 TF control strategy ADVISOR simulation results



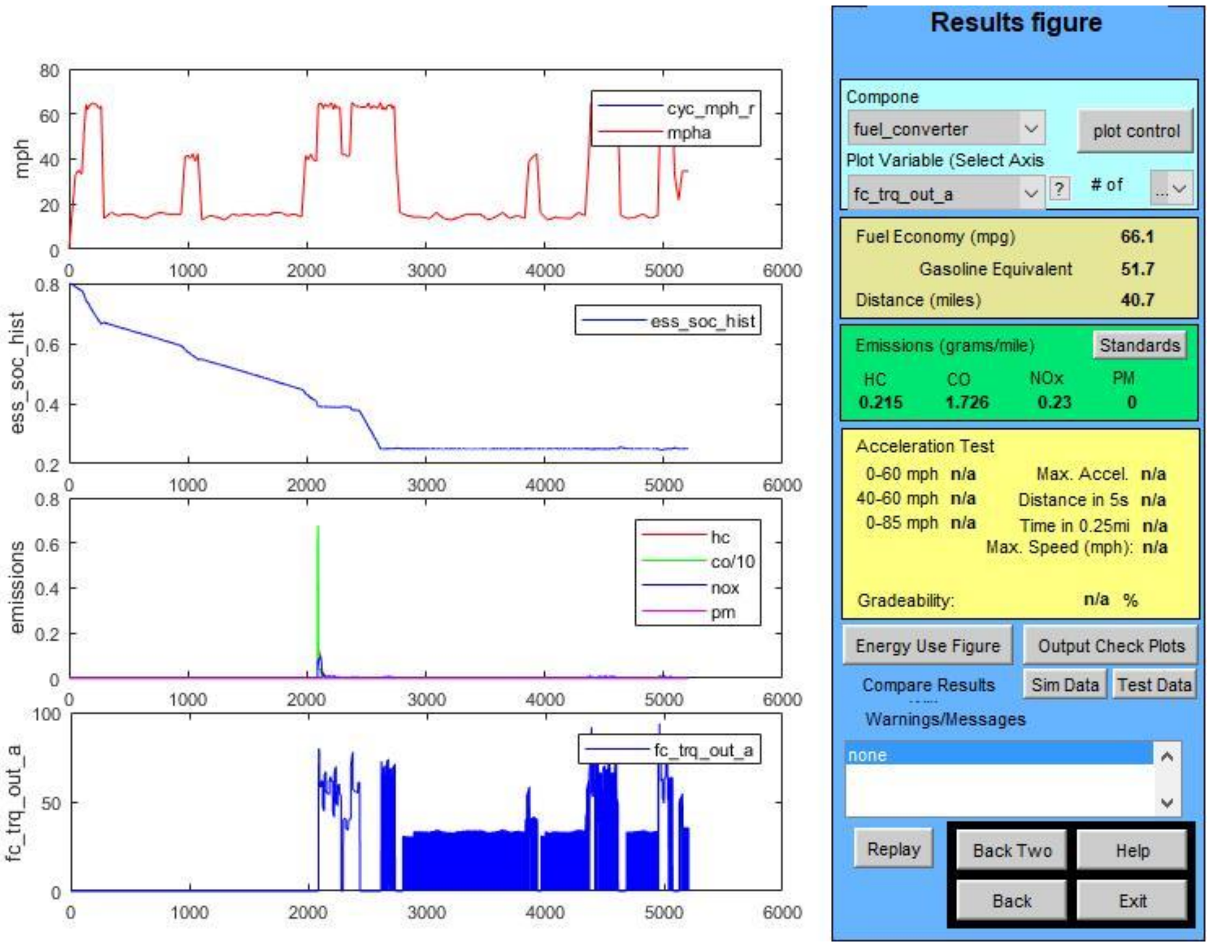
Fuel Economy (mpg) 68.2
Gasoline Equivalent 53
Distance (miles) 40.7

Emissions (grams/mile)

HC	CO	NOx	PM
0.212	0.991	0.236	0

Acceleration Test
0-60 mph n/a
40-60 mph n/a
0-85 mph n/a
Max. Accel. n/a
Distance in 5s n/a
Time in 0.25mi n/a
Max. Speed (mph): n/a
Gradeability: n/a %

Figure 49: Route 6 CS control strategy ADVISOR simulation results



Results figure

fuel_converter

fc_trq_out_a

Fuel Economy (mpg)	66.1
Gasoline Equivalent	51.7
Distance (miles)	40.7

Emissions (grams/mile)

HC	CO	NOx	PM
0.215	1.726	0.23	0

Acceleration Test

0-60 mph	n/a	Max. Accel.	n/a
40-60 mph	n/a	Distance in 5s	n/a
0-85 mph	n/a	Time in 0.25mi	n/a
		Max. Speed (mph):	n/a

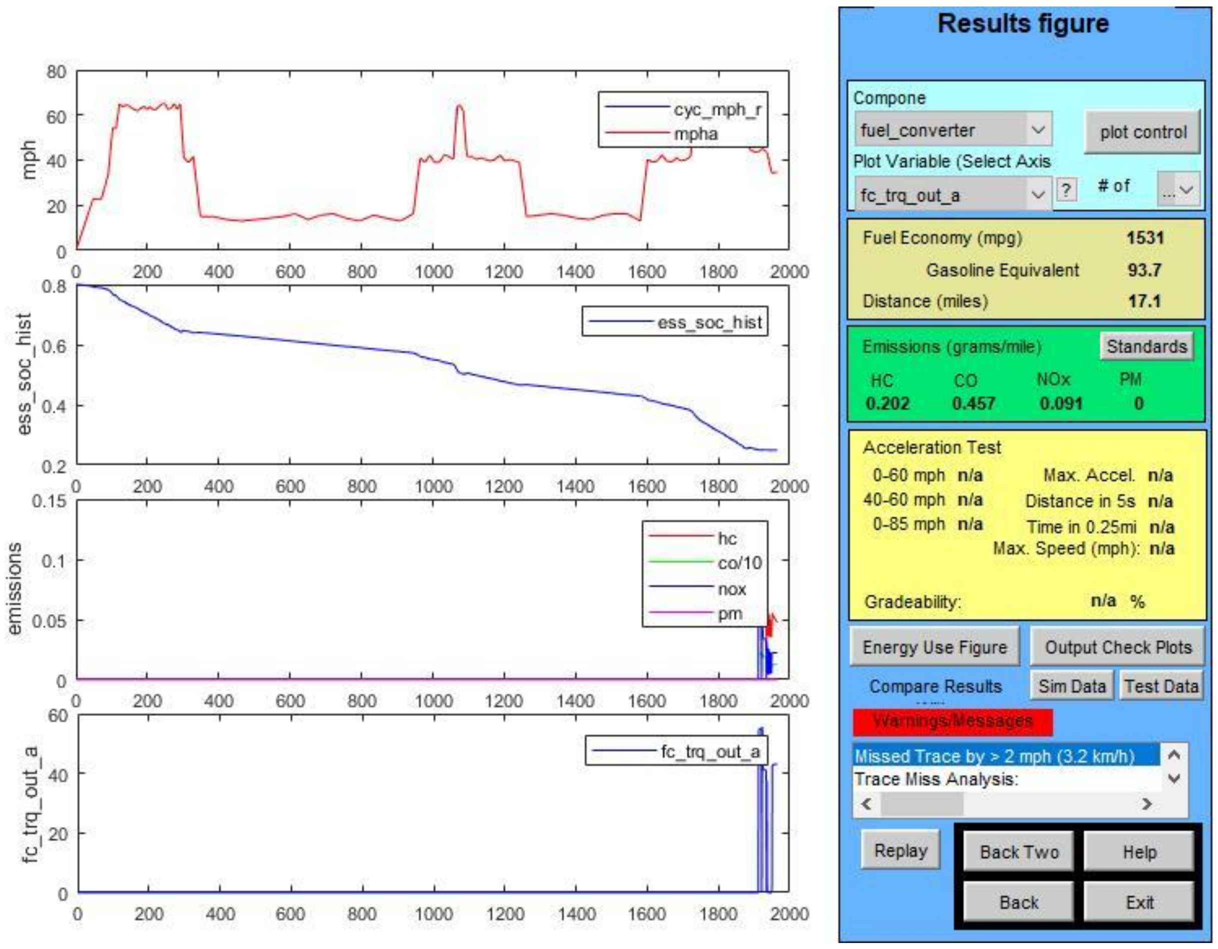
Gradeability: n/a %

Warnings/Messages

none

Figure 50: Route 6 CD control strategy ADVISOR simulation results

7.1.1.7. Route 7



Compone: fuel_converter

Plot Variable (Select Axis: fc_trq_out_a

Fuel Economy (mpg)	1531
Gasoline Equivalent	93.7
Distance (miles)	17.1

Emissions (grams/mile)

HC	CO	NOx	PM
0.202	0.457	0.091	0

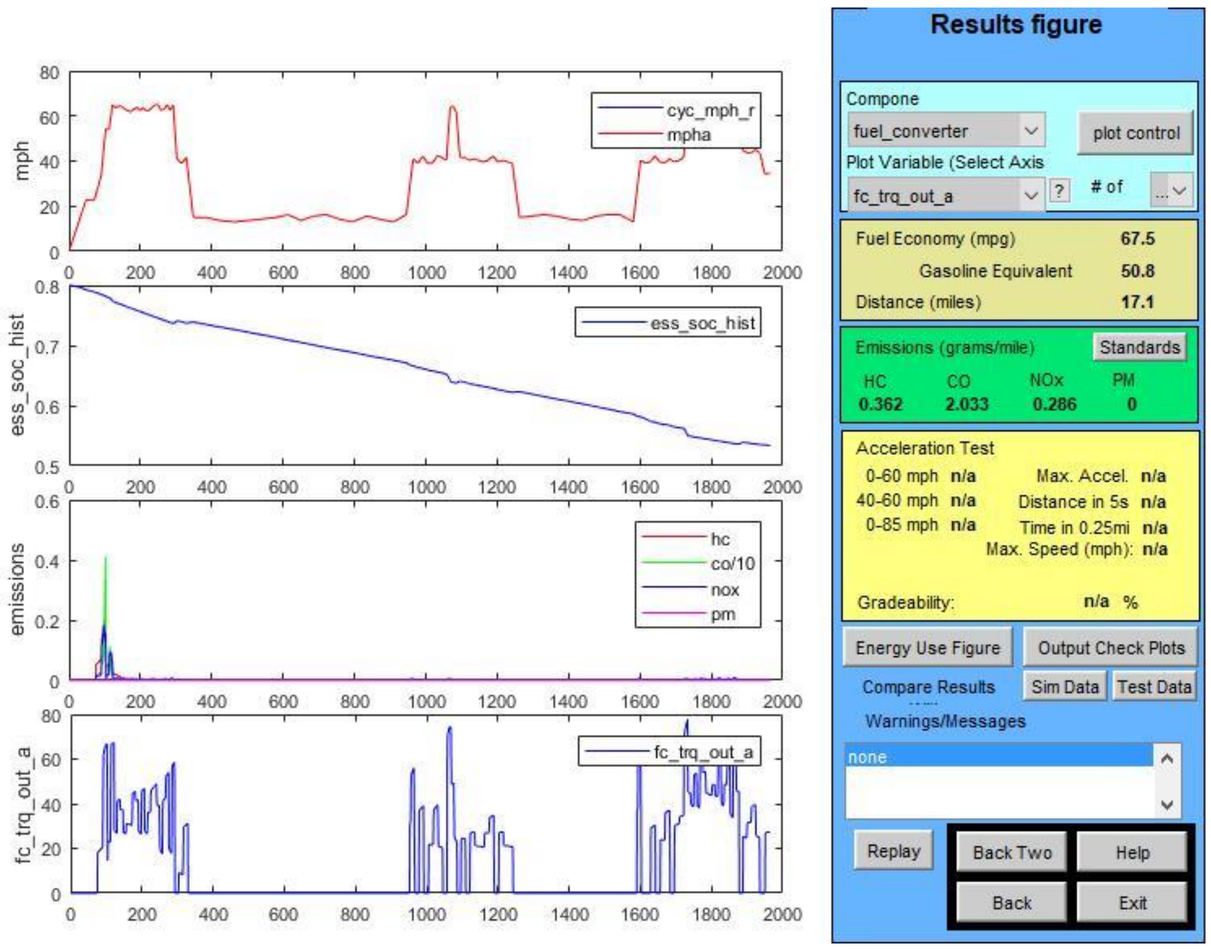
Acceleration Test

0-60 mph	n/a	Max. Accel.	n/a
40-60 mph	n/a	Distance in 5s	n/a
0-85 mph	n/a	Time in 0.25mi	n/a
		Max. Speed (mph):	n/a

Gradeability: n/a %

Missed Trace by > 2 mph (3.2 km/h)
Trace Miss Analysis:

Figure 51: Route 7 TF control strategy ADVISOR simulation results



Compone: fuel_converter

Plot Variable (Select Axis: fc_trq_out_a

Fuel Economy (mpg)	67.5
Gasoline Equivalent	50.8
Distance (miles)	17.1

Emissions (grams/mile)

HC	CO	NOx	PM
0.362	2.033	0.286	0

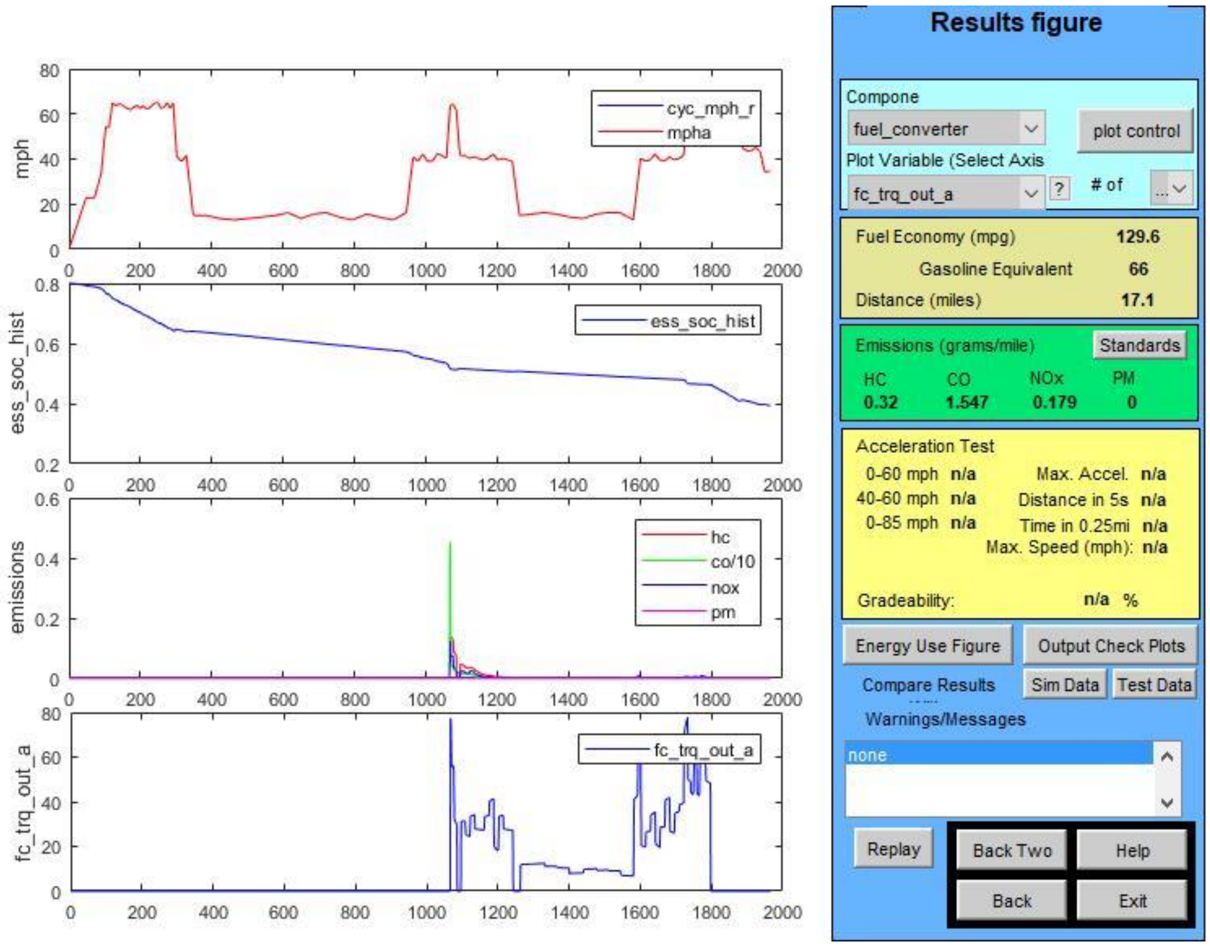
Acceleration Test

0-60 mph	n/a	Max. Accel.	n/a
40-60 mph	n/a	Distance in 5s	n/a
0-85 mph	n/a	Time in 0.25mi	n/a
		Max. Speed (mph):	n/a

Gradeability: n/a %

Warnings/Messages: none

Figure 52: Route 7 CS control strategy ADVISOR simulation results



fuel_converter

fc_trq_out_a

Fuel Economy (mpg) 129.6

Gasoline Equivalent 66

Distance (miles) 17.1

Emissions (grams/mile)

HC	CO	NOx	PM
0.32	1.547	0.179	0

Acceleration Test

0-60 mph n/a
40-60 mph n/a
0-85 mph n/a
Max. Accel. n/a
Distance in 5s n/a
Time in 0.25mi n/a
Max. Speed (mph): n/a

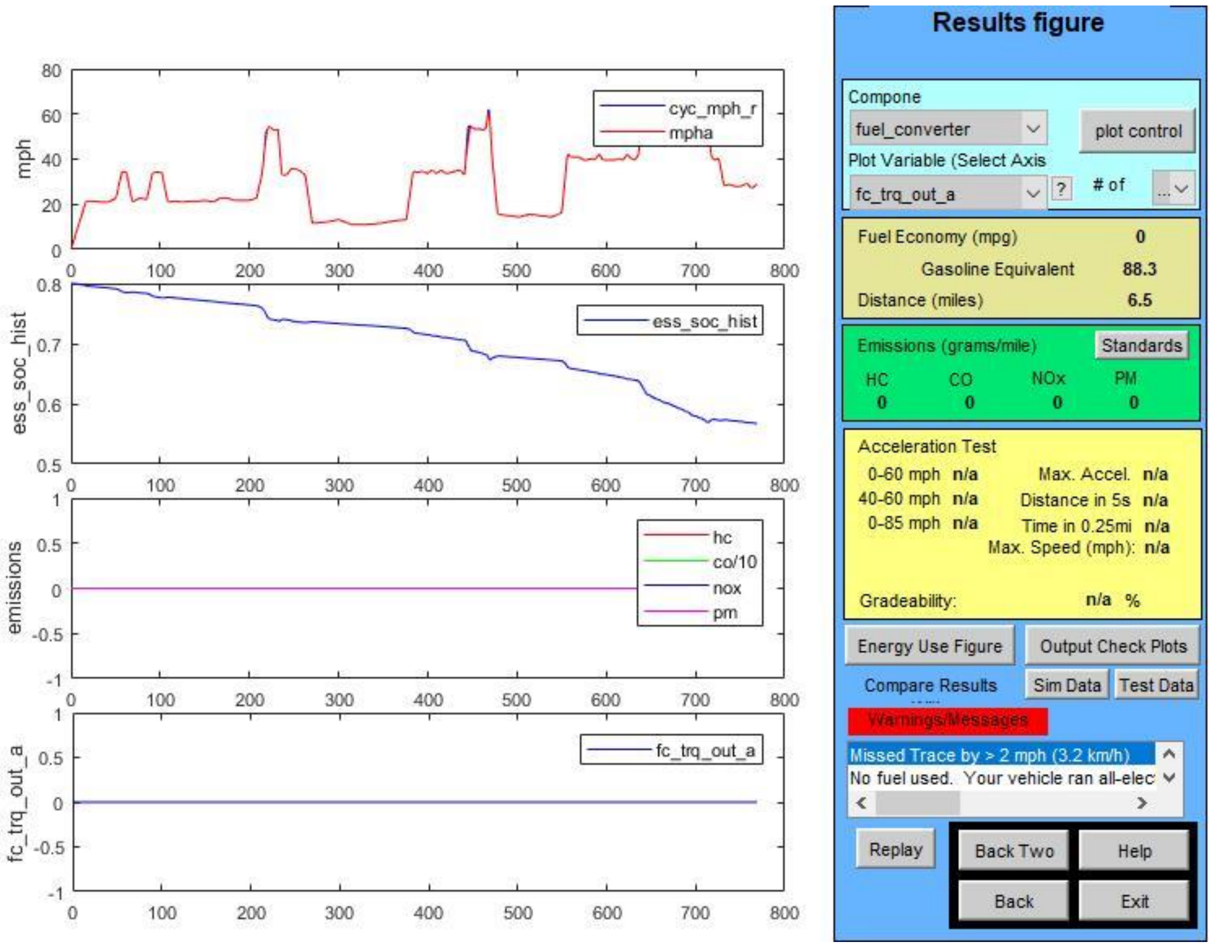
Gradeability: n/a %

Warnings/Messages

none

Figure 53: Route 7 CD control strategy ADVISOR simulation results

7.1.1.8. Route 8



Compone: fuel_converter

Plot Variable (Select Axis: fc_trq_out_a

Fuel Economy (mpg)	0
Gasoline Equivalent	88.3
Distance (miles)	6.5

Emissions (grams/mile)

HC	CO	NOx	PM
0	0	0	0

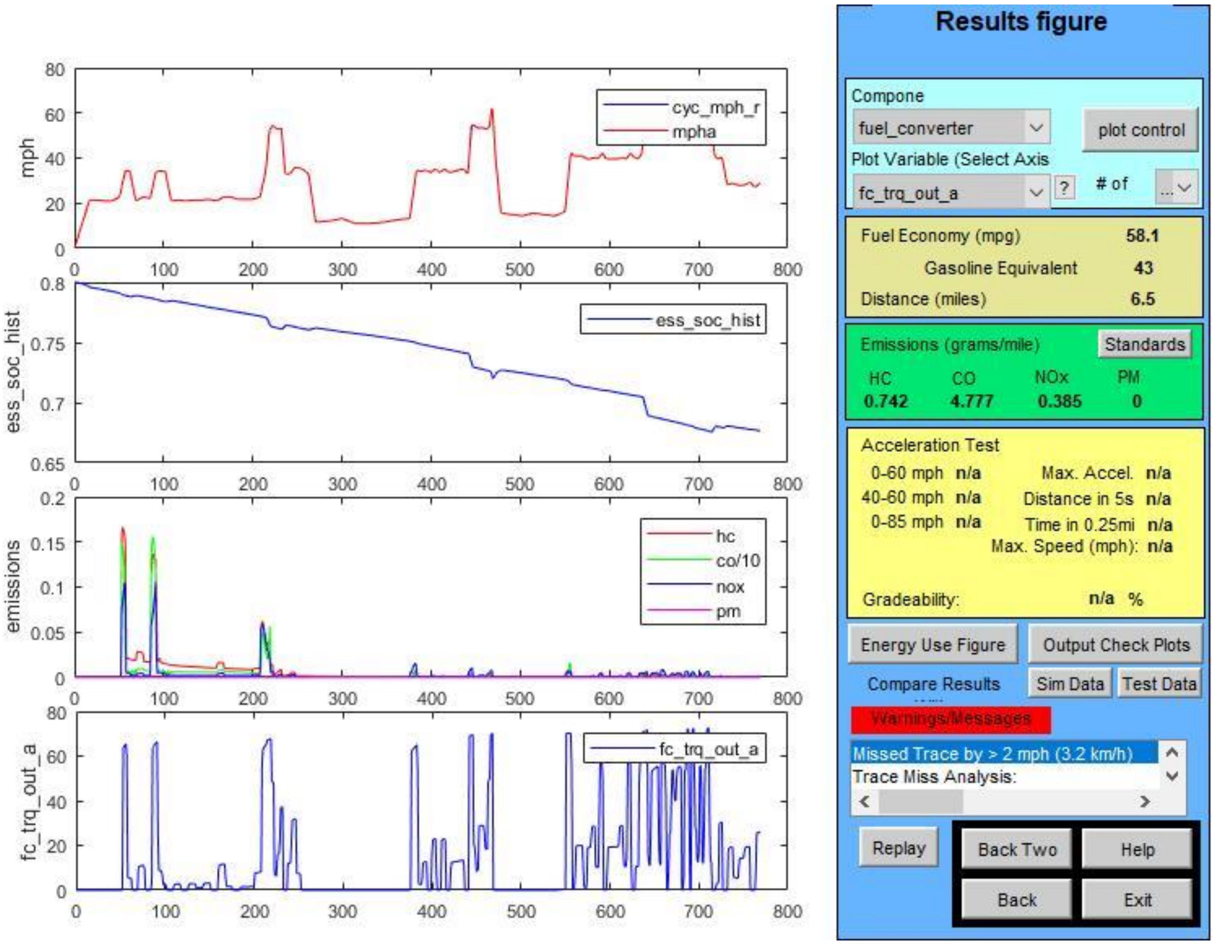
Acceleration Test

0-60 mph	n/a	Max. Accel.	n/a
40-60 mph	n/a	Distance in 5s	n/a
0-85 mph	n/a	Time in 0.25mi	n/a
		Max. Speed (mph):	n/a

Gradeability: n/a %

```
Missed Trace by > 2 mph (3.2 km/h)
No fuel used.  Your vehicle ran all-elec
```

Figure 54: Route 8 TF control strategy ADVISOR simulation results



Fuel Economy (mpg) 58.1
Gasoline Equivalent 43
Distance (miles) 6.5

Emissions (grams/mile)

HC	CO	NOx	PM
0.742	4.777	0.385	0

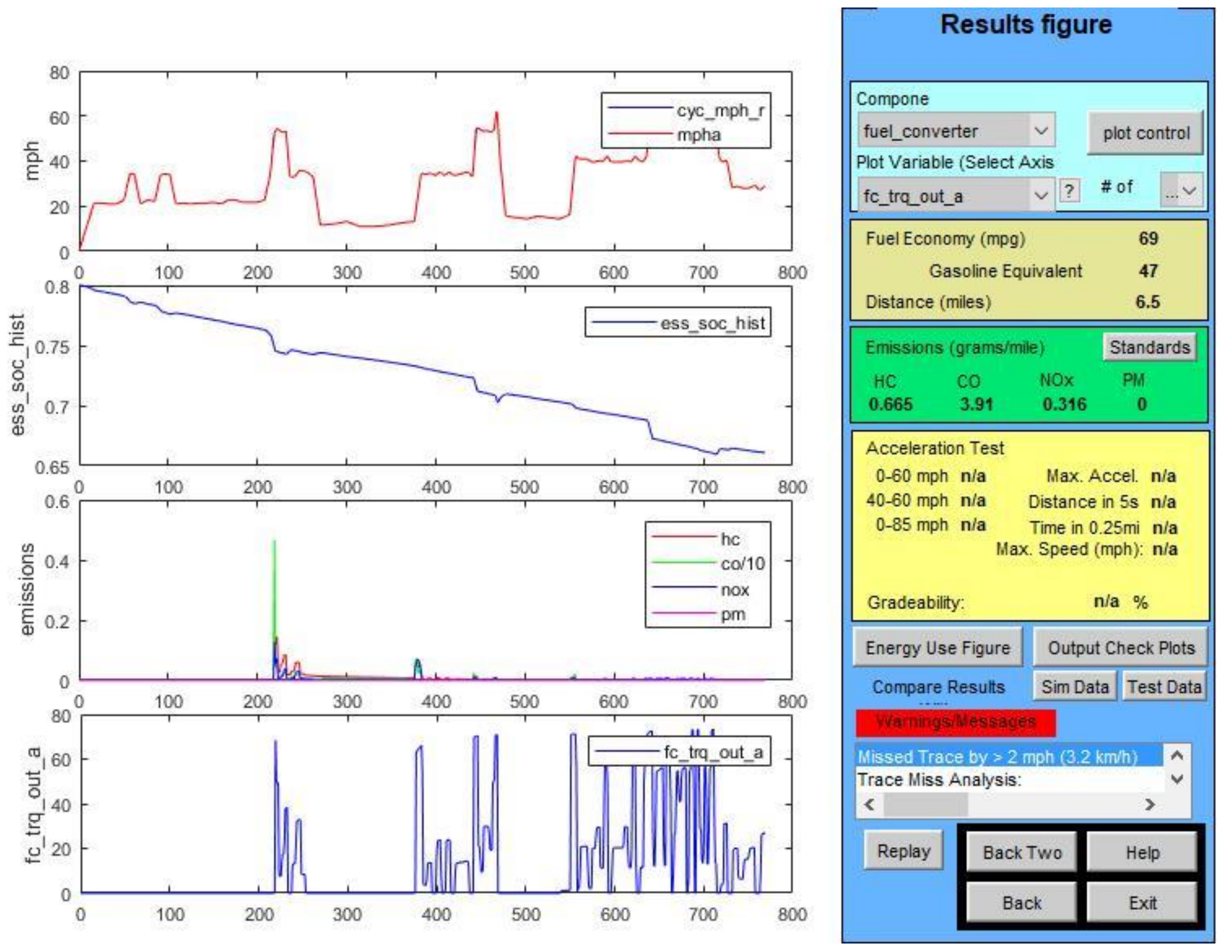
Acceleration Test

0-60 mph	n/a	Max. Accel.	n/a
40-60 mph	n/a	Distance in 5s	n/a
0-85 mph	n/a	Time in 0.25mi	n/a
		Max. Speed (mph):	n/a

Gradeability: n/a %

Missed Trace by > 2 mph (3.2 km/h)
Trace Miss Analysis:

Figure 55: Route 8 CS control strategy ADVISOR simulation results



Compone: fuel_converter
Plot Variable (Select Axis: fc_trq_out_a

Fuel Economy (mpg)	69
Gasoline Equivalent	47
Distance (miles)	6.5

Emissions (grams/mile)

HC	CO	NOx	PM
0.665	3.91	0.316	0

Acceleration Test

0-60 mph	n/a	Max. Accel.	n/a
40-60 mph	n/a	Distance in 5s	n/a
0-85 mph	n/a	Time in 0.25mi	n/a
		Max. Speed (mph):	n/a

Gradeability: n/a %

Warnings/Messages

Missed Trace by > 2 mph (3.2 km/h)
Trace Miss Analysis:

Figure 56: Route 8 CD control strategy ADVISOR simulation results

7.1.2. Original Trajectory Forecasting Code

```
%% Route Setup Version 5 %%
clear; clc;

%%Inputs
dist = input('Length of route in miles: ');

if dist < 11
    interval_dist = 0.05;
elseif (dist >= 11) && (dist < 26)
    interval_dist = 0.15;
else
    interval_dist = 0.25;
end

fprintf('\n')
while 1
    speed_lim_per = input('Percentages of route speed limit sections: ')/100;
    if (sum(speed_lim_per)-1) > 0.01
        disp('Percentages must add to 100%')
    else
        break
    end
end

fprintf('\n')
while 1
    speed_lim_cond = input('Corresponding speed limit conditions: ');
    if length(speed_lim_cond) ~= length(speed_lim_per)
        disp('Number of speed limit conditions must match number of speed
limit sections')
    else
        break
    end
end

fprintf('\n')
while 1
    traffic_per = input('Percentages of route traffic sections: ')/100;
    if (sum(traffic_per)-1) > 0.01
        disp('Percentages must add to 100%')
    else
        break
    end
end

fprintf('\n')
while 1
    traffic_cond = input('Corresponding traffic conditions: ');
    if length(traffic_cond) ~= length(traffic_per)
        disp('Number of traffic conditions must match number of traffic
sections')
    else
        break
    end
end

fprintf('\n')
while 1
    slope_per = input('Percentages of route slope sections: ')/100;
    if (sum(slope_per)-1) > 0.01
        disp('Percentages must add to 100%')
    else
        break
    end
end

fprintf('\n')
while 1
    slope_cond = input('Corresponding slope conditions: ');
    if length(slope_cond) ~= length(slope_per)
        disp('Number of slope conditions must match number of slope
sections')
    else
```

```
        break
    end
end

fprintf('\n')
Slope_Option = input('Slope consideration, 1 for On or 2 for Off: ');

%%Route Data (miles)
if (rem(dist,interval_dist) ~= 0)
    segments = (dist-rem(dist,interval_dist))/interval_dist + 1;
else
    segments = round(dist/interval_dist);
end

route = zeros(1,(segments+1));
route(end) = dist;

for i = 1:(segments)
   route(i) = (i-1)*interval_dist;
end

%%Speed Limit Data (mph)
speed_lim = zeros(1,segments+1);
speed_lim_marker = 0;

Car_speed = zeros(1,segments+1);

for i = 1:length(speed_lim_per)
    for j = 1:length(speed_lim)
        if i == 1
           if route(j) <= round((dist*speed_lim_per(1)),2)
               speed_lim(j) = speed_lim_cond(1);
               if j > 1
                   if speed_lim(j) == 1
                       Car_speed(j) = 15;
                   elseif speed_lim(j) == 2
                       Car_speed(j) = 25;
                   elseif speed_lim(j) == 3
                       Car_speed(j) = 35;
                   elseif speed_lim(j) == 4
                       Car_speed(j) = 45;
                   elseif speed_lim(j) == 5
                       Car_speed(j) = 55;
                   elseif speed_lim(j) == 6
                       Car_speed(j) = 65;
                   elseif speed_lim(j) == 7
                       Car_speed(j) = 75;
                   end
               end
           end
        else
           if (route(j) > (speed_lim_marker)) && (route(j) <=
round((speed_lim_marker + dist*speed_lim_per(i)),2))
               speed_lim(j) = speed_lim_cond(i);
               if speed_lim(j) == 1
```

```
                    Car_speed(j) = 15;
                elseif speed_lim(j) == 2
                    Car_speed(j) = 25;
                elseif speed_lim(j) == 3
                    Car_speed(j) = 35;
                elseif speed_lim(j) == 4
                    Car_speed(j) = 45;
                elseif speed_lim(j) == 5
                    Car_speed(j) = 55;
                elseif speed_lim(j) == 6
                    Car_speed(j) = 65;
                elseif speed_lim(j) == 7
                    Car_speed(j) = 75;
                end
            end
        end
    end
    speed_lim_marker = speed_lim_marker + dist*speed_lim_per(i);
end

%%Traffic Data
traffic = zeros(1,segments+1);
traf_marker = 0;

for i = 1:length(traffic_per)
    for j = 1:length(traffic)
        if i == 1
            if route(j) <= round((dist*traffic_per(1)),2)
                traffic(j) = traffic_cond(1);
                if traffic(j) == 1
                    Car_speed(j) = Car_speed(j)*(randi([95,100],1,1)/100);
                elseif traffic(j) == 2
                    Car_speed(j) = Car_speed(j)*(randi([60,65],1,1)/100);
                else
                    Car_speed(j) = Car_speed(j)*(randi([20,25],1,1)/100);
                end
            end
        else
            if (route(j) > (traf_marker)) && (route(j) <= round((traf_marker +
dist*traffic_per(i)),2))
                traffic(j) = traffic_cond(i);
                if traffic(j) == 1
                    Car_speed(j) = Car_speed(j)*(randi([95,100],1,1)/100);
                elseif traffic(j) == 2
                    Car_speed(j) = Car_speed(j)*(randi([60,65],1,1)/100);
                else
                    Car_speed(j) = Car_speed(j)*(randi([20,25],1,1)/100);
                end
            end
        end
    end
    traf_marker = traf_marker + dist*traffic_per(i);
end

%%Slope Data
slope = zeros(1,segments+1);
```

```
slope_marker = 0;
road_slope = zeros(1,segments+1);

for i = 1:length(slope_per)
    for j = 1:length(slope)
        if i == 1
           if route(j) <= round((dist*slope_per(1)),2)
               slope(j) = slope_cond(1);
               if slope(j) == 1
                   road_slope(j) = -0.05;
               elseif slope(j) == 2
                   road_slope(j) = 0;
               else
                   road_slope(j) = 0.05;
               end
           end
        else
           if (route(j) > (slope_marker)) && (route(j) <= round((slope_marker
+ dist*slope_per(i)),2))
               slope(j) = slope_cond(i);
               if slope(j) == 1
                   road_slope(j) = -0.05;
               elseif slope(j) == 2
                   road_slope(j) = 0;
               else
                   road_slope(j) = 0.05;
               end
           end
        end
    end
    slope_marker = slope_marker + dist*slope_per(i);
end

%%Time Calculation
Time = zeros(1,segments+1);

for i = 2:length(Time)
    Time(i) = Time(i-1) + interval_dist/((1/3600)*((Car_speed(i)+Car_speed(i-
1))/2));
end

cyc_mph = [Time' Car_speed'];
cyc_slope = [1609.34*route' road_slope'];
save('C:\Users\Joseph\Documents\advisor\data\drive_cycle\CYC_TrajFore_Fuz.mat
','cyc_mph','cyc_slope','Slope_Option')
save('route.mat')

%%Plots
figure
plot(route,Car_speed)
title('Car Speed vs. Distance')
xlabel('Distance (miles)')
ylabel('Car Speed (mph)')

figure
plot(Time,Car_speed)
```

```
title('Car Speed vs. Time')
xlabel('Time (secs)')
ylabel('Car Speed (mph)')

figure
plot(route,road_slope)
title('Grade vs. Distance')
xlabel('Distance (miles)')
ylabel('Grade (mph)')

%% Path Identifier Version 5 %%
clear; clc;

fprintf('\n')
Bat_Type = input('1 if Pb battery, 2 if Li battery: '); fprintf('\n')
Ah_Cap = input('Battery Capacity (ah): '); fprintf('\n')
Max_SOC = input('Max SOC: '); fprintf('\n')
Initial_SOC = input('Initial SOC: '); fprintf('\n')
Lowest_SOC = input('Minimum SOC: '); fprintf('\n')
SOC_Coeff = input('SOC Coefficient Adjustment: '); fprintf('\n')
Drive_Type = input('1 for EV/ICE capability, 2  EV/Blended/ICE capability:
'); fprintf('\n')
load('route.mat')

%%Assigning Section Priority
Ant_Spd_Matrix = [15 9.75 3.75; % used numbers in routeV4 code that give the
carspeed profile
                  25 16.25 6.25   %   1  0.65   0.25
                  35 22.75 8.75   %15 -  ----   ----
                  45 29.25 11.25  %25 -  ----   ----
                  55 35.75 13.75  %35 -  ----   ----
                  65 42.25 16.25  %45 -  ----   ----
                  75 48.75 18.75];%55 -  ----   ----
                                  %65 -  ----   ----
                                  %75 -  ----   ----

Priority_Matrix = [3 3 3; % 1- least priority (ICE); 2- medium priority
(Blended); 3- highest priority (EV)
                   3 3 3; % v <= 25 is EV Mode, 25 < v < 55 Blended Mode, v
>= 55 ICE Mode
                   2 3 3;
                   2 2 3;
                   1 2 3;
                   1 2 3;
                   1 2 3];

Matrix = [route; traffic; speed_lim; slope; zeros(size(route));
zeros(size(route))];

for i = 1:length(route)
    if Matrix(4,i) == 3
        Matrix(5,i) = 1; %high road grade means this should be of least
priority
    else
        Matrix(5,i) = Priority_Matrix(Matrix(3,i),Matrix(2,i));
    end
end

%%Calculating Section Size and Distance
Section_Size = 0;

j = 0;
for i = 2:length(Matrix(5,:))
    if Matrix(5,i) == Matrix(5,i-1)
        j = j + 1;
    else
        Section_Size = [Section_Size, j]; %#ok<AGROW>
        j = 1;
    end
end

Section_Size = [Section_Size(2:end), (i-sum(Section_Size)-1)];
Section_Dist = interval_dist*Section_Size;

%%SOC Needed per Section
ampHrs_per_Segment = zeros(size(route));
%%%%%%%%%%%%%%%%%%
for i = 2:length(ampHrs_per_Segment)
    if Matrix(5,i) == 3 %SOC equation for priority 3
        if Matrix(4,i) == 3
            ampHrs_per_Segment(i) =
interval_dist*2*0.87*(0.0198*(Ant_Spd_Matrix(Matrix(3,i),Matrix(2,i))) +
0.1689); %amount required double for uphill
        else
            ampHrs_per_Segment(i) =
interval_dist*0.87*(0.0198*(Ant_Spd_Matrix(Matrix(3,i),Matrix(2,i))) +
0.1689);
        end
    elseif Matrix(5,i) == 2 %SOC equation for priority 2
        if Matrix(4,i) == 3
            ampHrs_per_Segment(i) =
interval_dist*2*0.42*(0.0198*(Ant_Spd_Matrix(Matrix(3,i),Matrix(2,i))) +
0.1689); %amount required double for uphill
        else
            ampHrs_per_Segment(i) =
interval_dist*0.42*(0.0198*(Ant_Spd_Matrix(Matrix(3,i),Matrix(2,i))) +
0.1689);
        end
    else
        if Matrix(4,i) == 3 %SOC equation for priority 1
            ampHrs_per_Segment(i) =
interval_dist*2*0.34*(0.0198*(Ant_Spd_Matrix(Matrix(3,i),Matrix(2,i))) +
0.1689); %amount required double for uphill
        else
            ampHrs_per_Segment(i) =
interval_dist*0.34*(0.0198*(Ant_Spd_Matrix(Matrix(3,i),Matrix(2,i))) +
0.1689);
        end
    end
end
%%%%%%%%%%%%%%%%%%%%%
```

```
ampHrs_per_Segment = ampHrs_per_Segment(2:end);

Section_AmpHrs = zeros(size(Section_Size));

for i = 1:length(Section_AmpHrs)
    if i == 1
        Section_AmpHrs(i) = sum(ampHrs_per_Segment(1:Section_Size(i)));
    else
        Section_AmpHrs(i) = sum(ampHrs_per_Segment((1+sum(Section_Size(1:(i-
1)))):sum(Section_Size(1:i))));
    end
end

if Bat_Type == 1
    Section_SOC = Section_AmpHrs/(Ah_Cap*0.55);
else
    Section_SOC = Section_AmpHrs/Ah_Cap;
end

%%Sorting Sections by Priority and Length
Section_Priority = size(Section_Size);

for i = 1:length(Section_Size)
    Section_Priority(i) = Matrix(5,(1+sum(Section_Size(1:i))));
end

Section_Sort = [Section_Priority; Section_Dist; Section_SOC]';
[Section_Sort,index] = sortrows(Section_Sort,'descend');

%%Setting Section SOC_min
Section_SOC_Mins = zeros(1,length(index));
if Initial_SOC == Max_SOC
    Initial_SOC = Initial_SOC - 0.005;
end

allowed_SOC = Initial_SOC - Lowest_SOC;

%%% Determine settings for determining sections' SOC_Min
if Drive_Type == 1
    SOC_setting = 1;
else
    if allowed_SOC >= (round(sum(Section_Sort(:,3)),3) - 0.125) %Enough for
whole route in all electric mode; launch speed that includes all section
speeds
        SOC_setting = 1;
    else
        k_EV = length(find(Section_Sort(:,1) == 3));
        k_Bl = length(find(Section_Sort(:,1) == 2));
        if (allowed_SOC < (round(sum(Section_Sort(:,3)),3)) - 0.125) &&
(allowed_SOC >= round(sum(Section_Sort(1:(k_EV+k_Bl),3)),3)) %Enough for EV
and Blended suggested sections in all electric mode; launch speed that
includes EV and Blended section speeds
            SOC_setting = 2;
        elseif (allowed_SOC < round(sum(Section_Sort(1:(k_EV+k_Bl),3)),3)) &&
(allowed_SOC >= round(sum(Section_Sort(1:k_EV,3)),3)) %Enough for just EV
```

```
suggested sections in all eletric mode; launch speed that includes EV section
speeds
            SOC_setting = 3;
        elseif allowed_SOC < round(sum(Section_Sort(1:k_EV,3)),3) %SOC will
run out in the EV suggested sections; launch speed that includes EV section
speeds
            SOC_setting = 4;
        end
    end
end

%%% Different SOC settings for choosing SOC_Min values
if SOC_setting == 1
    flip_index = fliplr(index');

    for i = 1:(length(Section_SOC_Mins) - 1)
        if (allowed_SOC - sum(Section_SOC(index(1:(end-i))))) <= 0
            Section_SOC_Mins(flip_index(i)) = 0;
        else
            Section_SOC_Mins(flip_index(i)) = allowed_SOC -
sum(Section_SOC(index(1:(end-i))));
            allowed_SOC = sum(Section_SOC(index(1:(end-i))));
        end
    end

    Section_SOC_Mins(index(1)) = allowed_SOC;
    Allowed_SOC = Section_SOC_Mins;

    Section_SOC_Mins = Lowest_SOC*ones(size(Section_SOC_Mins));

elseif SOC_setting == 2
    flip_index = fliplr(index');

    for i = 1:(length(Section_SOC_Mins) - 1)
        if (allowed_SOC - sum(Section_SOC(index(1:(end-i))))) <= 0
            Section_SOC_Mins(flip_index(i)) = 0;
        else
            Section_SOC_Mins(flip_index(i)) = allowed_SOC -
sum(Section_SOC(index(1:(end-i))));
            allowed_SOC = sum(Section_SOC(index(1:(end-i))));
        end
    end

    Section_SOC_Mins(index(1)) = allowed_SOC;
    residual = sum(Section_SOC_Mins(sort(index((k_EV+k_Bl+1):end))));
    blend_ratios =
Section_Dist(sort(index((k_EV+k_Bl+1):end)))/sum(Section_Dist(sort(index((k_E
V+k_Bl+1):end))));
    Section_SOC_Mins(sort(index((k_EV+k_Bl+1):end))) = residual*blend_ratios;
    Allowed_SOC = Section_SOC_Mins;

    for i = 1:length(Section_SOC_Mins)
        if i == 1
            Section_SOC_Mins(i) = Initial_SOC - Section_SOC_Mins(i);
        else
```

```
            Section_SOC_Mins(i) = Section_SOC_Mins(i-1) -
Section_SOC_Mins(i);
        end
    end

    blended_Section_SOC_Mins =
Section_SOC_Mins(sort(index((k_EV+k_Bl+1):end)));
    blended_Section_distances = Section_Dist(sort(index((k_EV+k_Bl+1):end)));

elseif SOC_setting == 3
    flip_index = fliplr(index');

    for i = 1:(length(Section_SOC_Mins) - 1)
        if (allowed_SOC - sum(Section_SOC(index(1:(end-i))))) <= 0
            Section_SOC_Mins(flip_index(i)) = 0;
        else
            Section_SOC_Mins(flip_index(i)) = allowed_SOC -
sum(Section_SOC(index(1:(end-i))));
            allowed_SOC = sum(Section_SOC(index(1:(end-i))));
        end
    end

    Section_SOC_Mins(index(1)) = allowed_SOC;
    residual = sum(Section_SOC_Mins(sort(index((k_EV+1):(k_EV+k_Bl)))));
    blend_ratios =
Section_Dist(sort(index((k_EV+1):(k_EV+k_Bl))))/sum(Section_Dist(sort(index((
k_EV+1):(k_EV+k_Bl)))));
    Section_SOC_Mins(sort(index((k_EV+1):(k_EV+k_Bl)))) =
residual*blend_ratios;
    Allowed_SOC = Section_SOC_Mins;

    for i = 1:length(Section_SOC_Mins)
        if i == 1
            Section_SOC_Mins(i) = Initial_SOC - Section_SOC_Mins(i);
        else
            Section_SOC_Mins(i) = Section_SOC_Mins(i-1) -
Section_SOC_Mins(i);
        end
    end

    blended_Section_SOC_Mins =
Section_SOC_Mins(sort(index((k_EV+1):(k_EV+k_Bl))));
    blended_Section_distances =
Section_Dist(sort(index((k_EV+1):(k_EV+k_Bl))));

elseif SOC_setting == 4
    flip_index = fliplr(index');

    for i = 1:(length(Section_SOC_Mins) - 1)
        if (allowed_SOC - sum(Section_SOC(index(1:(end-i))))) <= 0
            Section_SOC_Mins(flip_index(i)) = 0;
        else
            Section_SOC_Mins(flip_index(i)) = allowed_SOC -
sum(Section_SOC(index(1:(end-i))));
            allowed_SOC = sum(Section_SOC(index(1:(end-i))));
        end
    end

    Section_SOC_Mins(index(1)) = allowed_SOC;
    Allowed_SOC = Section_SOC_Mins;

    for i = 1:length(Section_SOC_Mins)
        if i == 1
            Section_SOC_Mins(i) = Initial_SOC - Section_SOC_Mins(i);
        else
            Section_SOC_Mins(i) = Section_SOC_Mins(i-1) -
Section_SOC_Mins(i);
        end
    end

end

%%SOC Throughout Route
SOC_Mins = zeros(1,(length(route)-1));
j = 1 + length(Section_SOC_Mins);

for i = 1:length(Section_SOC_Mins)
    if i == 1
        SOC_Mins(1:Section_Size(i)) =
Section_SOC_Mins(i)*ones(1,Section_Size(i));
    else
        SOC_Mins((1+sum(Section_Size(1:(i-1)))):sum(Section_Size(1:i))) =
Section_SOC_Mins(i)*ones(1,Section_Size(i));
    end
end

SOC_Mins = [SOC_Mins(1) SOC_Mins];
Matrix(6,:) = SOC_Mins;
Section_SOC_Min_Dist = [Section_SOC_Mins; Section_Dist];
if SOC_setting == 1 || SOC_setting == 4
    blended_SOC_and_dist = [0;0];
else
    blended_SOC_and_dist = [blended_Section_SOC_Mins;
blended_Section_distances];
end

save('C:\Users\Joseph\Documents\advisor\models\Traj_Fore_Fuzzy_SOC_Variables.
mat','Section_SOC_Min_Dist','Matrix','SOC_setting','blended_SOC_and_dist')
save('Traj_Fore_Fuzzy_SOC_Variables.mat')

disp(Section_Dist)
disp(Section_Priority)
disp(Section_SOC)
disp(SOC_setting)
disp(Allowed_SOC)
disp(Section_SOC_Mins)
disp(blended_SOC_and_dist)

%% Fuzzy Block Controller Setup Version 5 %%
if Drive_Type == 2
```

```
    clear; clc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
    %%Meant for Two Inputs and One Output with same # of membership functions
for all%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%

    %%Number of Membership Functions
    fprintf('\n')

    while 1
        tri = input('Choose 5 or 7 membership functions for variables: ');
        if ((tri == 5) || (tri == 7))
            break
        else
            disp('Choose either 5 or 7.')
        end
    end

    if (tri == 5)
        rule = [5 5 5 4 3;
                5 5 4 3 2;
                5 4 3 2 1;
                4 3 2 1 1;
                3 2 1 1 1];
    elseif (tri == 7)
        rule = [7 7 7 7 6 5 4;
                7 7 7 6 5 4 3;
                7 7 6 5 4 3 2;
                7 6 5 4 3 2 1;
                6 5 4 3 2 1 1;
                5 4 3 2 1 1 1;
                4 3 2 1 1 1 1];
    end

    %%Error Input Triangles
    fprintf('\n')

    sat1 = input('Upper/Lower point of saturation for error range: ');
    b1 = (2*sat1)/(((tri+1)/2)-1);
    limit1 = sat1 + (b1/2);

    x1 = zeros(2*tri,10);
    y1 = zeros(2*tri,10);

    for i = 1:(tri+1)
         x1(i,:) = linspace((-limit1+((i-1)*(b1/2))),(-
limit1+(i*(b1/2))),10);
    end

    x1((tri+2):(2*tri),:) = x1(2:tri,:);
    for i = 1:(2*tri)
        if ((i == 1) || (i == (tri+1)))
            y1(i,:) = ones(1,10);
        else
            if (rem(i,2) ~= 0)
                y1(i,:) = (2/b1)*(x1(i,:) - x1(i,1));
            else
                y1(i,:) = -(2/b1)*(x1(i,:) - x1(i,1)) + 1;
            end
        end
    end

    %%d(Error)/dt Triangles Variables
    fprintf('\n')

    sat2 = input('Upper/Lower point of saturation for change in error range:
');
    b2 = (2*sat2)/(((tri+1)/2)-1);
    limit2 = sat2 + (b2/2);

    x2 = zeros(2*tri,10);
    y2 = zeros(2*tri,10);

    for i = 1:(tri+1)
         x2(i,:) = linspace((-limit2+((i-1)*(b2/2))),(-
limit2+(i*(b2/2))),10);
    end

    x2((tri+2):(2*tri),:) = x2(2:tri,:);

    for i = 1:(2*tri)
        if ((i == 1) || (i == (tri+1)))
            y2(i,:) = ones(1,10);
        else
            if (rem(i,2) ~= 0)
                y2(i,:) = (2/b2)*(x2(i,:) - x2(i,1));
            else
                y2(i,:) = -(2/b2)*(x2(i,:) - x2(i,1)) + 1;
            end
        end
    end

    %%Output Triangles Variables
    sat3 = 0.4; % hard set so x-axis allows [-0.4 0.4] for possible K output
    b3 = (2*sat3)/(((tri+1)/2)-1);
    limit3 = sat3 + (b3/2); %input('Upper/Lower limit for output range: ')

    x3 = zeros(2*tri,10);
    y3 = zeros(2*tri,10);

    for i = 1:(tri+1)
         x3(i,:) = linspace((-limit3+((i-1)*(b3/2))),(-
limit3+(i*(b3/2))),10);
    end
    x3((tri+2):(2*tri),:) = x3(2:tri,:);

    x3 = x3 + sat3; %shift possible K output to [0 0.8]

    for i = 1:(2*tri)
        if (rem(i,2) ~= 0)
            y3(i,:) = (2/b3)*(x3(i,:) - x3(i,1));
        else
            y3(i,:) = -(2/b3)*(x3(i,:) - x3(i,1)) + 1;
        end
    end

    out_midpoint = zeros(1,tri);

    for i = 1:tri
        out_midpoint(i) = -limit3 + i*(b3/2);
    end

    out_midpoint = out_midpoint + sat3; %%shift possible K output to [0 0.8]

    %%Controller (Determining Premise and Conclusion Certainty)
    fprintf('\n')
    while 1
        Prem_Method = input('Choose either Minimum or Product for premise
certainty: ','s');
        if (Prem_Method == 'Minimum')
            Prem_Method = 1;
            break
        elseif (Prem_Method == 'Product')
            Prem_Method = 2;
            break
        else
            disp('Must choose Minimum or Product.')
        end
    end
    fprintf('\n')
    while 1
        Defuzz_Method = input('Choose either COG or C_A defuzzification
method: ','s');
        if (Defuzz_Method == 'COG')
            Defuzz_Method = 1;
            break
        elseif (Defuzz_Method == 'C_A')
            Defuzz_Method = 2;
            break
        else
           disp('Must choose COG or C_A.')
        end
    end


save('C:\Users\Joseph\Documents\advisor\models\Traj_Fore_Fuzzy_Calc_Variables
.mat')
    save('Traj_Fore_Fuzzy_Calc_Variables.mat')
    %%Plots
    figure
    subplot(3,1,1)
    for m = 1:1:(2*tri)
    plot(x1(m,:),y1(m,:),'b')
    hold on
    end
    grid on
    ylabel('Certainty')
    xlabel('e(t)')
    ylim([0 1])

    subplot(3,1,2)
    for m = 1:1:(2*tri)
    plot(x2(m,:),y2(m,:),'b')
    hold on
    end
    grid on
    ylabel('Certainty')
    xlabel('delta e(t)')
    ylim([0 1])


    subplot(3,1,3)
    for m = 1:1:(2*tri)
    plot(x3(m,:),y3(m,:),'b')
    hold on
    end
    grid on
    ylabel('Certainty')
    xlabel('u(t)')
    ylim([0 1])
    clear;
end
clear;
advisor;
```

7.1.3. ADVISOR Code Additives

7.1.3.1. Parallel TF Vehicle Control Block: Engine Shutoff for EV Only Routes Function Code

```
function y = fcn(u, SOC)
FSC = load('Traj_Fore_Fuzzy_SOC_Variables.mat');
SOC_setting = FSC.SOC_setting;

if (SOC_setting == 1 && SOC >= 0.25)
    y = 0;
else
    y = u;
```

```
end
```

7.1.3.2. Fuzzy TF Control Block: SOC Min Assignment Function Code

```
function soc_min = fcn(distance,time)
FSC = load('Traj_Fore_Fuzzy_SOC_Variables.mat');
Matrix = FSC.Matrix;
[rows,~] = size(Matrix);

if (rows == 6)
    distance = distance*0.000621371;
    if distance == 0
        soc_min = Matrix(6,1);
    else
        a = find(Matrix(1,:) > distance);
        if sum(a) == 0
            soc_min = Matrix(6,end);
        else
            soc_min = Matrix(6,(a(1)-1));
        end
    end
elseif (rows == 2)
    soc_min = Matrix(2,(time+1));
end
```

7.1.3.3. Fuzzy TF Control Block: Control Strategy Function Code

```
function
[SOC_init_cur,Dist_init_cur,K,avail_trq,element_cur,priority_cur,desired,erro
r] =
fcn(SOC_init_prev,Dist_init_prev,req_trq,distance,time,element_prev,SOC,prior
ity_prev,SOC_goal,error_prev)
%% Initialize Variables
a = 0;
element_cur = 0;
desired = 0;
K = 0;
Switch = 0;
error = 0;
SOC_init_cur = 0;
Dist_init_cur = 0;

%%%% Fuzzy SOC Variables C:\Users\Joseph\Documents\advisor\models
FSC = load('Traj_Fore_Fuzzy_SOC_Variables.mat');
SOC_setting = FSC.SOC_setting;
Matrix = FSC.Matrix;
[rows,~] = size(Matrix);
blended_SOC_and_dist = FSC.blended_SOC_and_dist;

%%%% Fuzzy Calculation Variables C:\Users\Joseph\Documents\advisor\models
FV = load('Traj_Fore_Fuzzy_Calc_Variables.mat');
```

```
b1 = FV.b1;
sat1 = FV.sat1;
x1 = FV.x1;
y1 = FV.y1;
b2 = FV.b2;
sat2 = FV.sat2;
x2 = FV.x2;
y2 = FV.y2;
b3 = FV.b3;
% x3 = FV.x3;
% y3 = FV.y3;
tri = FV.tri;
Prem_Method = FV.Prem_Method;
rule = FV.rule;
Defuzz_Method = FV.Defuzz_Method;
out_midpoint = FV.out_midpoint;

%% Determine Priority Value
if (rows == 6)
    distance = distance*0.000621371;

    if distance == 0
        priority = Matrix(5,1);
        SOC_Min = Matrix(6,1);
    else
        a = find(Matrix(1,:) > distance);
        if sum(a) == 0
            priority = Matrix(5,end);
            SOC_Min = Matrix(6,end);
        else
            priority = Matrix(5,(a(1)-1));
            SOC_Min = Matrix(6,(a(1)-1));
        end
    end
elseif (rows == 2)
    priority = Matrix(1,(time+1));
    SOC_Min = Matrix(2,(time+1));
end

%% Determine Which Blended Section and SOC usage goal
if SOC_setting == 1
    element_cur = 0;
%     z = 0;
    K = 0; %Consider using K = 1

%     Switch = 0;
elseif SOC_setting == 2
    if priority == 1
        if distance == 0 && element_prev == 0
            element_cur = element_prev + 1;
            desired = (SOC -
blended_SOC_and_dist(1,element_cur))/blended_SOC_and_dist(2,element_cur);
            SOC_init_cur = SOC;
            Dist_init_cur = distance;
        elseif distance ~= 0 && priority_prev ~= 1
            element_cur = element_prev + 1;
            desired = (SOC - 
blended_SOC_and_dist(1,element_cur))/blended_SOC_and_dist(2,element_cur);
            SOC_init_cur = SOC;
            Dist_init_cur = distance;
%         elseif distance ~= 0 && (a(1)-2) > 0
%             if priority_prev ~= 1 %Matrix(5,(a(1)-2)) ~= 1
%                 w = element + 1;
%                 z = (SOC - 
blended_SOC_and_dist(1,w))/blended_SOC_and_dist(2,w);
%             end
        else
            element_cur = element_prev;
            desired = SOC_goal;
            SOC_init_cur = SOC_init_prev;
            Dist_init_cur = Dist_init_prev;
        end

        Switch = 1;
    else
        element_cur = element_prev;
%         z = 0;
        K = 0; %Consider using K = 1

%         Switch = 0;
    end
elseif SOC_setting == 3
    if priority == 1
        element_cur = element_prev;
%         z = 0;
        if ((SOC - SOC_Min) >= 0.01)
            K = 0.85;
        elseif ((SOC - SOC_Min) >= 0.005)
            K = 0.9;
        else
            K = 1;
        end

%         Switch = 0;
    elseif priority == 2
        if distance == 0 && element_prev == 0
            element_cur = element_prev + 1;
            desired = (SOC - 
blended_SOC_and_dist(1,element_cur))/blended_SOC_and_dist(2,element_cur);
            SOC_init_cur = SOC;
            Dist_init_cur = distance;
        elseif distance ~= 0 && priority_prev ~= 2
            element_cur = element_prev + 1;
            desired = (SOC - 
blended_SOC_and_dist(1,element_cur))/blended_SOC_and_dist(2,element_cur);
            SOC_init_cur = SOC;
            Dist_init_cur = distance;
%         elseif distance ~= 0 && (a(1)-2) > 0
%             if priority_prev ~= 2 %Matrix(5,(a(1)-2)) ~= 2
%                 w = element + 1;
%                 z = (SOC - 
blended_SOC_and_dist(1,w))/blended_SOC_and_dist(2,w);
```

```
%              end
        else
            element_cur = element_prev;
            desired = SOC_goal;
            SOC_init_cur = SOC_init_prev;
            Dist_init_cur = Dist_init_prev;
        end

        Switch = 1;
    else
        element_cur = element_prev;
%         z = 0;
        K = 0; %Consider using K = 1

%         Switch = 0;
    end
elseif SOC_setting == 4 %possibly combine with SOC_setting 3 with if
statement for EV section K depending on setting 3 or 4
    element_cur = 0;
%     z = 0;

%     Switch = 0;
    if priority == 1 || priority == 2
        if ((SOC - SOC_Min) >= 0.01)
            K = 0.85;
        elseif ((SOC - SOC_Min) >= 0.005)
            K = 0.9;
        else
            K = 1;
        end
    else
        K = 0;
    end
end

if (SOC <= SOC_Min)
    if (SOC_Min == 0.25)
        Switch = 0;
    end
    K = 1;
    Switch = 0;
end

%% Determine Available Torque K Coefficient
if Switch == 1
    if distance == 0 || distance == Dist_init_cur
        error = 0;
        del_error = error;
    else
        error = (desired - (SOC_init_cur-SOC)/(distance-
Dist_init_cur))/desired;
        del_error = error - error_prev;
    end

%%%%%%%%FUZZY LOGIC%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
    %%Meant for Two Inputs and One Output with same # of membership functions
for all%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%

    %%%% Controller (Finding Input Certainties and Triangle Numbers)
    i = [0 0]; % Error triangle numbers, i(1) for odd and i(2) for even
    j = [0 0]; % Change-in-Error triangle numbers, j(1) for odd and j(2) for
even

    mf_cer1 = zeros(1,2); % Error triangle certainties, mf_cer1(1) for odd
and mf_cer1(2) for even
    mf_cer2 = zeros(1,2); % Change-in-Error triangle certainties, mf_cer2(1)
for odd and mf_cer2(2) for even

    for k1 = 1:(2*tri)%Checks saturation and greater; checks for when
centertainties are less than 1
        if (error <= -sat1)
            mf_cer1(1) = 1;
            i(1) = 1;
        elseif (error >= sat1)
            mf_cer1(1) = 1;
            i(1) = tri;
        elseif ((error > x1(k1,1)) && (error < x1(k1,end)) && (abs(error) <
sat1))
            if (rem(k1,2) ~= 0)
                if (mf_cer1(1) == 0)
                    mf_cer1(1) = (2/b1)*(error - x1(k1,1));
                    if (k1 <= (tri + 1))
                        i(1) = k1;
                    elseif (k1 > (tri + 1))
                        i(1) = k1 - tri;
                    end
                else
                    mf_cer1(2) = (2/b1)*(error - x1(k1,1));
                    if (k1 <= (tri + 1))
                        i(2) = k1;
                    elseif (k1 > (tri + 1))
                        i(2) = k1 - tri;
                    end
                end
            else
                if (mf_cer1(1) == 0)
                    mf_cer1(1) = -(2/b1)*(error - x1(k1,1)) + 1;
                    if (k1 <= (tri + 1))
                        i(1) = k1 - 1;
                    elseif (k1 > (tri + 1))
                        i(1) = k1 - tri - 1;
                    end
                else
                    mf_cer1(2) = -(2/b1)*(error - x1(k1,1)) + 1;
                    if (k1 <= (tri + 1))
```

```
                        i(2) = k1 - 1;
                    elseif (k1 > (tri + 1))
                        i(2) = k1 - tri - 1;
                    end
                end
            end
        end
    end

    if ((rem(error,(b1/2)) == 0) && (abs(error) < sat1))%Checks when
certainties are one
       mf_cer1(1) = 1;
       for k1 = 1:(2*tri)
           if ((error == x1(k1,1)) && (1 == y1(k1,1)))
               if (k1 <= (tri + 1))
                   i(1) = k1 - 1;
               elseif (k1 > (tri + 1))
                   i(1) = k1 - tri - 1;
               end
           end
       end
    end

    for k1 = 1:(2*tri) %Checks saturation and greater; checks for when
centertainties are less than 1
        if (del_error <= -sat2)
            mf_cer2(1) = 1;
            j(1) = 1;
        elseif (del_error >= sat2)
            mf_cer2(1) = 1;
            j(1) = tri;
        elseif ((del_error > x2(k1,1)) && (del_error < x2(k1,end)) &&
(abs(del_error) < sat2))
            if (rem(k1,2) ~= 0)
                if (mf_cer2(1) == 0)
                    mf_cer2(1) = (2/b2)*(del_error - x2(k1,1));
                    if (k1 <= (tri + 1))
                        j(1) = k1;
                    elseif (k1 > (tri + 1))
                        j(1) = k1 - tri;
                    end
                else
                    mf_cer2(2) = (2/b2)*(del_error - x2(k1,1));
                    if (k1 <= (tri + 1))
                        j(2) = k1;
                    elseif (k1 > (tri + 1))
                        j(2) = k1 - tri;
                    end
                end
            else
                if (mf_cer2(1) == 0)
                    mf_cer2(1) = -(2/b2)*(del_error - x2(k1,1)) + 1;
                    if (k1 <= (tri + 1))
                        j(1) = k1 - 1;
                    elseif (k1 > (tri + 1))
                        j(1) = k1 - tri - 1;
                    end
                else
                    mf_cer2(2) = -(2/b2)*(del_error - x2(k1,1)) + 1;
                    if (k1 <= (tri + 1))
                        j(2) = k1 - 1;
                    elseif (k1 > (tri + 1))
                        j(2) = k1 - tri - 1;
                    end
                end
            end
         end
    end

    if ((rem(del_error,(b2/2)) == 0) && (abs(del_error) < sat2))%Checks when
certainties are one
       mf_cer2(1) = 1;
       for k1 = 1:(2*tri)
           if ((del_error == x2(k1,1)) && (1 == y2(k1,1)))
               if (k1 <= (tri + 1))
                   j(1) = k1 - 1;
               elseif (k1 > (tri + 1))
                   j(1) = (k1 - (tri + 1));
               end
           end
       end
    end

    %%%% Controller (Determining Premise and Conclusion Certainty)
    prem = zeros(2,2);

    for k1 = [1 2]
        for k2 = [1 2]
            if (Prem_Method == 1)
                prem(k1,k2) = min(mf_cer1(k1),mf_cer2(k2));
            elseif (Prem_Method == 2)
                prem(k1,k2) = (mf_cer1(k1)*mf_cer2(k2));
            end
        end
    end

    leng = length(find(prem~=0));
    u = zeros(leng,2);
    k = 1;

    for k1 = [1 2]
        for k2 = [1 2]
            if prem(k1,k2) ~= 0
               u(k,:) = [prem(k1,k2) rule(i(k1),j(k2))];
               k = k + 1;
            end
        end
    end

    Denom = zeros(1,leng);
    Num = zeros(1,leng);
    if (Defuzz_Method == 1)
        if (Prem_Method == 1)
            Denom = b3.*(u(:,1)' - 0.5.*(u(:,1)'.^2));
        elseif (Prem_Method == 2)
            Denom = 0.5.*b3.*u(:,1)';
        end
        Num = out_midpoint(u(:,2)).*Denom;
    elseif (Defuzz_Method == 2)
        Denom = u(:,1)';
        Num = out_midpoint(u(:,2)).*Denom;
    end
    K = sum(Num)/sum(Denom);
%%%%%%%%%%%%%%%%%%%%%%%%%
end

if K < 0
    K = 0;
end

avail_trq = K*req_trq;
priority_cur = priority;
```

7.2. Convolutional Neural Network Code

7.2.1. Data Preprocessing

```
close all; clear; clc;

file_list = dir('..\drive_cycles_advisor_results_equal_num_1_0L_PB25');
file_list = file_list(5:end);

V_pos = 0;
V_neg = 0;
A_pos = 0;
A_neg = 0;

for cycle = 1:length(file_list)
    %% CNN Input Data Range Extraction %%

    cycle_data = 
['..\drive_cycles_advisor_results_equal_num_1_0L_PB25\',file_list(cycle).name
];
    load(cycle_data)

    if max(drive_cycle(:,2)) > V_pos
        V_pos = max(drive_cycle(:,2));
    end

    if min(drive_cycle(:,2)) < V_neg
        V_neg = min(drive_cycle(:,2));
    end

    if max(drive_cycle(:,3)) > A_pos
        A_pos = max(drive_cycle(:,3));
    end

    if min(drive_cycle(:,3)) < A_neg
        A_neg = min(drive_cycle(:,3));
    end
end

for output_choice = 1:1
    clearvars -except V_neg V_pos A_neg A_pos output_choice file_list

    i = round((resolution_rounding(A_pos,0.1) -
resolution_rounding(A_neg,0.1))/0.1 + 1);
    j = round((resolution_rounding(V_pos,0.5) -
resolution_rounding(V_neg,0.5))/0.5 + 1);

    Inputs = zeros(i,j,3,length(file_list));
    Outputs = zeros(length(file_list),1);
    distance = zeros(length(file_list),1);%%%
    engine = zeros(length(file_list),1);%%%

%     output_choice = input('Choice of CNN output (1-4): ');
%     fprintf('\n')

    for cycle = 1:length(file_list)
        %% CNN Input Preprocessing %%

        cycle_data =
['..\drive_cycles_advisor_results_equal_num_1_0L_PB25\',file_list(cycle).name
];
        load(cycle_data)

        engine_on = 0;
        distance(cycle) = trapz(drive_cycle(:,1),(drive_cycle(:,2)/3600));%%%
        Inputs(:,:,:,cycle) = Inputs(:,:,:,cycle) + Initial_SOC;
        if output_choice ~= 5
            Outputs(cycle) = results(output_choice);
        else
            Outputs(cycle) = max(gal);
        end

        rounded_vel = resolution_rounding(drive_cycle(:,2),0.5);
        rounded_accel = resolution_rounding(drive_cycle(:,3),0.1);

        section = 1;
        prev_layer = 0;
        for ii = 1:length(drive_cycle(:,1))
            row = round((rounded_accel(ii) -
resolution_rounding(A_neg,0.1))/0.1 + 1);
            col = round((rounded_vel(ii) -
resolution_rounding(V_neg,0.5))/0.5 + 1);
```

```
            if ii == 1
                if Section_SOC(section) == 0
                    layer = 1;
                else
                    ratio = 
round((Allowed_SOC(section)/Section_SOC(section)),3);%1 - 
round((Allowed_SOC(section)/Section_SOC(section)),3);
                    if ratio >= 1%<= 0%>= 1
                        layer = 3;
                    elseif (ratio < 1) && (ratio > 0)
                        layer = 2;
                    elseif ratio == 0%1%0
                        layer = 1;
                    end
                end
            else
                if Matrix(1,ii) ~= Matrix(1,(ii-1))
                    section = section + 1;
                    if Section_SOC(section) == 0
                        layer = 1;
                    else
                        ratio = 
round((Allowed_SOC(section)/Section_SOC(section)),3);%1 - 
round((Allowed_SOC(section)/Section_SOC(section)),3);
                        if ratio >= 1%<= 0%>= 1
                            layer = 3;
                        elseif (ratio < 1) && (ratio > 0)
                            layer = 2;
                        elseif ratio == 0%1%0
                            layer = 1;
                        end
                    end
                end
            end

            if layer == 1
                engine_on = engine_on + 1;%%%
                if prev_layer == 3
                    Inputs(row,col,layer,cycle) = Inputs(row,col,layer,cycle) 
+ 1;
                elseif prev_layer == 2
                    Inputs(row,col,layer,cycle) = Inputs(row,col,layer,cycle) 
+ 1;
                elseif prev_layer == 1
                    Inputs(row,col,layer,cycle) = Inputs(row,col,layer,cycle) 
+ 1;
                end
            elseif layer == 2
                engine_on = engine_on + 1;%%%
                if prev_layer == 3
                    Inputs(row,col,layer,cycle) = Inputs(row,col,layer,cycle) 
+ (1-ratio);
                else
                    Inputs(row,col,layer,cycle) = Inputs(row,col,layer,cycle) 
+ (1-ratio);
```

```
                end
            elseif layer == 3
                Inputs(row,col,layer,cycle) = Inputs(row,col,layer,cycle) +
1;
            end

            prev_layer = layer;
        end
        engine(cycle) = (engine_on - length(T_w_sat(:,1)));
    end

    XTrain = Inputs;
    YTrain = Outputs;
    Dist_Train = distance;%%%

    %%%%%%%%% Might Have to take out %%%%%%%%%%%
    id = 0;
    if output_choice ~= 1
        id = find(YTrain == 0);
        XTrain(:,:,:,id) = [];
        YTrain(id) = [];
        Dist_Train(id) = [];%%%
        engine(id) = [];
%         temp = find(YTrain > 3);
%         XTrain(:,:,:,temp) = [];
%         YTrain(temp) = [];
%         Dist_Train(temp) = [];%%%
%         idd = find(Dist_Train < 6);%%%
%         XTrain(:,:,:,idd) = [];%%%
%         YTrain(idd) = [];%%%
%         Dist_Train(idd) = [];%%%
%         plot(1:length(YTrain),YTrain)%%%
%         YTrain = YTrain.*Dist_Train;%%%
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%     if output_choice == 3
% %         XTrain = XTrain(:,:,1:2,:);
% %         YTrain = ((YTrain - min(YTrain))./(max(YTrain) - min(YTrain))) *
(4 - 0.05) + 0.05;
% %         YTrain = ((YTrain - min(YTrain))./(max(YTrain) - min(YTrain))) *
(2 - 0.03) + 0.03;
% %         YTrain = ((YTrain - min(YTrain))./(max(YTrain) - min(YTrain))) *
(10 - 1) + 1;
% %         YTrain = (YTrain - min(YTrain))./(max(YTrain) - min(YTrain));
% %         YTrain = YTrain.^(1/1.5);
% %         YTrain = YTrain.^(1/2);
% %         YTrain = YTrain.^(1/3);
% %         temp = find(YTrain > 25);
% %         XTrain(:,:,:,temp) = [];
% %         YTrain(temp) = [];
% %         YTrain = YTrain/20;
%     end

    num_test = 0.15;
    num_val = (1 - num_test)*0.2;
    idx = randperm(size(XTrain,4),round(num_test*size(XTrain,4)));
    XTest = XTrain(:,:,:,idx);
    XTrain(:,:,:,idx) = [];
    YTest = YTrain(idx);
    YTrain(idx) = [];
    Dist_Test = Dist_Train(idx);%%%
    Dist_Train(idx) = [];%%%

    idy = randperm(size(XTrain,4),round(num_val*size(XTrain,4)));
    XValidation = XTrain(:,:,:,idy);
    XTrain(:,:,:,idy) = [];
    YValidation = YTrain(idy);
    YTrain(idy) = [];
    Dist_Valid = Dist_Train(idy);%%%
    Dist_Train(idy) = [];%%%

    CNN_Data = ['CNN_New_Data_Output_',num2str(output_choice),'.mat'];

save(CNN_Data,'output_choice','Inputs','Outputs','distance','XTrain','YTrain'
,'Dist_Train', ...

'XValidation','YValidation','Dist_Valid','XTest','YTest','Dist_Test','i','j',
'id', ...
         'idx','idy','-v7.3')
End
```

7.2.2. Training

```
close all; clear; clc;

for output_choice = 1:5

    CNN_Data = ['CNN_New_Data_Output_',num2str(output_choice),'.mat'];
    load(CNN_Data)
    temp = 0;

    if output_choice ~= 1
        XTrain = XTrain(:,:,1:2,:);
        XValidation = XValidation(:,:,1:2,:);
        XTest = XTest(:,:,1:2,:);
    end

    k = size(XTrain,3);


    TF_CNN = Build_CNN(i,j,k); %#ok<IJCL>
    options = trainingOptions('adam', ...
                              'MiniBatchSize',50, ...
                              'MaxEpochs',1000, ...
                              'InitialLearnRate',0.0001, ...
                              'Shuffle','every-epoch', ...
                              'ValidationData',{XValidation,YValidation}, ...
                              'ValidationFrequency',25, ...
                              'ValidationPatience',30, ...
                              'ExecutionEnvironment','gpu', ...
                              'Verbose',true);

    for i = 1:5
        [net, info] = trainNetwork(XTrain,YTrain,TF_CNN,options);
        YPred = round(predict(net,XTest),4);

        rmse = sqrt(mean((YTest - YPred).^2));
        fprintf('Root Mean Square Error: %4.2f\n',rmse)
        avg_error = mean(100*(abs(YTest - YPred)./YTest));
        avg_accuracy = 100 - avg_error;
        fprintf('Average Percent Accuracy: %4.2f\n',avg_accuracy)

        if avg_accuracy > temp
            file_name = ['Output_',num2str(output_choice),'_Linear.mat'];
            save(file_name,'-v7.3')
            temp = avg_accuracy;
        end
    end

    clc;
    clearvars -except output_choice
end

function TF_CNN = Build_CNN(i,j,k)
    TF_CNN = layerGraph();

    tempLayers = [
        imageInputLayer([i j k],'Name','CNN_Inputs')%,'Normalization','none')
        convolution2dLayer(3,3,'Name','Conv2D_1, 3@3x3','Padding','same')
        reluLayer('Name','relu_1')
        convolution2dLayer(3,3,'Name','Conv2D_2, 3@3x3','Padding','same')
        reluLayer('Name','relu_2')
        maxPooling2dLayer(2,'Name','MaxPool2D_1, 2x2_2','Stride',1)
        convolution2dLayer(3,6,'Name','Conv2D_1, 6@3x3','Padding','same')
        reluLayer('Name','relu_3')
        convolution2dLayer(3,6,'Name','Conv2D_2, 6@3x3','Padding','same')
        reluLayer('Name','relu_4')
        maxPooling2dLayer(2,'Name','MaxPool2D_2, 2x2_2','Stride',1)
        convolution2dLayer(3,6,'Name','Conv2D_1, 6@3x3','Padding','same')
        reluLayer('Name','relu_5')
        convolution2dLayer(3,6,'Name','Conv2D_2, 6@3x3','Padding','same')
        reluLayer('Name','relu_6')
        maxPooling2dLayer(2,'Name','MaxPool2D_3, 2x2_2','Stride',1)
        convolution2dLayer(3,12,'Name','Conv2D_3, 12@3x3','Padding','same')
        reluLayer('Name','relu_7')
        convolution2dLayer(3,12,'Name','Conv2D_4, 12@3x3','Padding','same')
        reluLayer('Name','relu_8')
        maxPooling2dLayer(2,'Name','MaxPool2D_4, 2x2_2','Stride',1)
        convolution2dLayer(2,12,'Name','Conv2D_5, 12@2x2','Padding','same')
        reluLayer('Name','relu_9')
        convolution2dLayer(2,12,'Name','Conv2D_6, 12@2x2','Padding','same')
        reluLayer('Name','relu_10')
        maxPooling2dLayer(2,'Name','MaxPool2D_5, 2x2_2','Stride',1)
        FlatteningLayer('Input Vector')
        fullyConnectedLayer(512,'Name','Fully Connected_1')
        reluLayer('Name','Output relu_1')
        fullyConnectedLayer(256,'Name','Fully Connected_2')
        reluLayer('Name','Output relu_2')
        fullyConnectedLayer(1,'Name','Fully Connected_5')
%         reluLayer('Name','Output relu_5')
        regressionLayer('Name','regressionoutput')];
    TF_CNN = addLayers(TF_CNN,tempLayers);
end
```

7.3. Complete Trajectory Forecasting Based Machine Learning Control Strategy

7.3.1. Original Route Data

7.3.1.1. Route 1

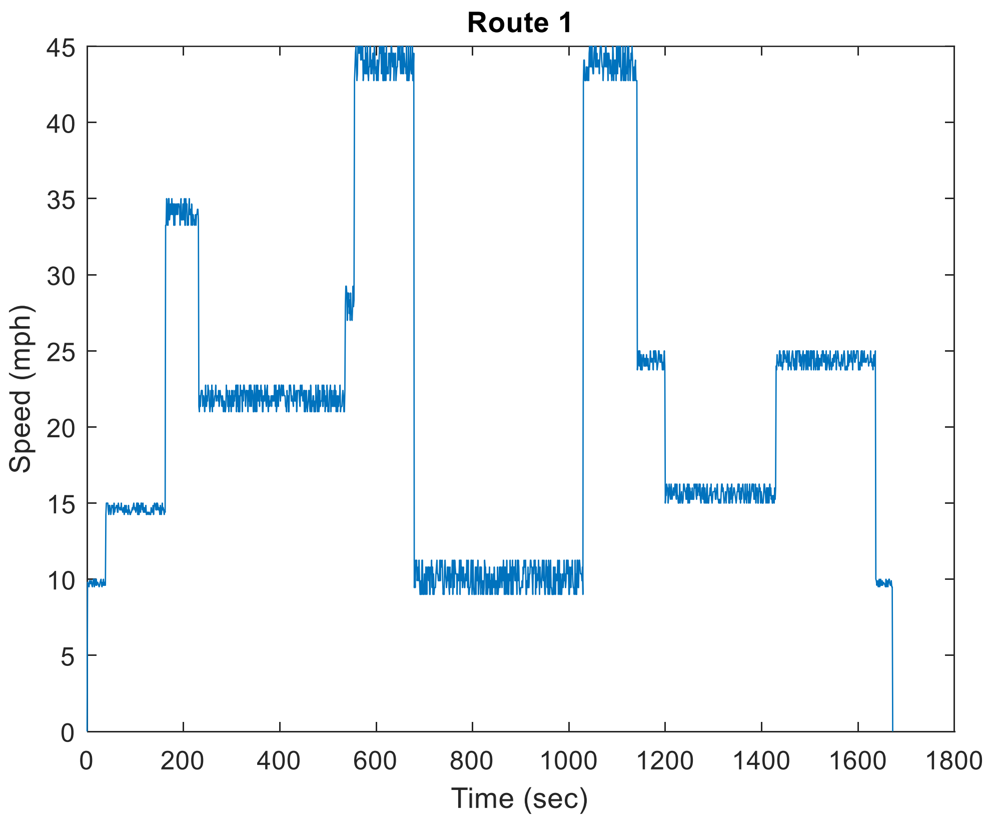


Figure 57: Original Route 1, 10 miles

7.3.1.2. Route 2

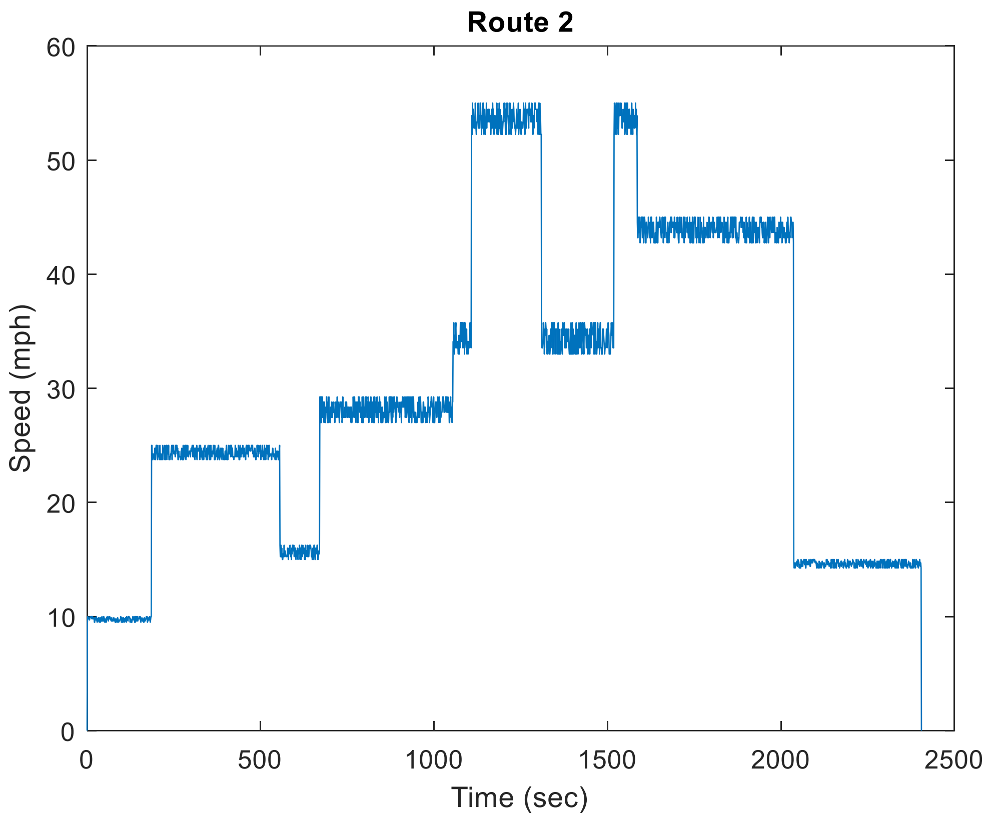


Figure 58: Original Route 2, 20 miles

7.3.1.3.Route 3

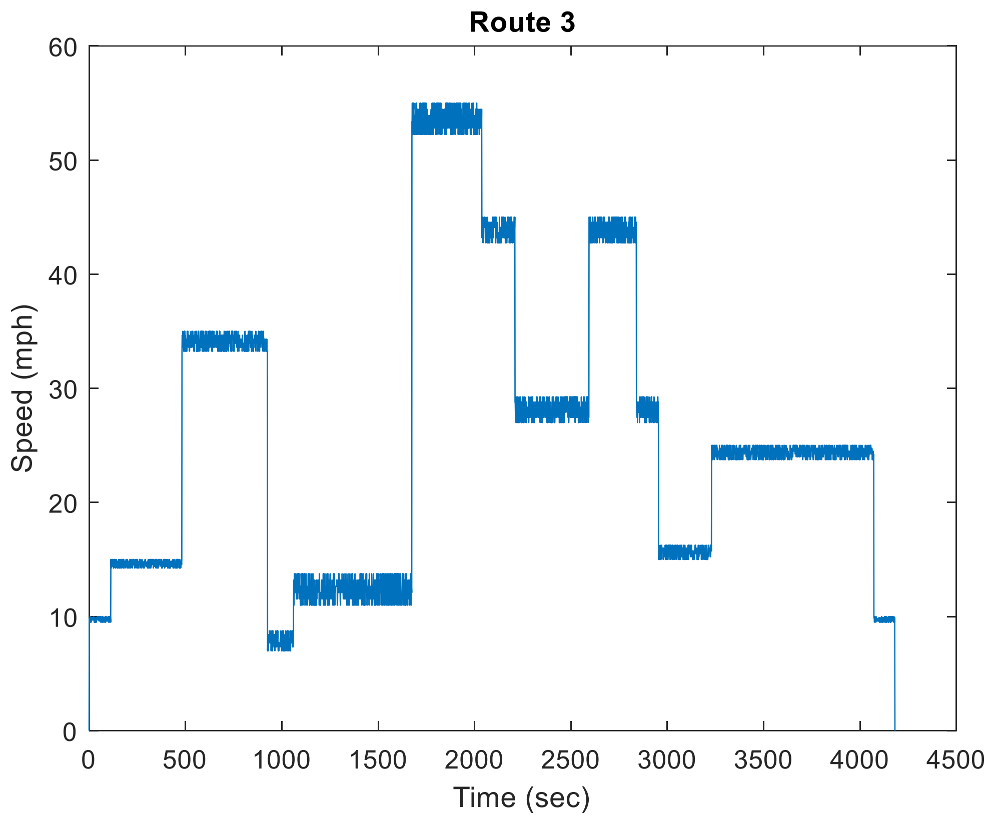


Figure 59: Original Route 3, 30 miles

7.3.1.4. Route 4

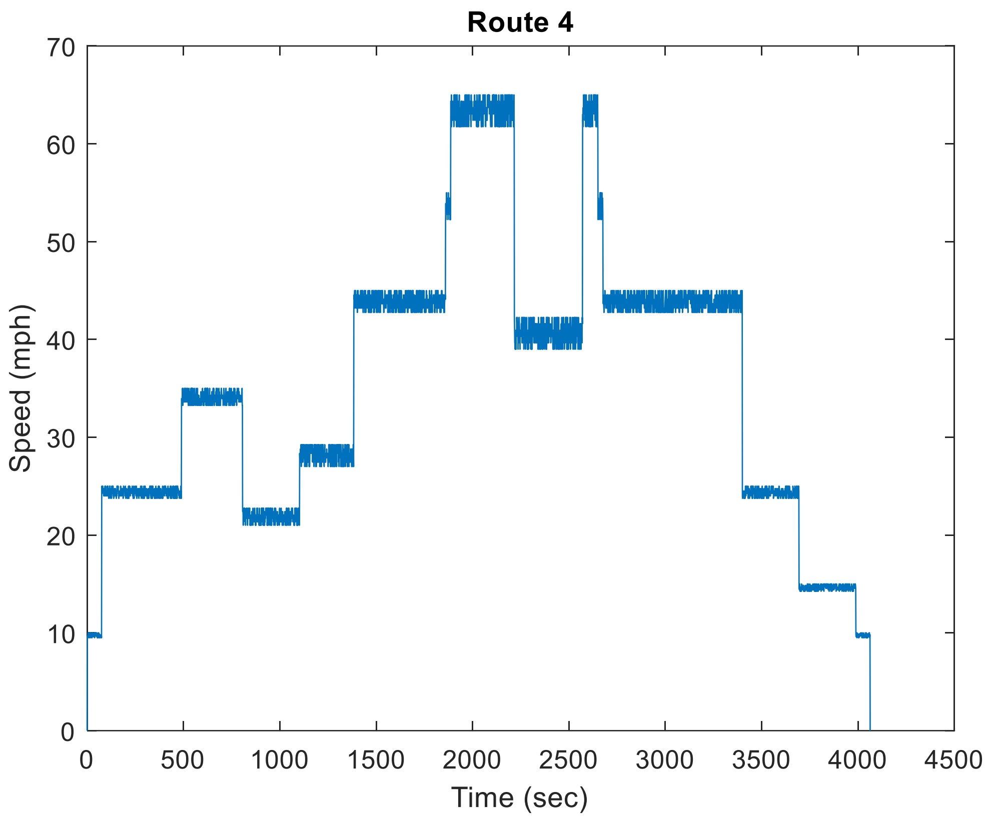


Figure 60: Original Route 4, 40 miles

7.3.1.5. Route 5

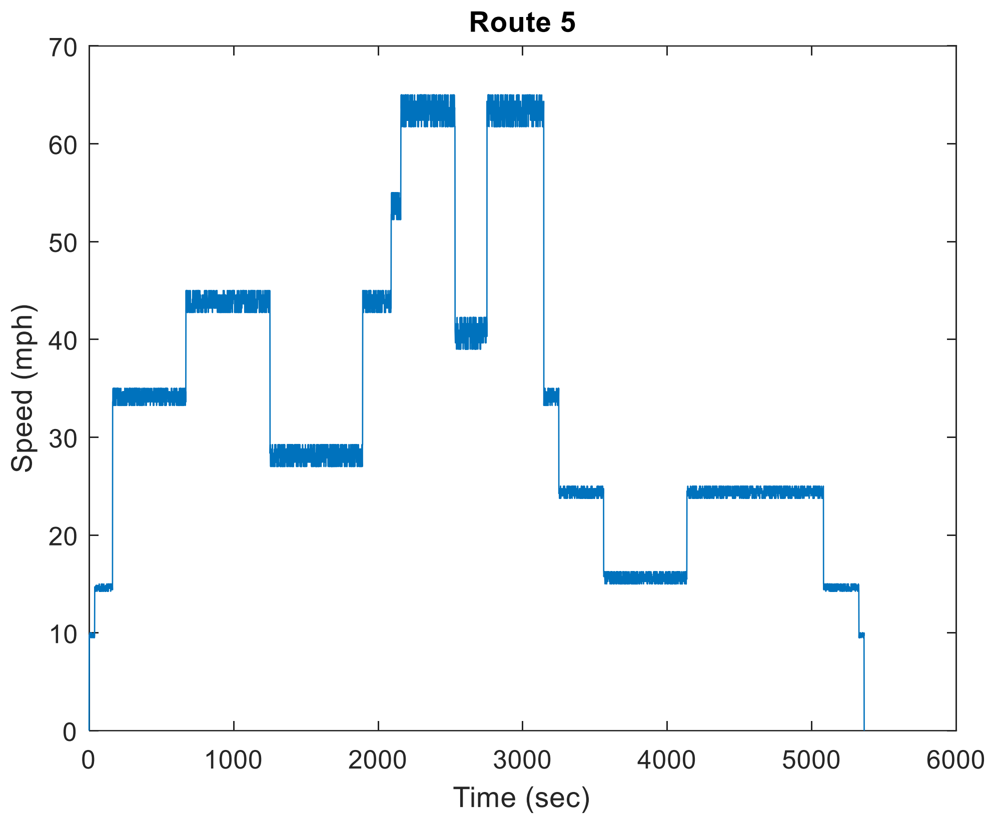


Figure 61: Original Route 5, 50 miles

7.3.1.6. Route 6

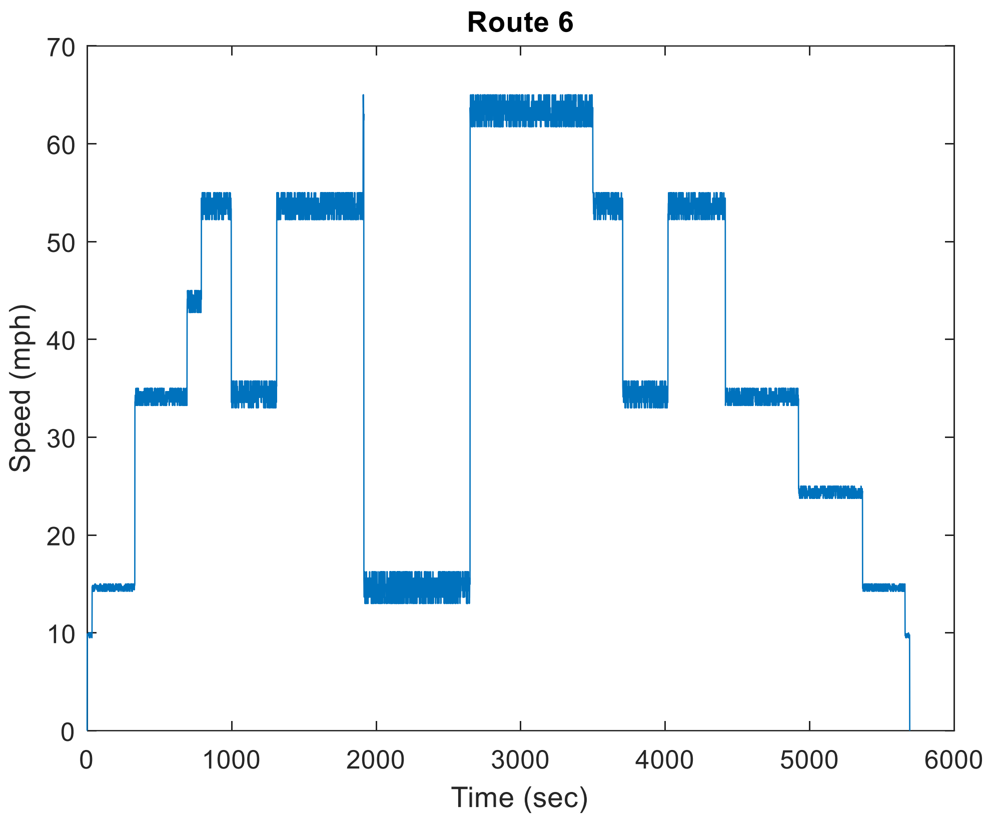


Figure 62: Original Route 6, 60 miles

7.3.2. MATLAB Code

```
for result = 1:5
    close all; clearvars -except result; clc;

    for route_num = 1:6
        while 1
            close all; clearvars -except route_num result; clc;

            %% Initial Test Variables %%
            %%% Choice of Vehicle and Output Parameters %%%
            engine = '1_0';
            battery = 28;
    %         result = input('Choose 1 (MPGGE), 2 (HC), 3 (CO), 4 (NOx), or 5
(gal) parameter: ');
            clc;

            switch result
                case 1
                    output_choice = 1;
                    output_name = 'MPGGE';
                case 2
                    output_choice = 2;
                    output_name = 'HC';
                case 3
                    output_choice = 3;
                    output_name = 'CO';
                case 4
                    output_choice = 4;
                    output_name = 'NOx';
                case 5
                    output_choice = 5;
                    output_name = 'gal';
            end

            switch battery
                case 25
                    SOC_ratio = 0.0319;
                case 28
                    SOC_ratio = 0.0288;
            end

            advisor_net = ['../CNN 2 - Main network, ',engine,'L
PB',num2str(battery),'/CNN_',output_name,'.mat'];
            load(advisor_net,'net')

            route.test_variables.engine = engine;
            route.test_variables.battery = battery;
            route.test_variables.output_choice = output_choice;
            route.test_variables.output_name = output_name;
            route.test_variables.SOC_ratio = SOC_ratio;
            route.test_variables.advisor_cnn = net;
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            %% Original Route %%
            %%% Choosing and Creating Route %%%
    %          route_num = input('Drive Cycle to Test, 1 (10 miles), 2 (20
miles), 3 (30 miles), 4 (40 miles), 5 (50 miles), 6 (60 miles): ');

            switch route_num
                case 1
                    dist = 10;
                    speed_lim_per = [1 5 25 40 28 1]/100;
                    speed_lim_cond = [1 2 4 5 3 1];
                    traffic_per = [12.5 20 15 10 17.5 10 15]/100;
                    traffic_cond = [1 2 1 3 1 2 1];
                case 2
                    dist = 20;
                    speed_lim_per = [2.5 15 15 32.5 27.5 7.5 2.5]/100;
                    speed_lim_cond = [1 3 5 6 5 2 1];
                    traffic_per = [5 10 20 15 10 15 10 15]/100;
                    traffic_cond = [1 1 2 1 2 1 1 1];
                case 3
                    dist = 30;
                    speed_lim_per = [1 5 15 25 30 23 1]/100;
                    speed_lim_cond = [1 2 4 6 5 3 1];
                    traffic_per = [20 8 25 10 10 7 20]/100;
                    traffic_cond = [1 3 1 2 1 2 1];
                case 4
                    dist = 40;
                    speed_lim_per = [0.5 7 12 20 1 28 1 22 5 3 0.5]/100;
                    speed_lim_cond = [1 3 4 5 6 7 6 5 3 2 1];
                    traffic_per = [15 10 30 10 35]/100;
                    traffic_cond = [1 2 1 2 1];
                case 5
                    dist = 50;
                    speed_lim_per = [0.2 1 9.6 29 2 32 2 22 2 0.2]/100;
                    speed_lim_cond = [1 2 4 5 6 7 4 3 2 1];
                    traffic_per = [25 10 20 5 10 10 5 10 5]/100;
                    traffic_cond = [1 2 1 2 1 1 2 1 1];
                case 6
                    dist = 60;
                    speed_lim_per = [0.15 2 5.7 2 25 30 20 8 5 2 0.15]/100;
                    speed_lim_cond = [1 2 4 5 6 7 6 4 3 2 1];
                    traffic_per = [15 5 15 5 30 5 7.5 10 7.5]/100;
                    traffic_cond = [1 2 1 3 1 2 1 1 1];
            end

            while 1
                if round(sum(speed_lim_per)) ~= 1
                    disp('Percentages must add to 100%')
                else
                    break
                end
            end

            while 1
                if length(speed_lim_cond) ~= length(speed_lim_per)
                    disp('Number of speed limit conditions must match number
of speed limit sections')
                else
                    break
                end
            end

            while 1
                if round(sum(traffic_per)) ~= 1
                    disp('Percentages must add to 100%')
                else
                    break
                end
            end

            while 1
                if length(traffic_cond) ~= length(traffic_per)
                    disp('Number of traffic conditions must match number of
traffic sections')
                else
                    break
                end
            end

            speed_lim_marker = dist*speed_lim_per;
```

```
            for i = 2:length(speed_lim_marker)
                speed_lim_marker(i) = sum(speed_lim_marker((i-1):i));
            end

            traffic_marker = dist*traffic_per;
            for i = 2:length(traffic_marker)
                traffic_marker(i) = sum(traffic_marker((i-1):i));
            end

            time = 1;
            speed_count = 1;
            traffic_count = 1;
            drive_cycle = zeros(1,2);

            while true
                if time > 1
                    if (trapz(drive_cycle(:,1),drive_cycle(:,2)/3600) >
speed_lim_marker(speed_count)) && (speed_count < length(speed_lim_cond))
                        speed_count = speed_count + 1;
                    end

                    if (trapz(drive_cycle(:,1),drive_cycle(:,2)/3600) >
traffic_marker(traffic_count)) && (traffic_count < length(traffic_cond))
                        traffic_count = traffic_count + 1;
                    end

                    if trapz(drive_cycle(:,1),drive_cycle(:,2)/3600) > dist
                        break;
                    end
                end

                if speed_lim_cond(speed_count) == 1
                    drive_cycle((time+1),1) = time;
                    drive_cycle((time+1),2) = 10;
                elseif speed_lim_cond(speed_count) == 2
                    drive_cycle((time+1),1) = time;
                    drive_cycle((time+1),2) = 15;
                elseif speed_lim_cond(speed_count) == 3
                    drive_cycle((time+1),1) = time;
                    drive_cycle((time+1),2) = 25;
                elseif speed_lim_cond(speed_count) == 4
                    drive_cycle((time+1),1) = time;
                    drive_cycle((time+1),2) = 35;
                elseif speed_lim_cond(speed_count) == 5
                    drive_cycle((time+1),1) = time;
                    drive_cycle((time+1),2) = 45;
                elseif speed_lim_cond(speed_count) == 6
                    drive_cycle((time+1),1) = time;
                    drive_cycle((time+1),2) = 55;
                elseif speed_lim_cond(speed_count) == 7
                    drive_cycle((time+1),1) = time;
                    drive_cycle((time+1),2) = 65;
                end

                if traffic_cond(traffic_count) == 1
```

```
                    drive_cycle((time+1),2) =
drive_cycle((time+1),2)*(randi([95,100],1,1)/100);
                elseif traffic_cond(traffic_count) == 2
                    drive_cycle((time+1),2) =
drive_cycle((time+1),2)*(randi([60,65],1,1)/100);
                elseif traffic_cond(traffic_count) == 3
                    drive_cycle((time+1),2) =
drive_cycle((time+1),2)*(randi([20,25],1,1)/100);
                end

                time = time + 1;
            end

            drive_cycle = [drive_cycle; (drive_cycle(end,1) + 1) 0];
%#ok<AGROW>
            accel = [0; 0.44704*(drive_cycle(2:end,2) - drive_cycle(1:(end-
1),2))];
            drive_cycle = [drive_cycle, accel]; %#ok<AGROW>

            route.name = ['Route_',num2str(route_num),'_Data.mat'];
            route.original_cycle.drive_cycle = drive_cycle;
            route.original_cycle.distance = dist;
            route.original_cycle.route_conditions.speed_lim_per =
speed_lim_per;
            route.original_cycle.route_conditions.speed_lim_cond =
speed_lim_cond;
            route.original_cycle.route_conditions.speed_lim_marker =
speed_lim_marker;
            route.original_cycle.route_conditions.traffic_per = traffic_per;
            route.original_cycle.route_conditions.traffic_cond =
traffic_cond;
            route.original_cycle.route_conditions.traffic_marker =
traffic_marker;
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            %%% Setting Up Trajectory Forecasting Variables for Advisor %%%

[route.original_cycle.TFF_Setup.Initial_SOC,route.original_cycle.TFF_Setup.SO
C_setting, ...

route.original_cycle.TFF_Setup.Section_SOC,route.original_cycle.TFF_Setup.All
owed_SOC, ...

route.original_cycle.TFF_Setup.Section_Size,route.original_cycle.TFF_Setup.Ma
trix, ...
             route.original_cycle.TFF_Setup.Section_SOC_Min_Dist, ...
             route.original_cycle.TFF_Setup.blended_SOC_and_dist] =
CNN_TFF_Setup(route.original_cycle.drive_cycle, ...

route.test_variables.battery,[]);

            save('temp_data.mat','route')
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            %%% Running Non-Gui Advisor For Needed Values From Original Cycle
%%%
```

```
            clearvars -except route

            bdclose('all')
            pause(1)

TF_advisor_func(route.original_cycle.TFF_Setup.Initial_SOC,route.test_variabl
es.engine,route.test_variables.battery);
            bdclose('all')
            pause(1)

            load('temp_data.mat')
            route.advisor_variables.gas_lhv = 42600;
            route.advisor_variables.gas_dens = 749;
            route.advisor_variables.fc_fuel_lhv = fc_fuel_lhv;
            route.advisor_variables.fc_fuel_den = fc_fuel_den;
            route.advisor_variables.kWhpgal = 33.44;
            route.advisor_variables.charger_eff = 0.85;
            route.advisor_variables.ess_tmp = ess_tmp;
            route.advisor_variables.ess_max_ah_cap = ess_max_ah_cap;
            route.advisor_variables.ess_module_num =  ess_module_num;
            route.advisor_variables.ess_coulombic_eff = ess_coulombic_eff;
            route.advisor_variables.amb_tmp = amb_tmp;
            route.advisor_variables.ess_soc = ess_soc;
            route.advisor_variables.ess_voc = ess_voc;
            route.advisor_variables.ess_r_dis = ess_r_dis;
            route.advisor_variables.ess_r_chg = ess_r_chg;
            route.advisor_variables.ess_init_soc = ess_init_soc;

            route.original_cycle.advisor_outputs.t = t;
            route.original_cycle.advisor_outputs.distance =
distance*0.000621371;
            route.original_cycle.advisor_outputs.gal = gal;
            route.original_cycle.advisor_outputs.emis = emis;
            route.original_cycle.advisor_outputs.ess_soc_hist = ess_soc_hist;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            %% Route Update %%
            clearvars -except route

            %%% Varibales For Route Updates %%%
            num_update = [1 1 1 2 2 2];
            num_update = num_update(str2double(route.name(7)));
    %         num_update = input('Number of times the route is updated: ');
fprintf('\n')

            if num_update == 1
                update_dist = route.original_cycle.distance*0.2;
            elseif num_update == 2
                update_dist = route.original_cycle.distance*[0.2 0.4];
            end
    %         while 1
    %             update_dist = input(['Distances in miles at which updates
occur within the original ', ...
    %
num2str(route.original_cycle.distance),' miles: ']); fprintf('\n')
```

```
    %              if length(update_dist) ~= num_update
    %                  disp('Number of update distances does not match the
number of updates')
    %              else
    %                  break
    %              end
    %          end
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


            %%% Variables to Take Into Account Traffic In Original Route %%%
            element = find(route.original_cycle.advisor_outputs.distance >=
update_dist(1));
            route.original_cycle_added_traffic.drive_cycle =
route.original_cycle.drive_cycle(1:(element(1)-1),1:2);

            section_dist = [update_dist, route.original_cycle.distance];
            section_dist = section_dist(2:end) - section_dist(1:(end-1));
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            %%% Compiling Final Driven Cycle Using CNN Chosen Route
Alternatives %%%
            route.final_cycle.drive_cycle = [];
            route.final_cycle.t = 0;
            route.final_cycle.distance = 0;
            route.final_cycle.gal = 0;
            route.final_cycle.emis = 0;
            route.final_cycle.ess_soc_hist = [];
            for i = 1:num_update
                if i == 1
                    prev_cycle = 'original_cycle';
                    temp = find(route.(prev_cycle).advisor_outputs.distance
>= update_dist(i));
                else
                    prev_cycle = ['cycle_update_',num2str(i-1)];
                    temp = find(route.(prev_cycle).advisor_outputs.distance
>= (update_dist(i) - update_dist(i-1)));
                end
                route.final_cycle.drive_cycle =
[route.final_cycle.drive_cycle; route.(prev_cycle).drive_cycle(1:(temp(1)-
1),2)];
                route.(prev_cycle).advisor_outputs.t =
route.(prev_cycle).advisor_outputs.t(1:temp(1));
                route.final_cycle.t = [route.final_cycle.t;
(route.(prev_cycle).advisor_outputs.t(2:end) + route.final_cycle.t(end))];
                route.(prev_cycle).advisor_outputs.distance =
route.(prev_cycle).advisor_outputs.distance(1:temp(1));
                route.final_cycle.distance = route.final_cycle.distance +
route.(prev_cycle).advisor_outputs.distance(end);
                route.(prev_cycle).advisor_outputs.gal =
max(route.(prev_cycle).advisor_outputs.gal(1:temp(1)));
                route.final_cycle.gal = route.final_cycle.gal +
route.(prev_cycle).advisor_outputs.gal;
                route.(prev_cycle).advisor_outputs.emis =
trapz(route.(prev_cycle).advisor_outputs.t,route.(prev_cycle).advisor_outputs
.emis(1:temp(1),:));
```

```
                route.final_cycle.emis = route.final_cycle.emis +
route.(prev_cycle).advisor_outputs.emis;
                route.(prev_cycle).advisor_outputs.ess_soc_hist =
route.(prev_cycle).advisor_outputs.ess_soc_hist(1:temp(1));
                route.final_cycle.ess_soc_hist =
[route.final_cycle.ess_soc_hist;
route.(prev_cycle).advisor_outputs.ess_soc_hist(1:(end-1))];

                cycle = ['cycle_update_',num2str(i)];
                route.(cycle).TFF_Setup.Initial_SOC =
route.(prev_cycle).advisor_outputs.ess_soc_hist(temp(1));

                %%% Creating and Choosing Route Alternatives Based On
Changing Traffic and Update Location %%%
                emis_gal = 1000;
                MPGGE = 0;
                alternative = 0;
                while alternative < 3
                    alternative = alternative + 1;
    %                 for alternative = 1:3
                    new_dist = route.original_cycle.distance -
update_dist(i);
                    speed_lim_marker_update =
route.original_cycle.route_conditions.speed_lim_marker(route.original_cycle.r
oute_conditions.speed_lim_marker > update_dist(i)) - update_dist(i);
                    speed_lim_cond_update =
route.original_cycle.route_conditions.speed_lim_cond((length(route.original_c
ycle.route_conditions.speed_lim_cond)-
length(speed_lim_marker_update)+1):end);
                    traffic_marker_update =
route.original_cycle.route_conditions.traffic_marker(route.original_cycle.rou
te_conditions.traffic_marker > update_dist(i)) - update_dist(i);
                    traffic_cond_update =
route.original_cycle.route_conditions.traffic_cond((length(route.original_cyc
le.route_conditions.traffic_cond)-length(traffic_marker_update)+1):end) + ...
                                          randi([(2-
alternative),2],1,length(traffic_marker_update));
                    traffic_cond_update(traffic_cond_update > 3) = 3;
                    traffic_cond_update(traffic_cond_update < 1) = 1;

                    time = 0;
                    speed_count = 1;
                    traffic_count = 1;
                    drive_cycle = zeros(1,2);

                    while true
                        if time > 1
                            if (trapz(drive_cycle(:,1),drive_cycle(:,2)/3600)
> speed_lim_marker_update(speed_count)) && (speed_count <
length(speed_lim_cond_update))
                                speed_count = speed_count + 1;
                            end

                            if (trapz(drive_cycle(:,1),drive_cycle(:,2)/3600)
> traffic_marker_update(traffic_count)) && (traffic_count <
length(traffic_cond_update))
```

```
                                traffic_count = traffic_count + 1;
                            end

                            if trapz(drive_cycle(:,1),drive_cycle(:,2)/3600)
>= new_dist
                                break;
                            end
                        end

                        if speed_lim_cond_update(speed_count) == 1
                            drive_cycle((time+1),1) = time;
                            drive_cycle((time+1),2) = 10;
                        elseif speed_lim_cond_update(speed_count) == 2
                            drive_cycle((time+1),1) = time;
                            drive_cycle((time+1),2) = 15;
                        elseif speed_lim_cond_update(speed_count) == 3
                            drive_cycle((time+1),1) = time;
                            drive_cycle((time+1),2) = 25;
                        elseif speed_lim_cond_update(speed_count) == 4
                            drive_cycle((time+1),1) = time;
                            drive_cycle((time+1),2) = 35;
                        elseif speed_lim_cond_update(speed_count) == 5
                            drive_cycle((time+1),1) = time;
                            drive_cycle((time+1),2) = 45;
                        elseif speed_lim_cond_update(speed_count) == 6
                            drive_cycle((time+1),1) = time;
                            drive_cycle((time+1),2) = 55;
                        elseif speed_lim_cond_update(speed_count) == 7
                            drive_cycle((time+1),1) = time;
                            drive_cycle((time+1),2) = 65;
                        end

                        if traffic_cond_update(traffic_count) == 1
                            drive_cycle((time+1),2) =
drive_cycle((time+1),2)*(randi([95,100],1,1)/100);
                        elseif traffic_cond_update(traffic_count) == 2
                            drive_cycle((time+1),2) =
drive_cycle((time+1),2)*(randi([60,65],1,1)/100);
                        elseif traffic_cond_update(traffic_count) == 3
                            drive_cycle((time+1),2) =
drive_cycle((time+1),2)*(randi([20,25],1,1)/100);
                        end

                        if (time == 0) && (alternative == 1)
                            original_cycle_added_traffic =
[(route.original_cycle_added_traffic.drive_cycle(end,1) + 1),
drive_cycle(end,2)];
                        elseif (time > 0) &&
(trapz(drive_cycle(:,1),drive_cycle(:,2)/3600) < section_dist(i)) &&
(alternative == 1)
                            original_cycle_added_traffic =
[original_cycle_added_traffic; (original_cycle_added_traffic(end,1) + 1),
drive_cycle(end,2)]; %#ok<AGROW>
                        end

                        time = time + 1;
```

```
                    end

                    if (i == num_update) && (alternative == 1)
                        original_cycle_added_traffic =
[original_cycle_added_traffic; (original_cycle_added_traffic(end,1) + 1) 0];
%#ok<AGROW>
    %                       accel = [0;
0.44704*(route.original_cycle_added_traffic.drive_cycle(2:end,2) -
route.original_cycle_added_traffic.drive_cycle(1:(end-1),2))];
    %                       route.original_cycle_added_traffic.drive_cycle =
[route.original_cycle_added_traffic.drive_cycle, accel];
    %                       clearvars accel;
                    end

                    drive_cycle = [drive_cycle; (drive_cycle(end,1) + 1) 0];
%#ok<AGROW>
                    accel = [0; 0.44704*(drive_cycle(2:end,2) -
drive_cycle(1:(end-1),2))];
                    drive_cycle = [drive_cycle, accel]; %#ok<AGROW>

                    [~,SOC_setting,Section_SOC,Allowed_SOC,~,Matrix,~,~] =
CNN_TFF_Setup(drive_cycle, ...

route.test_variables.battery, ...

route.(cycle).TFF_Setup.Initial_SOC);

                    if (route.test_variables.output_choice ~= 1) &&
(((route.(cycle).TFF_Setup.Initial_SOC - 0.25)/new_dist) >=
route.test_variables.SOC_ratio) && (SOC_setting == 1)
                        Output = 0;
                    else
                        CNN_Input =
CNN_Data_Preprocessing(drive_cycle,route.(cycle).TFF_Setup.Initial_SOC, ...

Matrix,Section_SOC,Allowed_SOC, ...

route.test_variables.output_choice);
                        Output =
round(predict(route.test_variables.advisor_cnn,CNN_Input),4);
                    end

                    file_name = [route.test_variables.output_name,'/Route
',route.name(7),'/', ...

route.name(1:7),'_Update_',num2str(i),'_Alternative_',num2str(alternative),'.
mat'];
                    Bat = route.test_variables.battery;
                    IS = route.(cycle).TFF_Setup.Initial_SOC;
                    OC = route.test_variables.output_choice;
                    save(file_name,'drive_cycle','Bat','IS','OC','Output')

                    if route.test_variables.output_choice ~= 1 && Output <
emis_gal
                        emis_gal = Output;
                        route.(cycle).alternative = alternative;
                        route.(cycle).drive_cycle = [];
                        route.(cycle).drive_cycle = drive_cycle;
                        route.(cycle).route_conditions.speed_lim_cond = [];
                        route.(cycle).route_conditions.speed_lim_cond =
speed_lim_cond_update;
                        route.(cycle).route_conditions.speed_lim_marker = [];
                        route.(cycle).route_conditions.speed_lim_marker =
speed_lim_marker_update;
                        route.(cycle).route_conditions.traffic_cond = [];
                        route.(cycle).route_conditions.traffic_cond =
traffic_cond_update;
                        route.(cycle).route_conditions.traffic_marker = [];
                        route.(cycle).route_conditions.traffic_marker =
traffic_marker_update;
                    elseif route.test_variables.output_choice == 1 && Output
> MPGGE
                        MPGGE = Output;
                        route.(cycle).alternative = alternative;
                        route.(cycle).drive_cycle = [];
                        route.(cycle).drive_cycle = drive_cycle;
                        route.(cycle).route_conditions.speed_lim_cond = [];
                        route.(cycle).route_conditions.speed_lim_cond =
speed_lim_cond_update;
                        route.(cycle).route_conditions.speed_lim_marker = [];
                        route.(cycle).route_conditions.speed_lim_marker =
speed_lim_marker_update;
                        route.(cycle).route_conditions.traffic_cond = [];
                        route.(cycle).route_conditions.traffic_cond =
traffic_cond_update;
                        route.(cycle).route_conditions.traffic_marker = [];
                        route.(cycle).route_conditions.traffic_marker =
traffic_marker_update;
                    elseif (alternative == 3) && (emis_gal == 1000) && (MPGGE
== 0)
                        alternative = 0;
                    end
                end

                route.original_cycle_added_traffic.drive_cycle =
[route.original_cycle_added_traffic.drive_cycle;
original_cycle_added_traffic];
                if i == num_update
                    accel = [0;
0.44704*(route.original_cycle_added_traffic.drive_cycle(2:end,2) -
route.original_cycle_added_traffic.drive_cycle(1:(end-1),2))];
                    route.original_cycle_added_traffic.drive_cycle =
[route.original_cycle_added_traffic.drive_cycle, accel];
                end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%

                %%% Setting Up and Running TF in Advisor For Chosen Route
Alternative %%%
                [~,route.(cycle).TFF_Setup.SOC_setting, ...
route.(cycle).TFF_Setup.Section_SOC,route.(cycle).TFF_Setup.Allowed_SOC, ...

route.(cycle).TFF_Setup.Section_Size,route.(cycle).TFF_Setup.Matrix, ...
                 route.(cycle).TFF_Setup.Section_SOC_Min_Dist, ...
                 route.(cycle).TFF_Setup.blended_SOC_and_dist] =
CNN_TFF_Setup(route.(cycle).drive_cycle, ...

route.test_variables.battery, ...

route.(cycle).TFF_Setup.Initial_SOC);


save('Route_Update_Params.mat','route','i','num_update','update_dist','prev_c
ycle','cycle','section_dist')
                clearvars -except route prev_cycle cycle
                bdclose('all')
                pause(1)

TF_advisor_func(route.(cycle).TFF_Setup.Initial_SOC,route.test_variables.engi
ne,route.test_variables.battery);
                bdclose('all')
                pause(1)

                clearvars -except t distance gal emis ess_soc_hist
                load('Route_Update_Params.mat')
                route.(cycle).advisor_outputs.t = t;
                route.(cycle).advisor_outputs.distance =
distance*0.000621371;
                route.(cycle).advisor_outputs.gal = gal;
                route.(cycle).advisor_outputs.emis = emis;
                route.(cycle).advisor_outputs.ess_soc_hist = ess_soc_hist;

                if i == num_update
                    route.final_cycle.drive_cycle =
[route.final_cycle.drive_cycle; route.(cycle).drive_cycle(:,2)];
                    route.final_cycle.drive_cycle =
[(0:(length(route.final_cycle.drive_cycle)-1))',
route.final_cycle.drive_cycle];
                    accel = [0;
0.44704*(route.final_cycle.drive_cycle(2:end,2) -
route.final_cycle.drive_cycle(1:(end-1),2))];
                    route.final_cycle.drive_cycle =
[route.final_cycle.drive_cycle, accel];
                    route.final_cycle.t = [route.final_cycle.t;
(route.(cycle).advisor_outputs.t(2:end) + route.final_cycle.t(end))];
                    route.final_cycle.distance = route.final_cycle.distance +
route.(cycle).advisor_outputs.distance(end);
                    route.(cycle).advisor_outputs.gal =
max(route.(cycle).advisor_outputs.gal);
                    route.final_cycle.gal = route.final_cycle.gal +
route.(cycle).advisor_outputs.gal;
                    route.(cycle).advisor_outputs.emis =
trapz(route.(cycle).advisor_outputs.t,route.(cycle).advisor_outputs.emis);
                    route.final_cycle.emis = route.final_cycle.emis +
route.(cycle).advisor_outputs.emis;
```

```
                    route.final_cycle.ess_soc_hist =
[route.final_cycle.ess_soc_hist; route.(cycle).advisor_outputs.ess_soc_hist];
                end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            %%% Running Recharge Sim to Calculate MPGGE %%%
            save('temp_data.mat','route')
            if (exist('C:\Users\Joseph\Documents\MATLAB\Sim
Cache\slprj','dir') ~= 0)
                rmdir('C:\Users\Joseph\Documents\MATLAB\Sim Cache\slprj','s')
                delete('C:\Users\Joseph\Documents\MATLAB\Sim Cache\*')
            end

            sim('recharge_kWh',100000)
            route.final_cycle.MPGGE =
route.final_cycle.distance/(route.final_cycle.gal/(route.advisor_variables.ga
s_lhv/route.advisor_variables.fc_fuel_lhv*route.advisor_variables.gas_dens/ro
ute.advisor_variables.fc_fuel_den) + ...

recharge_kWh/route.advisor_variables.charger_eff/route.advisor_variables.kWhp
gal);
            route.final_cycle.CNN_TF_Outputs =
[route.final_cycle.MPGGE,(route.final_cycle.emis(1:(end-
1))/route.final_cycle.distance),route.final_cycle.gal];
            save('temp_data.mat','route')
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            %% CS and CD Results Comparison %%
            %%% Running CS and CD Strategy For Traffic Filled Original Cycle
%%%
            clearvars -except route

            Slope_Option = 2;
            cyc_mph = route.original_cycle_added_traffic.drive_cycle(:,1:2);

save('..\..\..\..\advisor\data\drive_cycle\CYC_TrajFore_Fuz.mat','cyc_mph','S
lope_Option')

            bdclose('all')
            pause(1)
            CS_outputs =
CS_advisor_func(route.advisor_variables.ess_init_soc,route.test_variables.eng
ine,route.test_variables.battery);
            bdclose('all')
            pause(1)
            load('temp_data.mat')
            route.final_cycle.CS_Outputs = [CS_outputs, max(gal)];
            clearvars -except route
            save('temp_data.mat','route')

            bdclose('all')
            pause(1)
            CD_outputs =
CD_advisor_func(route.advisor_variables.ess_init_soc,route.test_variables.eng
ine,route.test_variables.battery);
            bdclose('all')
            pause(1)
            load('temp_data.mat')
            route.final_cycle.CD_Outputs = [CD_outputs, max(gal)];
            clearvars -except route

            result = route.test_variables.output_choice;
            route_num = str2double(route.name(7));
            if (result == 1) && (route.final_cycle.CNN_TF_Outputs(result) >=
route.final_cycle.CS_Outputs(result)) && ...
                (route.final_cycle.CNN_TF_Outputs(result) >=
route.final_cycle.CD_Outputs(result))
                file_name = [route.test_variables.output_name,'/Route
',route.name(7),'/',route.name];
                save(file_name,'route')
                break;
            elseif (result ~= 1) && (route.final_cycle.CNN_TF_Outputs(result)
<= route.final_cycle.CS_Outputs(result)) && ...
                    (route.final_cycle.CNN_TF_Outputs(result) <=
route.final_cycle.CD_Outputs(result))
                file_name = [route.test_variables.output_name,'/Route
',route.name(7),'/',route.name];
                save(file_name,'route')
                break;
            end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        end
    end
end

close all; clear; clc
%% Function Calls %%
function [Initial_SOC,SOC_setting,Section_SOC,Allowed_SOC, ...
          Section_Size,Matrix,Section_SOC_Min_Dist, ...
          blended_SOC_and_dist] =
CNN_TFF_Setup(drive_cycle,battery,Initial_SOC)
    if battery == 25
        Ah_Cap = 25;
    elseif battery == 28
        Ah_Cap = 28;
    end

    %% Path Identifier Version 5 %%
    Slope_Option = 2;
    cyc_mph = drive_cycle(:,1:2);

save('..\..\..\..\advisor\data\drive_cycle\CYC_TrajFore_Fuz.mat','cyc_mph','S
lope_Option')

    interval_distance = zeros(1,length(cyc_mph(:,1))-1);
    for i = 2:length(cyc_mph(:,1))
```

```
        interval_distance(i-1) = trapz(cyc_mph((i-1):i,1),cyc_mph((i-
1):i,2)/3600);
    end
    total_distance = sum(interval_distance);

    Max_SOC = 0.8;
    Lowest_SOC = 0.25;

    if isempty(Initial_SOC) == 1
        Initial_SOC = 0.8;

        if total_distance < 5
            Initial_SOC = 0.40;
        elseif total_distance < 10
            Initial_SOC = 0.50;
        elseif total_distance < 25
            Initial_SOC = 0.60;
        elseif total_distance < 40
            Initial_SOC = 0.70;
        end
    end

    Matrix = zeros(2,length(cyc_mph(:,1))); % [Priority; SOC]

    for i = 1:length(cyc_mph(:,2))
        if cyc_mph(i,2) <= 25
            Matrix(1,i) = 3;
        elseif (cyc_mph(i,2) > 25) && (cyc_mph(i,2) < 55)
            Matrix(1,i) = 2;
        elseif (cyc_mph(i,2) >= 55)
            Matrix(1,i) = 1;
        end
    end

    %%Calculating Section Size and Distance
    Section_Size = 0;
    Section_Dist = 0;

    j = 0;
    for i = 2:length(Matrix(1,:))
        if Matrix(1,i) == Matrix(1,i-1)
            j = j + 1;
            Section_Dist(end) = Section_Dist(end) + interval_distance(i-1);
        else
            Section_Size = [Section_Size, j]; %#ok<AGROW>
            Section_Dist = [Section_Dist, interval_distance(i-1)];
%#ok<AGROW>
            j = 1;
        end
    end

    if j == (length(Matrix(1,:)) - 1)
        Section_Size = j;
    else
        Section_Size = [Section_Size(2:end), (i-sum(Section_Size)-1)];
    end

            %%SOC Needed per Section
        ampHrs_per_Segment = zeros(1,length(cyc_mph(:,1))-1);
        %%%%%%%%%%%%%%%%%%
        for i = 2:length(cyc_mph(:,1))
            if Matrix(1,i) == 3 %SOC equation for priority 3
                    ampHrs_per_Segment(i-1) = interval_distance(i-
1)*0.87*(0.0198*cyc_mph(i,2) + 0.1689);
            elseif Matrix(1,i) == 2 %SOC equation for priority 2
                    ampHrs_per_Segment(i-1) = interval_distance(i-
1)*0.42*(0.0198*cyc_mph(i,2) + 0.1689);
            elseif Matrix(1,i) == 1 %SOC equation for priority 1
                    ampHrs_per_Segment(i-1) = interval_distance(i-
1)*0.34*(0.0198*cyc_mph(i,2) + 0.1689);
            end
        end
        %%%%%%%%%%%%%%%%%%%%%

        Section_AmpHrs = zeros(size(Section_Size));

        for i = 1:length(Section_AmpHrs)
            if i == 1
                Section_AmpHrs(i) =
sum(ampHrs_per_Segment(1:Section_Size(i)));
            else
                Section_AmpHrs(i) =
sum(ampHrs_per_Segment((1+sum(Section_Size(1:(i-
1)))):sum(Section_Size(1:i))));
            end
        end

        if Bat_Type == 1
            Section_SOC = Section_AmpHrs/(Ah_Cap*0.55);
        else
            Section_SOC = Section_AmpHrs/Ah_Cap;
        end

    %%Sorting Sections by Priority and Length
    Section_Priority = zeros(size(Section_Size));

    for i = 1:length(Section_Size)
        Section_Priority(i) = Matrix(1,(1+sum(Section_Size(1:i))));
    end

    Section_Sort = [Section_Priority; Section_Dist; Section_SOC]';
    [Section_Sort,index] = sortrows(Section_Sort,'descend');

    %%Setting Section SOC_min
    Section_SOC_Mins = zeros(1,length(index));
    if Initial_SOC == Max_SOC
        Initial_SOC = Initial_SOC - 0.005;
    end

    allowed_SOC = Initial_SOC - Lowest_SOC;
    %%% Determine settings for determining sections' SOC_Min
    if allowed_SOC >= (round(sum(Section_Sort(:,3)),3) - 0.125) %Enough for
whole route in all electric mode; launch speed that includes all section
speeds
        SOC_setting = 1;
    else
        k_EV = length(find(Section_Sort(:,1) == 3));
        k_Bl = length(find(Section_Sort(:,1) == 2));
        if (allowed_SOC < (round(sum(Section_Sort(:,3)),3)) - 0.125) &&
(allowed_SOC >= round(sum(Section_Sort(1:(k_EV+k_Bl),3)),3)) %Enough for EV
and Blended suggested sections in all electric mode; launch speed that
includes EV and Blended section speeds
            SOC_setting = 2;
        elseif (allowed_SOC < round(sum(Section_Sort(1:(k_EV+k_Bl),3)),3)) &&
(allowed_SOC >= round(sum(Section_Sort(1:k_EV,3)),3)) %Enough for just EV
suggested sections in all eletric mode; launch speed that includes EV section
speeds
            SOC_setting = 3;
        elseif allowed_SOC < round(sum(Section_Sort(1:k_EV,3)),3) %SOC will
run out in the EV suggested sections; launch speed that includes EV section
speeds
            SOC_setting = 4;
        end
    end


    %%% Different SOC settings for choosing SOC_Min values
    if SOC_setting == 1
        flip_index = fliplr(index');

        for i = 1:(length(Section_SOC_Mins) - 1)
            if (allowed_SOC - sum(Section_SOC(index(1:(end-i))))) <= 0
                Section_SOC_Mins(flip_index(i)) = 0;
            else
                Section_SOC_Mins(flip_index(i)) = allowed_SOC -
sum(Section_SOC(index(1:(end-i))));
                allowed_SOC = sum(Section_SOC(index(1:(end-i))));
            end
        end

        Section_SOC_Mins(index(1)) = allowed_SOC;
        Allowed_SOC = Section_SOC_Mins;

        Section_SOC_Mins = Lowest_SOC*ones(size(Section_SOC_Mins));

    elseif SOC_setting == 2
        flip_index = fliplr(index');

        for i = 1:(length(Section_SOC_Mins) - 1)
            if (allowed_SOC - sum(Section_SOC(index(1:(end-i))))) <= 0
                Section_SOC_Mins(flip_index(i)) = 0;
            else
                Section_SOC_Mins(flip_index(i)) = allowed_SOC -
sum(Section_SOC(index(1:(end-i))));
```

```
                allowed_SOC = sum(Section_SOC(index(1:(end-i))));
            end
        end

        Section_SOC_Mins(index(1)) = allowed_SOC;
        residual = sum(Section_SOC_Mins(sort(index((k_EV+k_Bl+1):end))));
        blend_ratios =
Section_Dist(sort(index((k_EV+k_Bl+1):end)))/sum(Section_Dist(sort(index((k_E
V+k_Bl+1):end))));
        Section_SOC_Mins(sort(index((k_EV+k_Bl+1):end))) =
residual*blend_ratios;
        Allowed_SOC = Section_SOC_Mins;

        for i = 1:length(Section_SOC_Mins)
            if i == 1
                Section_SOC_Mins(i) = Initial_SOC - Section_SOC_Mins(i);
            else
                Section_SOC_Mins(i) = Section_SOC_Mins(i-1) -
Section_SOC_Mins(i);
            end
        end

        blended_Section_SOC_Mins =
Section_SOC_Mins(sort(index((k_EV+k_Bl+1):end)));
        blended_Section_distances =
Section_Dist(sort(index((k_EV+k_Bl+1):end)));

    elseif SOC_setting == 3
        flip_index = fliplr(index');

        for i = 1:(length(Section_SOC_Mins) - 1)
            if (allowed_SOC - sum(Section_SOC(index(1:(end-i))))) <= 0
                Section_SOC_Mins(flip_index(i)) = 0;
            else
                Section_SOC_Mins(flip_index(i)) = allowed_SOC -
sum(Section_SOC(index(1:(end-i))));
                allowed_SOC = sum(Section_SOC(index(1:(end-i))));
            end
        end

        Section_SOC_Mins(index(1)) = allowed_SOC;
        residual = sum(Section_SOC_Mins(sort(index((k_EV+1):(k_EV+k_Bl)))));
        blend_ratios =
Section_Dist(sort(index((k_EV+1):(k_EV+k_Bl))))/sum(Section_Dist(sort(index((
k_EV+1):(k_EV+k_Bl)))));
        Section_SOC_Mins(sort(index((k_EV+1):(k_EV+k_Bl)))) =
residual*blend_ratios;
        Allowed_SOC = Section_SOC_Mins;

        for i = 1:length(Section_SOC_Mins)
            if i == 1
                Section_SOC_Mins(i) = Initial_SOC - Section_SOC_Mins(i);
            else
                Section_SOC_Mins(i) = Section_SOC_Mins(i-1) -
Section_SOC_Mins(i);
```

```
            end
        end

        blended_Section_SOC_Mins =
Section_SOC_Mins(sort(index((k_EV+1):(k_EV+k_Bl))));
        blended_Section_distances =
Section_Dist(sort(index((k_EV+1):(k_EV+k_Bl))));

    elseif SOC_setting == 4
        flip_index = fliplr(index');

        for i = 1:(length(Section_SOC_Mins) - 1)
            if (allowed_SOC - sum(Section_SOC(index(1:(end-i))))) <= 0
                Section_SOC_Mins(flip_index(i)) = 0;
            else
                Section_SOC_Mins(flip_index(i)) = allowed_SOC -
sum(Section_SOC(index(1:(end-i))));
                allowed_SOC = sum(Section_SOC(index(1:(end-i))));
            end
        end

        Section_SOC_Mins(index(1)) = allowed_SOC;
        Allowed_SOC = Section_SOC_Mins;

        for i = 1:length(Section_SOC_Mins)
            if i == 1
                Section_SOC_Mins(i) = Initial_SOC - Section_SOC_Mins(i);
            else
                Section_SOC_Mins(i) = Section_SOC_Mins(i-1) -
Section_SOC_Mins(i);
            end
        end

    end

    %%SOC Throughout Route
    SOC_Mins = zeros(1,(length(cyc_mph(:,1))-1));
    j = 1 + length(Section_SOC_Mins);

    for i = 1:length(Section_SOC_Mins)
        if i == 1
            SOC_Mins(1:Section_Size(i)) =
Section_SOC_Mins(i)*ones(1,Section_Size(i));
        else
            SOC_Mins((1+sum(Section_Size(1:(i-1)))):sum(Section_Size(1:i))) =
Section_SOC_Mins(i)*ones(1,Section_Size(i));
        end
    end

    SOC_Mins = [SOC_Mins(1) SOC_Mins];
    Matrix(2,:) = SOC_Mins;
    Section_SOC_Min_Dist = [Section_SOC_Mins; Section_Dist];

    if SOC_setting == 1 || SOC_setting == 4
        blended_SOC_and_dist = [0;0];
    else
        blended_SOC_and_dist = [blended_Section_SOC_Mins;
blended_Section_distances];
    end


save('..\..\..\..\advisor\models\Traj_Fore_Fuzzy_SOC_Variables.mat','Section_
SOC_Min_Dist', ...
         'Matrix','SOC_setting','blended_SOC_and_dist')
end

function outputs = TF_advisor_func(Initial_SOC,engine,battery)
    if (exist('C:\Users\Joseph\Documents\MATLAB\Sim Cache\slprj','dir') ~= 0)
        rmdir('C:\Users\Joseph\Documents\MATLAB\Sim Cache\slprj','s')
        delete('C:\Users\Joseph\Documents\MATLAB\Sim Cache\*')
    end

    if Initial_SOC == 0.7950
        Initial_SOC = 0.8;
    end

input.init.saved_veh_file=['PARALLEL_TF_FUZ_defaults_',engine,'L_PB',num2str(
battery),'_in'];
    [~,~]=adv_no_gui('initialize',input);
    input.modify.param={'ess_init_soc'};
    input.modify.value={Initial_SOC};
    [~,~]=adv_no_gui('modify',input);
    input.cycle.param = {'cycle.name','cycle.soc'};
    input.cycle.value={'CYC_TrajFore_Fuz','off'};
    [~,b]=adv_no_gui('drive_cycle',input);

    if ~isempty(b)
        outputs = [b.cycle.MPGGE, b.cycle.hc_gpm, b.cycle.co_gpm,
b.cycle.nox_gpm];
    else
        outputs = [];
    end
end

function Input =
CNN_Data_Preprocessing(drive_cycle,Initial_SOC,Matrix,Section_SOC, ...
                                        Allowed_SOC,output_choice)
    load('Input_Boundaries.mat','V_neg','V_pos','A_neg','A_pos');

    i = round((resolution_rounding(A_pos,0.1) -
resolution_rounding(A_neg,0.1))/0.1 + 1);
    j = round((resolution_rounding(V_pos,0.5) -
resolution_rounding(V_neg,0.5))/0.5 + 1);

    Input = zeros(i,j,3);
    Input(:,:,:) = Input(:,:,:) + Initial_SOC;

    rounded_vel = resolution_rounding(drive_cycle(:,2),0.5);
    rounded_accel = resolution_rounding(drive_cycle(:,3),0.1);
    section = 1;
    prev_layer = 0;
    for ii = 1:length(drive_cycle(:,1))
        row = round((rounded_accel(ii) - resolution_rounding(A_neg,0.1))/0.1
+ 1);
        if row <= 0
            row = 1;
        elseif row > i
            row = i;
        end

        col = round((rounded_vel(ii) - resolution_rounding(V_neg,0.5))/0.5 +
1);
        if col <= 0
            col = 1;
        elseif col > j
            col = j;
        end

        if ii == 1
            if Section_SOC(section) == 0
                layer = 1;
            else
                ratio =
round((Allowed_SOC(section)/Section_SOC(section)),3);%1 -
round((Allowed_SOC(section)/Section_SOC(section)),3);
                if ratio >= 1%<= 0%>= 1
                    layer = 3;
                elseif (ratio < 1) && (ratio > 0)
                    layer = 2;
                elseif ratio == 0%1%0
                    layer = 1;
                elseif ratio < 0
                    layer = 3;
                end
            end
        else
            if Matrix(1,ii) ~= Matrix(1,(ii-1))
                section = section + 1;
                if Section_SOC(section) == 0
                    layer = 1;
                else
                    ratio =
round((Allowed_SOC(section)/Section_SOC(section)),3);%1 -
round((Allowed_SOC(section)/Section_SOC(section)),3);
                    if ratio >= 1%<= 0%>= 1
                        layer = 3;
                    elseif (ratio < 1) && (ratio > 0)
                        layer = 2;
                    elseif ratio == 0%1%0
                        layer = 1;
                    elseif ratio < 0
                        layer = 3;
                    end
                end
            end
        end

        if layer == 1
            if prev_layer == 3
                Input(row,col,layer) = Input(row,col,layer) + 1;
            elseif prev_layer == 2
                Input(row,col,layer) = Input(row,col,layer) + 1;
            elseif prev_layer == 1
                Input(row,col,layer) = Input(row,col,layer) + 1;
            end
        elseif layer == 2
            if prev_layer == 3
                Input(row,col,layer) = Input(row,col,layer) + (1-ratio);
            else
                Input(row,col,layer) = Input(row,col,layer) + (1-ratio);
            end
        elseif layer == 3
            if ratio < 0
                Input(row,col,layer) = Input(row,col,layer) - 1;
            else
                Input(row,col,layer) = Input(row,col,layer) + 1;
            end
        end

        prev_layer = layer;
    end

    if output_choice ~= 1
        Input = Input(:,:,1:2);
    end
end

function outputs = CS_advisor_func(Initial_SOC,engine,battery)
    if (exist('C:\Users\Joseph\Documents\MATLAB\Sim Cache\slprj','dir') ~= 0)
        rmdir('C:\Users\Joseph\Documents\MATLAB\Sim Cache\slprj','s')
        delete('C:\Users\Joseph\Documents\MATLAB\Sim Cache\*')
    end

    if Initial_SOC == 0.7950
        Initial_SOC = 0.8;
    end

input.init.saved_veh_file=['PARALLEL_defaults_',engine,'L_PB',num2str(battery
),'_in'];
    input.init.comp_files.comp={'powertrain_control'};
    input.init.comp_files.name={'PTC_PAR_Charge_Sustain'};
    input.init.comp_files.ver={'par'};
    input.init.comp_files.type={'man'};
    [~,~]=adv_no_gui('initialize',input);
    input.modify.param={'ess_init_soc'};
    input.modify.value={Initial_SOC};
    [~,~]=adv_no_gui('modify',input);
    input.cycle.param = {'cycle.name','cycle.soc'};
    input.cycle.value={'CYC_TrajFore_Fuz','off'};
    [~,b]=adv_no_gui('drive_cycle',input);
    if ~isempty(b)
        outputs = [b.cycle.MPGGE, b.cycle.hc_gpm, b.cycle.co_gpm,
b.cycle.nox_gpm];
    else
        outputs = [];
    end
end

function outputs = CD_advisor_func(Initial_SOC,engine,battery)
    if (exist('C:\Users\Joseph\Documents\MATLAB\Sim Cache\slprj','dir') ~= 0)
        rmdir('C:\Users\Joseph\Documents\MATLAB\Sim Cache\slprj','s')
        delete('C:\Users\Joseph\Documents\MATLAB\Sim Cache\*')
    end

    if Initial_SOC == 0.7950
        Initial_SOC = 0.8;
    end

input.init.saved_veh_file=['PARALLEL_defaults_',engine,'L_PB',num2str(battery
),'_in'];
    input.init.comp_files.comp={'powertrain_control'};
    input.init.comp_files.name={'PTC_PAR_Charge_Deplete'};
    input.init.comp_files.ver={'par'};
    input.init.comp_files.type={'man'};
    [~,~]=adv_no_gui('initialize',input);
    input.modify.param={'ess_init_soc'};
    input.modify.value={Initial_SOC};
    [~,~]=adv_no_gui('modify',input);
    input.cycle.param = {'cycle.name','cycle.soc'};
    input.cycle.value={'CYC_TrajFore_Fuz','off'};
    [~,b]=adv_no_gui('drive_cycle',input);

    if ~isempty(b)
        outputs = [b.cycle.MPGGE, b.cycle.hc_gpm, b.cycle.co_gpm,
b.cycle.nox_gpm];
    else
        outputs = [];
    end
end
```