Implementation of Radial Basis Function Artificial Neural Network into an Adaptive Equivalent Consumption Minimization Strategy for Optimized Control of a Hybrid Electric Vehicle

Thomas P. Harris
West Virginia University, th0036@mix.wvu.edu

IMPLEMENTATION OF RADIAL BASIS FUNCTION ARTIFICIAL NEURAL NETWORK INTO AN ADAPTIVE EQUIVALENT CONSUMPTION MINIMIZATION STRATEGY FOR OPTIMIZED CONTROL OF A HYBRID ELECTRIC VEHICLE

Thomas Harris

Thesis submitted to the Statler College of Engineering and Mineral Resources at West Virginia University in Partial fulfillment of the requirements for the degree of

Master of Science
in
Mechanical Engineering

Department of Mechanical and Aerospace Engineering

Morgantown, West Virginia
2020

Keywords: Hybrid-Electric Vehicle, Torque Split Algorithm, Artificial Neural Network, Equivalent Consumption Minimization Strategy, Optimal Control Strategy

Abstract

IMPLEMENTATION OF RADIAL BASIS FUNCTION ARTIFICIAL NEURAL NETWORK INTO AN ADAPTIVE EQUIVALENT CONSUMPTION MINIMIZATION STRATEGY FOR OPTIMIZED CONTROL OF A HYBRID ELECTRIC VEHICLE

Thomas Harris

Continued increases in the emission of greenhouse gases by passenger vehicles has accelerated the production of hybrid electric vehicles. With this increase in production, there has been a parallel demand for continuously improving strategies of hybrid electric vehicle control. The goal of an ideal control strategy is to maximize fuel economy while minimizing emissions. The design and implementation of an optimized control strategy is a complex challenge. Methods exist by which the globally optimal control strategy may be found. However, these methods are not applicable in real-world driving applications since these methods require *a priori* knowledge of the upcoming drive cycle. Real-time control strategies use the global optimal as a benchmark against which performance can be evaluated. Real-time strategies incorporate methods such as drive cycle prediction algorithms, parameter feedback, driving pattern recognition algorithms, etc. The goal of this work is to use a previously defined strategy which has been shown to closely approximate the global optimal and implement a radial basis function (RBF) artificial neural network (ANN) that dynamically adapts the strategy based on past driving conditions. The strategy used is the Equivalent Consumption Minimization Strategy (ECMS) [1], which uses an equivalence factor to define the control strategy. The equivalence factor essentially defines the torque split between the electric motor and internal combustion engine. Consequently, the equivalence factor greatly affects fuel economy. An equivalence factor that is optimal (with respect to fuel economy) for a single drive cycle can be found offline – with *a priori* knowledge of the drive cycle. The RBF ANN is used to dynamically update the equivalence factor by examining a past time window of driving characteristics. A total of 30 sets of training data are used to train the RBF ANN, each set contains characteristics from a different drive cycle. Each drive cycle is characterized by 9 parameters. For each drive cycle, the optimal equivalence factor is determined and included in the training data. The performance of the RBF ANN is evaluated against the fuel economy obtained with the optimal equivalence factor from the ECMS. For the majority of drive cycles examined, the RBF ANN implementation is shown to produce fuel economy values that are within +/- 2.5% of the fuel economy obtained with the optimal equivalence factor. The advantage of the RBF ANN is that it does not require *a priori* drive cycle knowledge and is able to be implemented real time while meeting or exceeding the performance of the optimal ECMS. Recommendations are made on how the RBF ANN could be improved to produce better results across a greater array of driving conditions.

Acknowledgements

I would like to acknowledge the members of my thesis committee. First, my committee chair and advisor, Dr. Andrew Nix, for his involvement and guidance in the EcoCAR program. His levels of support and dedication are astounding. Personally, I thank Dr. Nix for encouraging and convincing me to finish my master's degree when I was considering quitting and joining the workforce. Dr. Nix truly fights for his students and is very loyal to them. Secondly, I think Dr. Scott Wayne for his willing support in the early years of the EcoCAR program. Many times, I would consult with Dr. Wayne and seek his technical expertise in vehicle modelling and simulation. Dr. Wayne was always willing to give valuable guidance. Thirdly, I thank Dr. Mario Perhinschi for his guidance in the creation of the artificial neural network that is used in this thesis. Dr. Perhinschi introduced me to artificial intelligence in mechanical engineering, and his guidance on the topic has been invaluable.

I would like to acknowledge the two primary mentors of the WVU EcoCAR team. I especially thank the General Motors mentor William Cawthorne for his involvement in the team. Bill has been instrumental in the controls and modeling efforts of the team, and I have learned much from him. His experience in industry has been invaluable. I thank Bill for his willingness to freely share his wealth of knowledge in the area of controls and modeling as well as in the area of team management. I also thank the WVU MathWorks mentor Walt Stuart. His help throughout the years has been invaluable to the EcoCAR program. I especially thank Walt for his help and guidance in some of the artificial neural network modelling contained in this thesis.

I also acknowledge and thank Dr. Brian Woerner for his involvement and leadership in the EcoCAR program.

I would also like to acknowledge the leadership team of the EcoCAR program. First, project manager Benton Morris for his efforts in management of the project. From a budget perspective, Benton has been good to the propulsion controls and modelling team – always finding room in the budget for computers and technical equipment we need. Second, engineering manager, Aaron Mull, for wiring the Blazer and being able to provide a wealth of knowledge across all areas of the vehicle. Aaron is one of the most hard working and knowledgeable members of the team. Third, Clay Vincent and Alex Colon for their efforts on the connected and automated vehicle team. Lastly, Jared Diethorn, my co-lead on the propulsion controls and modelling team. I especially thank Jared for his help in my last semester. He stepped up and took on a lot of the leadership role, giving me the time I needed to finish this thesis.

I also acknowledge the undergraduate students who were very dedicated in working on the propulsion controls and modelling team. I especially thank Rose Webber for her help in year one of the competition. I could not have survived the year without her help. I also thank Jeremiah Schollar, and Calvin Wolfes, for their help in year 2 of the competition. Lastly, I thank Holden Frasier for his involvement in the team and his willingness to step up and accept the role of a graduate team-lead.

Dedication

This thesis is dedicated especially to my mom, dad, and sister. I thank them for their unwavering support in all my decisions regarding my education. I especially thank Dad for keeping me grounded when things were rough.

This thesis is also dedicated to my extended family. I thank them for all their continuous encouragement and support. I especially thank my aunt, Rebecca Bland, and my uncle, Brian Richards, for their wisdom and direction regarding my education.

Contents

Table of Figures

List of Acronyms

A-ECMS	Adaptive-Equivalent Consumption Minimization Strategy
ANN	Artificial Neural Network
Art_Rural	Artemis Rural Road
AVTC	Advanced Vehicle Technical Competition

AWD	All-Wheel Drive
BMS	Battery Management System
Braun_City	Braunschweig City Driving Cycle
BSFC	Brake Specific Fuel Consumption
CAN	Controller Area Network
CAV	Connected and Automated Vehicle
CD	Charge Depleting
CS	Charge Sustaining
DP	Dynamic Programming
ECE_Extra	ECE Extra-Urban Driving Cycle
ECMS	Equivalent Consumption Minimization Strategy
FWD	Front-Wheel Drive
HEV	Hybrid Electric Vehicle
HEV_SM	Hybrid Electric Vehicle Single-Mode Control
HV	High Voltage
ICE	Internal Combustion Engine
ICV	Internal Combustion Vehicle
Jap1015	Japanese 10-15 Mode

Jap15	Japanese 15 Mode
MIL	Model in the Loop
MMS	Motor Management System
mpg	Miles per Gallon
NO_x	Nitrogen Oxide
PCM	Propulsion Controls & Modeling
PHEV	Plug-in Hybrid Electric Vehicle
PI	Proportional Integral
PM	Particulate Matter
PSI	Propulsion System Integration
RDP	Representative Driving Patterns
RNN	Recurrent Neural Network
SI	Spark Ignition
Sim	Simulation
SOE_{ref}	State of Energy Reference
SOC	State of Charge
TSA	Torque-Split Algorithm
VIL	Vehicle In the Loop

List of Symbols

σ^2	variance
T_e	engine torque
T_m	motor torque
ω_e	engine speed
$\dot{m}_{elec}$	electrical consumption
P_{eng}	engine power
Q_{lhv}	lower heating value of fuel
t_o	initial time
t_f	final time
μ	weighting factor for NO_x
v	weighting factor for PM
SOC_{target}	target SOC
max	maximum
min	minimum
η	efficiency
$\sum$	summation
λ	optimal control value

z	matrix of hidden layer weights
N_H	number of hidden layers
k	neuron number
$\bar{c}$	center vector
O_k	output neuron k
$\bar{x}$	input data set
K_p	tuning parameter
s^o	tuning parameter
x^{sp}	reference SOC
φ_k	basis function
w_k	weights
$\lVert\ \rVert$	normalization
$\otimes$	element wise product
π	pi
$\bar{y}$	output of training data
H	cost vector
T_w	time window

List of Units

mpg	miles per gallon
%	percent
ga	gravitational acceleration
s	seconds
m	meters
kWh	kilowatt hour
g	grams
radps	radians per second

1. Introduction

The objective of this research is to design an artificial neural network for implementation with an adaptive control strategy for a hybrid electric vehicle. The main goal of the control strategy is to maximize fuel economy over an unknown drive cycle. The general purpose of a hybrid electric vehicle control strategy is to split the torque between the electric motor and internal combustion engine (ICE) in a way that maximizes efficiency. These control strategies are colloquially known as torque-split algorithms (TSA). A multitude of hybrid electric vehicle control strategies exist; however, not all are created equally.

The best performing strategies (globally optimal) are only implementable if the future driving conditions are known *a priori*. In general, everyday driving scenarios, this information is not available. To overcome this lack of knowledge, the best performing control strategies have been augmented with predictive and/or learning capabilities with comparable results to those obtained when the future conditions are known *a priori* [1].

This work explains the implementation of a radial basis function (RBF) artificial neural network (ANN) with an optimal control strategy. The ANN is used to examine a past time window of driving conditions and make assumptions of future driving conditions – for which control parameters are estimated. All the analysis and modeling described in this work is performed exclusively in a simulation environment. The following pages describe the full vehicle model used for training data generation, the control algorithm model, and the design and implementation of the artificial neural network. Results gathered from the controller with the artificial neural network (ANN) implementation are then presented and analyzed. Lastly, conclusions are drawn, and recommendations are made for improvements in any future work.

1.1. Hybrid Electric Vehicles

As emission regulations continue to tighten, hybrid electric vehicles have become increasingly ubiquitous. Heightened awareness of global warming and the harmful environmental impact of traditional combustion engines has accelerated the drive towards all-electric transportation. In the U.S. alone, the economic sector of transportation accounted for a staggering 28.2% of the 2018 greenhouse gas emissions [2]. More than 90% of the fuel used in transportation is petroleum based. This includes primarily gasoline and diesel. The largest section - over half - of transportation greenhouse gas production comes from passenger cars and light-duty trucks [2].

Hybrid electric vehicles (HEVs) are standing in the gap between conventional and all-electric transportation. Until all-electric transportation is facilitated and accepted, the HEV will likely continue to bridge this gap. Electric hybridization is so appealing because it has been shown to produce significantly less emissions than conventional gas-powered vehicles [1]. Emissions are reduced in two primary ways: regenerative braking and engine operation optimization. Regenerative braking can be used to convert the energy used during deceleration to electrical energy which is stored in the high-voltage (HV) battery for later use. The engine operation can be optimized by using the electric motor to push the engine into more efficient areas of operation in which fewer emissions are produced and fuel consumption is minimized.

High fuel economy also contributes to the appeal of electric hybridization. One reason for the increase in fuel economy is the inherent efficiency advantage that an electric motor has over a traditional internal combustion engine (ICE). At peak performance, an ICE can, at best, operate around 40% efficiency. Conversely, an electric motor can have efficiencies greater than 90%. This, coupled with the more efficient engine operation, greatly improves fuel economy. Additionally, the generation of electric power is more efficient than internal combustion engines, even considering grid transmission losses.

Hybrid electric vehicles can be classified as either a plug-in hybrid-electric vehicle (PHEV) or a hybrid electric vehicle (HEV). A PHEV is characterized by a large HV battery which gives the vehicle the ability to function both as a completely electric vehicle and a hybrid. A PHEV can be plugged in each night to be charged. If, during the course of driving, the HV battery state of charge (SOC) drops below a predefined threshold, or the torque demanded by the driver exceeds what the electric motor is able to produce, then the ICE can be used in tandem with the electric motor. Both the electric motor and ICE can be used as primary sources of propulsion either individually or in tandem.

A HEV is characterized by a smaller HV battery. The battery is not necessarily large enough to support a fully electric driving mode. A HEV cannot be plugged in to recharge the HV battery. The motor is used to augment and support the engine operation. Both PHEVs and HEVs allow for energy recapture through regenerative braking.

1.2. Vehicle Architectures

There are two primary HEV architectures: parallel and series (Figure 1). In a series architecture, the engine is not mechanically coupled to the wheels. It is connected to a generator which supplies electrical energy to an inverter. The inverter provides electricity to the motor which drives the wheels. A HV battery also supplies electricity to the inverter. The generator can be used to charge the HV battery if the SOC drops below a predefined threshold. As shown in Figure 1, the power flow of the series architecture originates from the battery and engine. Power flow from the engine travels through the generator and inverter. At the inverter, power flow from the HV battery is combined with the power flow from the generator. The power flow then travels to the motor and through a differential before arriving at the wheels. The most significant difference in the power flow of the engine and HV battery is that the power flow from the HV battery works in reverse. Not only can the engine/generator system be used to reverse the power flow through the inverter, but regenerative braking can also be used. Instead of using friction brakes to slow the vehicle, negative torque can be commanded from the motor – effectively reversing the electrical power flow. The reversal of power from regenerative braking, allows for the capture of free energy – energy supplied by the inertia of the vehicle instead of the fuel.

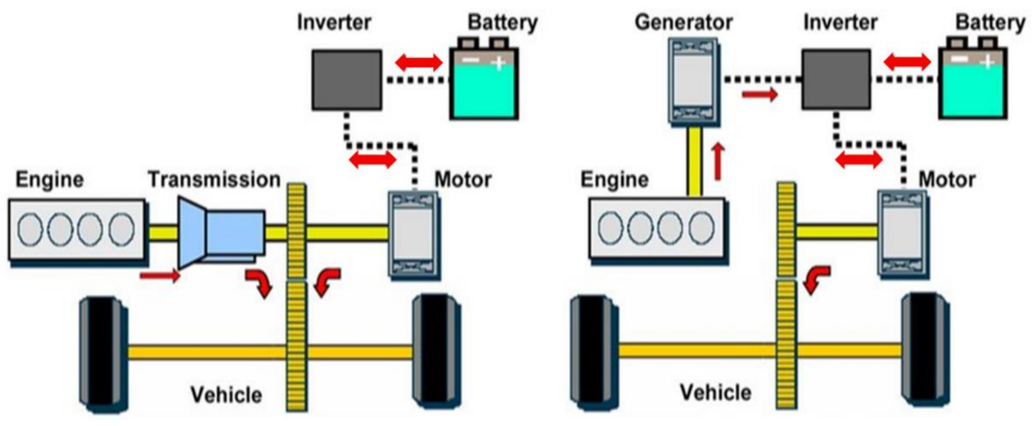


Figure 1: Left Parallel Architecture. Right - Series Architecture [3]

In a parallel architecture, the engine and motor are both mechanically coupled to the wheels. There are five different parallel architecture configurations: P0, P1, P2, P3, and

P4. Each configuration has the electric motor integrated in a different location. There are also power-split parallel architectures which combine aspects of both parallel and series architectures. For this work, focus will be placed purely on parallel architectures. In a P0, the electric motor is located on the front of the engine. The motor is usually small and is typically only used for start/stop functionality. There is usually not a HV battery in a P0. The P0 motor uses a 48V battery that is also used for the normal vehicle accessories. In P1 through P3 architectures, the electric motor is located between the engine and transmission clutch; between the clutch and the transmission; and after the transmission, respectively. P0 through P3 configurations all contribute to drive the same axle. In a P4 architecture, the electric motor is usually used to drive the rear axle while the engine is used to drive the front axle. This type of architecture is also known as “electric all-wheel drive.” However, there are no restrictions that the motor and engine be confined to the rear and front axles. A P4 architecture could potentially have a motor driving the front axle and an engine driving the rear axle.

A parallel architecture can also benefit from regenerative braking. By commanding negative motor torque, the power flow is reversed, and the battery is charged while decelerating the vehicle. Alternatively, the battery can also be charged using opportunity charging.

Opportunity charging occurs when the engine produces more wheel torque than the driver requests, but the motor commands a negative amount of wheel torque equal to the amount of excess that the engine is producing. In this way, the driver demanded torque is still satisfied, and the HV battery is charged. Opportunity charging can be used to move the engine into a more efficient operating point, while also charging the HV battery.

1.3. EcoCAR Mobility Challenge

The EcoCAR Mobility Challenge is the present-day challenge in a series of Advanced Vehicle Technical Competitions (AVTCs). AVTCs are sponsored primarily by Argonne National Laboratories, MathWorks Inc, and General Motors. These AVTC's started in 1988, with the Methanol Marathon [4]. The competitions are comprised of multiple universities across North America. In the EcoCAR Mobility Challenge, 11 universities are competing. All competing universities have been provided with new, 2019 Chevrolet Blazers and have been tasked with redesigning and turning this conventional vehicle into an HEV. The main goals of the competition are to improve fuel economy and reduce emissions, all while maintaining a safe and fully functioning vehicle. The competition takes place over four years. At the end of each year, a final competition is held. The final competition gives teams the opportunity to present their work over the past year and show their vehicle off in dynamic testing events.

There are four primary sub-teams within each university's EcoCAR team: mechanical, electrical, propulsion controls and modeling (PCM), and connected and automated vehicles (CAVs). Each sub-team is responsible for a particular area of integration. The mechanical team is responsible for mechanical integration – i.e. engine/transmission/drive-train installation. The electrical team is responsible for low voltage and high voltage wiring. The PCM team is responsible for vehicle architecture modeling and vehicle controls. Modeling results are used to drive vehicle architecture/component selection decisions. The PCM team is also responsible for designing, programming, and implementing energy management strategies, diagnostic systems, and fault mitigation strategies. The PCM team flashes these systems and strategies onto a controller which interfaces with the stock controllers in the Blazer. The CAVs team is responsible for designing driver alert systems, adaptive cruise control algorithms, and vehicle-to-vehicle and vehicle-to-infrastructure communication strategies.

Over the course of four years, all the sub-teams work together to produce a functioning, road-legal, fully integrated hybrid electric vehicle. The competition facilitates a team-oriented and results-driven environment. Skills learned in the EcoCAR competition readily translate to industry-level jobs.

1.4. Team-Selected Architecture

The architecture selection process for the WVU team was driven primarily by fuel economy modeling results from the PCM team and integration considerations from the propulsion system integration (PSI) team. Different vehicle architectures have their unique benefits and drawbacks. Some vehicles even combine architectures, i.e. P0/P4 and P0/P3 architectures are common. The PCM team modeled, analyzed, and compared multiple vehicle architectures. For each architecture that was examined, a specific control strategy was optimized. This architecture-specific optimization gave the WVU team a true understanding of the potential contained in each architecture. Based on considerations of vehicle control complexity and mechanical integration, the WVU EcoCAR team ultimately selected a P4 architecture.

The HV battery selected is a General Motors HEV4 battery pack. This battery pack has a total energy storage capacity of 1.5 kWh with a nominal voltage of 300 V and a peak power of 50 kW. The engine selected is a 4-cylinder GM 2.5L LCV engine rated for a maximum torque of 255 Nm and a maximum power of 148 kW. The transmission is a GM 9-speed

M3D (9T50) with an accumulator. The electric motor selected is an electric axle with an integrated electric motor and differential gearbox. This system is known as an eAWD and is manufactured by Magna Powertrain. This eAWD is used in the Volvo V60 hybrid in Europe. The eAWD has a peak power of 50 kW, and a peak torque of 200 Nm, maximum continuous power of 20 kW, maximum continuous torque of 90 Nm, and a maximum motor speed of 12000 rpm. The gear ratio of the integrated differential gearbox is 9.17. The system comes with an inverter specifically intended for use with the eAWD.

The P4 architecture has three primary modes of power flow: FWD with opportunity charging, FWD with regenerative braking, and AWD. These three modes are shown in Figure 2, with arrows indicating the direction of power flow for each mode. In FWD with opportunity charging, the engine supplies excess torque to the front axle, while the electric motor “drags” the rear axle by producing negative torque on the rear axle equal to the amount of excess that the front axle is supplying. The negative torque reverses the power flow direction of the electric propulsion system and charges the HV battery. In FWD with regenerative braking, the engine is supplying the front axle with power, while the motor is producing negative torque to brake the vehicle. The application of negative torque by the motor captures free energy from the vehicle’s inertia. A greater amount of energy is able to be recaptured from regenerative braking if the deceleration happens slowly.

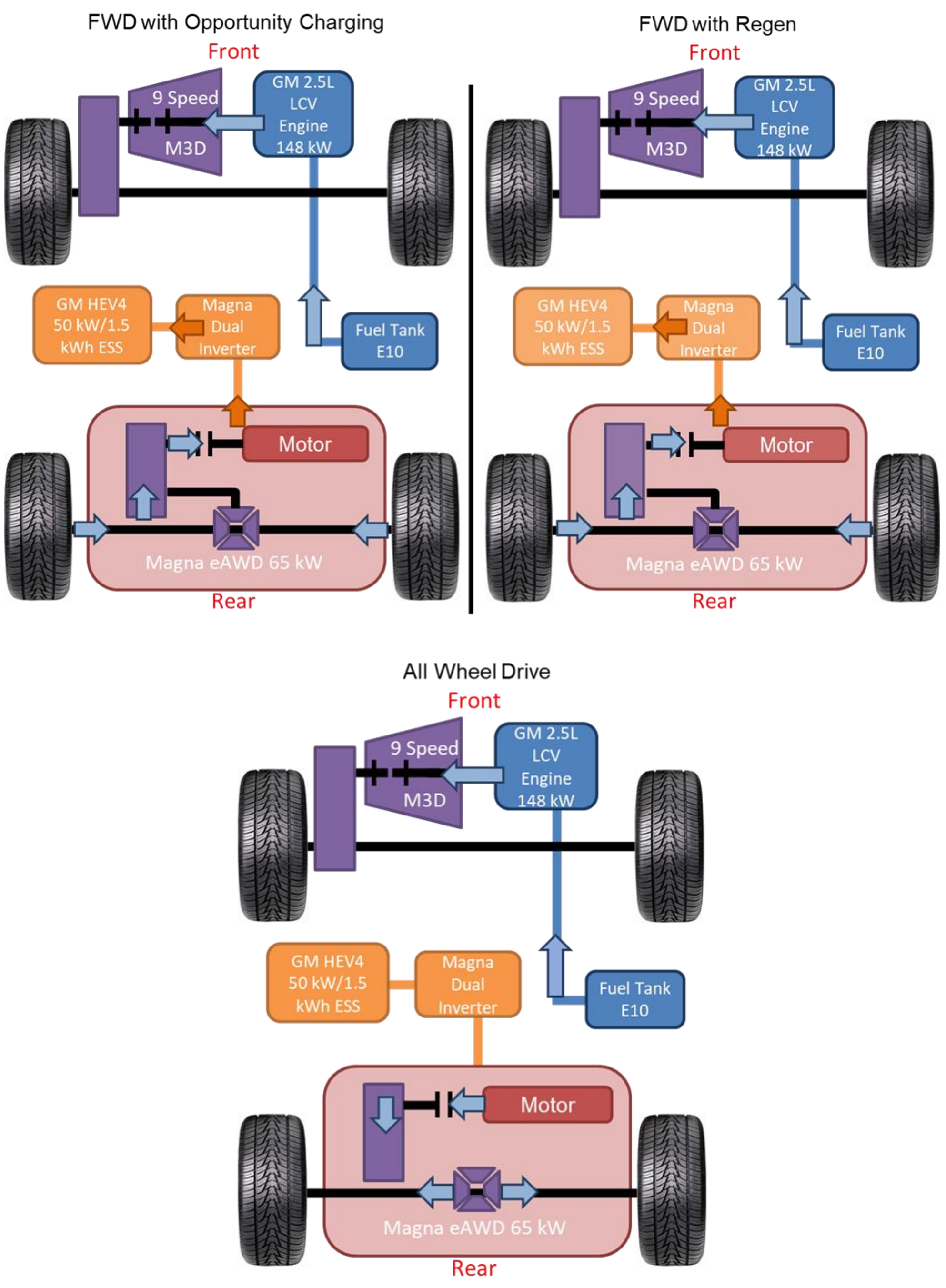


Figure 2: P4 Architecture Power Flow Methods

In AWD, both the engine and the motor are supplying power to their respective axles.

The available power of the engine far outweighs the power available from the electric motor and battery. The engine can provide 148 kW, while the motor and HV battery are matched at 50 kW for maximum power – merely a third of the engine power. This mismatch in power, in addition to the small energy capacity of the HV battery (1.5 kWh), led to the decision to not have an electric-only operating mode. An electric-only mode is also known as charge depleting (CD) mode. A hybrid vehicle with a larger electrical energy capacity may operate in CD mode until a state of charge (SOC) threshold is reached, at which point the vehicle would enter charge sustaining (CS) mode. In CS mode, the vehicle maintains the SOC around a setpoint without significant variation from the setpoint. The benefit of having CD and CS modes is that the vehicle is able to function as a fully electric vehicle (in CD mode) and a hybrid vehicle (in CS mode).

With the electric powertrain component sizing of the team designed P4 architecture, a CD mode would not make sense. The motor would only be able to supply a limited amount of power. Plus, it would not be able to supply this independent power for any meaningful length of time.

Based on the power comparison, it makes more sense to use the electric motor to augment the operation of the engine. For instance, to push the engine into a more efficient operating region of lower brake specific fuel consumption (BSFC). This type of operation is equivalent to operating exclusively in CS mode. When considering an engine, BSFC is essentially a measure of efficiency given a fuel flow rate, the efficiency can be calculated. The BSFC is a function of engine speed and torque. The engine speed is determined by the speed of the vehicle, and the gear ratios going from the wheel speed to the output shaft of the engine. The total gear ratio is defined by the transmission and differential gear ratios. The engine torque is directly affected by the accelerator pedal position. An accelerator pedal map is used to map accelerator pedal positions to a wheel torque. The challenge is determining the most efficient torque split between the engine and motor. A control strategy cannot simply command the most efficient split. The most efficient torque split would be to simply command all the torque from the motor – because of its inherently greater efficiency. However, this would result in quickly draining the HV battery – especially if the HV battery has a small energy storage capacity and cannot support an electric only mode. Such is the case with the vehicle architecture in this work.

In the overall consideration of the powertrain system, the greatest losses will come from the engine. Therefore, a better option is to use the motor to push the engine into its regions of highest efficiency.

Consider the engine map in Figure 3. The engine speed (x-axis) is directly related to the vehicle speed, tire size, and gear ratios of the transmission and differential. The engine torque (y-axis) is requested by the driver through the accelerator pedal. If the engine is operating at point “A”, it is in a relatively high region of BSFC. However, if it could be moved to point “A′”, it would be in a more efficient region of lower BSFC.

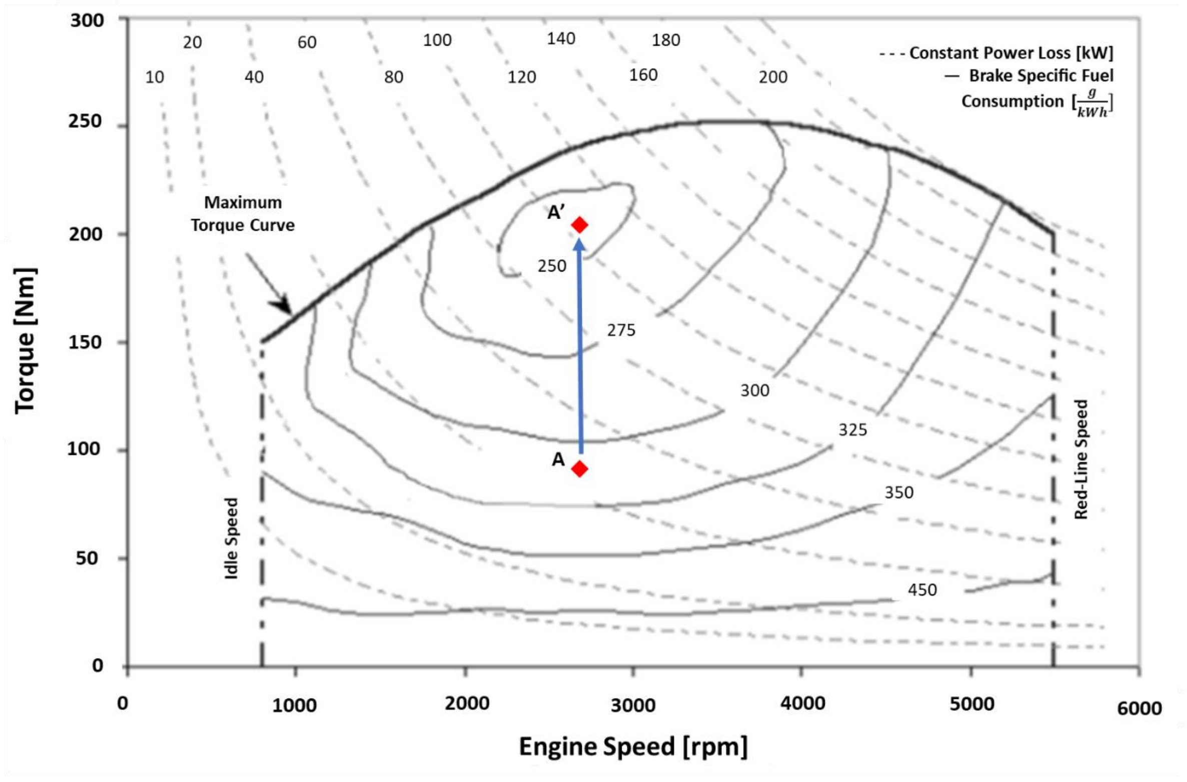


Figure 3: Engine Map Showing Increased Efficiency with Increase in Torque Production

How can the operation point of the engine be moved from A to A′? First, it is important to understand the two different realms of torque: component and wheel. Component torque is the torque the component is actually producing. Wheel torque is the torque a component is producing at the wheels. Wheel torque is the component torque reflected through the drivetrain gear ratios. If the driver is requesting a certain amount of wheel torque trough the accelerator pedal, this is essentially a request for two wheel torques – one from the engine and one from the motor. These wheel torques translate to two component torques between the engine and motor. To move the engine from A to A′, the engine can produce more torque than requested, at a reduced fuel consumption, and the electric motor can apply negative wheel torque to generate electricity, with the net torque meeting the driver request. The greater production of engine wheel torque translates to a greater production of component torque, thereby moving the engine operation point from A to A’. In doing so, not only is the engine more efficient, but the motor is opportunity charging and increasing the HV battery SOC.

The opposite scenario could also be encountered. The engine could be producing torque B but be more efficient at point B′ (Figure 4).

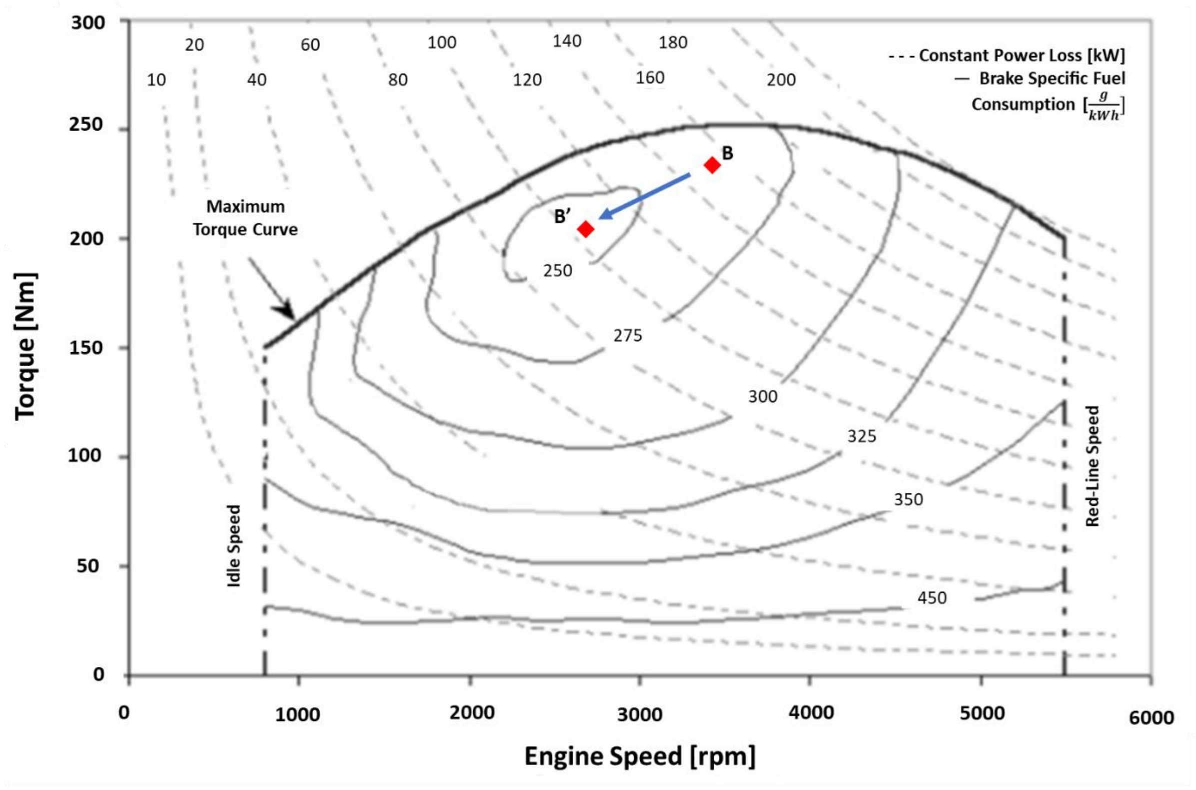


Figure 4: Engine Map Showing Increased Efficiency with Decrease in Torque Production

The engine could produce less wheel torque and the motor could assist by making up the difference in the driver requested wheel torque. However, notice that this involves a translation in the engine speed as well as the torque. The speed of the engine is directly a function of the vehicle speed being reflected through the tire size and differential and transmission gear ratios. To translate horizontally along the speed profile of the engine map, a transmission gear can be commanded that would facilitate the change in speed. To maintain a constant, “cruise”, speed, the required torque is only a function of the vehicle aerodynamics, road friction resistances, and powertrain efficiency losses.

The motor efficiency is inherently much higher than the engine efficiency. The motor efficiency is also a function of speed and torque. However, unlike the engine, the motor operates at a constant gear ratio. When the motor is producing propulsive torque, it is said to be “motoring.” When it is producing negative, charging torque, it is said to be “generating.” In Figure 5, “generating” torque is in the lower half of the map, and “motoring” torque is in the upper half of the map. The motor map shows regions of “pure” efficiency and not a related parameter like the engine and BSFC.

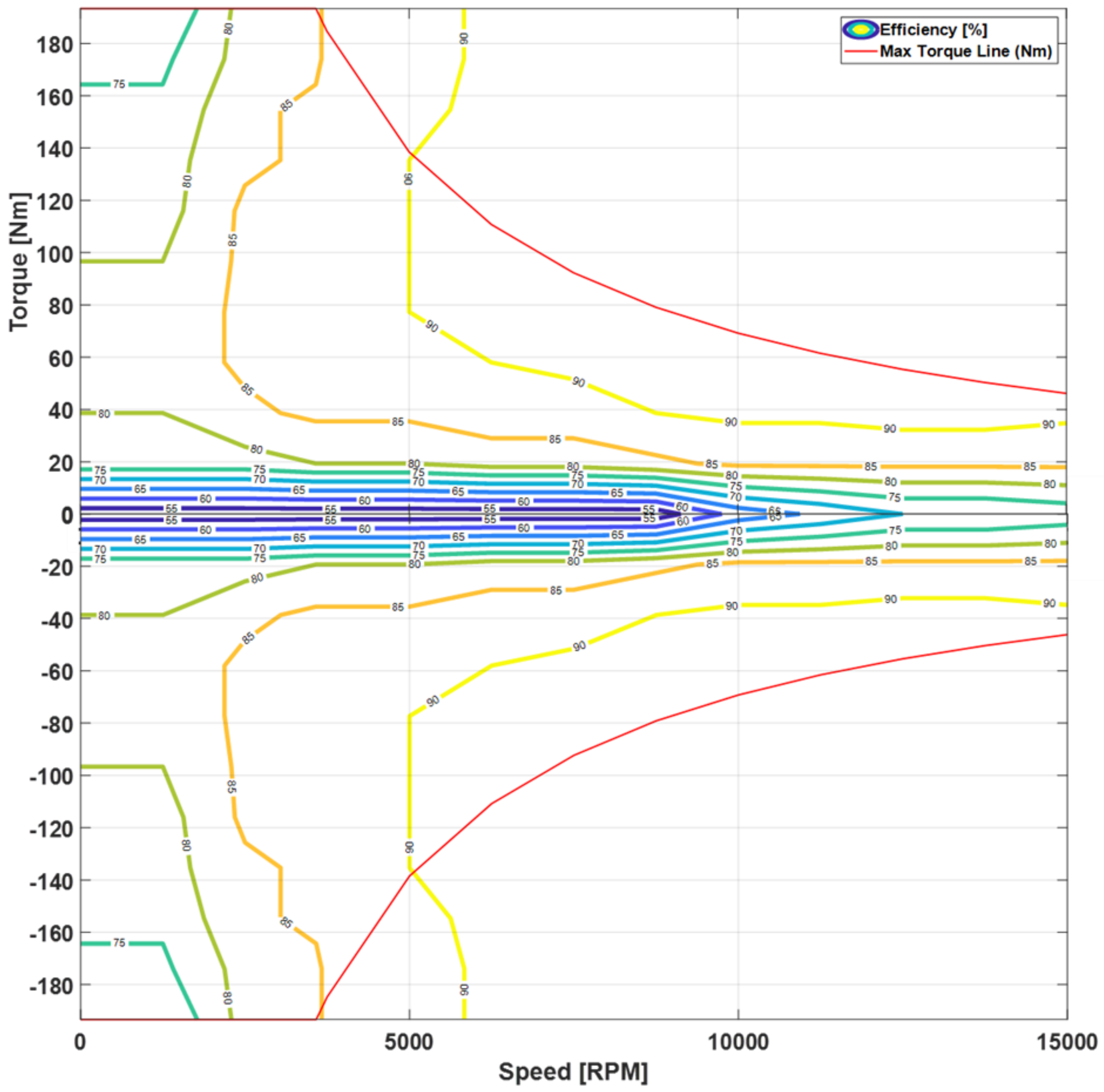


Figure 5: Efficiency Regions of Motor Map

The generating efficiencies are generally a close reflection of the motoring efficiencies. However, this may not always be the case. In order to fully characterize the losses of the entire system, battery and inverter losses would also need to be considered. The battery is subject to losses due to internal resistances. The inverter is also not 100% efficient in its operation.

Understanding the engine and motor operating points in conjunction with their respective efficiencies provides a true understanding of the overall effectiveness of a hybrid electric vehicle control strategy. An optimal control will yield operating points of the engine and motor that reflect the most efficient torque split while still meeting the driver-requested wheel torque.

The goal of this work is to implement an optimal control strategy with an ANN. The optimal control strategy the work focuses on is known as the equivalent consumption minimization strategy (ECMS). The ECMS is only truly optimal if the future driving conditions are known. Since this is not feasible in the real world, the ANN is used to analyze past driving conditions and optimize the control parameter of ECMS for those past driving conditions. The ECMS control parameter is known as the equivalence factor. The ANN examines past driving conditions over a sliding 3-minute time window and optimizes the equivalence factor for those driving conditions. The assumption is that future driving conditions will not substantially change over 3 minutes. Even if the conditions do change, there will only be, at most, a 3-minute time window over which the equivalence factor is not optimal.

The following section outlines recent work concerning control strategies and ANNs.

2. Literature Review

The following section outlines recent work in the development of control strategies which produce optimal torque splits to maximize fuel economy. Additionally, instances of ANN augmented control strategies in general are also examined.

2.1. Overview of HEV Control Strategies

The design of an optimal control strategy for an HEV is a complex problem. The goal is to create a strategy which optimally splits the driver commanded torque between the ICE and motor in a way which maximizes fuel economy and minimizes emission production. The work described in this paper focuses on the design of a control strategy which solely endeavors to maximize fuel economy.

An optimal control strategy cannot simply command the most efficient torque split between the ICE and motor. If it did, all the driver's torque command would be produced by the motor because of its inherently greater efficiency. This would result in the HV battery being quickly depleted. A strategy is needed which allows the HV battery SOC to maintain self-sustainability so that the motor may continually be able to assist the ICE operation.

Many different control strategies exist for hybrid-electric vehicles. Broadly speaking, control strategies may be categorized as either rules based or optimal based strategies. Rules based strategies are effective for real-time implementation. A rules-based strategy determines the control based on heuristics, intuition, or an optimally discovered solution which is determined offline [1].

In an optimal strategy, an appropriate cost function is created, which is minimized at each time step. Ideally, the dynamically changing cost function is supposed to be equivalent to the globally optimal cost function. However, the true global cost function is only known if the drive cycle is known *a priori*. If the cost function is appropriately defined, its minimization results in the minimization of the global cost function. Strategies employing appropriate cost functions have been shown to closely approximate the global optimal solution [1].

A drive cycle is simply a trace of vehicle speed vs. time. An example is shown below:

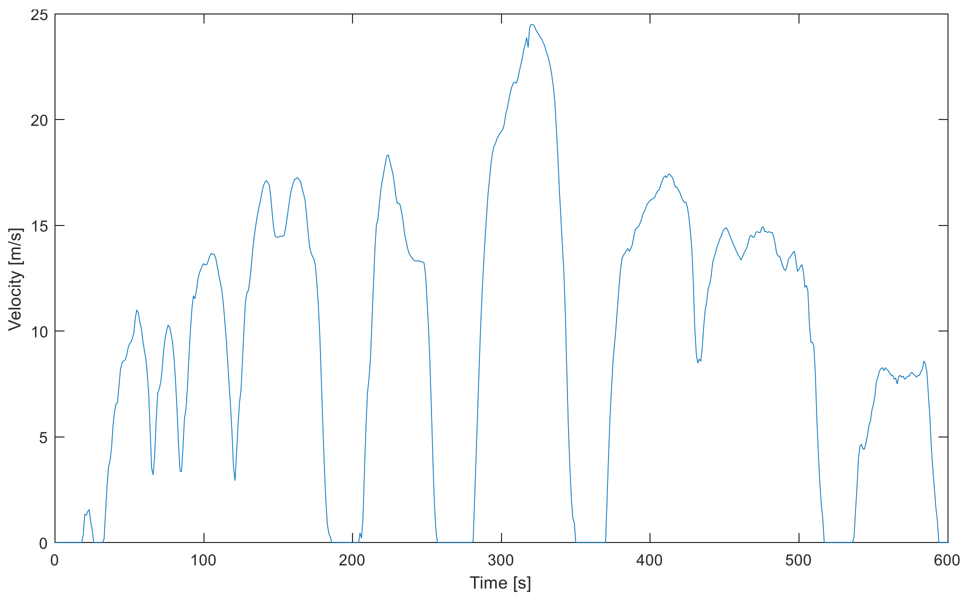


Figure 6: Example Drive Cycle

At a given time, the expected speed of the vehicle is specified. Different drive cycles are created to test different characteristics of a vehicle. For example, a drive cycle can be created that is used primarily to evaluate a vehicles fuel economy. Drive cycles can be created to test vehicle performance under different conditions such as city and/or highway driving.

A drive cycle will require a certain power and torque demand from the vehicle. Knowing the power and torque capabilities of a vehicle, it can be determined whether or not a drive cycle can be completed. This can be determined using the relationship between speed and power:

$$P = \omega T \tag{2-1}$$

Where P is power, ω is angular velocity, and T is torque. It should be noted that if this equation is used when evaluating a drive cycle, it must be applied at the wheels, i.e. power at the wheel, torque at the wheels, and angular velocity of the wheels. If the vehicle specifications are given in terms of the powertrain components, then the power, torque, and angular velocity must be reflected through the associated powertrain gear ratios and tire radius to arrive at the wheels.

2.2. Dynamic Programming

Dynamic programming (DP) is a numeric method of solving the optimal energy management problem over the course of an entire drive cycle. However, before DP can be implemented, an equation which represents the energy management problem must be formulated.

To truly optimize an energy management strategy over the course of a drive cycle, the entire drive cycle must be completely known beforehand. Once the drive cycle is known, an equation can be formulated over which the drive cycle can be optimized. G. Rizzoni has presented a global optimal energy management formula as follows [1]:

$$J = \int_{t_0}^{t_f} \dot{m}_{f,eqv}\big(u(t)\big), t)dt \tag{2-2}$$

Where $\dot{m}_{f,eqv}$ is the equivalent fuel consumption. The goal is to find the control u(t) which minimizes this non-linear cost function (J) over the course of a drive cycle, from the initial time (t_0) to the final time (t_f). The above equation can be unique between vehicles. For example, additional costs based on SOC constraints, vehicle speed constraints, component power limitations, emissions, etc. may be added to the overall cost function. For hybrid electric vehicles, a typical constraint is that the starting and ending SOC be within a certain threshold of each other. Additionally, it is usually necessary to impose constraints that would prevent the vehicle speed from deviating too far from the drive cycle speed trace that is being followed.

A numerical method of solving this problem is dynamic programming. DP provides the optimal solution to (2-2). However, DP is limited to the simulation environment because of the need for *a priori* knowledge of the drive cycle. In fact, DP requires that the solution be calculated starting at the end of the cycle and be worked backwards to the beginning. The method of DP is also rather computationally intensive. For these reasons, DP is not practical to implement in a real-time controller during normal driving [1]. The main theory behind DP is that the continuous cost function be discretized over time. Then, for each discretized point, the cost be minimized – while obeying any imposed constraints.

Many different research groups have used fuel economy results obtained from DP as a baseline against which they can compare their own control strategies [5], [6], [7]. The concept of DP is the same across the different research groups. DP represents the best possible solution. The actual formulation of DP varies across research work, because the overall cost functions vary in terms of system constrains and overall system performance goals. In work performed by H. Peng et. al [8], an overall cost function was defined for a parallel hybrid electric truck. The overall cost function (J) to be minimized was given as:

(2-3)

$$J = \sum_{k=0}^{N-1} L\big(x(k), u(k)\big) = \sum_{k=0}^{N-1} fuel(k) + \mu * NO_x(k) + v * PM(k)$$

Where N is the driving cycle duration, and L is the instantaneous cost including engine NOx and PM (particulate matter) emissions and fuel use. The goal is to find the control actions $u(k)$ which minimize J. In the formulation of this equation, the factors μ and v are weights which define how much cost emissions are to be given in the overall cost function. If fuel was the only consideration, both μ and v would be zero. For a problem in which both fuel and emissions were being considered, μ and v would be > 0 [8].

Before the optimization was performed, there were 8 constraints imposed:

$$\omega_{e\_min} \leq \omega_e(k) \leq \omega_{e\_max} \tag{2-4}$$

$$T_{e\_min}(\omega_e(k)) \leq T_e(k) \leq T_{e\_max}(\omega_e(k)) \tag{2-5}$$

$$T_{m\_min}(\omega_m(k), SOC(k)) \leq T_m(k) \leq T_{m\_max}(\omega_e(k), SOC(k)) \tag{2-6}$$

$$SOC_{min} \leq SOC(k) \leq SOC_{max} \tag{2-7}$$

Where ω_e is engine speed, T_e is engine torque, T_m is motor torque, and SOC is the battery state of charge.

An additional constraint was also imposed on the SOC so that not only would the SOC remain between the maximum and minimum bounds, but that the ending SOC be equal to a defined value. The final cost function is given by:

$$J = \sum_{k=0}^{N-1} [fuel(k) + \mu * NO_x(k) + v * PM(k)] + \alpha(SOC(N) - SOC_f)^2 \qquad (2\text{-}8)$$

Where SOC_f is the ending SOC and α is a weighing factor.

The final equation above is an example of how a global optimization problem can be defined for a specific vehicle architecture and specific system goals.

The next step is to solve the cost function using DP. As with all DP methods, this equation was solved from time N to time 0. A standard DP method was used in which the control and state values are discretized into a finite grid. For each step in the optimization search $J(k)$, the cost function is evaluated at the state variable grid points [8].

A simplified example by Rizzoni et. al, shows how a grid can be used and how costs are associated with moving from one point to another within the grid (Figure 7):

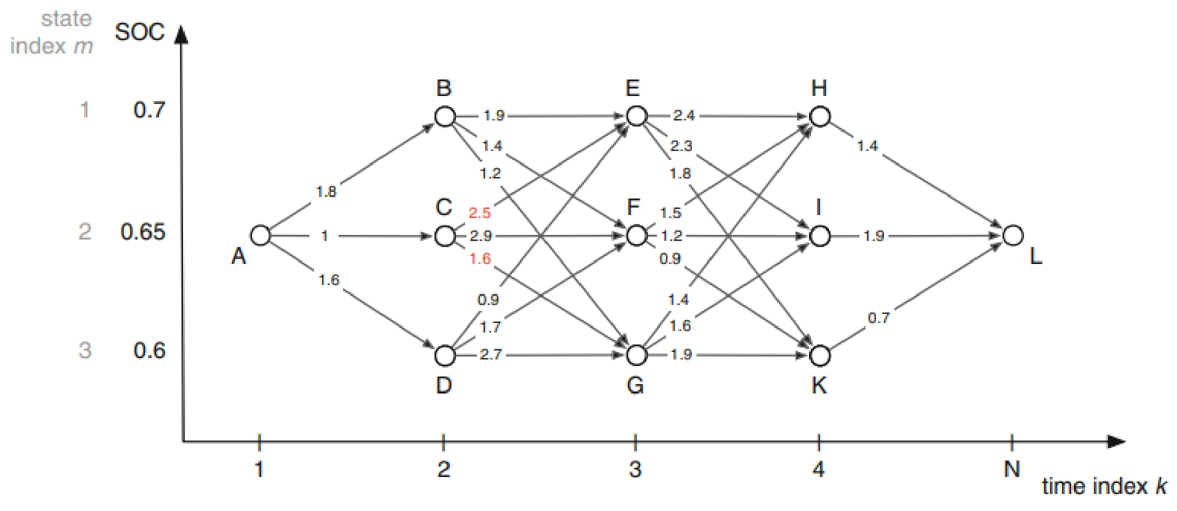


Figure 7: Simplified DP example of costs associated with moving from one point to another on grid of operating points [1]

As shown in Figure 7, costs are associated with moving from one operating point to the next. The goal is to select the path through the grid which minimizes the overall cost.

2.3. Equivalent Consumption Minimization Strategy (ECMS)

Other methods exist in which the optimal results obtained from DP can be closely approximated. The method of the ECMS is used to convert the global optimal solution (Equation (2-2)) into a series of instantaneous minimization problems. As such, ECMS is less computationally intensive. The results from ECMS have been shown to closely approximate the global optimal solution [1].

The ECMS is formulated based on the premise that, for a charge-sustaining hybrid, the starting and ending SOC is approximately the same. If this is the case, then the battery is essentially an energy buffer: if electrical energy is used, it will eventually need to be replenished, either by opportunity charging or regenerative braking. Generally speaking, the ECMS operates by equating fuel energy consumption with electrical energy consumption using an equivalence factor. The ECMS algorithm then selects control outputs between the engine and motor(s) that minimize the equivalent fuel consumption. The control outputs are engine and motor torque commands. The challenge in designing and calibrating an ECMS is choosing an appropriate equivalence factor to equate the electrical consumption to the fuel consumption.

The equivalent fuel consumption is based on the following equation:

$$\dot{m}_{f,eqv}(t) = \dot{m}_f(t) + \dot{m}_{elec}(t) \tag{2-9}$$

Where $\dot{m}_{f,eqv}(t)$ is the instantaneous equivalent fuel consumption, $\dot{m}_f(t)$ is the instantaneous fuel consumption of the engine, and $\dot{m}_{elec}(t)$ is the equivalent instantaneous electrical consumption.

The instantaneous fuel consumption of the engine is defined as follows:

$$\dot{m}_f(t) = \frac{P_{eng}(t)}{\eta_{eng}(t) * Q_{lhv}} \tag{2-10}$$

Where $P_{eng}(t)$ is the instantaneous power of the engine, $\eta_{eng}(t)$ is the instantaneous efficiency of the engine, and Q_{lhv} is the lower heating value of the fuel.

The instantaneous electrical consumption is equivalent to the fuel consumption in that it has the same units. It is, however, scaled by an equivalence factor. The instantaneous electrical consumption is given by the following equation:

$$\dot{m}_{elec}(t) = \frac{s(t)}{Q_{lhv}} * P_{batt}(t) \tag{2-11}$$

Where $s(t)$ is the equivalence factor and $P_{batt}(t)$ is the instantaneous battery power. The equivalence factor $s(t)$ can be thought of as a cost that is applied to the electrical power that equates it to a fuel power [1]. The convention is negative battery power propels the vehicle (motoring) and positive battery power charges the battery (generating). Positive power increases the equivalent fuel consumption and negative power decreases the equivalent fuel consumption.

To prevent the battery SOC from being depleted, a penalty factor is assigned to $\dot{m}_{elec}(t)$, based on the instantaneous SOC. The penalty factor makes electrical energy cheap if the SOC is near the maximum SOC of the battery and makes electrical energy expensive if the SOC is near the minimum SOC. This bounds the SOC and keeps it around the target SOC. The target SOC is specified based on the efficiencies of the HV battery. The penalty (p) is in the form of a sigmoid (Figure 8), and has the following equation:

$$p(SOC) = 1 - \left(\frac{SOC(t) - SOC_{target}}{0.5 * (SOC_{max} - SOC_{min})}\right)^{a} \qquad (2\text{-}12)$$

Where $SOC(t)$ is the instantaneous SOC, SOC_{target} is the target SOC, SOC_{max} is the maximum allowed SOC, SOC_{min} is the minimum allowed SOC, and a is the penalty factor which affects the curvature of the sigmoid. In Figure 8, SOC_{target} = 60%, SOC_{max} = 80%, and SOC_{min} = 40%.

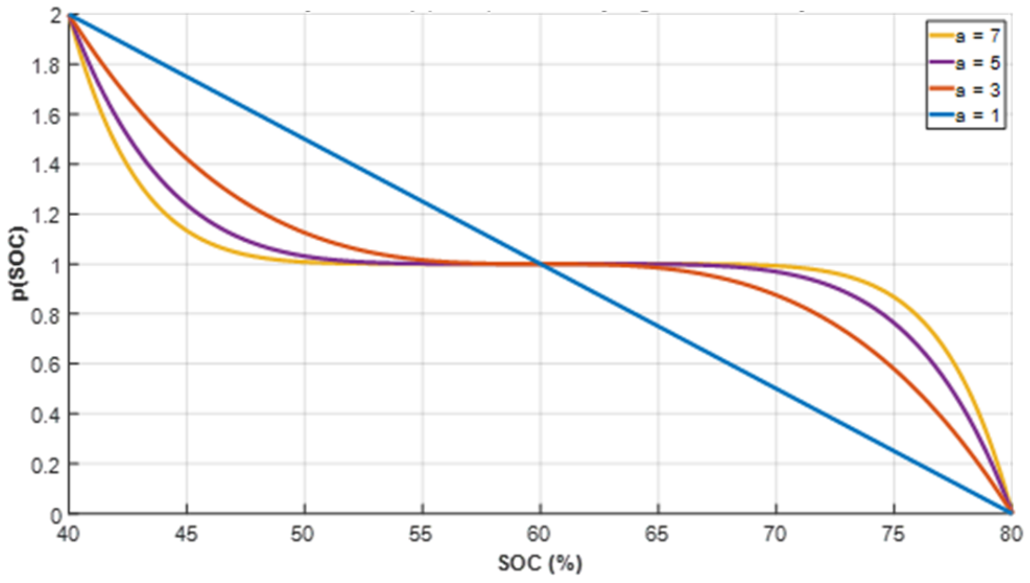


Figure 8: SOC Penalty Function with Varying SOC Penalty Factors

The SOC penalty factor (a) affects the range of SOC that is used. If a = 7, then there is little to no cost change until the SOC approaches the min and max bounds. If a = 1, then the electrical energy cost changes even if there is a slight deviation from the target.

Based on the previous equations, the ECMS can be written as follows:

$$\dot{m}_{f,eqv}(t) = \frac{P_{eng}(t)}{\eta_{eng}(t) * Q_{lhv}} + \frac{s(t)}{Q_{lhv}} * P_{batt}(t) * p(SOC) \qquad (2\text{-}13)$$

This equation can be multiplied by Q_{lhv} to arrive at an equation of equivalent power:

$$P_{f,eqv}(t) = \frac{P_f(t)}{\eta_{eng}(t)} + s(t) * P_{batt}(t) * p(SOC) \qquad (2\text{-}14)$$

Where $P_{f,eqv}(t)$ is the instantaneous equivalent power, $P_f(t)$ is the instantaneous fuel power, and $P_{batt}(t)$ is the instantaneous power of the battery.

To avoid large torque command oscillations between consecutive time steps, an additional term is added to Equation (2-14). This additional term takes the absolute value of the difference between the last engine power and the current engine power. Added to Equation (2-14), this term acts as a cost, making it more expensive to select an engine power that differs greatly from the last commanded engine power. Equation 2-15 shows this term:

$$P_{eng,rate\ limit} = \left|P_{eng}^{t} - P_{eng}^{t-1}\right| \qquad 2\text{-}15$$

Where $P_{eng,rate\ limit}$ is the cost which limits the rate at which the engine can switch between power levels, P_{eng}^{t} is the power of the engine at the current time, and P_{eng}^{t-1} is the power of the engine at the last time step. In conclusion, the final equation is given as:

$$P_{f,eqv}(t) = \frac{P_f(t)}{\eta_{eng}(t)} + s(t) * P_{batt}(t) * p(SOC) + P_{eng,rate\ limit} \qquad 2\text{-}16$$

Implemented into the vehicle controller, $P_{f,eqv}(t)$ is a vector of costs, with each index representing operating conditions of the components. The minimum value of $P_{f,eqv}(t)$ is selected, and the associated torques of the engine and motor are commanded.

Despite ECMS being computationally practical and providing results close to the global optimal solution, there is still a problem. Like DP, the ECMS method needs to have *a priori* knowledge of the drive cycle in order to produce results close to the global optimal solution.

2.4. Adaptive-ECMS (A-ECMS)

Research has been ongoing to create and adaptive-ECMS (A-ECMS) algorithm which can adapt to provide results close to the global optimum without having *a priori* knowledge of the drive cycle. There are three main methods which have been examined in reference to the A-ECMS: drive cycle prediction, driving pattern recognition, and SOC feedback [1]. Each of these methods implement a dynamically varying equivalence factor.

The work described in later sections of this thesis is focused on creating an adaptive implementation of ECMS. The adaptive part is added using an RBF ANN. As driving conditions change, the RBF ANN updates the equivalence factor based on the data with which it was trained.

2.4.1. Drive Cycle Prediction

The drive cycle prediction method uses current driving conditions to try and estimate what the future driving conditions will be. Based on the estimations, the equivalence factor is updated accordingly. The results from this method are inferior to an ECMS method tuned over an *a priori* drive cycle, nevertheless, the results are still good [1], [9], [10], [11].

Work has been done by Z. Chen et. al [10], in which a convolutional neural network was developed to make drive cycle predictions for a plug-in hybrid electric bus. The bus route was classified into 6 unique drive cycles using a k-Shape clustering algorithm. The clustered data was used to train the CNN. A comparison of fuel economy was made between a baseline ECMS and the implementation of the CNN with ECMS. The implementation of the CNN with ECMS is referred to as A-ECMS. The A-ECMS showed an improvement of 14.86% over the ordinary ECMS [10]. It should be noted that the CNN was trained for a particular bus route. It would not be expected to see such significant improvements over all driving conditions.

In the work of Chasse et al [11] an A-ECMS is applied to a parallel HEV. The work is based on the idea that the electrical energy usage at the end of a drive cycle can be converted into an equivalent fuel energy using a discharge equivalency factor s_{dis} or a charging equivalency factor s_{chg}. During a real-time drive cycle, the use of s_{dis} or s_{chg} depends on the final sign of the total energy usage. However, for a real-time application, it is not known if the ending electrical energy will be positive or negative. Because of this, the equivalency factor cannot be set with certainty. To compensate, the equivalency factor is evaluated by introducing a probability factor which estimates whether the equivalency factor will be s_{dis} or s_{chg} during the course of the drive cycle [11].

The probability factor is such that unity indicates s_{dis} is necessary and a value of zero indicates a s_{chg} should be used. The value of the probability factor is defined based on the "energy horizon", which is the required energy at the wheels [11]. The probability factor is assigned a value, for each time step, based on an estimate of whether the end of the drive cycle will result in positive or negative net electrical usage. The estimates are obtained using a predefined function and knowledge of what the maximum positive and negative values of the net electrical energy usage can be at the end of the drive cycle [11].

In work performed by Sorrentino et al [12], a DP technique is performed on-line by using an ANN to predict what the upcoming vehicle load will be. DP is performed over a known time horizon, in which the upcoming control parameters can be set to minimize the fuel economy over the known road load. In this work, the future load is predicted as a function of current and past load states. The load prediction is estimated using a recurrent neural network [12].

The results from this work show that the load prediction implementation with a hybrid electric vehicle greatly outperforms a conventional vehicle while also coming close to the reference fuel economy obtained with DP (Table 1).

Table 1: Fuel Economy Results of Predictive Road Load A-ECMS

Fuel Economy [km/l]		
Hybrid		Conventional
Predictive Model	Reference (DP)	--
16.1	16.8	11.1

Another method of drive cycle prediction was investigated by Jeon et al [13]. In this work, an A-ECMS algorithm was made for a multi-mode parallel HEV. Optimal control values were found for 6 representative driving patterns (RDP). The optimal control values were determined off-line. The optimal values were obtained using the Taguchi method with orthogonal arrays. This method was chosen because of its low computational cost and ease of analysis. The optimal values for each RDP were stored in the vehicle controller's memory. An ANN was then used to recognize one of the six RDPs during real-time driving.

A total of 24 inputs were fed to the ANN for the purpose of identifying the RDP. These inputs included average acceleration, average deceleration, average running velocity, maximum velocity, maximum grade, minimum grade etc. Inputs were sampled every 1 second, and the control algorithm was updated every 300 seconds. The ANN used was the Hamming network. The results of this research showed improvements over both a HEV single-mode control (HEV_SM) algorithm without driving pattern recognition and an internal combustion vehicle (ICV) [13]. The results are shown in Table 2.

Table 2: Fuel Economy Results of Drive Cycle Predictive A-ECMS

	Vehicle Control Configuration	**Fuel Economy (km/liter)**
Test Pattern 1 1 Cycle	HEV_MM	11.590
	HEV_SM	10.656
	ICV	9.208
FTP-75 5 Cycles	HEV_MM	16.604
	HEV_SM	13.808
	ICV	10.414
NEDC 10 Cycles	HEV_MM	14.643
	HEV_SM	13.552
	ICV	10.855

Work performed by Hadi Kazemi et. al of WVU, proposed a predictive ECMS (PECMS) method in which a future time horizon was modelled to redefine the equivalence factor of traditional ECMS. The prediction horizon was a function of three parameters: estimated energy required in the prediction horizon, amount of energy recaptured through regenerative braking, and a charge and discharge cost factor [14], [15].

To investigate the effect of the proposed method on fuel economy and charge sustainability, the method was implemented and simulated in a hybridized Chevrolet Camaro vehicle model. The simulations were performed with 4 different prediction horizon windows. In a comparison of the PECMS and A-ECMS methods, the fuel economy was seen to increase substantially over a particular few drive cycles. The comparison over three drive cycles is shown in Table 3.

Table 3: PECMS and A-ECMS Comparison

Method	MPG	Time Horizon	Improvement Over A-ECMS
UDDS			
A-ECMS	54.77	-	-
PECMS	55.47	60	1.3%
	55.75	30	1.8%
	55.87	15	2.0%
	55.61	5	1.5%
HWFET			
A-ECMS	39.77	-	-
PECMS	40.1	60	0.8%
	40.93	30	2.9%
	41.4	15	4.1%
	40.91	5	2.9%
US06			
A-ECMS	24.99	-	-
PECMS	25.8	60	3.2%
	25.92	30	3.7%
	25.87	15	3.5%
	25.76	5	3.1%

From Table 3, it can be seen that some of the improvements of the PECMS are as high as 3.7%. This is a significant improvement, considering that the A-ECMS is already a highly optimized method.

This work of Hadi Kazemi et. al, differs from the work described in this thesis in that Kazemi used an assumed input of vehicle-to-infrastructure communication. In other words, the time horizon was able to be predicted by assuming that the upcoming driving conditions were externally communicated to the vehicle. In this way, Kazemi was able to examine the effect of the time window length. Conversely, the work described in this thesis updates the ECMS by examining a time window of past driving conditions. The assumption is that the future conditions, over relatively short time windows, will be equivalent to the past conditions. In this work, there is no external communication providing pre-knowledge of upcoming driving conditions.

Work performed by W. Vaz et. al [16], used a neural network to both classify the driving pattern and differentiate between city and highway driving conditions using accelerator and brake pedal positions [16]. The neural network developed was able to successfully distinguish between aggressive and defensive driving styles as well as city and highway driving cycles in 11 different cases.

2.4.2. Driving Pattern Recognition

Work done in driving pattern recognition has also been performed in an effort to improve fuel economy [17], [18], [19]. Of particular interest for this work is research in driving pattern recognition by Bo Gu, in which a total of 21 different metrics were used to characterize a single driving cycle. Eighteen different drive cycles are used to be representative of all real-world driving conditions. Some of these 18 cycles are combined and then clustered and further categorized into 4 classes, over which the mean of the optimal equivalence factor is determined for each class. Table 4 shows the final categorizations [18].

Table 4: Drive Cycle Categorization [18]

Class 1 (Optimal EQF) Extra-urban	**Class 2 (Optimal EQF) Highway**	**Class 3 (Optimal EQF) Sub-urban**	**Class 4 (Optimal EQF) Urban**
EUDC (2.1) **EUDC_LP (2.1)**	HWFET (2.25)	City1 (2.5) FTP (2.55) US06 (2.75) SC03 (2.4) LA92 (2.55) LA92 Short (2.2)	City2 (3) NYCC (3.3) ECE (3) 11-MODE (3.15)
Class 1 Mean EQF	Class 2 Mean EQF	Class 3 Mean EQF	Class 4 Mean EQF
2.1	2.25	2.5	3.11

The A-ECMS algorithm views a sliding time window of past driving conditions. Based on how closely the past operating conditions match one of the 4 classes, one of the 4 equivalence factors is selected. This equivalence factor is then used in the ECMS algorithm. Figure 9 shows a block diagram of the A-ECMS algorithm.

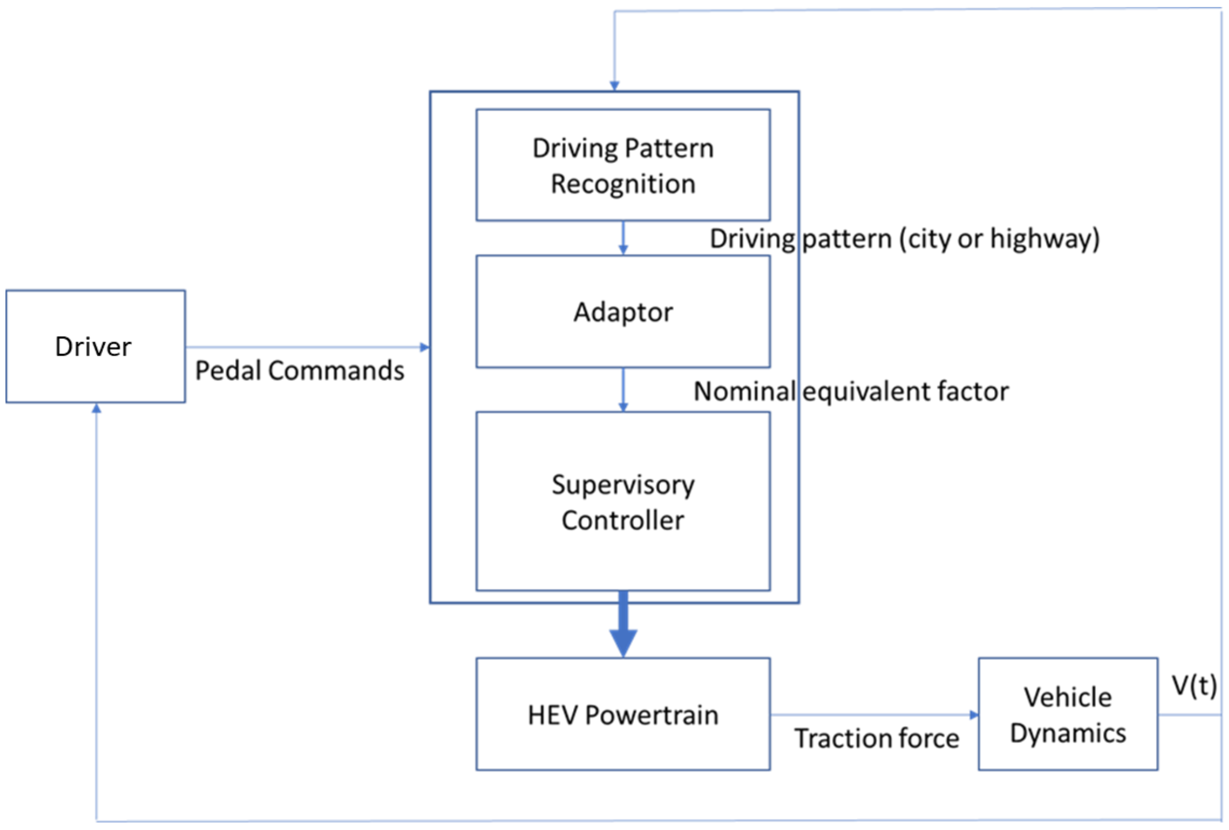


Figure 9: Adaptive ECMS Algorithm Diagram [18]

The assumption is made that a driver will not switch back and forth between city and highway driving over the course of a few minutes. Based on this assumption, a sliding time window of 200 seconds is used to determine the current driving conditions and set the equivalence factor accordingly [18].

The results of the A-ECMS algorithm can be seen in Figure 10. Figure 10 shows equivalent fuel consumption comparisons for each of the 18 drive cycles. The comparison is between the adaptive ECMS, the best equivalence factor, and the worst equivalence factor.

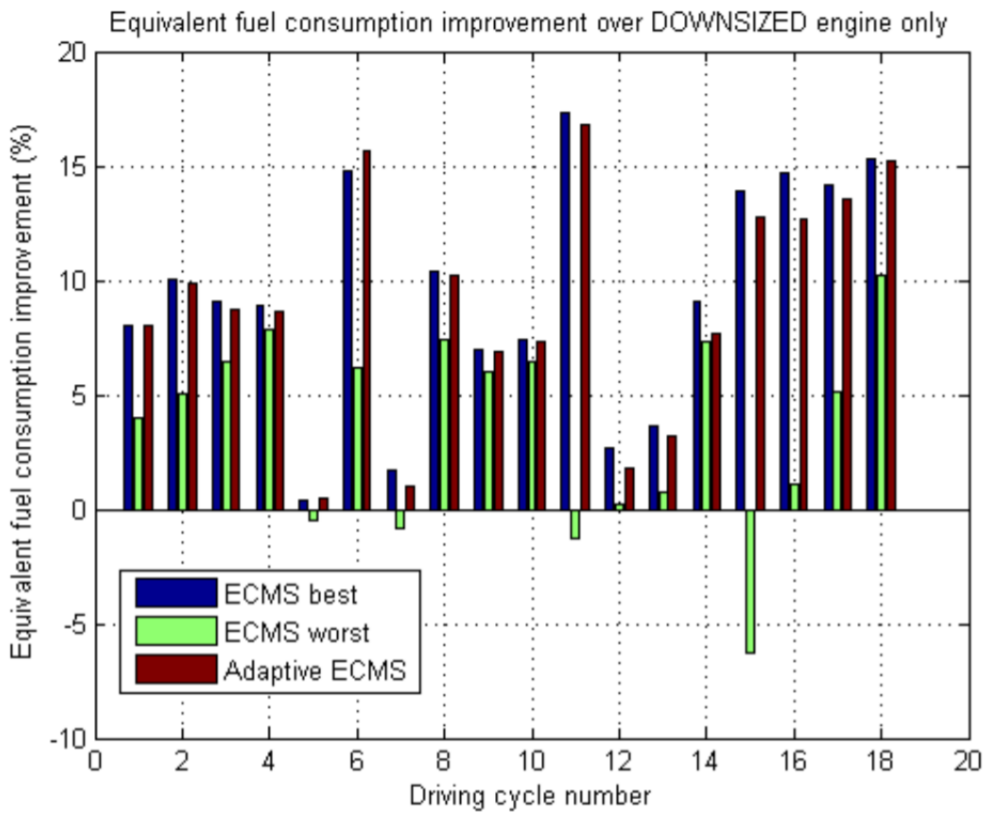


Figure 10: A-ECMS Results [18]

The best and worst equivalence factors are shown to highlight the importance of the selected equivalence factor. If the equivalence factor is chosen without thought, fuel economy results can be worse than the original non-hybridized vehicle. The results show that the adaptive algorithm is very close to the fuel economy obtained using the best equivalence factor. The adaptive method even outperforms the best equivalence factor over some drive cycles.

2.4.3. SOC Feedback

In work done by Kessels et al. [20], a battery SOC feedback controller is used to adapt the energy management strategy. This strategy is implemented in a parallel vehicle with an integrated starter motor [20]. In this work, it was shown that the strategy used achieve performance close to the maximum performance as determined by the optimal strategy. This is done without the need for *a priori* knowledge of the drive cycle [20]. This method defines a reference state of energy (SOE_{ref}) for the battery. Using external vehicle signals, an optimal control value λ is selected and fed into the control strategy. It is assumed that the battery SOC is an indication of whether or not the correct estimation value for λ has been selected. If the current battery SOC starts to deviate too far from the reference SOC, then λ is adapted. To keep the SOC around the reference value, a proportional integral (PI) controller is used (Figure 11).

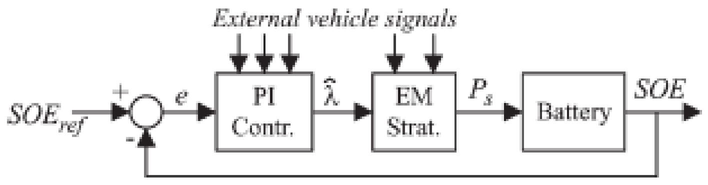


Figure 11: SOE Feedback Controller for Selecting Optimal Control Value [20]

Results for the NEDC drive cycle are shown below in Table 5.

Table 5: Fuel Economy Results for SOC-Adaptive Energy Management Strategy [20]

Simulation	**Absolute fuel use [g]**	**Relative fuel use [%]**
(Sim_1) Baseline	577	100
(Sim_2) Dynamic Programming	436	75.6
(Sim_3) On-line Strategy, λ fixed	435	74.5
(Sim_4) On-line Strategy, PI-control for λ	441	76.4

Figure 12 shows the SOE trace over the NEDC cycle for the DP, fixed λ, and PI-controlled λ strategies.

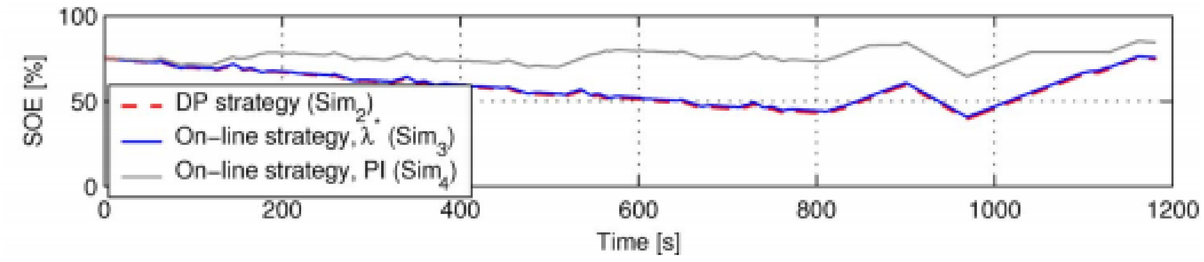


Figure 12: SOC Trace Comparison for 3 Strategies [20]

It can be seen that the PI-controlled strategy allows for less variation in the SOC compared to the other strategies. This makes intuitive sense since it is being tracked to remain near the SOE_{ref} value. Despite the lower variation in SOC, results in Table 5 show that the strategy compares very closely with the results obtained from DP.

Work done by Chasse et al. also demonstrates the online ability to adapt an optimal control strategy equivalence factor using SOC feedback as a control parameter. The expression for the adaptive equivalence factor is shown below:

$$s(t) = s^o + K_p(x^{sp} - x(t)) \tag{2-17}$$

Where $s(t)$ is the equivalence factor, K_p and s^o are tuning parameters, x^{sp} is the reference SOC, and $x(t)$ is the real-time SOC value. The purpose of the expression is to impose penalties if the SOC becomes too high or too low [11].

Work performed by Han et. al, [21], uses a recurrent neural network (RNN) in conjunction with battery SOC tracking to update the equivalence factor of A-ECMS for use in a PHEV.

The goal of this work was twofold: to maximize fuel economy while also maintaining the HV battery life. The second goal of maintaining HV battery life is implemented by adding an additional weighting factor to the A-ECMS.

The equivalence factor of A-ECMS is indirectly updated using the RNN. The RNN is first trained offline using the optimal SOC trajectory, which is obtained from dynamic programming. In the online implementation, the RNN receives current traffic information and vehicle states. Specifically, the RNN receives average speed, distance, and the SOC of the previous step. The RNN then outputs the ideal reference SOC – based on the training from the optimal SOC trajectory. The reference SOC is then compared to the actual SOC, and a PI controller is used to update the equivalence factor of the ECMS based on the deviation of the actual SOC from the reference SOC [21]. Results from the work of Han et. al showed that the A-ECMS with the RNN implementation was able reduce fuel consumption by 18.1% when compared to a simple CS-CD control strategy.

2.5. Alternative Methods

In work done by Connelly, et. al at WVU, [22], [23], SOC dependent and SOC independent shift maps were developed and analyzed for implementation into the redesigned powertrain of a 2016 Chevrolet Camaro. The shift schedules were tested in both MIL and VIL environments. The original Camaro was redesigned with a P3 PHEV architecture. Analysis of the shift maps were performed to determine the effect on engine and vehicle fuel economy. The original stock shift schedule was a function of vehicle speed and accelerator pedal position. This two-parameter stock shift schedule was improved using an exhaustive sensitivity analysis which took into account the additional power available from the electric powertrain components.

The developed two-parameter shift schedule (SOC independent shift schedule) resulted in an increase in both engine and full vehicle fuel economy. Additionally, a three-parameter shift schedule was developed that added the HV SOC as the third parameter (SOC dependent shift schedule). With the SOC-dependent shift schedule, an attempt was made to explore a greater amount of solution space by varying the initial starting SOC in the MIL environment. Ultimately, in a comparison between the SOC dependent and SOC independent shift schedules, there was not a noticeable difference in overall fuel economy - although they both produced improvements over the stock shift schedule [23].

In work done by George et. al of WVU [22], [24], a base power-loss minimization torque-split algorithm was developed for a PHEV, against which an algorithm focused on reducing engine transients was developed and compared. Both torque split algorithms were developed using cost functions. The torque split algorithm which focused on reducing engine transients used an updated and improved cost function. In MIL, an 8.25% decrease in engine transients was observed over the base power-loss algorithm [24]. In VIL, a 14.6% decrease in engine torque transients was measured, with 4.84% of the reduction being attributed to the reduction of engine torque transients. Additionally, a 10.4% reduction in CO emissions was observed in VIL when compared to the base power-loss algorithm [24].

In a vehicle dynamometer test of fuel economy, an increase in fuel economy of 1.70% was observed. However, this increase was within the 3% margin of uncertainty of the CAN collected data. George recommended that the method of engine torque reduction performed in his work be applied in future to an A-ECMS.

2.6. ANN Control Systems

ANN’s have been shown to produce desirable and stable control results across a wide variety of areas. A frequent area in which neural networks are implemented is that of aircraft control [25], [26], [27]. Of particular interest is the work done by Furquan et. al, in the use of a neural network to control landing, roll, pitch and altitude hold. In this work, the neural network was trained using available flight data, including control actions from human intervention. The goal of implementing the neural network is to improve the performance of conventional controllers present on an aircraft. Simulation results showed that the neural network controller provided robustness to variation of system parameters [25].

Additionally, work done by M. Perhinschi et. al, showed positive results using a neural network to develop an adaptive flight controller. The neural network compensation was able to requite inversion errors and changes in aircraft dynamics – even including actuator failures. In all scenarios investigated, simulations showed that neural network augmentation provided overall robustness and good stability and performance characteristics [26].

3. Methodology

The objective of the current work is to present an A-ECMS control strategy using an on-board artificial neural network (ANN) which dynamically updates the equivalence factor based on a sliding time window of past driving parameters. This implementation of A-ECMS most closely aligns with that of driving pattern recognition. This work will describe the vehicle model, vehicle control algorithm, generation of ANN training data, ANN design, validation data, and testing methodology. All the results shown and analyzed have been obtained purely in the model-in-the-loop (MIL) environment.

3.1. Full Vehicle Model

The full vehicle model used in this research was developed in MATLAB Simulink. The full vehicle model consists of three primary models: the driver model, plant model, and controller model. The plant model contains all the physical component models of the car: the engine, motor, battery, drivetrain, transmission, and torque converter. The controller model contains the control algorithms needed for interaction between the plant components and the driver model. Many of the component models used were initially created by MathWorks. In the following sections, credit is given to MathWorks regarding those component models which have remained virtually unchanged.

The top level of the full vehicle model is shown below in Figure 13. The top level of the model shows the Model-in-the-Loop (MIL) Plant and MIL Controller subsystems. The plant model contains all the models of the physical systems of the vehicle, i.e. engine, motor, transmission, vehicle body, torque converter, wheels, and brakes. The controller model contains all the algorithms used to control the plant components. Physical signals (i.e. component speeds, torques and temperatures) are passed from the plant model to the controller model, while commands (i.e. torque, current, and speed commands) are passed from the controller model to the plant model.

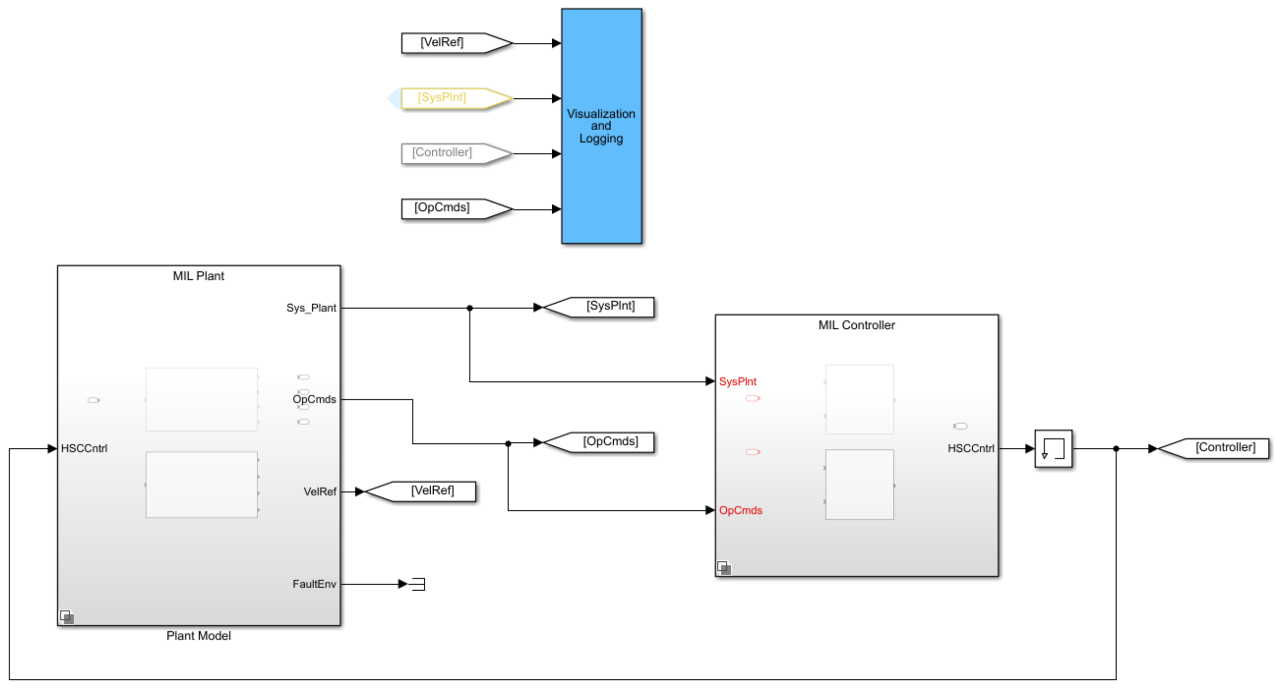


Figure 13: Top Level of Full Vehicle Model

The top level also contains the visualization and logging subsystem. The outputs of the plant and controller models are fed to this subsystem. The visualization and logging subsystem contains a variety of scopes and logging blocks that are used to view vehicle parameters such as speed, component torque, HV battery voltage and current, SOC, etc.

3.1.1. Battery Model

The battery is modeled based on the General Motors HEV4 battery pack. This battery pack has a nominal voltage of 300 V and a peak power of 50 kW. The total usable energy storage is 1 kWh. The Datasheet Battery from the Powertrain Blockset from MathWorks is used to model the battery. The HV battery is model is shown below in Figure 14.

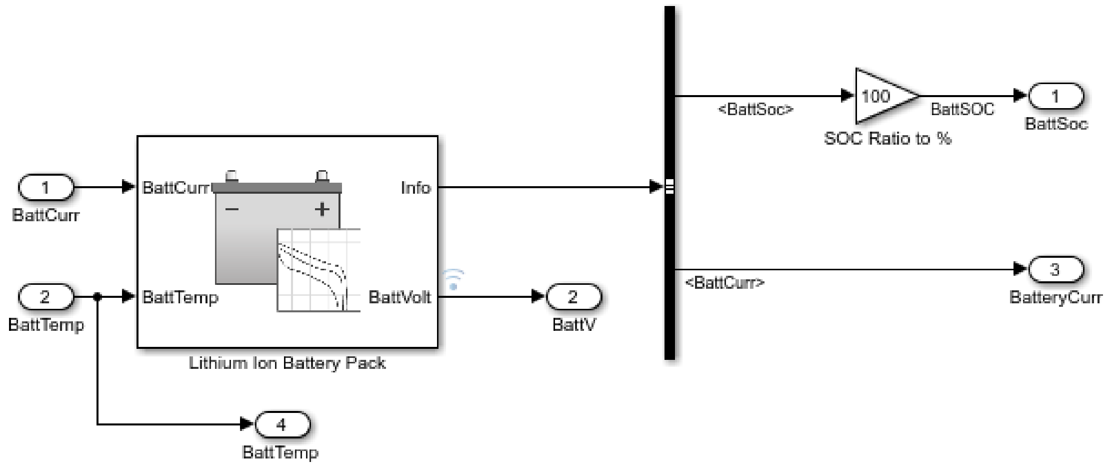


Figure 14: HV Battery Model

In the Datasheet Battery block, the initial battery capacity can be set based on the simulation setup, thus defining the starting SOC of a simulation. The block uses the battery current load and the battery housing temperature as inputs and uses basic electrical relationships to output parameters such as combined and normalized battery current, state of charge, output voltage, and output power. The battery model uses internal resistance and open-circuit voltage lookup tables to report the battery voltage. This block determines the battery SOC based on the total battery capacity (Ah) and the integration of battery current over each time step [28].

To reflect the peak power outputs, a battery management system (BMS) was created. The BMS outputs the maximum voltages, powers, and currents that the battery can provide. The BMS is shown below in Figure 15.

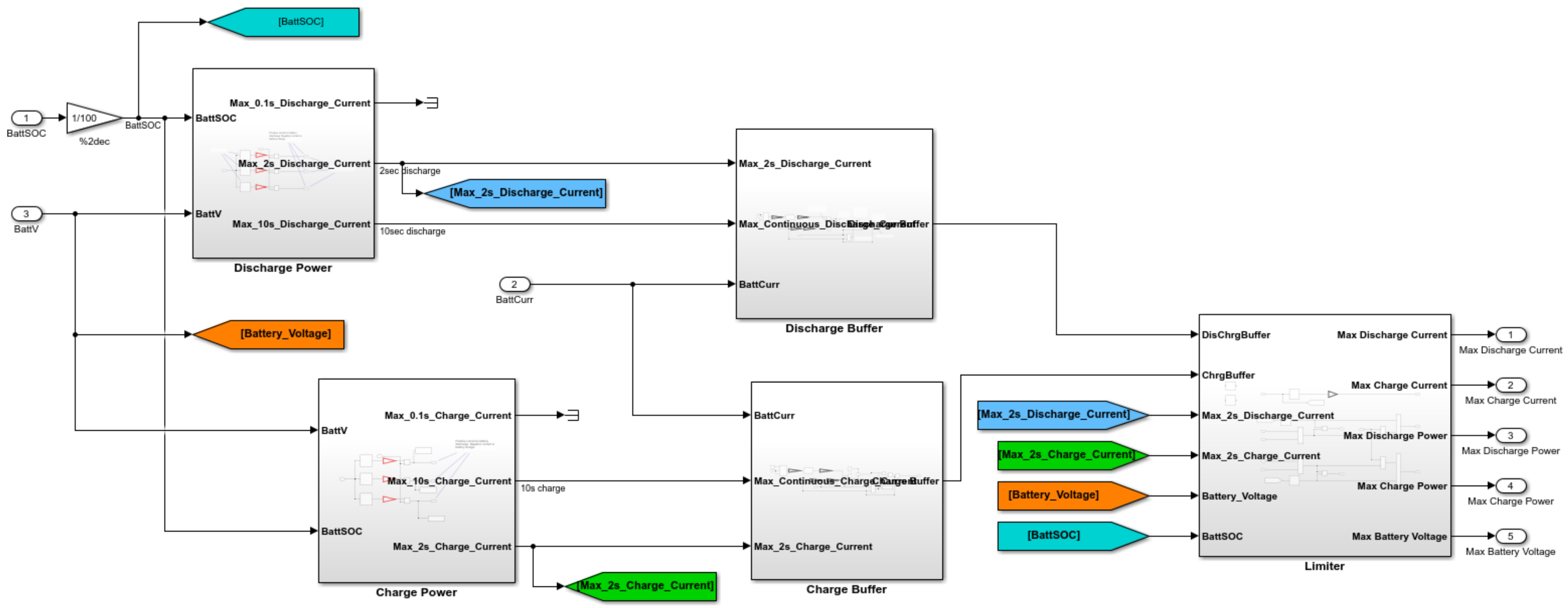


Figure 15: Battery Management System

The power discharge and charge limits were provided by GM. The limits are given as functions of battery SOC and are in terms of maximum discharge and charge powers over set time intervals. Maximum powers are provided as maximum 10 second, 2 second, and 0.1 second charge and discharge powers. For simplicity, the 10-second maximum is considered to be the continuous maximum discharge/charge power, and the 2-second maximum is considered to be the peak maximum discharge/charge power. The discharge/charge powers are divided by the battery voltage to arrive at battery charge/discharge currents.

To model these time-varying limits, a windowed integrator is used to calculate a buffer. The windowed integrator integrates over a receding 2-second time window. The buffers are used to reflect the time-based limits. The buffer values vary between 0 and 1 and are multiplied by the 2-second maximum currents. The buffer value starts out as 1 whenever the 2-second maximum current is drawn. If the maximum current is held for 2 seconds, then the buffer will be zero, thus reducing the maximum battery current to the 10-second (continuous) current. The logic calculating the discharge buffer is shown in Figure 16. The discharge and charge buffers are similarly calculated: the primary differences resulting from the negative sign of the battery current.

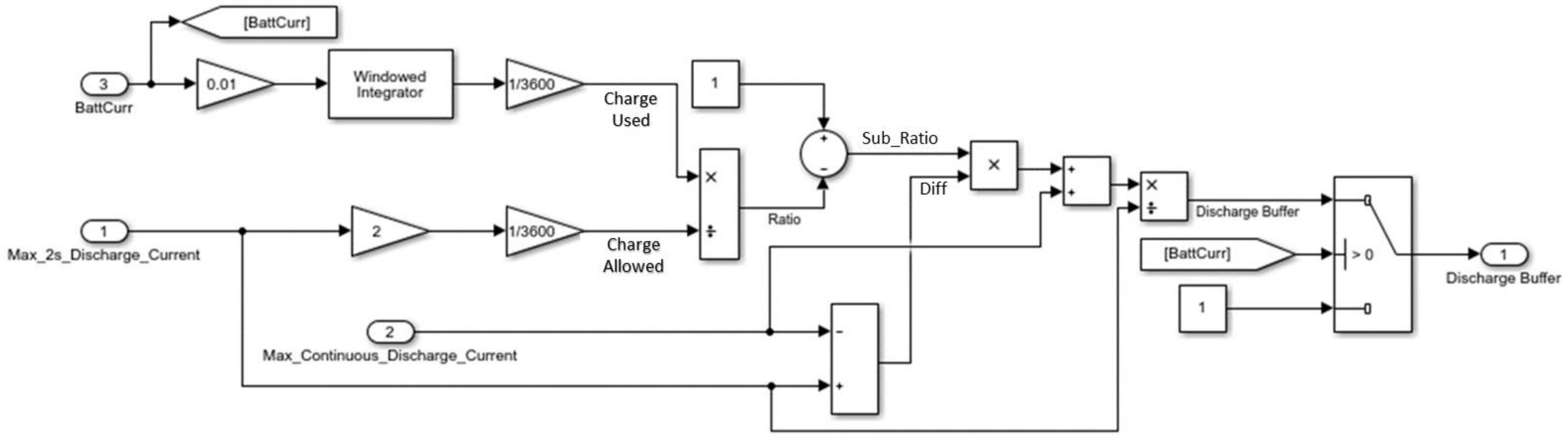


Figure 16: Discharge Buffer Determination

The input battery current (‘BattCurr’) is scaled by a value equivalent to the BMS model time step before being fed into the windowed integrator block. After integration, the signal is divided by 3600 to obtain the amount of charge used (‘Charge Used’) in Ah. The energy used is divided by the amount of charge available (‘Charge Allowed’). ‘Charge Allowed’ is the 2-second maximum discharge current multiplied by 2 (seconds) and divided by 3600 to arrive at units of charge (Ah). If this ratio between ‘Charge Used’ and ‘Charge Allowed’ is ever 1, then all of the available charge of the 2-second maximum power limit has been consumed.

The ratio (‘Ratio’) of used charge (‘Charge Used’) to available charge (‘Charge Allowed’) is subtracted from 1. Now, if the available charge of the 2-second maximum power is reached, the signal value will be zero. This signal (‘Sub_Ratio’) is multiplied by the difference between the 2-second maximum discharge current and the continuous discharge current (‘Diff’). This difference between the 2-second maximum discharge power and the continuous discharge power is a value of how many additional amps are available above the continuous discharge power.

Multiplying ‘Sub_Ratio’ and ‘Diff’ results in a measure of how far above the continuous discharge current the maximum discharge current is allowed to be. If the value is zero, then no additional current is allowed above the continuous discharge current. The amount of current allowed above the continuous current is added to the continuous current to arrive at the maximum allowable discharge current. The maximum allowable current is then divided by the 2-second continuous current to arrive at the discharge buffer.

If the battery current is ever negative, this indicates a charging event. A charging event will reset the discharge buffer to 1, allowing the maximum 2-second discharge current to be drawn.

The “Limiter” subsystem on the right-hand-side of Figure 15 uses the calculated charge and discharge buffers and the SOC to output the maximum battery voltage, maximum charge and discharge currents, and maximum charge and discharge powers. This logic is shown in Figure 17. The two tables “Over Charge Foldback” and “Over Discharge Foldback” output a gain value between 0 and 1, depending on the SOC level. The HV battery is required to stay between 80% and 20% SOC. If these bounds are exceeded, then the HV battery contactors will open. To prevent the SOC from going out of the prescribed bounds, the lookup tables limit the current and power if the SOC gets too close to a boundary. For example, until the SOC reaches 30%, the output of the “Over Discharge Foldback” table will be 1 – not inhibiting the discharge current or power. However, if the SOC reaches 30%, the output from the “Over Discharge Foldback” table will linearly approach zero, reaching zero at 22%. If the output value is zero, then the maximum discharge current and power is zero. The same procedure applies for the “Over Charge Foldback” table: if the SOC approaches 80%, the output will reach zero, setting the maximum charge current and power to 0.

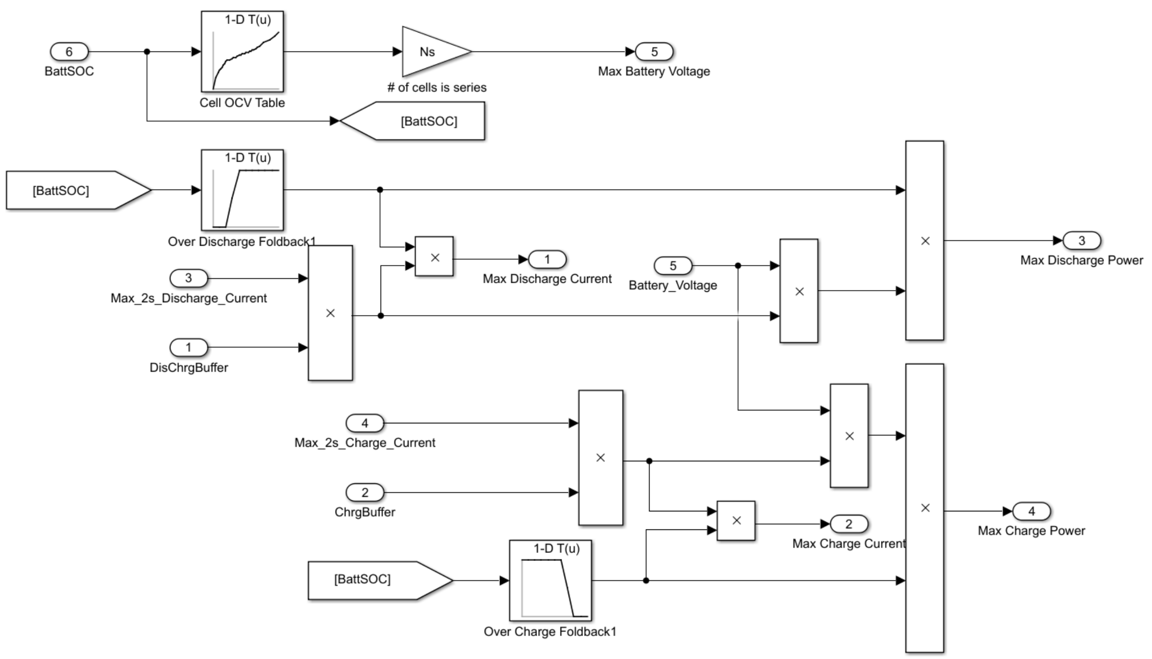


Figure 17: Battery Limit Logic

3.1.2. Motor Model

The electric motor is an electric axle with an integrated electric motor and gearbox (Figure 18). This system is known as the eAWD and is manufactured by Magna. The Magna eAWD has a gearbox ratio of 9.17, peak maximum power (20 seconds) of 50 kW, peak maximum torque (20 seconds) of 200 Nm, maximum continuous power of 20 kW, maximum continuous torque of 90 Nm, and a max motor speed of 12000 rpm.

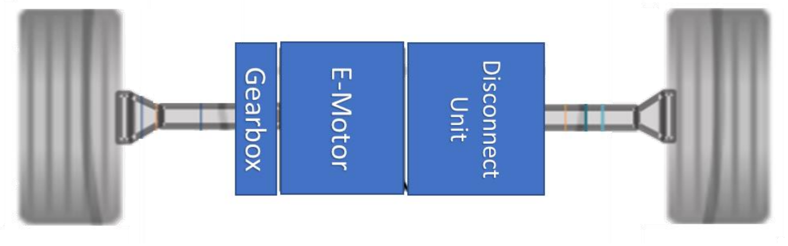


Figure 18: MAGNA eAWD

The electric motor is modeled using the Mapped Motor block of the Powertrain Blockset from MathWorks (Figure 19). The Mapped Motor Block uses lookup tables to determine the maximum torque available from the machine. If necessary, the torque command is clipped according to the maximum torque-speed envelope. A power-loss lookup table is used to determine the motor efficiency at given operating speeds and commanded torques. The motor current is determined based on the calculated electrical power and the provided battery voltage [29]. The Mapped Motor block receives inputs of battery voltage (BattVolt), motor speed (MtrSpd), and torque command (TrqCmd). The block outputs battery current (BattCurr), motor torque produced (MtrTrq), and an information signal (info) that contains a variety of other signals related to the operating conditions of the motor.

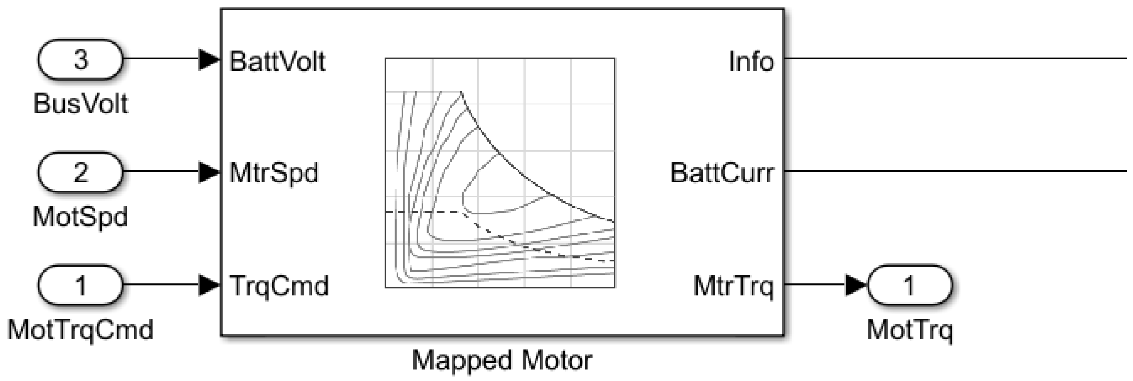


Figure 19: Mapped Motor Model

A motor management system (MMS) was modelled to impose the 20 second peak torque values. The MMS (Figure 20) is modeled similarly to the BMS described in the battery section. The only difference is the peak times.

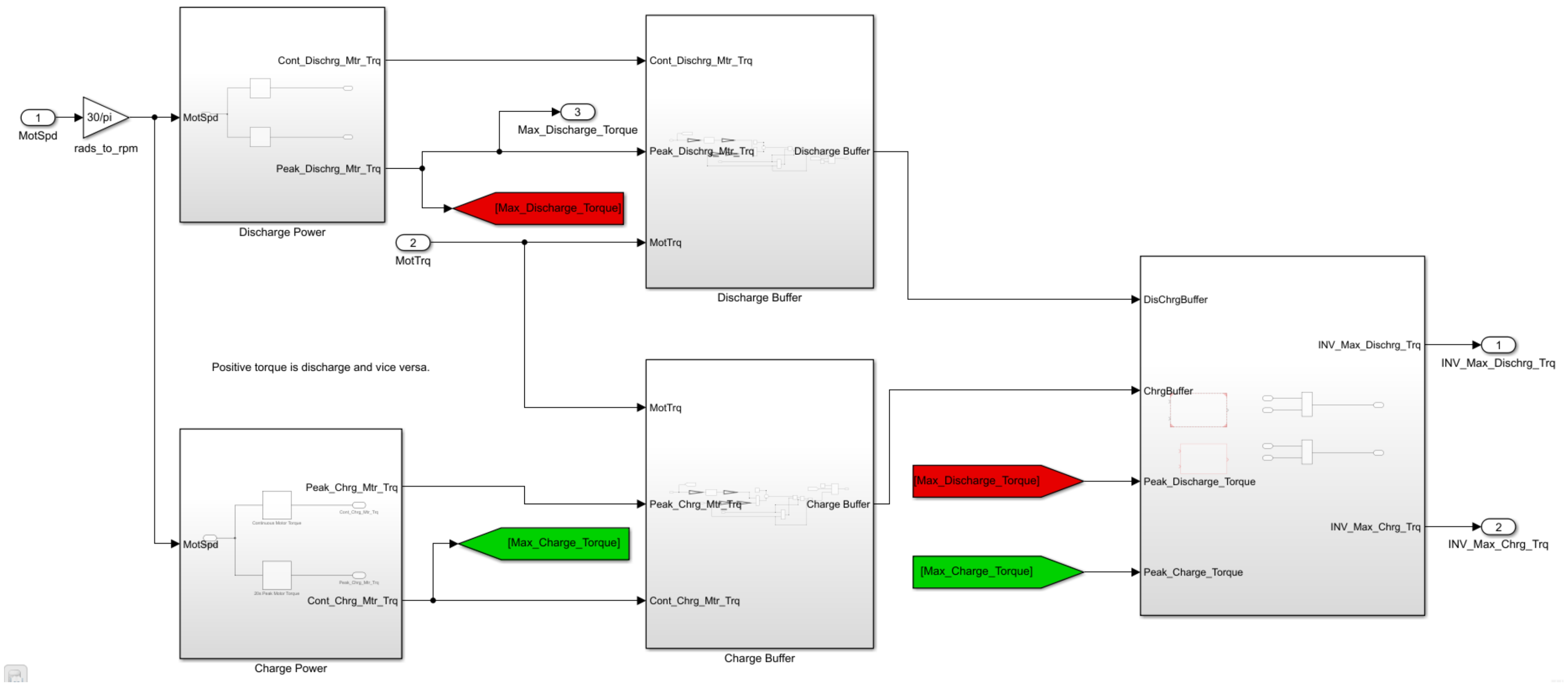


Figure 20: Motor Management System

The maximum and minimum motor torque are calculated in the subsystem on the far right-hand-side of Figure 20. In this subsystem, the maximum motor torque is calculated by multiplying the maximum discharge torque by the discharge buffer. Similarly, the maximum charge torque is calculated by multiplying the maximum charge torque by the charge buffer.

3.1.3. Engine Model

The engine being modeled is a GM 2.5L LCV in-line 4 cylinder engine rated for a maximum torque of 255 Nm and a maximum power of 148 kW. To model the engine, the Mapped Spark Ignition (SI) Engine of the Powertrain Blockset from MathWorks is used (Figure 21).

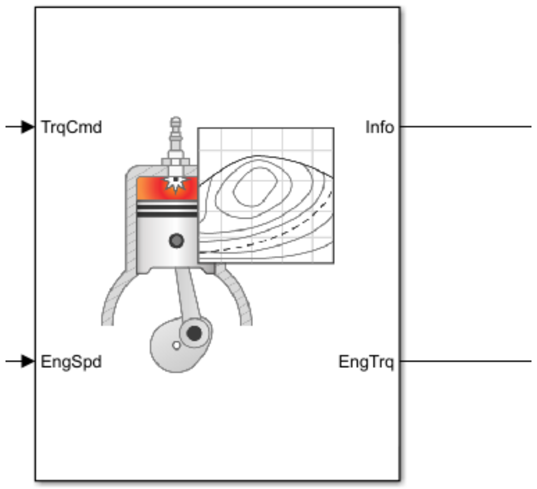


Figure 21: Mapped Spark Ignition Engine Model

The engine model was provided by MathWorks and was parameterized using tabulated steady-state operating conditions data provided by GM. This block receives torque command (TrqCmd) and engine speed (EngSpd) as inputs and outputs an actual engine torque produced value (EngTrq) and additional engine operating parameters (info) such as air mass flow, fuel flow, exhaust temperature, etc. The engine model uses lookup tables to determine the engine operating values [30]. There is a first order transfer function included in this model that acts as a low pass filter to provide engine smoothing and delay. This low pass filer represents airflow and combustion dynamics.

One limitation of this engine model is that the table data is derived from steady-state operating conditions. However, in reality, the engine frequently operates in a dynamic/transient state. It is assumed that, although the steady-state operating parameters are used in the lookup tables, these parameters adequately represent the parameters necessary for modeling the dynamic operation of the engine.

3.1.4. Transmission Model

The transmission being modeled is a GM 9-speed M3D (9T50) with an accumulator. The M3D is modeled using the Ideal Fixed Gear Transmission of the Powertrain Blockset from MathWorks (Figure 22).

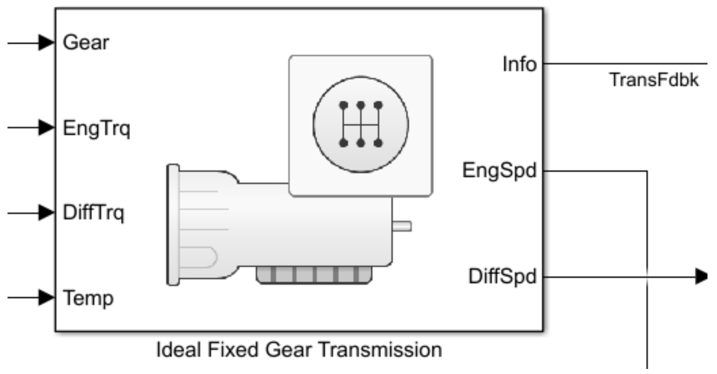


Figure 22: Ideal Fixed Gear Transmission Model

The Ideal Fixed Gear Transmission block was provided by MathWorks and was parameterized using data relevant to the M3D transmission provided by GM. The transmission model receives a gear command (Gear), engine torque (EngTrq), differential torque (DiffTrq), and an oil temperature value (Temp). Using mathematic relationships, the model outputs engine speed (EngSpd), differential speed (DiffSpd), and other operational values (info) such as transmission speed and transmission gear etc. [31]. Transmission efficiency is also included in the transmission model and determined using a four-dimensional lookup table parameterized with tabulated data provided by GM.

3.1.5. Torque Converter Model

The torque converter is also modeled using the Powertrain Blockset from MathWorks (Figure 23).

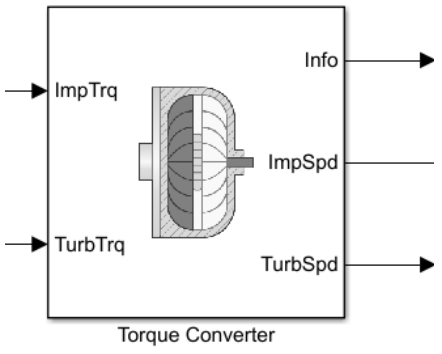


Figure 23: Torque Converter Model

This model was provided by MathWorks and is parameterized with drive shaft dynamic coefficients, converter characterization coefficients, and clutch parameter coefficients. The torque converter model receives inputs of impeller torque (ImpTrq), which is from the engine crankshaft, and turbine torque (TurbTrq), which is from the transmission. The model outputs impeller speed (ImpSpd) and turbo speed (TurbSpd). The output of impeller speed is equivalent to the engine speed.

3.1.6. Friction Brakes and Wheels Model

The friction brakes and wheels are modeled together using the Powertrain Blockset (Figure 24). The wheels are a longitudinal model, and the brakes are modeled as disk brakes.

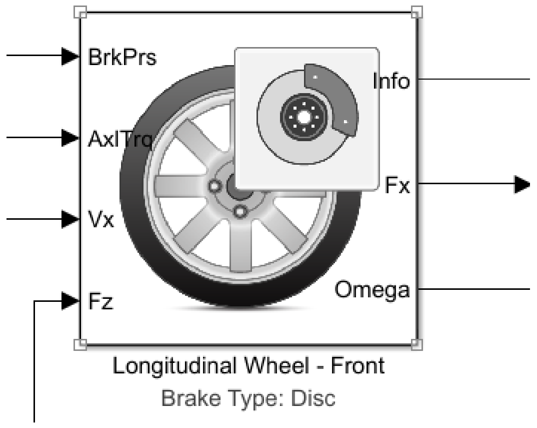


Figure 24: Friction Brake and Wheel Model

The friction brakes and wheels model was provided by MathWorks. The wheel dynamics are modeled using a "Magic Formula," which uses constant coefficients for calculations. The block is parameterized with wheel parameters such as inertia, angular velocity, damping coefficient, radius, velocity force components, etc. Disc brake parameters include static and kinetic coefficients, pad radius, actuator bore, etc. The rolling resistance of the wheels are calculated as a function of velocity, normal force, and tire pressure [32].

The inputs to the model are brake pressure (BrkPrs), axle torque (AxlTrq), velocity (Vx), and normal force (Fz). Outputs of the model are longitudinal axle force (Fx), angular wheel velocity (Omega), and an information signal (info) that contains additional wheel information.

This model of the brakes does not include regenerative braking. Regenerative braking is done entirely by the motor commanding negative torque. The logic for commanding negative motor torque is discussed in section 3.2.1, Driver Torque Request Determination and Arbitration.

3.1.7. Driver Model

The driver is also modeled using the Powertrain Blockset (Figure 25). This longitudinal driver model simulates a human driver by using a predictive closed-loop controller to minimize the error between a reference drive trace velocity and the actual vehicle velocity. In this model, the driver has the ability to "look ahead" and preview the drive trace which allows the driver to smoothly adapt to a wide range of varying drive traces, similar to a human driver. This model was provided by MathWorks.

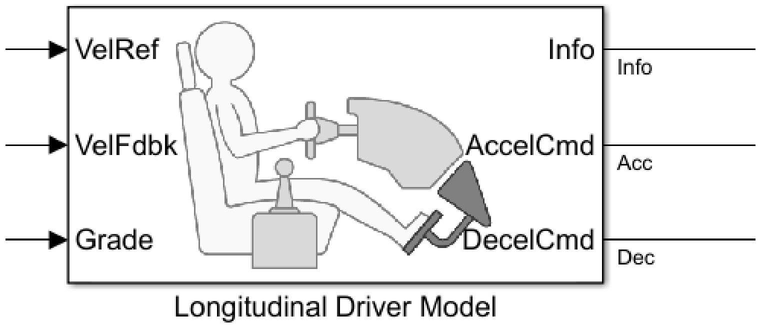


Figure 25: Longitudinal Driver Model

The driver subsystem receives three input signals: the reference velocity of the drive trace (VelRef), the actual vehicle velocity (VelFdbk), and the road grade (Grad). Based on the error between the actual and trace velocity, the driver model outputs acceleration commands (AccelCmd) and deceleration commands (DecelCmd) and an information signal (info) that contains additional information related to the driver. Configuration parameters to the driver model include driver response time, preview distance, vehicle aerodynamic drag coefficient, rolling resistance coefficient, vehicle weight, and driveline resistances. Since receiving the model from MathWorks, the only thing that has been modified has been vehicle weight and driver response times. The driver model was initially parameterized with a weight that represented the original weight of the Blazer. Since hybridizing the Blazer, this weight has changed. The driver response time has been modified to more closely match the reference velocity (VelRef).

The driver model is able to anticipate the upcoming drive trace by using future trace information to determine the future accelerations which will minimize a performance index. The performance index is a projected weighted mean squared error between the previewed drive path and previewed plant output. The previewed path input is defined by a time T*, which defines the previewed path input T* seconds ahead. T* is derived from the preview distance parameter. The previewed output is related to, and defined by, the present state of the vehicle dynamics [33].

By previewing the drive trace, the predictive driver model can make smooth acceleration/deceleration transitions, much as a human driver can when viewing road conditions ahead of them. Simpler driver models, such as those based on PI controllers, only rely on the instantaneous error between desired and actual vehicle speed. As such, a PI-based driver does not have as smooth responses when compared to a predictive driver model.

3.1.8. Longitudinal Vehicle Body Model

The vehicle body is modeled using the Powertrain Blockset (Figure 26). This block is parameterized with mass, drag coefficient, frontal area, vehicle dimensions, etc. The block receives inputs of longitudinal forces on the front (FwF) and rear (FwR) axles, road grade (Grade), and longitudinal wind speed (WindX).

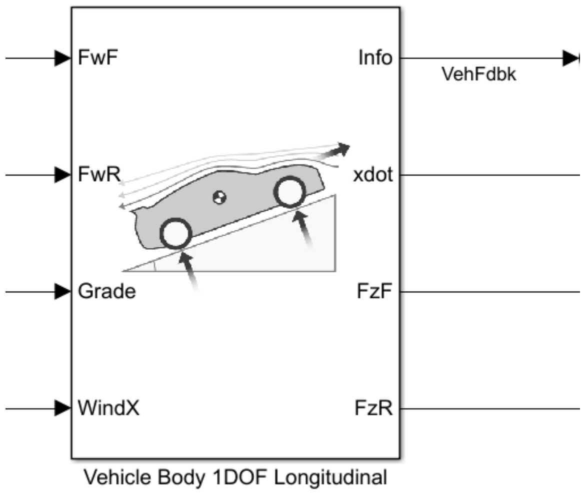


Figure 26: Vehicle Body Model [34]

This model was provided by MathWorks. The block outputs the normal forces on the front (FzF) and rear (FzR) axles, and vehicle speed (xdot). Additionally, the "info" signal outputs additional information related to the vehicle such as acceleration, drag, etc. The outputs of speed (xdot) and acceleration are used as inputs to the ANN in the controller model.

3.2. Control Algorithm

The control algorithm receives inputs from the driver and feedback from the drivetrain components to ultimately determine an appropriate torque split between the powertrain components. The control algorithm includes torque determination and arbitration algorithms, powertrain constraint algorithms, as well as the implementation of the ECMS algorithm and ANN.

3.2.1. Driver Torque Request Determination and Arbitration

The driver torque request determination and arbitration block (Figure 27) receives the accelerator and brake pedal percentages and the vehicle speed.

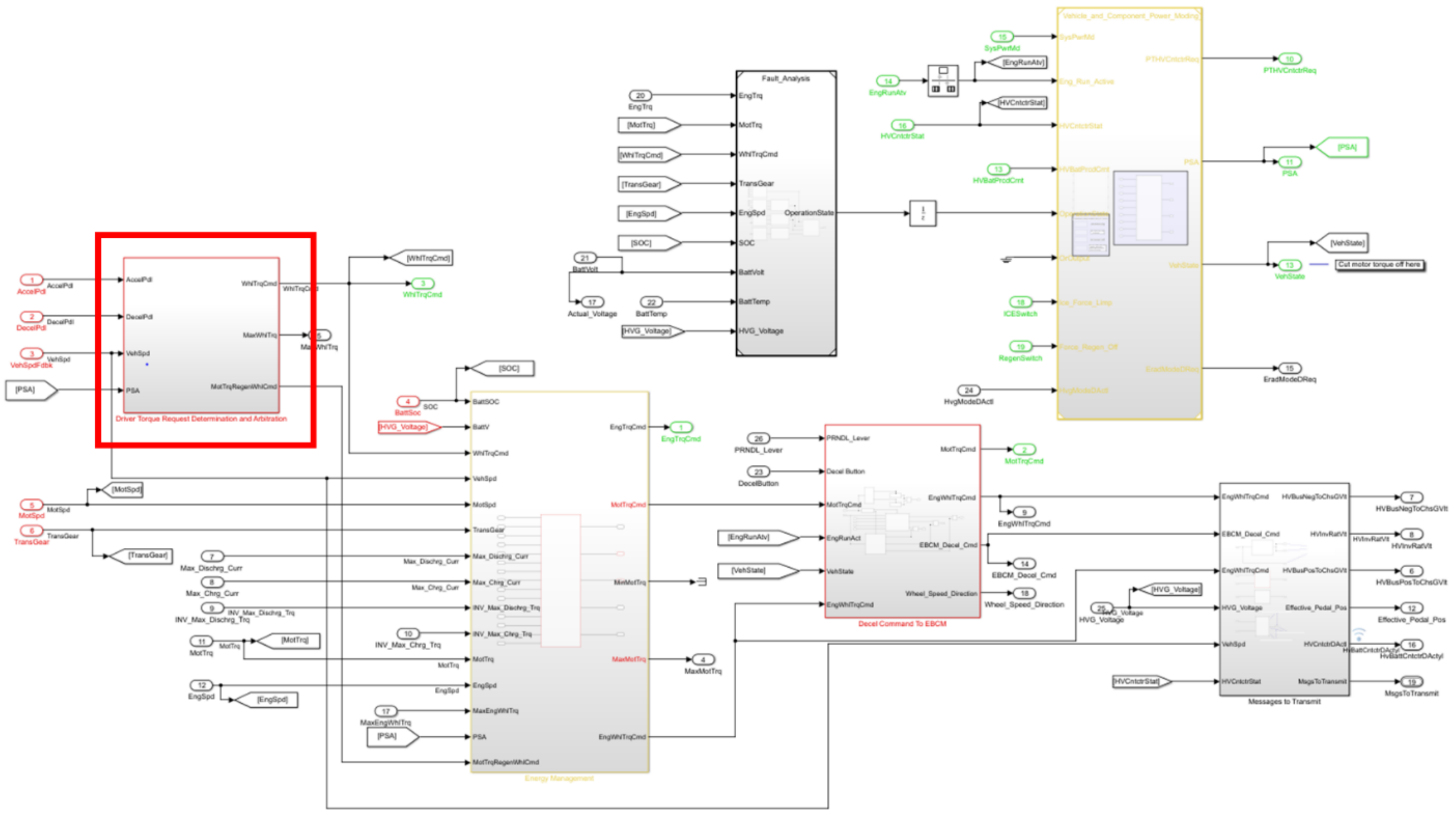

Driver Torque Request Determination and Arbitration
Fault_Analysis
Vehicle_and_Component_Power_Moding
Energy Management
Decel Command To EBCM
Messages to Transmit
Cut motor torque off here

Figure 27: Driver Torque Request Determination and Arbitration (Highlighted in Red) of the Control Algorithm

Based on these signals a wheel torque command and a regenerative braking wheel torque command are generated. A picture of the logic for the determination of the regenerative braking wheel torque command is shown in Figure 28.

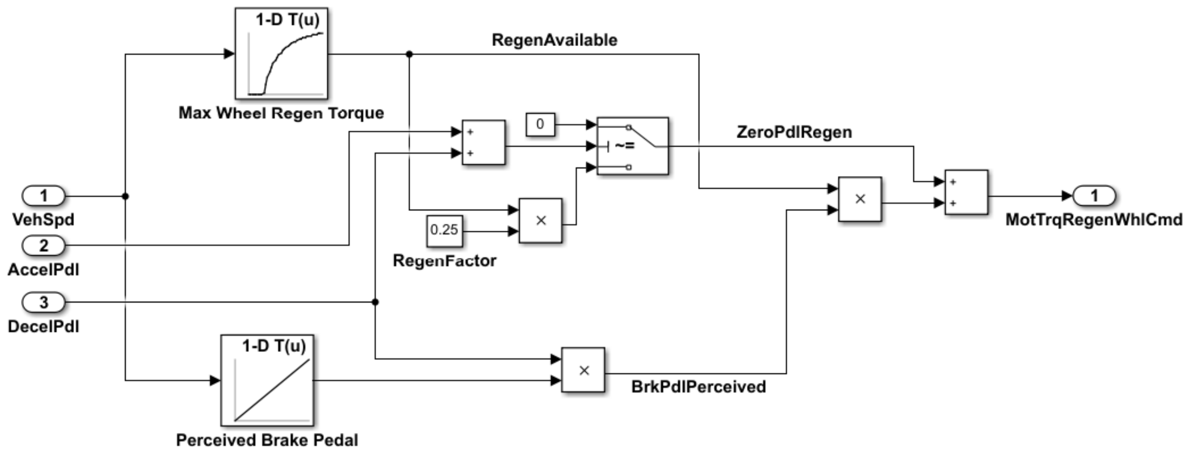


Figure 28: Determination of Regenerative Braking Wheel Torque Command

Based on vehicle speed, the maximum available wheel regen torque is found using a lookup table. This lookup table is parameterized with the Magna motor torque specifications.

A typical feature in HEV's is zero-pedal regen. This feature applies a small amount of negative torque whenever the brake or accelerator pedal is not being pressed. Zero-pedal regen is determined here by first checking to see if the brake or accelerator pedal are being depressed. If the value of these signals, when added together, is greater than a small threshold (approximately zero), then the maximum amount of wheel regen torque available is multiplied by a calbratable regen factor. This factor can be calibrated to balance between driver comfort and energy recapture. If the addition of the brake and accelerator pedal is greater than the threshold (i.e. one of them is being pressed), then a zero is passed for the zero-pedal regen toruqe.

The vehicle speed is also fed into the “Percieved Brake Pedal” lookup table. This lookup table is primarily for dirver comfort. This lookup table outputs a gain value between zero and one. At low vehicle speeds, the output is close to zero. This is because the driver comfort is low if the full amout of regenerative braking is aplied at low vehicle speeds. As the vehicle speed increases, the percentage of allowable regenerative braking is increased. Past a calibratable vehicle speed threshold, the gain output is saturated at 1.

The brake pedal fraction is multiplied by the gain output from the “Perceived Brake Pedal” lookup table. This creates the perceived brake pedal (BrkPdlPerceived) signal. The perceived brake pedal signal is multiplied by the maximum amount of regen wheel torque. This is then added to the amount of zero-pedal regen torque. If the zero-pedal wheel torque is nonzero, then the perceived brake pedal will be zero and vice versa. The final signal is output as the motor regen wheel torque command (MotTrqRegenWhlCmd).

The driver demanded wheel torque is calculated as a function of vehicle speed and accelerator and brake pedal positions. First, the maximum allowable vehicle torque is determined as a function of vehicle speed. The maximum allowable vehicle torque comes from the combined available wheel torque of the engine and motor, based on their individual torque profiles. The vehicle speed is fed into an offline-generated lookup table that outputs maximum wheel torque.

Once the maximum wheel torque is known, logic shown in Figure 29 is used to determine the driver demanded wheel torque.

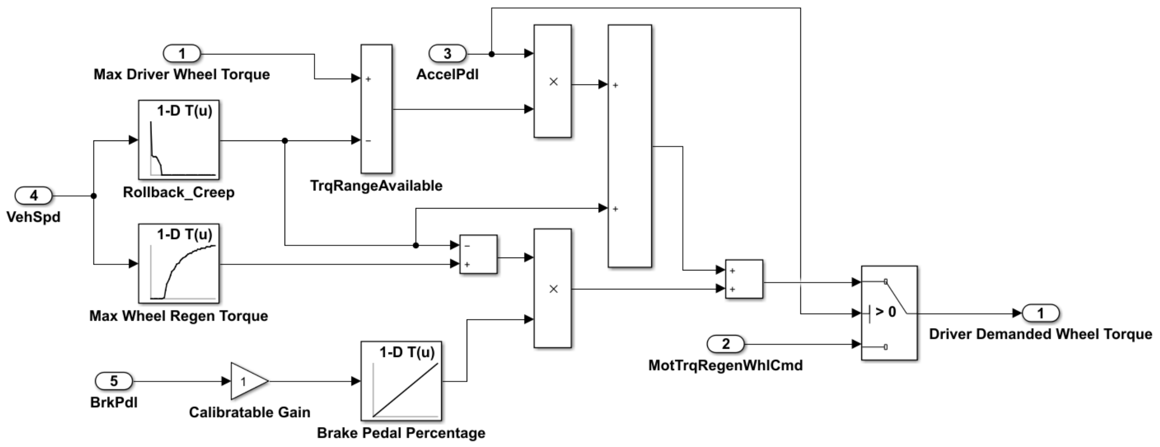


Figure 29: Driver Torque Request Determination

Vehicle speed is fed into a lookup table that defines the amount of creep torque for the vehicle. The creep torque is subtracted from the maximum torque available to get a range of available torque. The available torque is multiplied by the fraction of accelerator pedal being applied. The resulting torque is that torque which is available to the accelerator pedal. Creep torque is added to arrive at a final driver demanded wheel torque.

3.2.2. Motor and Battery Constraints

The motor needs to be carefully monitored and controlled to avoid over charging or discharging the battery. The battery will provide whatever load is commanded by the motor. It does not have any internal limitations that will prevent it from trying to supply power if it is close, or even past, its limits. In other words, the battery will not protect itself. Consequently, its limits must be understood and be reflected in the torque that the motor requests from the battery. In the following logic description, a "battery torque" limit is calculated using the motor efficiency tables.

First, the battery constraints are calculated based on the SOC, voltage, and temperature limits. The logic for determining the maximum available battery power is shown in Figure 30.

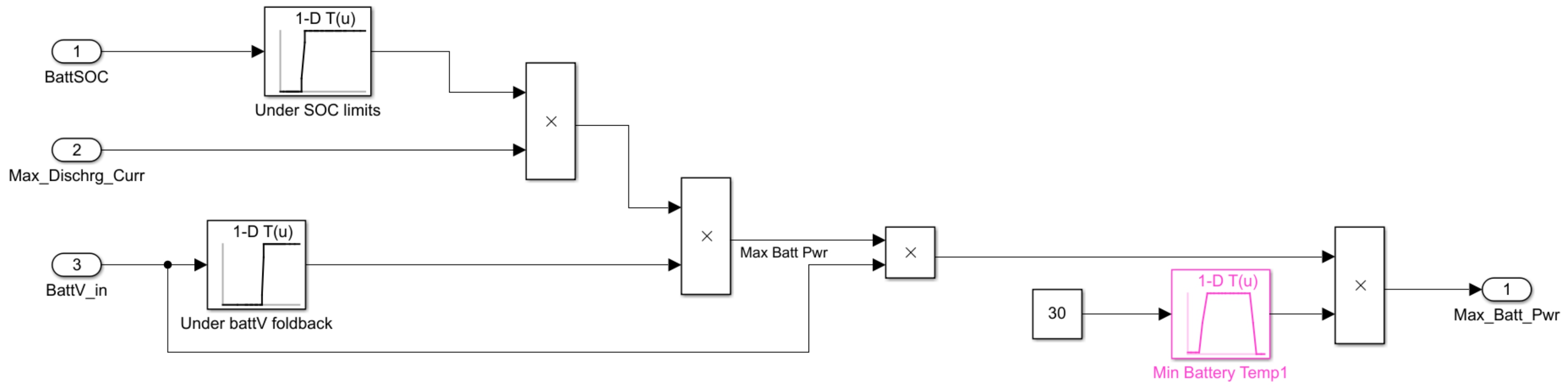


Figure 30: Maximum Battery Discharge Power Calculations

The SOC (BattSOC) is fed into a lookup table which outputs a gain value between zero and one. The lookup table is parameterized with threshold limits based on acceptable range of SOC. While the battery SOC is above a specified threshold, the gain output is 1. As the SOC approaches the lower threshold, the gain will linearly approach 0 until the threshold is passed, at which point the gain will be zero. With an output of zero, the maximum battery power is 0.

Voltage is also used to determine the maximum battery discharge power. The same type of lookup table is used for voltage as is used for SOC – except the table is parameterized with thresholds pertaining to voltage. While the voltage is above a prescribed threshold, the gain output is 1. As the voltage approaches the lower threshold, the gain value linearly approaches zero, until the threshold is reach, at which point the gain is 0. With a gain of 0, the maximum battery power is zero.

Lastly, the maximum battery discharge power is also determined using the battery temperature. The battery temperature (30 C as shown in Figure 30) is fed into a lookup table similar to those described for the SOC and voltage. The temperature lookup table will output a value of 1 while the battery is in a safe operating temperature. If the temperature becomes too hot or cold, the output will eventually reach zero, at which point the maximum battery power is reduced to zero. In real-time implementation, the constant temperature of 30 C will be replaced by a real-time temperature signal.

A similar method is used to determine the maximum battery charge power. The essential difference is that the lookup table values are swapped, i.e. limits are imposed as the SOC and voltage becomes too high.

To determine the motor efficiency, two efficiency lookup tables were created based on power loss tables supplied by Magna (Figure 31). The efficiency tables are function of motor speed (MotSpd) and motor torque (MotTrq).

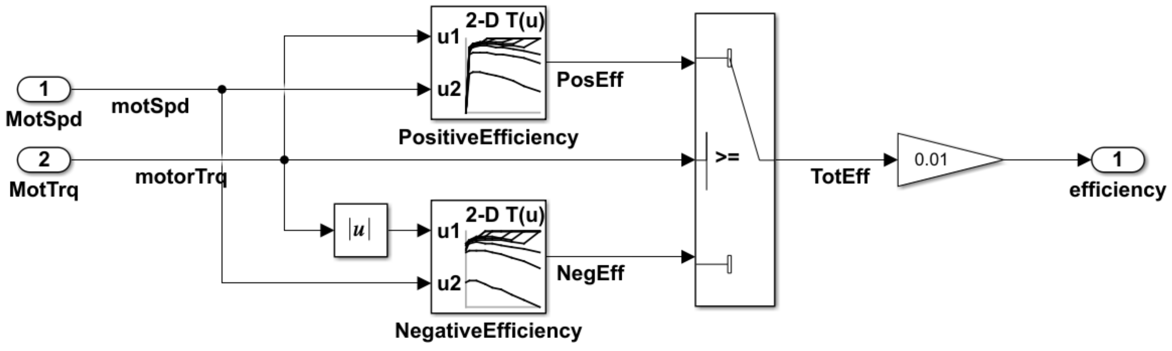


Figure 31: Motor Efficiency Determination

The efficiency that is passed to the out port is dependent on whether the motor is providing positive (motoring) or negative (generating) torque. If the motor is generating, then the generating efficiency (Negative Efficiency) signal (NegEff) is passed. Otherwise, the positive efficiency (PositiveEfficiency) signal (PosEff) is passed.

To arrive at the “battery torque,” the maximum battery powers, either charging (Max_Batt_Charge_Pwr) or discharging (Max_Batt_Discharge_Pwr) - are divided by the motor speed (MotSpd_radps) as shown in Figure 32.

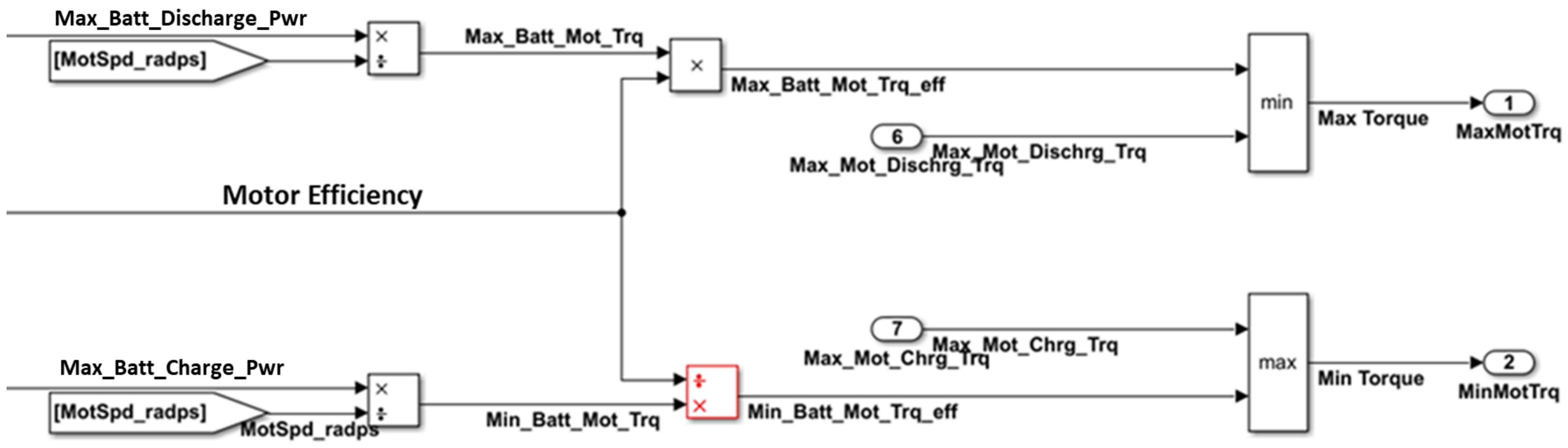


Figure 32: Maximum and Minimum Motor Torque

Thus, we have a maximum and minimum battery torque (Max_Batt_Mot_Trq and Min_Batt_Mot_Trq respectively). Motor efficiency is then applied to the maximum battery torque. For the discharge torque (Max_Batt_Mot_Trq), the motor efficiency is multiplied, for the charge torque (Min_Batt_Mot_Trq), the motor efficiency is divided.

The final maximum motor discharge torque ('MaxMotTrq') is then the minimum between the maximum motor torque (Max_Mot_Discharge_Trq) and the maximum battery torque (Max_Batt_Mot_Trq_eff). The maximum motor charge torque (MinMotTrq) is then the maximum between the maximum motor charging torque ('Max_Mot_Chrg_Trq') and the minimum battery torque (Min_Batt_Mot_Trq_eff).

Thus, the maximum and minimum motor torque available to be commanded reflect the limits of the battery.

3.2.3. ECMS Algorithm Implementation

Now that the maximum motor torques are known, the ECMS algorithm can be implemented. The full implementation is shown in Figure 32.

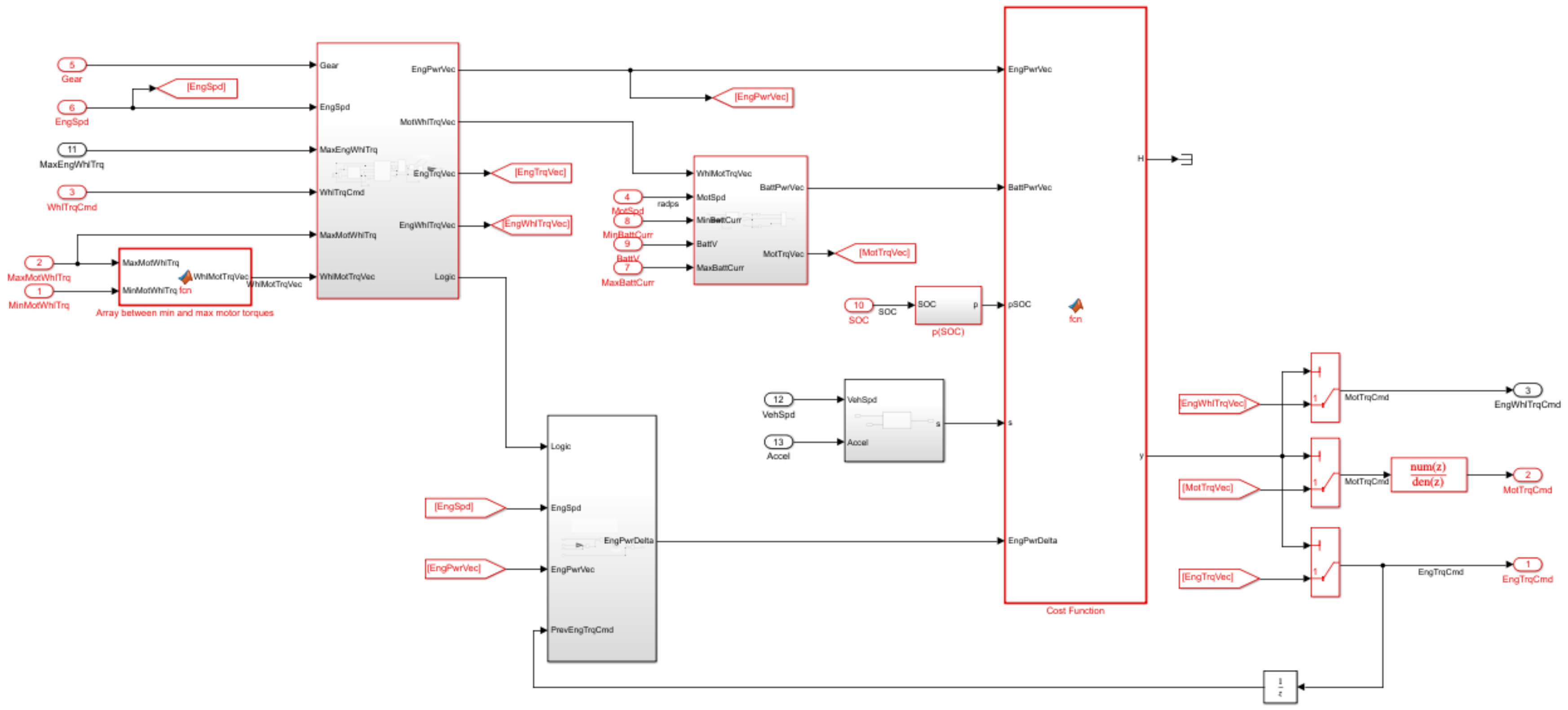


Figure 33: A-ECMS Implementation

Inputs to the ECMS algorithm are: transmission gear (Gear), engine speed (EngSpd), wheel torque command (WhlTrqCmd), maximum motor wheel torque (MaxMotWhlTrq), minimum motor wheel torque (MinMotWhlTrq), maximum engine wheel torque (MaxEngWhlTrq), motor speed (MotSpd), minimum battery current (MinBattCurr), maximum battery current (MaxBattCurr), battery voltage (BattV), SOC (SOC), and vehicle speed (VehSpd) and acceleration (Accel). Outputs of the ECMS algorithm are motor torque command (MotTrqCmd), engine torque command (EngTrqCmd), and engine wheel torque command (EngWhlTrqCmd). The actual ECMS equation is contained in the MATLAB function block labeled “Cost Function”. The ECMS equation requires only 5 inputs: engine power vector (EngPwrVec), battery power vector (BattPwrVec), SOC penaly (pSOC), equivalence factor (s), and a vector of engine power differences (EngPwrDelta).

In the ECMS implementation, a vector of available operating points are made for the motor. This vector consists of 100 evenly spaced elements ranging between the minimum and maximum motor wheel torques. This vector is subtracted from the wheel torque command to arrive at a vector of potential operating points for the engine. At this point, there are two vectors of component torques. Each pair of elements, with the same index, in the two vectors add up to equal the wheel torque command.

At this point in the logic, a potential problem is that some elements in the vector of engine wheel torques might exceed the current operating capabilities of the engine. The operating points in engine wheel torque vector (WhlEngTrqVec) must be evaluated and clipped at the actual maximum torque limits of the engine (MaxEngWhlTrq). This is done using dynamic saturation blocks (Figure 34).

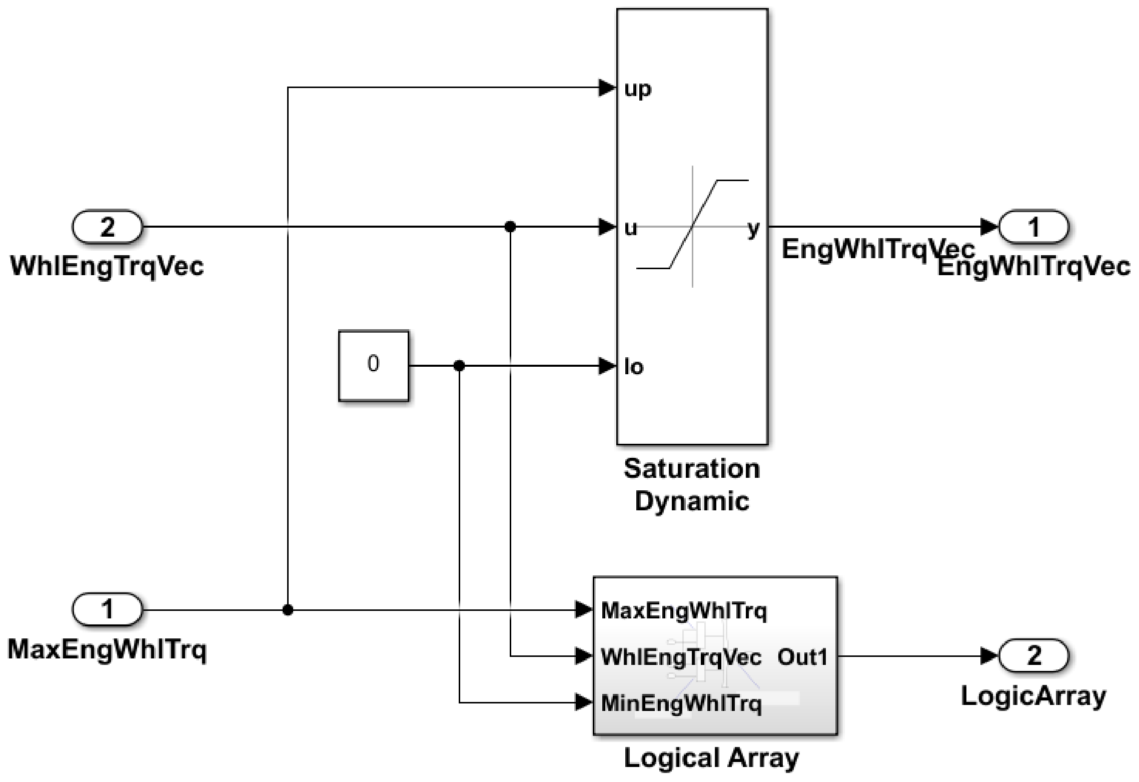


Figure 34: Engine Operating Limits

In the Logical Array subsystem, the maximum engine wheel torque (MaxEngWhlTrq) is compared to each element in engine wheel torque vector (MaxEngWhlTrq). From this comparison, a logic array is created (LogicArray). The logic array only consists of ones and zeros. If any element value in the engine wheel torque vector is greater than the maximum engine wheel torque, then the corresponding element in LogicArray is a zero, else, the corresponding element in LogicArray is a one. This logic array is used further downstream in the algorithm to prevent the ECMS algorithm from calculating the costs of and selecting infeasible operating points for the engine and motor.

At this stage, there is a potential problem in the implementation. With the engine having been constrained to its allowable operation conditions, it is possible that the wheel torque command may no longer be able to be produced. To account for this, logic is added as shown in Figure 35.

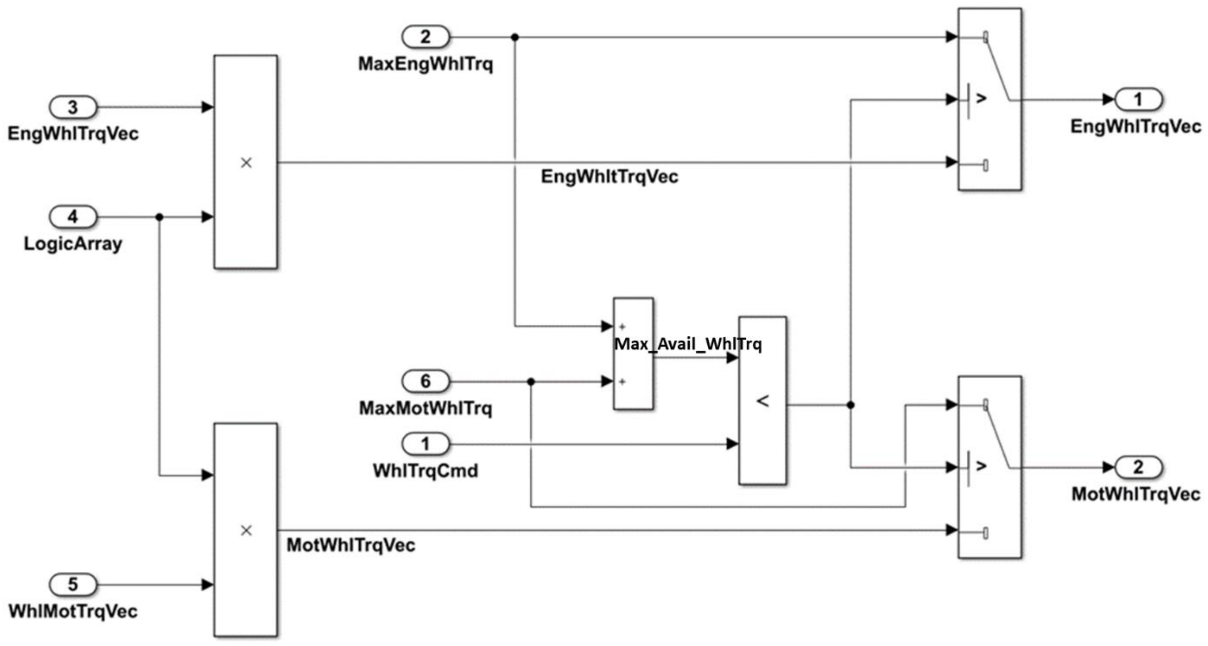


Figure 35: Additional Constraints Logic

In this logic, two things happen, the engine and motor wheel torque vectors (EngWhlTrqVec and MotWhlTrqVec) are multiplied by 'LogicArray' to arrive at two new engine and motor wheel torque vectors that have element values of zero where an operating point is not valid. The wheel torque command is compared to the combination of maximum engine and motor wheel torques, representing the maximum available wheel torque to the vehicle (Max_Avail_WhlTrq). If the wheel torque command (WhlTrqCmd) is greater than the combined maximum engine and motor wheel torques (Max_Avail_WhlTrq), then the maximum motor and engine wheel torques (MaxEngWhlTrq and MaxMotWhlTrq) are passed through the switches as shown in Figure 35. This means that if the wheel torque command is greater than the combination of the maximum engine and motor wheel torques, then the maximum engine and motor wheel torques will be the only elements in engine and motor wheel torque vectors. Thus, the maximum allowable torques are supplied to try and meet the diver demanded wheel torque.

The engine wheel torque vector (WhlTrq) is converted to an engine torque vector (EngTrqVec) by stepping through the transmission gear and differential gear ratios. The engine torque vector is then fed through a brake torque lookup table and fuel flow table to arrive at a vector of engine fuel flow rates. This vector is then multiplied by the lower heating value of the fuel to arrive at a vector of engine powers (EngPwrVec) as shown in Figure 36.

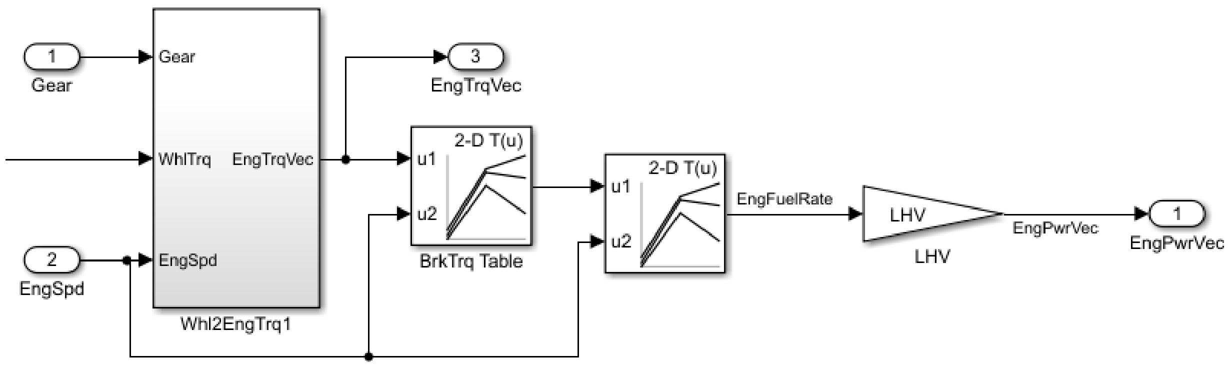


Figure 36: Creation of Engine Power Vector

Next, the vector of motor wheel torques is converted to a vector of available battery powers. This logic is shown in Figure 37.

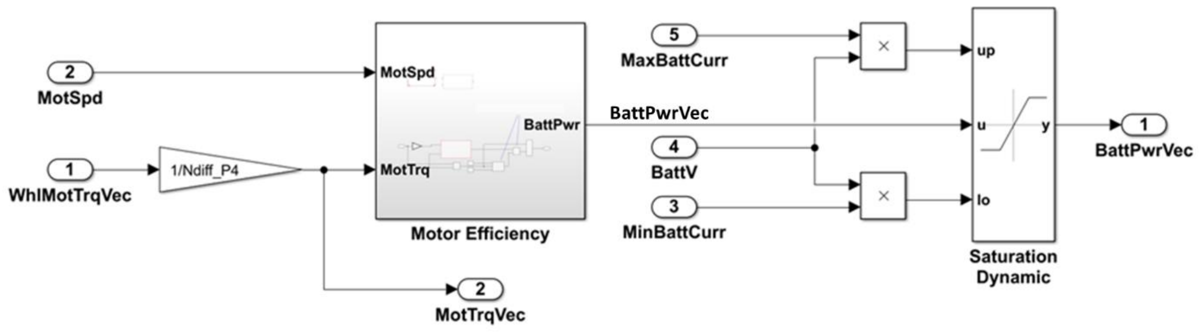


Figure 37: Battery Power Vector Calculation

the motor wheel torque vector (WhlMotTrqVec) is converted to a component torque vector through division of the motor's differential gear ratio (Ndiff_P4). The motor efficiency is used to obtain the battery power vector (BattPwrVec). The battery power vector is dynamically saturated with the maximum and minimum battery powers. The minimum and maximum powers are determined by multiplying the operating battery voltage (BattV) by the maximum and minimum battery currents (MaxBattCurr and MinBattCurr). The resulting vector (BattPwrVec) contains the allowable powers that the battery can produce.

At this point, we have two of the five inputs necessary to define the ECMS equation: an engine and battery power vector. The algorithm still needs the equivalence factor (s), engine power difference vector (EngPwrDelta), and the SOC penalty factor (pSOC).

The engine power difference vector is calculated as shown in Figure 38. This logic was provided by MathWorks. The purpose of the "EngPwrDelta" vector is to assign higher costs to engine torque values that differ significantly from the last commanded engine torque. This helps prevent drastic oscillations in the commanded engine torque.

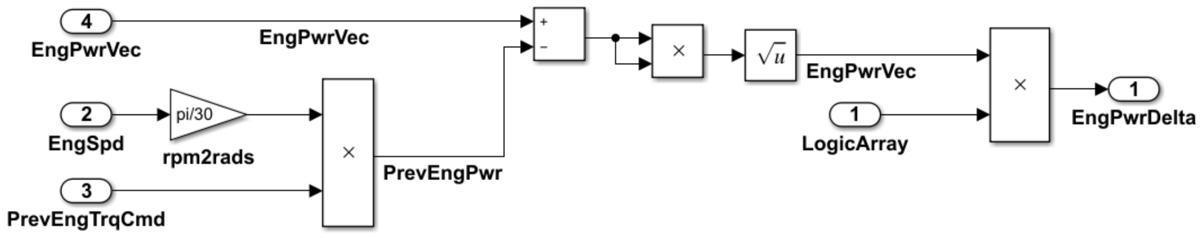
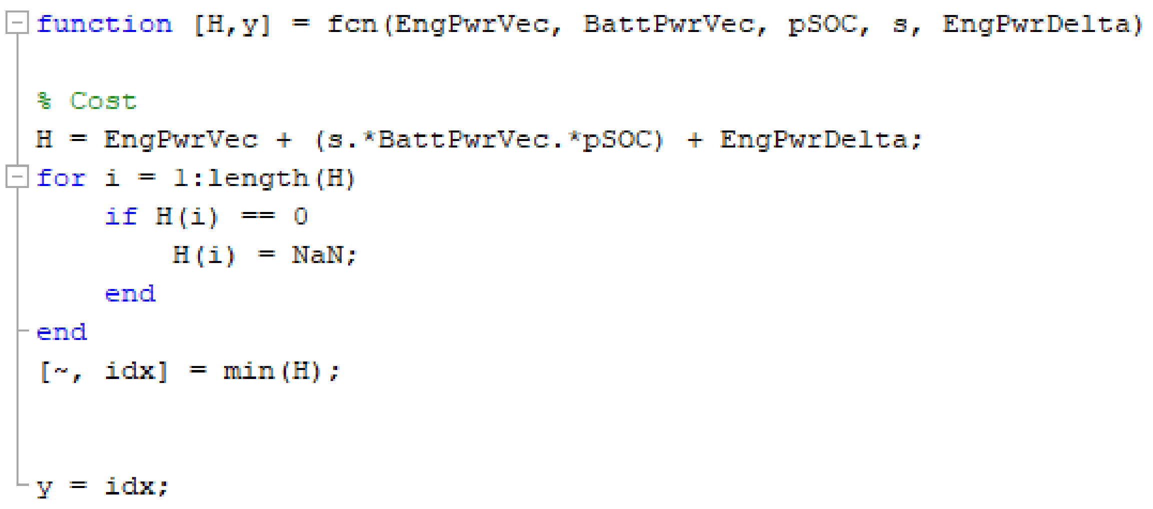


Figure *38: Engine Torque Rate-Limiter Logic*

The previous engine torque command (PrevEngTrqCmd) is converted to a power by multiplying by the engine speed (EngSpd) in radians per second. The resulting signal (PrevEngPwr) is subtracted from the current engine power vector (EngPwrVec). The absolute value of this difference is multiplied by the "LogicArray" signal to arrive at the engine power delta vector (EngPwrDelta). This vector is added to the overall ECMS equation to increase the cost of selecting engine torques that vary largely from the previously commanded torque.

Next, the penalty factor (pSOC) based on the battery SOC is created. The penalty factor is assigned by implementing the curve given by Equation (2-12). Once this is done, the equivalence factor (s) is ready to be defined. In ordinary ECMS, the equivalence factor is simply a static value.

In this model, the final ECMS equation is implemented using a MATLAB function block. The function is shown below in Figure 39.



```
function [H,y] = fcn(EngPwrVec, BattPwrVec, pSOC, s, EngPwrDelta)

% Cost
H = EngPwrVec + (s.*BattPwrVec.*pSOC) + EngPwrDelta;
for i = 1:length(H)
    if H(i) == 0
        H(i) = NaN;
    end
end
[~, idx] = min(H);


y = idx;
```

Figure 39: MATLAB Function for ECMS Algorithm

In the function, the cost, "H", is the implementation of the ECMS (Equation 2-16). "H" is a vector of costs. The minimum of "H" is determined using the implicit MATLAB "min" function, and the associated element index of the minimum cost is defined as "idx". To avoid selecting the index where the cost is associated with the zero elements of the vectors, all zero values of the cost function vector "H" are set to NaN. The "min" function ignores NaN values, and will therefore only select elements that represent feasible operating points.

The index value "idx" is assigned to variable "y" and output from the function block as shown in Figure 40.

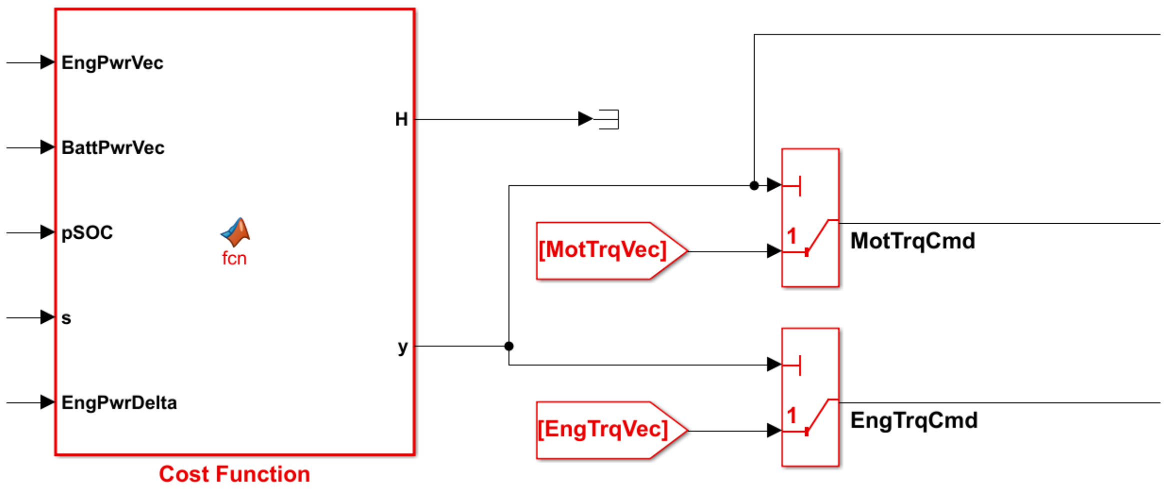


Figure 40: ECMS MATLAB Function Block and Component Torque Selections

The index “y” is fed into a switch block that also receives the engine and motor torque vectors (MotTrqVec and EngTrqVec). The output of the switches is the component torque (MotTrqCmd and EngTrqCmd) associated with the index “y”. This is the implementation of the ECMS algorithm.

4. Artificial Neural Network (ANN) Description/Implementation

A radial basis function (RBF) ANN is used to implement the adaptive portion of the ECMS algorithm. From this point forward, the ANN implementation with ECMS will be called ANN-ECMS. The RBF method was chosen because it can be trained very quickly by exposure to the entire set of training data at once. This is unlike other ANN methods that are trained with one data set at a time, which can take considerable time. The RBF ANN consists of a single hidden layer and an output layer. The weights between the hidden and output layer are updated during training. The structure of the RBF ANN is shown in Figure 41.

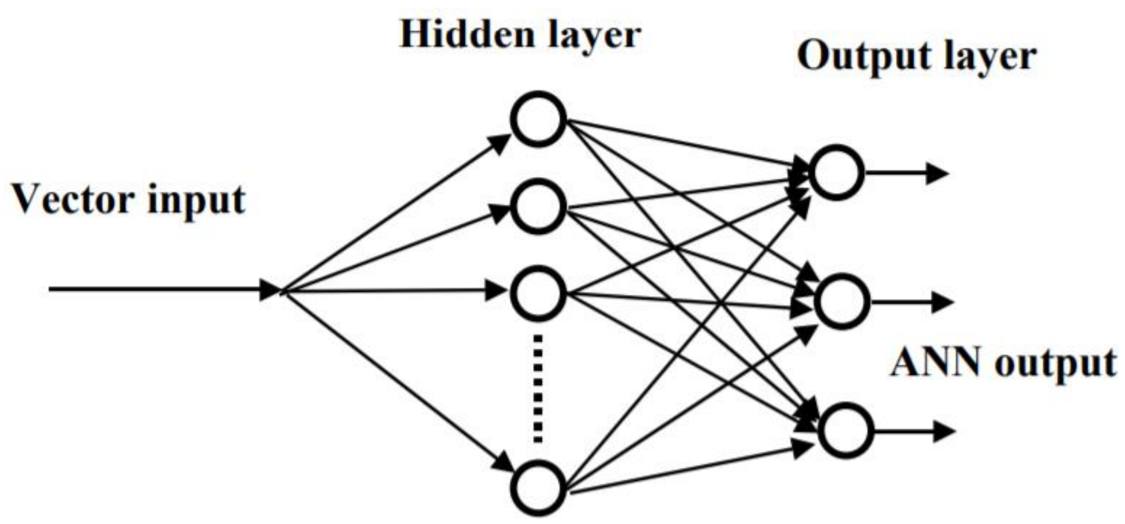


Figure 41: Single Hidden Layer ANN Diagram [35]

Given a non-linear function, it is approximated as the weighted sum of a few non-linear functions known as basis functions:

$$f(\bar{x}) \approx \sum_{k} w_k \varphi_k(\bar{x}) \tag{3-1}$$

Where $\bar{x}$, is a set of input training data, $f(\bar{x})$ is the function approximation, w_k are the weights, and $\varphi_k(\bar{x})$ is the basis function.

The vector inputs are not used directly in the basis function. A set of “centers” are defined and the distance between $\bar{x}$ and the “centers” are used as the inputs to the basis function [35]. A diagram of a hidden layer neuron of the RBF ANN is shown in Figure 42.

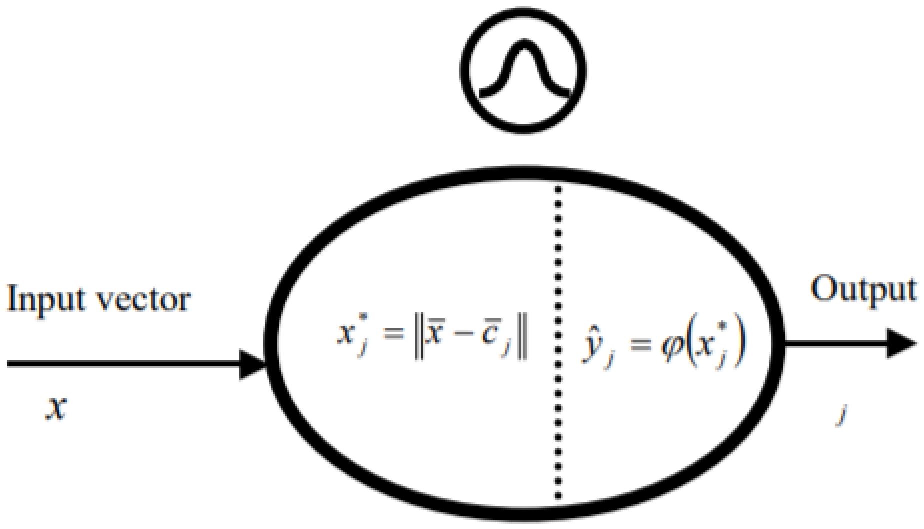

Input vector
x
$x_j^* = \|\bar{x} - \bar{c}_j\|$
$\hat{y}_j = \varphi(x_j^*)$
Output
j

Figure 42: RBF Neuron in the Hidden Layer [35]

Where x_j^* is the distance between the input vector $\bar{x}$ and the center $\bar{c}_j$ for the jth hidden layer neuron. A center vector $\bar{c}_j$ is defined for each neuron in the hidden layer. The center vector can be defined arbitrarily, or it can be made equal to the training data itself. In this work, the center vectors were made equal to the training data.

The basis function is given by:

$$\varphi(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-x^2}{2\sigma^2}} \tag{3-2}$$

Where σ^2 is the Gaussian distribution variance. This is an internal parameter of the ANN and can be constant for each neuron in the hidden layer, or it can be defined explicitly for each neuron.

The output of the RBF ANN is the sum of the multiplication of the weights and the outputs of each hidden neuron (Figure 43).

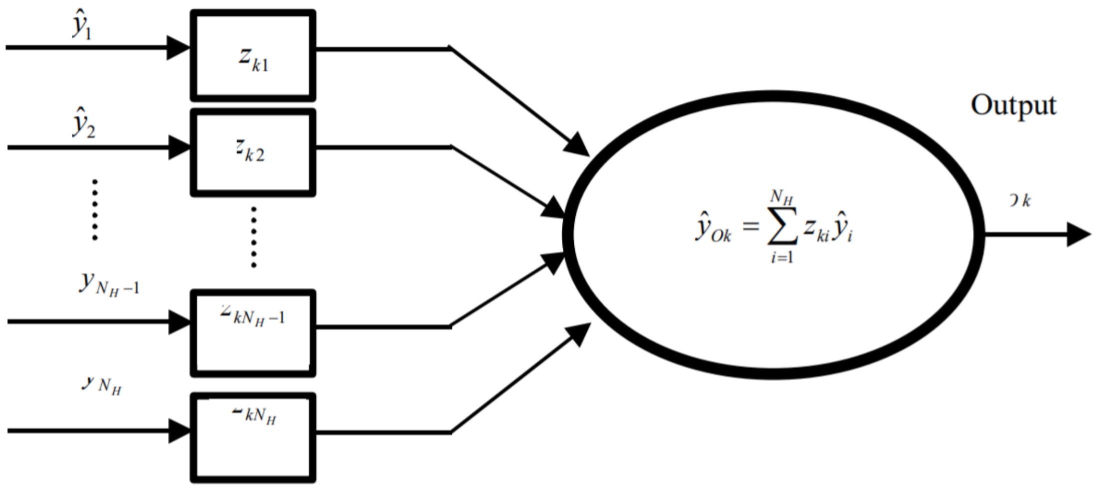


Figure 43: Output Neuron in RBF ANN [35]

Where $z_{1\ldots N_H}$ are the weights associated with each hidden layer and O_k is the output of the kth output neuron.

The variance has a large impact on the behavioral characteristics of the ANN. Particularly, the variance affects the interpolation and extrapolation properties of the ANN. Figure 44 represents a single neuron in the hidden layer of the ANN. The input vector - x - goes from 1 to N, where N is the number of inputs to the ANN. There are the same number of center vectors as there are inputs ($\bar{c}_2$ ... $\bar{c}_N$).

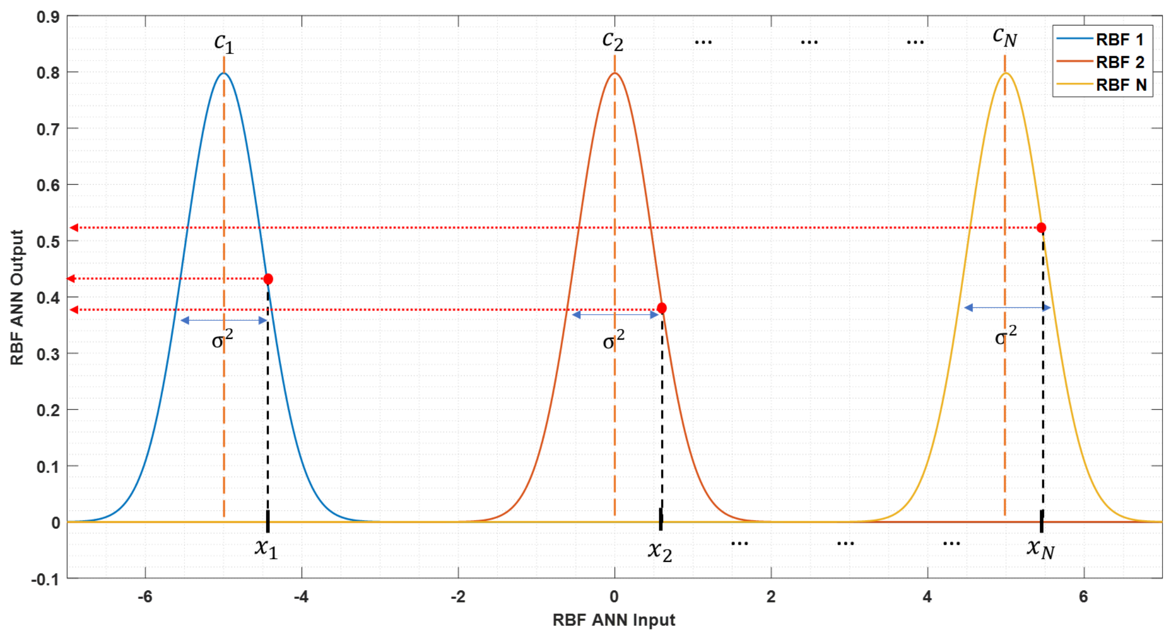


Figure 44: Well-Bounded Set of Input Data to RBF ANN

Figure 44 shows that when the elements of the input vector are near the centers, the corresponding output values will be non-zero. However, if the inputs are far from the center vectors and the variance is small, then the output of the RBF tends to zero. The variance for each neuron in the hidden layer can be uniquely defined. However, for this work, the variance is equal for all neurons in the hidden layer. Of course, the output is also dependent on the placement of the center vectors. If the center vectors are close to one another, then a small variance can produce non-zero values. However, if the centers are far apart, then a large variance is needed to achieve non-zero outputs.

The opposite problem could also occur. If the value of the variance is too large, the corresponding outputs will also be too large. A variance value should be selected based on a sensitivity analysis in which multiple values are tested. The variance which produces the most desirable results should be selected for use in the RBF ANN.

The input ($\bar{x}$) to the ANN is an 8-element vector. Each element of the input vector is a characterization of the past driving conditions.

For this work, 30 hidden neurons are used in the ANN. The number of hidden neurons was selected based on the size of the set of training data, which was a set of 30. An equal number of hidden neurons and training sets allows for the inversion of a square matrix when training the ANN. The ANN training is discussed in a later section.

Only one neuron is used in the output layer. There is only one parameter of interest being changed – the equivalence factor. Therefore, only one output neuron is needed.

4.1. Training Data Generation

To train the RBF ANN, a total of 30 drive cycles were evaluated. Each drive cycle is characterized by 9 parameters. For each drive cycle, the optimal equivalence factor which maximizes fuel economy is determined. This optimal equivalence factor is used in conjunction with the drive cycle characteristic parameters to train the ANN. The characteristic parameters of the drive cycle are the inputs to the ANN.

To find the optimal equivalence factor, an array of equivalence factors was tested over each drive cycle. To accurately report the fuel economy, a requirement was imposed that the ending SOC be within +/- 1% of the starting SOC. There are existing methods used to relate a delta SOC over a drive cycle by converting from electrical energy to fuel energy, but since the HV battery has a relatively low capacity and a purely electric-only mode is not modeled, the bounded SOC condition was used. To achieve the SOC balance, each equivalence factor value was used in a cyclically repeating drive cycle – the drive cycle was repeated 3 times for each equivalence factor. At the end of each drive cycle, the ending

SOC was set to be the starting SOC for the next cycle. For instance, if the equivalence factor varied from 0.5 to 0.9 in increments of 0.05, then the drive cycle over which the equivalence factor was being optimized would be run a total of 27 times.

With each run of a drive cycle, all the input parameters (drive cycle characteristics) were saved. The input parameters are listed below:

- Average Acceleration [ga]
- Average Deceleration [ga]
- Average Positive Jerk [$\frac{ga}{s}$]
- Average Negative Jerk [$\frac{ga}{s}$]
- Total Distance [mile]
- Idle Time [sec]
- Average Speed [$\frac{m}{s}$]
- Maximum Speed [$\frac{m}{s}$]

It should be noted, that from this point forward, the input parameters are often referred to as the drive cycle characteristics. Drive cycle characteristics are a reference to the input parameters listed above.

In post-processing the data from each drive cycle, those equivalence factors which were either too low or too high to achieve charge sustainability were ignored. Out of those equivalence factors which achieved charge sustainability, those which achieved the highest fuel economy were selected to use in the training data. The input parameters associated with those equivalence factors were also selected to use as training data.

To increase the hyperspace of the training data, two different driver models were used. One driver model used a fast response time (normal driver), which resulted in the driver closely following the drive trace. The other driver used a slow response time (smooth driver), resulting in smaller acceleration and jerk values.

The normal driver follows the drive trace more closely, resulting in more aggressive accelerations, producing greater average acceleration and jerk values. Figure 45 shows a section of the HUDDS cycle using the normal driver. This figure shows a close match between the reference velocity and the vehicle velocity. The driver closely follows the reference trace – capturing the acceleration and jerk values implicit to the drive cycle. Figure 46 shows a linear regression plot of the normal driver over the entire HUDDS drive cycle.

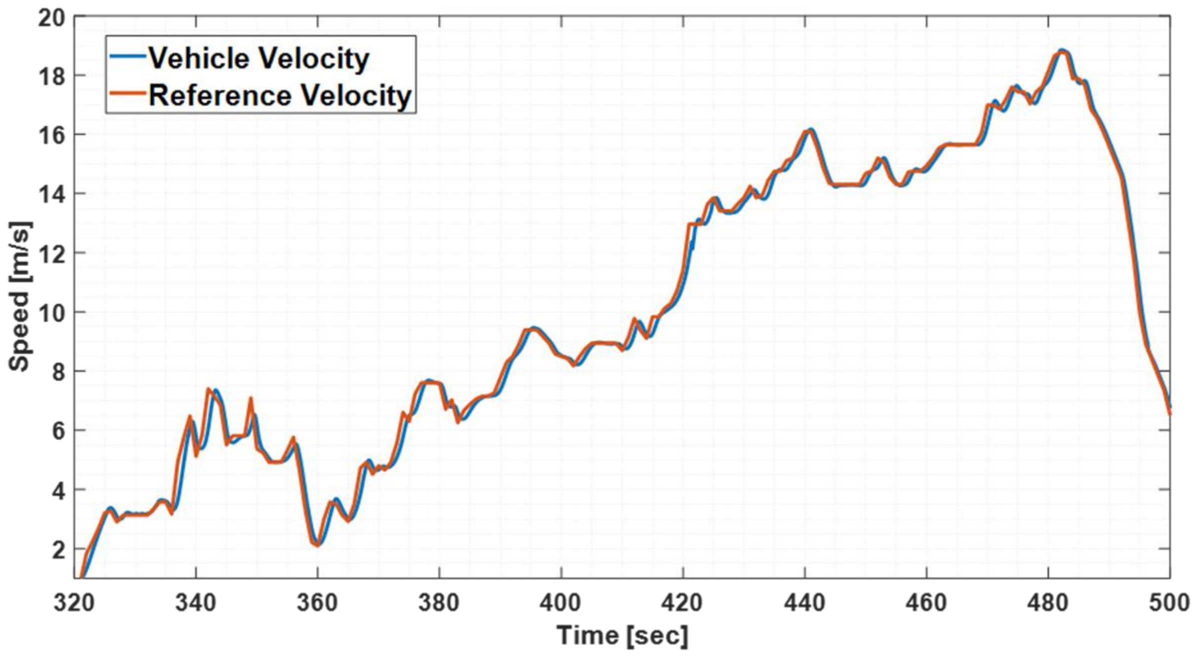


Figure 45: Normal Driver Speed Trace

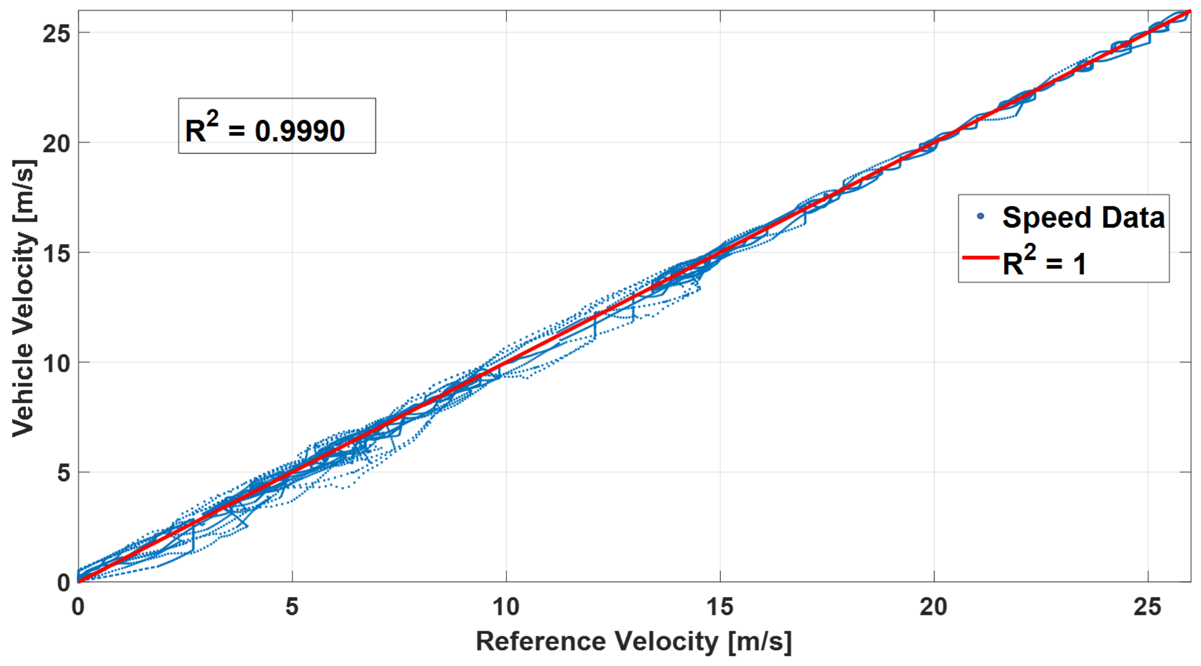


Figure 46: Linear Regression of Normal Driver

Figure 47 shows the same section of the HUDDS cycle using the smooth driver. In this figure, the vehicle velocity does not follow the reference velocity as rigorously as the normal driver. Instead, areas of rapid speed change in the drive cycle are smoothed over by the driver. This results in lower values of acceleration and jerk when compared to the rough driver. Figure 48 shows a linear regression plot of the smooth driver over the entire HUDDS drive cycle.

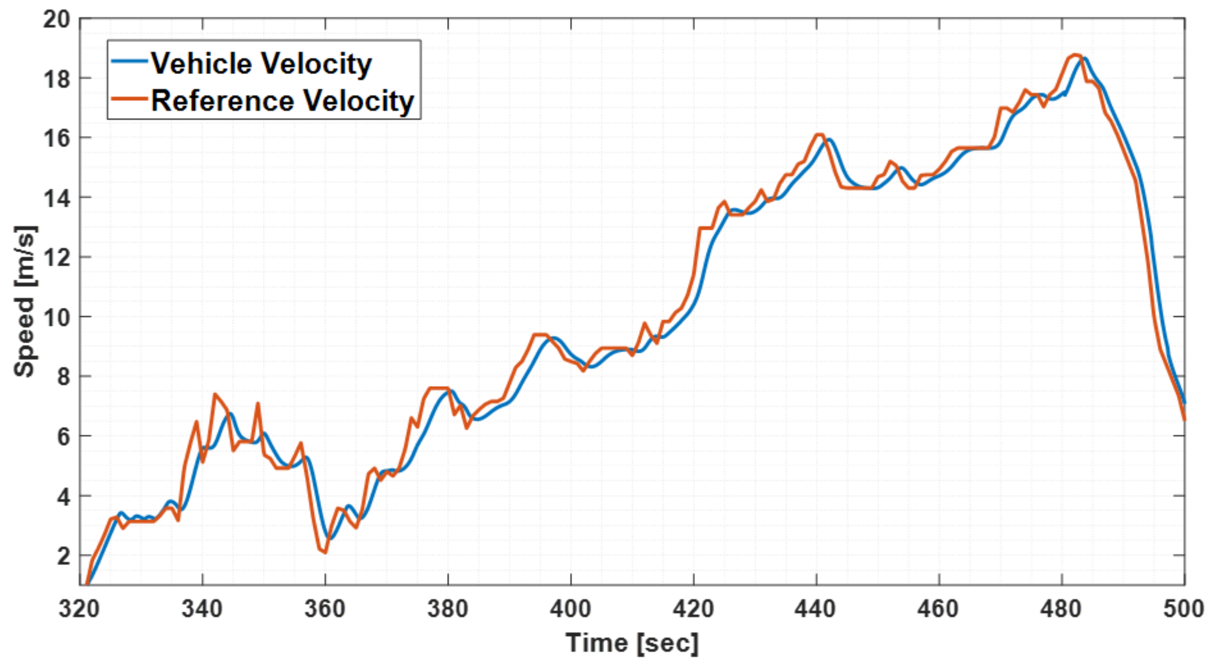


Figure 47: Smooth Driver Speed Trace

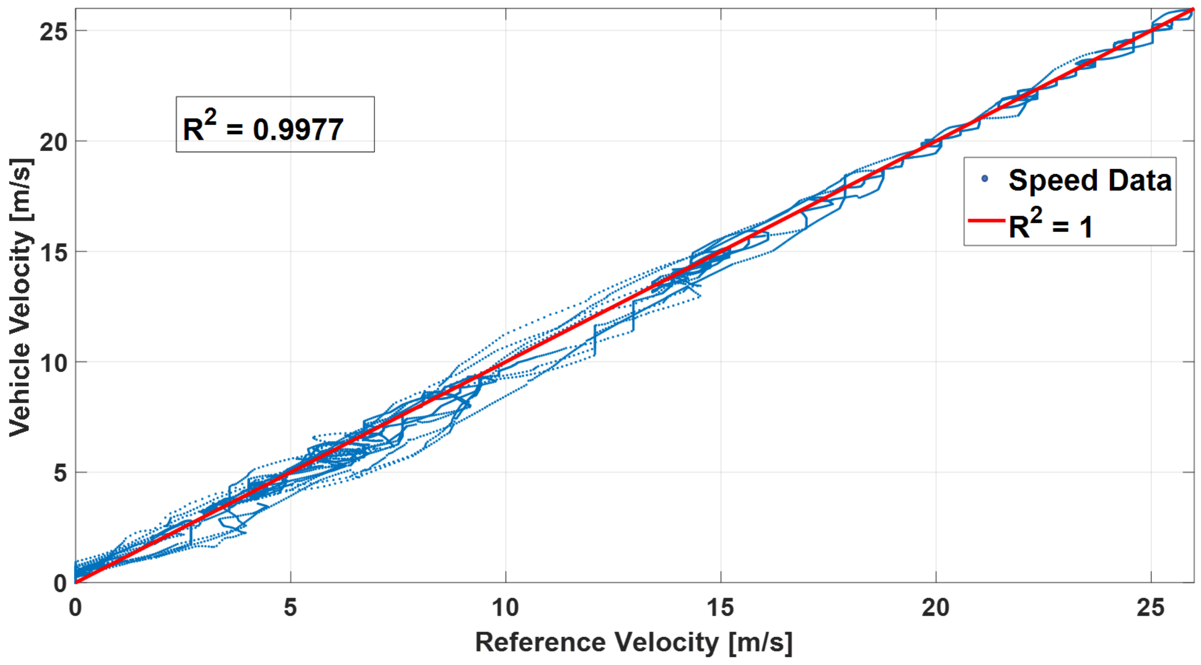


Figure 48: Linear Regression of Smooth Driver

Figure 48 shows an R^2 value of 0.9977, which is lower than the R^2 value of the normal driver (0.9990). This indicates that the smooth driver deviates from the drive cycle more than the normal driver.

The significance of having two different drivers is that a single drive trace can result in two different sets of training data with different parameter characteristics.

Table 6 and Table 7 show all the drive cycles that were evaluated. For each drive cycle, the inputs are displayed, along with the associated equivalence factor and fuel economy. The drive cycles selected for training vary in length, speed, acceleration, and idle time. A wide range of characteristics were desired to capture as much of the input hyperspace as possible.

The drive cycles can be characterized as city, highway, or an amalgamation of both. City cycles are characterized by sporadic speeds, aggressive accelerations, high idle times, and relatively low average speeds. Conversely, highway cycles are characterized by more consistent speeds, passive accelerations, little to no idle time, and higher average speeds. The advantage of using two different driver models to evaluate the same drive cycles is that implicit cycle characteristics like speed and idle time can be preserved, while the acceleration and jerk can be varied. For example, a city cycle with aggressive accelerations, low speeds, and high idle times, can be evaluated using both drivers. The normal driver will capture the true acceleration and jerk of the cycle, while the smooth driver will preserve the speeds and idle times but will change the acceleration and jerk. The smooth driver makes a city cycle more characteristic of a highway cycle – essentially creating a new drive cycle. This is how the hyperspace of the training data is able to be expanded.

Table 6: Drive Cycle Characteristics – Normal Driver

Drive Cycle	**Average Acceleration [ga]**	**Average Deceleration [ga]**	**Average Positive Jerk [$\frac{ga}{s}$]**	**Average Negative Jerk [$\frac{ga}{s}$]**	**Total Distance [mile]**	**Idle Time [sec]**	**Average Speed [$\frac{m}{s}$]**	**Maximum Speed [$\frac{m}{s}$]**	**Equivalence Factor**	**Fuel Economy [mpg]**
Artemis Rural Road	0.023	-0.023	0.032	-0.032	11.781	25.455	17.309	32.424	0.9	38.164
Artemis Urban	0.034	-0.034	0.077	-0.078	4.165	22.664	6.647	18.631	0.9	24.768
Braunschweig City Driving Cycle	0.027	-0.027	0.054	-0.054	8.185	109.166	7.484	18.223	0.6	30.843
City Suburban Heavy Vehicle Cycle (CSC)	0.021	-0.021	0.046	-0.046	8.306	64.167	7.771	21.645	0.6	34.720
ECE Extra-Urban Driving Cycle (Low Powered Vehicles)	0.010	-0.010	0.013	-0.013	4.520	42.346	17.970	27.021	0.7	41.272

EUDC	0.012	-0.012	0.015	-0.015	4.691	42.773	18.640	35.276	0.9	37.486
FTP72	0.026	-0.026	0.047	-0.048	7.846	301.031	9.098	26.648	0.6	35.847
FTP75	0.019	-0.019	0.035	-0.036	11.608	1013.081	7.486	26.744	0.9	32.496
HUDDS	0.017	-0.017	0.037	-0.037	7.158	28.337	10.742	29.192	0.9	34.175
Japanese 10 Mode	0.019	-0.018	0.024	-0.023	0.467	38.842	5.514	12.341	0.7	31.646
Japanese 15 Mode	0.014	-0.014	0.014	-0.014	1.480	77.258	10.222	20.965	1.0	37.211

Japanese 10-15 Mode	0.020	-0.020	0.040	-0.040	3.232	46.245	7.788	21.826	0.6	33.643
US06	0.032	-0.032	0.031	-0.032	8.824	9.902	23.455	38.534	0.9	26.837
Business Arterial Commuter (BAC) – Arterial Cycle	0.028	-0.028	0.023	-0.023	2.074	39.923	12.132	19.542	0.6	31.642
Business Arterial Commuter (BAC) – Commuter Cycle	0.008	-0.008	0.006	-0.006	4.217	27.828	21.598	25.955	0.5	37.037
Central Business District (CBD) Cycle	0.026	-0.026	0.037	-0.036	2.257	103.752	6.403	10.039	0.9	28.542
HWFET	0.009	-0.009	0.007	-0.007	10.981	11.036	22.900	28.341	0.6	40.701

New York Composite Cycle	0.018	-0.018	0.034	-0.034	3.499	78.552	5.418	18.469	0.9	25.897

Table 7: Drive Cycle Characteristics – Smooth Driver

Drive Cycle	**Average Acceleration [ga]**	**Average Deceleration [ga]**	**Average Positive Jerk [$\frac{ga}{s}$]**	**Average Negative Jerk [$\frac{ga}{s}$]**	**Total Distance [mile]**	**Idle Time [sec]**	**Average Speed [$\frac{m}{s}$]**	**Maximum Speed [$\frac{m}{s}$]**	**Equivalence Factor**	**Fuel Economy [mpg]**
Business Arterial Commuter (BAC) – Arterial Cycle	0.027	-0.028	0.022	-0.022	2.056	35.936	12.026	19.496	0.9	30.452
Business Arterial Commuter (BAC) – Commuter Cycle	0.008	-0.008	0.006	-0.006	4.245	27.109	21.734	26.160	1.1	36.481
Central Business District (CBD) Cycle	0.024	-0.024	0.034	-0.033	2.187	87.638	6.202	9.837	0.6	29.534

ECE Extra-Urban Driving Cycle (Low Powered Vehicles)	0.008	-0.008	0.006	-0.006	4.453	41.478	17.825	26.692	0.8	41.726
EUDC	0.010	-0.010	0.006	-0.005	4.656	41.829	18.638	34.961	0.9	37.711
HWFET	0.009	-0.009	0.006	-0.006	10.853	11.385	22.644	28.063	0.8	40.767
Japanese 15 Mode	0.013	-0.013	0.010	-0.010	1.471	77.118	10.180	20.921	0.9	38.940
Japanese 10 Mode	0.025	-0.024	0.063	-0.060	0.514	38.128	6.006	13.207	0.7	32.292
Japanese 10-15 Mode	0.015	-0.015	0.018	-0.017	3.016	54.429	7.304	21.223	0.7	33.778

LA92Short	0.024	-0.024	0.024	-0.024	7.278	173.079	12.004	30.671	0.9	30.593
New York Composite Cycle	0.014	-0.013	0.023	-0.022	3.251	174.688	5.043	17.850	0.6	30.089
US06	0.029	-0.029	0.024	-0.024	8.679	10.249	23.104	37.907	0.9	27.564

Not all of the drive cycles evaluated with the normal driver were evaluated with the smooth driver. For the smooth driver, most of the dive cycles selected had small distances. Since most of the inputs involve an average, there would be little difference between averages in long drive cycles. Also, since the sliding time window will not be long, the input distances will be short. Therefore, most of the drive cycles selected for the smooth driver are relatively short.

4.2. Risks

There is a high risk associated with the outlined approach. If the equivalence factor is examined and updated based on past driving conditions, there is no guarantee that it will be optimal for future driving conditions. The underlying assumption is that driving conditions will remain relatively consistent over a window of a few minutes. Also, if the driving conditions do change, there will only have been a few minutes over which the "optimal" value was not being applied.

The other risk is that the input parameters to the ANN will violate the hyperspace of inputs used to train the ANN. A violation of the hyperspace will result in the ANN performing extrapolation, which can produce undesirable results. This is why it is important to cover as much hyperspace with the training data as possible.

Despite the risk involved, the approach is still worthy of investigation. There is still potential for good results.

4.3. RBF Training

After the training data was generated, the ANN was trained. Training was performed in a single step by exposing the ANN to all of the training data at once. First, a matrix of distances between each input, $\bar{x}_i$ and each center $\bar{c}_j$ was defined. Where $i = 1,2,3, \dots N_K$ and $j = 1,2,3, \dots N_H$. N_K is the number of training data sets, and N_H is the number of hidden neurons.

The vectors $\bar{x}_i$ and $\bar{c}_j$ are used to create a matrix of differences:

$$D = \{d_{ji}\} = \begin{bmatrix} \|\bar{x}_1 - \bar{c}_1\| & \|\bar{x}_2 - \bar{c}_1\| & \cdots & \|\bar{x}_{N_K} - \bar{c}_1\| \\ \|\bar{x}_1 - \bar{c}_2\| & \|\bar{x}_2 - \bar{c}_2\| & \cdots & \|\bar{x}_{N_K} - \bar{c}_2\| \\ \vdots & \vdots & \ddots & \vdots \\ \|\bar{x}_1 - \bar{c}_{N_H}\| & \|\bar{x}_2 - \bar{c}_{N_H}\| & \cdots & \|\bar{x}_{N_K} - \bar{c}_{N_H}\| \end{bmatrix} \tag{3-3}$$

The centers $\bar{c}_j$ are selected to be equivalent to the input data sets $\bar{x}_i$. This results in a zero diagonal in the D matrix. The D matrix is used in the activation function to determine the output of the hidden layers:

$$\varphi(D) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-D \otimes D}{2\sigma^2}} \tag{3-4}$$

Where $D \otimes D$ is the element wise product and $\varphi(D)$ will be a N_H x N_K sized matrix.

The product of the output of the hidden layer and the weights are supposed to approximate the training data, so the weights (Z) are determined as follows:

$$Z = [\varphi(D)^T]^{-1} * \bar{y} \tag{3-5}$$

Where $\bar{y}$ is the known output of the training data.

The MATLAB code used to train the ANN can be found in Appendix A.

4.4. RBF ANN Implementation

The on-line RBF ANN implemented into Simulink examines the input vector over a specified time window. At the end of every time window, the equivalence factor is updated.

The average values of acceleration and deceleration are determined from the acceleration signal (Accel) that originates in the plant model from the longitudinal vehicle body model. The acceleration is fed into two switch blocks (Figure 49).

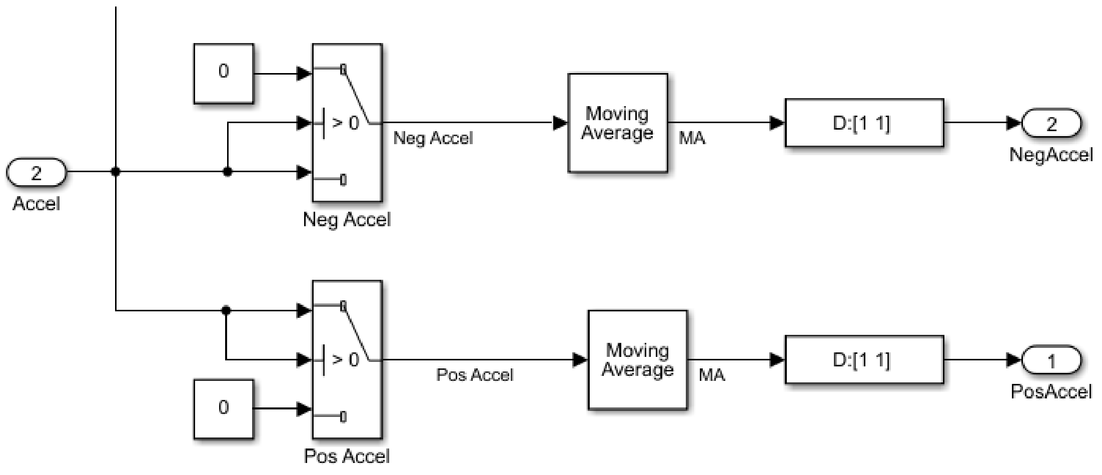


Figure 49: Acceleration/Deceleration Logic

To determine positive acceleration, if the “Accel” signal is positive, then it is passed as positive acceleration (Pos Accel). If “Accel” is negative, then a zero is passed for “Pos Accel”. The same logic is used for determining the deceleration, except the acceleration signal “Accel” is passed if it is negative.

The acceleration (Pos Accel) and deceleration (Neg Accel) signals are fed into a moving average block. The moving average block determines the average over a defined amount of timesteps. The running timestep of the controller model is 25 ms. For example, to determine the average over a window of, say, 2 minutes, the moving average block calculates the average over 4800 timesteps.

The positive jerk (Pos Jerk) and negative jerk (Neg Jerk) values are determined (Figure 50) by differentiating the vehicle acceleration signal and using a switch to separate positive and negative values in exactly the same way as the acceleration and deceleration.

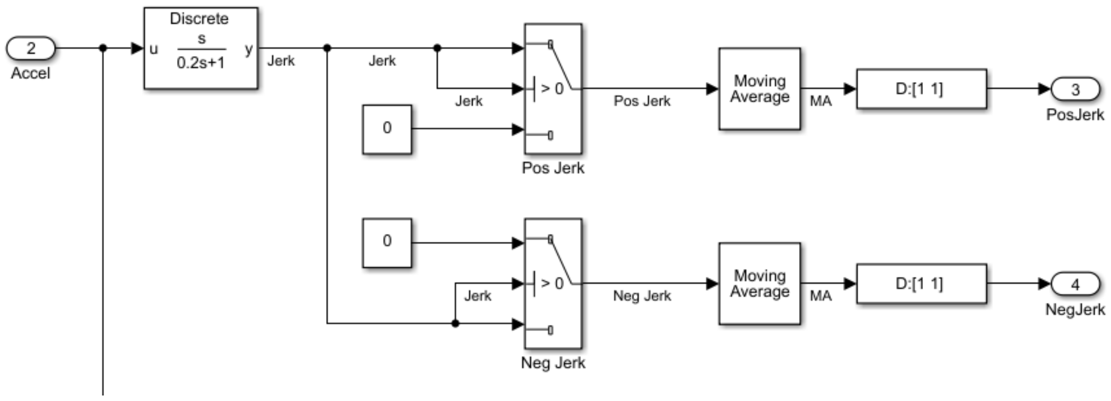


Figure 50: Positive and Negative Jerk Calculation

A filtered derivative is used to differentiate the acceleration signal (Accel). The filtered derivative was used to eliminate noise that was seen when using an ordinary derivative. The filter time constant was selected by comparing the filtered derivative with the ordinary derivative. The time constant was selected such that the filtered derivative mimicked the trend of the ordinary derivative minus the noise.

The average vehicle speed (AvgSpd) is determined using another moving average block (Figure 51). The maximum vehicle speed (MaxVel) is determined using a moving maximum block. The moving maximum block works in the same way as the moving average block, except it determines the maximum value over a number of time steps instead of the average (Figure 51).

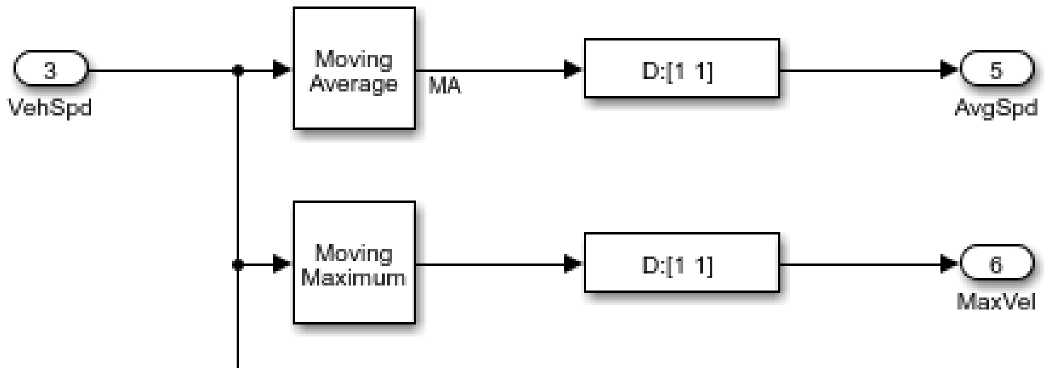


Figure 51: Average Vehicle Speed and Maximum Vehicle Speed

The distance is determined using a discrete-time integrator (Figure 52), which recieves the vehicle speed (VehSpd), and a trigger (gen).

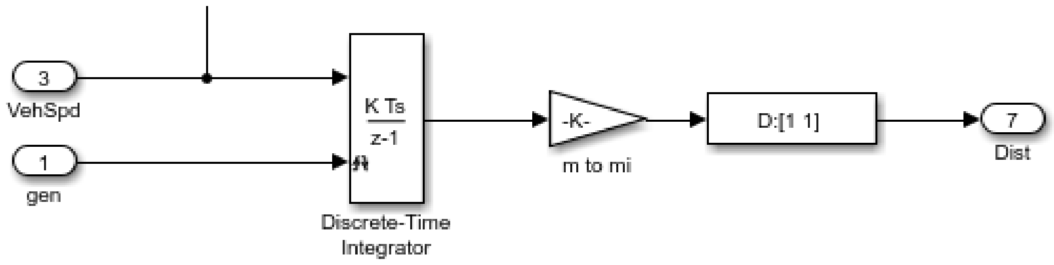


Figure 52: Distance Calculation

The trigger is activated by either a rising or a falling edge. The signal (gen) feeding the trigger is a square wave that rises and falls with a frequency set to match the time of the time window. The distance is then converted from meters to miles using a gain block.

The idle time (StopTime) is also determined using a discrete-time integrator block (Figure 53).

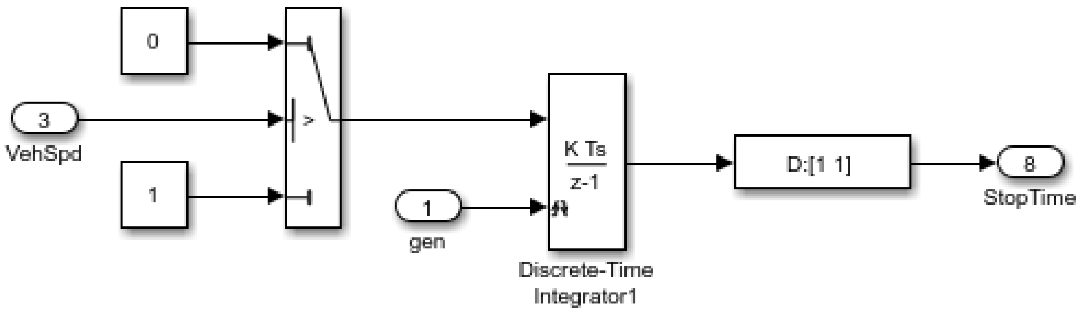


Figure 53: Idle Time Calculation

The vehicle speed (VehSpd) is fed into a switch, which passes a “1” if the speed is below a predefined value. The predefined value indicates when the vehicle is idling. The output of the switch block is fed into the discrete-time integrator that is triggered by the same square wave signal (gen) described in Figure 52. The integration results in a cumulative sum of the idle time, which is reset at the beginning of every new time window.

Once the input signals are all determined, they are combined using a multiplexer block. A sample and hold block is then used to output the inputs once at the beginning of every time window (Figure 54). The sample and hold block is triggered using the rising and falling edge of the “gen” signal described from Figure 52.

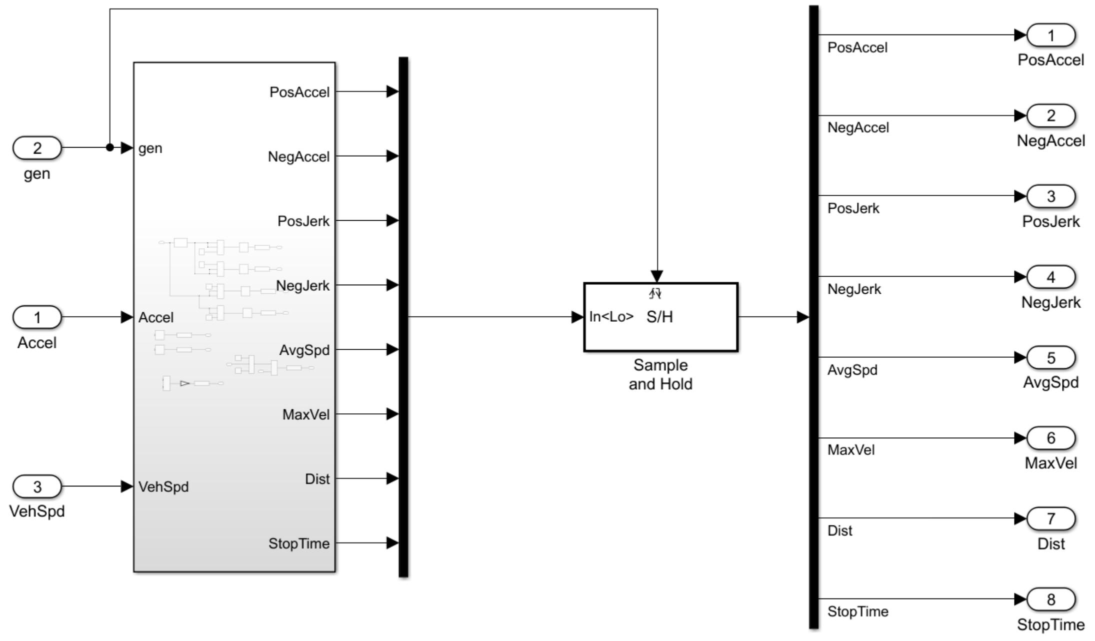


Figure 54: Sample and Hold of Input Signals

With the sample and hold block, the input signals stay constant over a period equal to the specified time window. After the signal is sampled and held, the signal is de-multiplexed back into its constituent inputs. Figure 55 shows a sample input of maximum vehicle speed over the course of a drive cycle. This figure shows how the input value remains constant over the specified time window. In this figure, the time window was set to be 3 minutes.

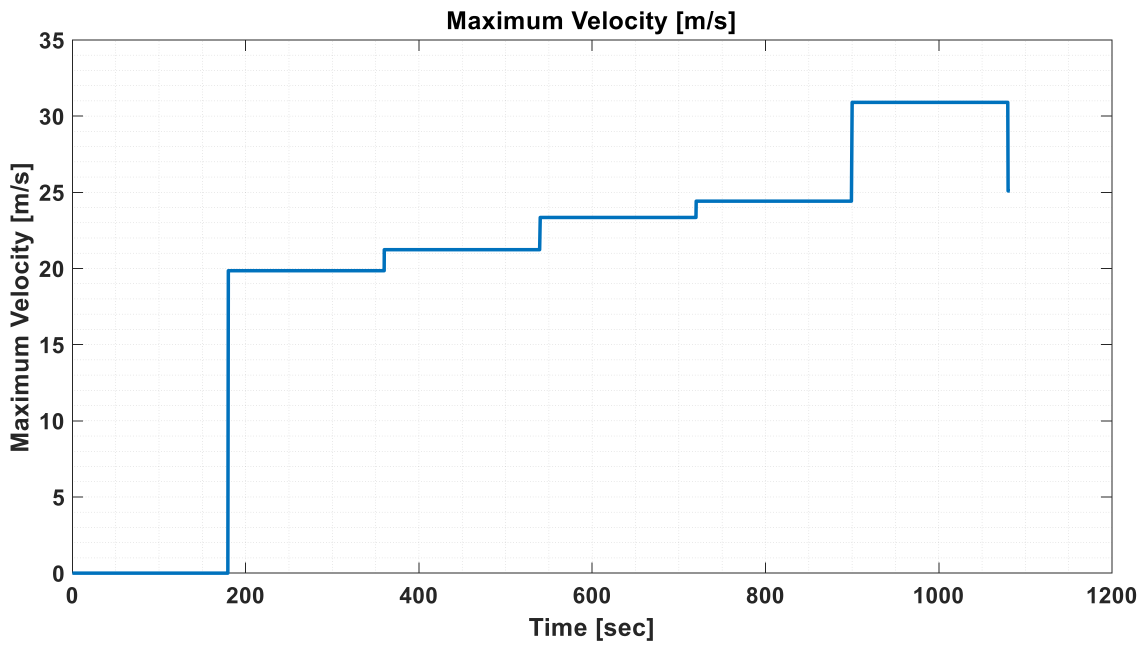


Figure 55: RBF ANN Maximum Velocity Input

The inputs are fed to a function block that implements the RBF ANN (Figure 56). The output of the function block is the equivalence factor (equiv_factor).

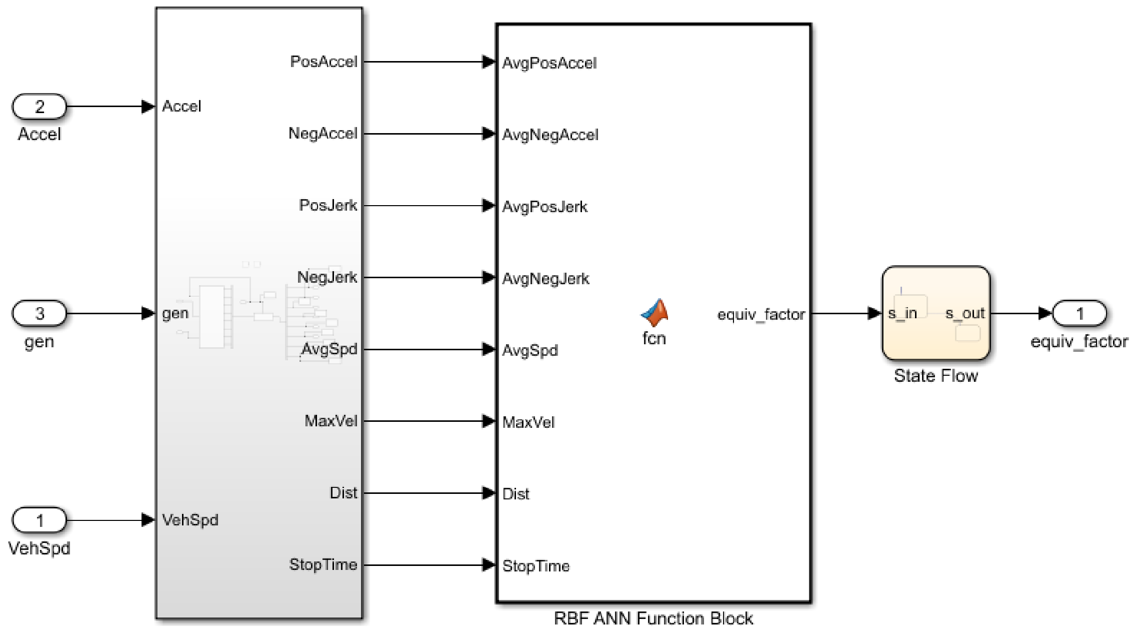


Figure 56: RBF ANN Function Block

Lastly, before being passed to the ECMS cost function block (from Figure 33), the equivalence factor is fed into a state flow block. The state flow block sets the equivalence factor to a constant until the first time window is passed. Otherwise, the equivalence factor would be a zero until the first time window was reached. Before the fuel economy is calculated, a drive cycle is run at least once, and the ending equivalence factor is used as the starting equivalence factor for the next run of the cycle. In an actual vehicle implementation, it is recommended that the last equivalence factor before the vehicle was shut off be saved and used as the starting value for the next key cycle. The code inside the RBF ANN Function Block can be seen in Appendix A.

5. Results and Analysis

The performance of the ANN-ECMS is evaluated using verification and validation. Verification is the process of evaluating the ANN performance using inputs that it has been trained with. In the case of this work, verification will involve selecting a few drive cycles that were used to train the ANN and running them using ANN-ECMS. The results from ANN-ECMS will then be compared to the fuel economy results obtained using the optimal equivalence factor from ordinary ECMS. From this point forward, results obtained using the optimal equivalence factor from ordinary ECMS will be known as optimal ECMS. Over the verification drive cycles, ANN-ECMS should produce results reasonably close to the optimal ECMS.

Validation is the process of evaluating the ANN performance using inputs that were not used the train the ANN. In the case of this work, validation will involve running some select drive cycles with ANN-ECMS that were not used to train the ANN. Then, the results of ANN-ECMS will be compared to optimal ECMS.

5.1. RBF ANN Parameter Selection and Performance Verification

Before performance of the RBF ANN can be verified, a value of variance (σ^2) and a time window (T_w) must be selected. To select σ^2 and T_w, the drive cycles to be used for verification are first selected. The performance of the RBF ANN over these drive cycles is then determined using different values of σ^2 and T_w. The values of σ^2 and T_w which produce fuel economy results closest to that of the optimal ECMS are selected. Fuel economy comparisons using the selected variance and time window are then presented.

The variance of the RBF ANN affects the value of the output. As shown again in Figure 57 (Note, Figure 57 is the same as Figure 44), if the variance is too small, this will result in the outputs of the RBF ANN tending towards zero. However, the placement of the centers (c) also come into play. If the centers are very close together, then a small value of variance will not push the output to zero. However, if the centers are far apart, a larger variance value will be needed.

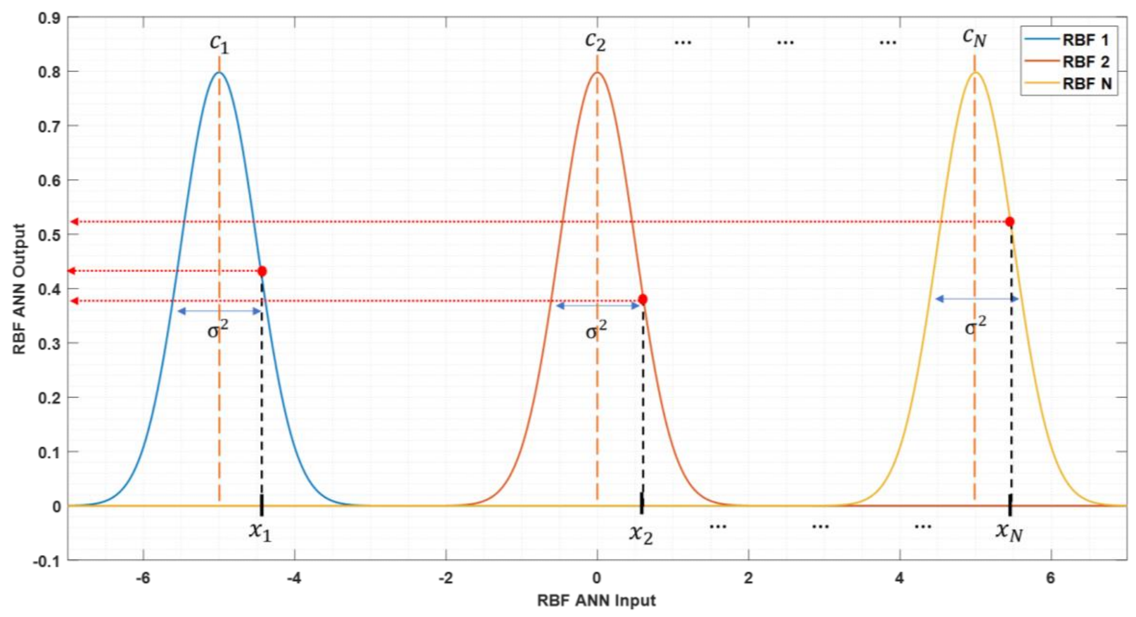


Figure 57: Set of Input Data to RBF ANN

Verification of the ANN-ECMS performance is performed using 6 drive cycles from the training data. The 6 drive cycles selected for verification are HWFET, US06, EUDC, New York Composite (NYCC), ECE Extra Urban Driving (ECEExtra), and Japanese 10-15 Mode (Jap1015). These cycles were selected because they are some of the cycles also analyzed in the work of Gu et. al [18]. As such, it will be informative to compare results. Additionally, these cycles offer a broad range of driving conditions.

The HWFET cycle is characterized by high speed with low aggression accelerations and practically no idle time. The HWFET cycle is shown below.

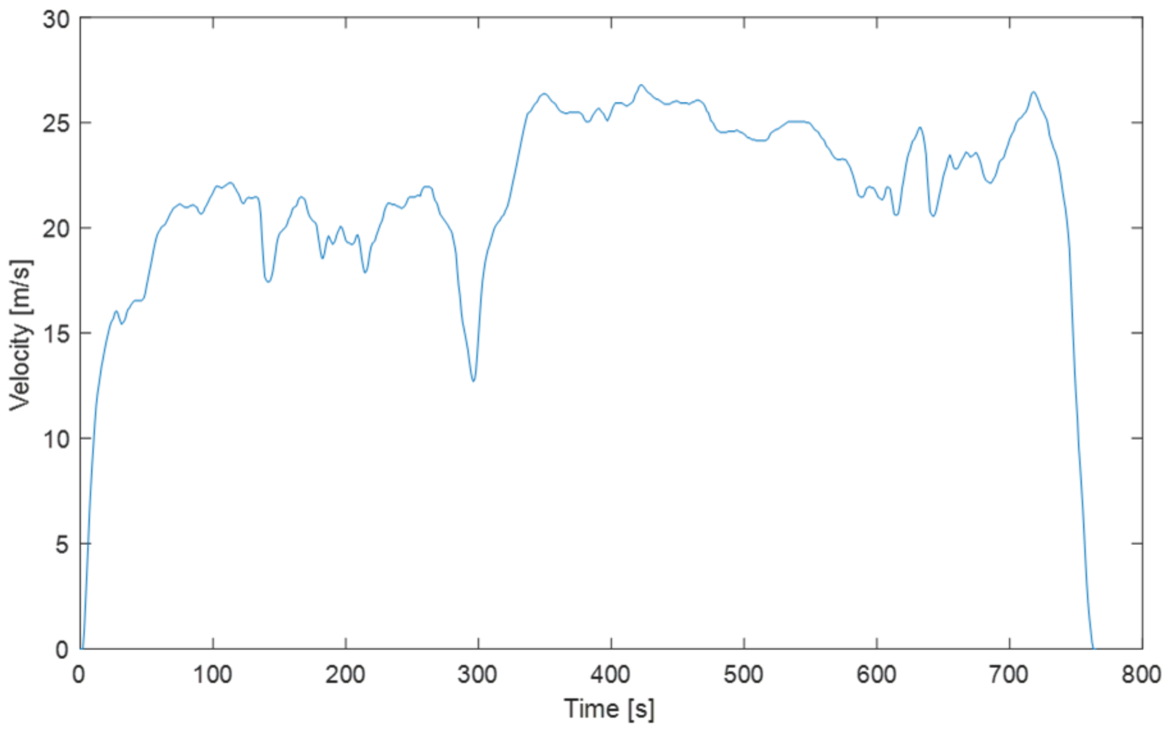


Figure 58: Highway Fuel Economy Driving Cycle (HWFET)

The US06 cycle is another high-speed cycle. However, unlike the HWFET cycle, the US06 cycle contains more aggressive accelerations, and instances of moderate idle time. The US06 cycle is shown below.

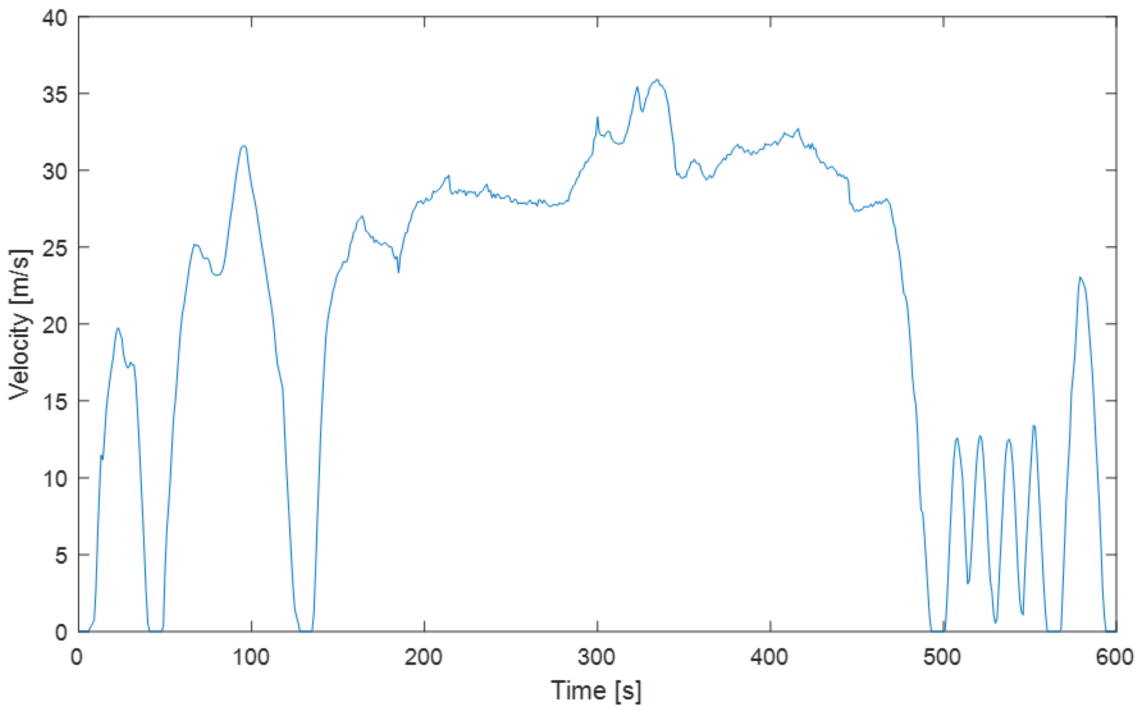


Figure 59: US06 Drive Cycle

The EUDC cycle contains moderately low speeds with very low acceleration and no idle time. The EUDC cycle is shown below.

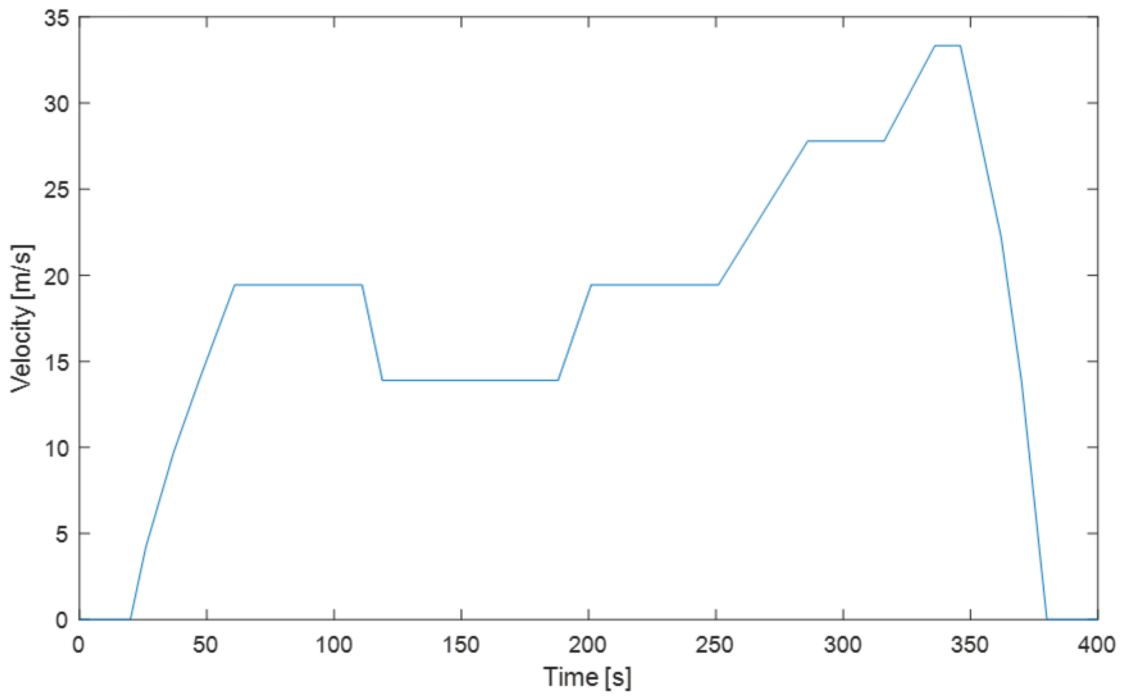


Figure 60: ECE Extra-Urban Driving Cycle

The NYCC cycle is characterized by low speeds, very aggressive acceleration, and frequent instances of idle time. The NYCC cycle is shown below.

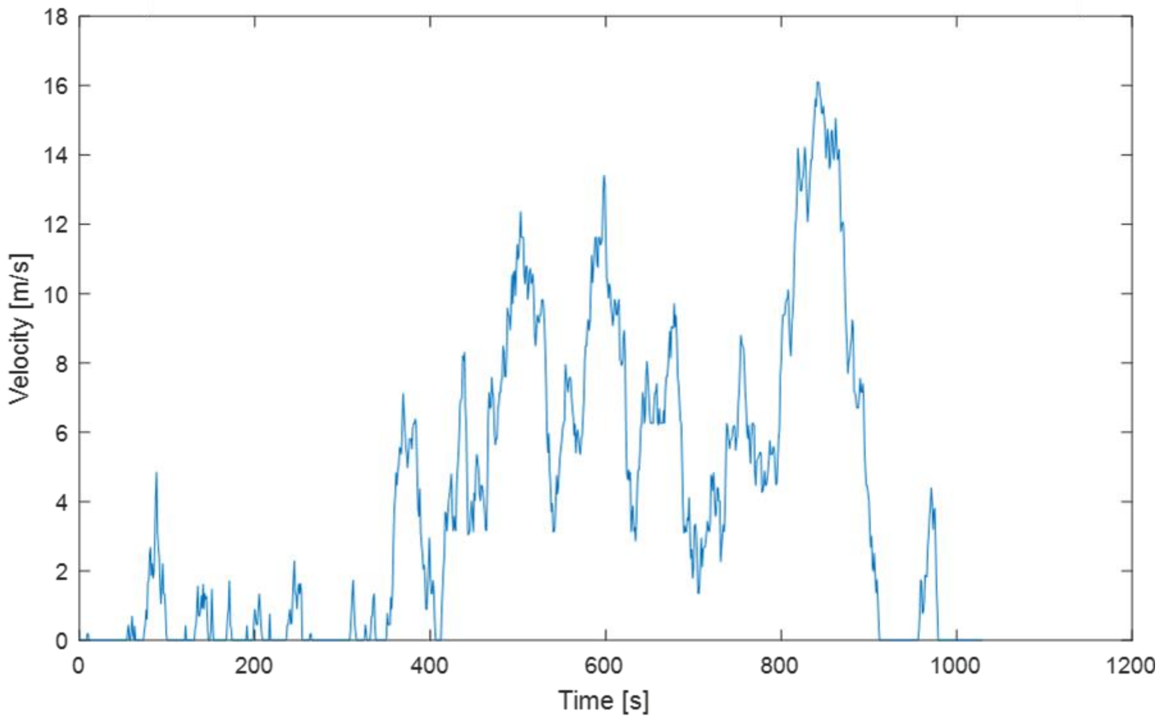


Figure 61: New York Composite Cycle

The ECEExtra cycle is similar to the EUDC cycle in aggression level and idle time. However, the ECEExtra cycle (shown below) does not reach as high of speeds as the EUDC cycle.

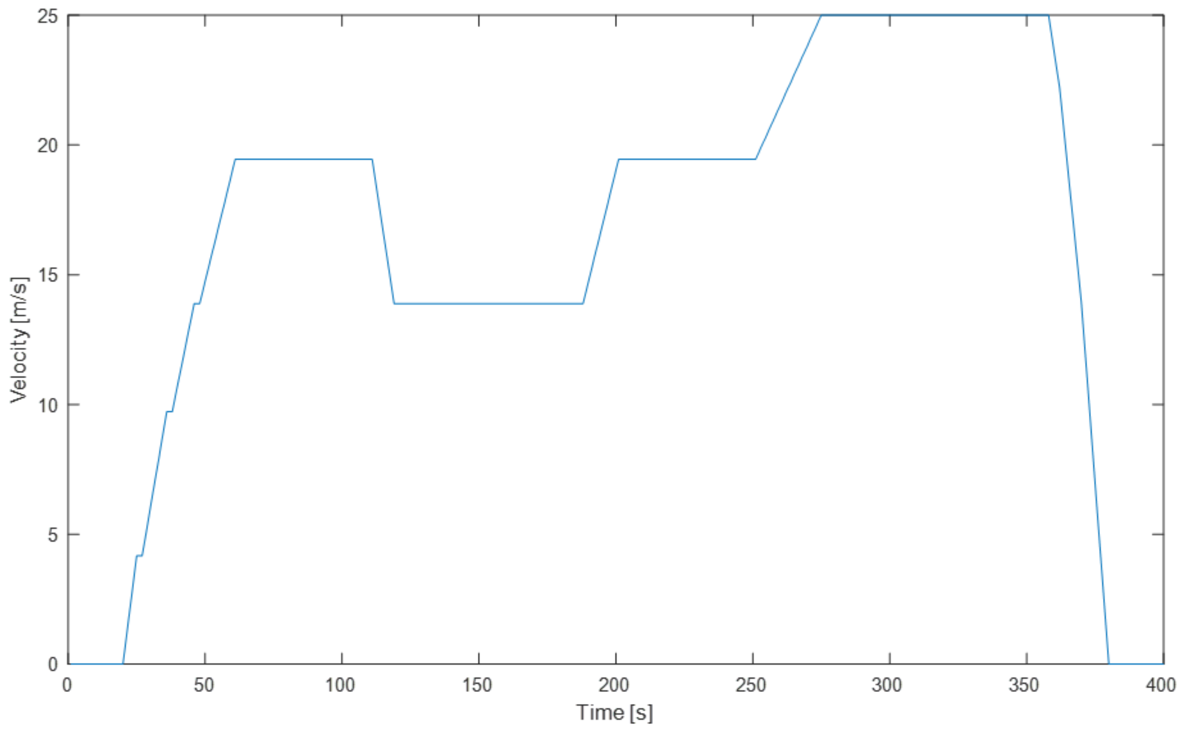


Figure 62: ECE Extra-Urban Driving Cycle (Low Powered Vehicles)

The Jap1015 cycle, shown below, is characterized by low speeds with a few areas of aggressive acceleration. The cycle also contains frequent instances of extended idle time.

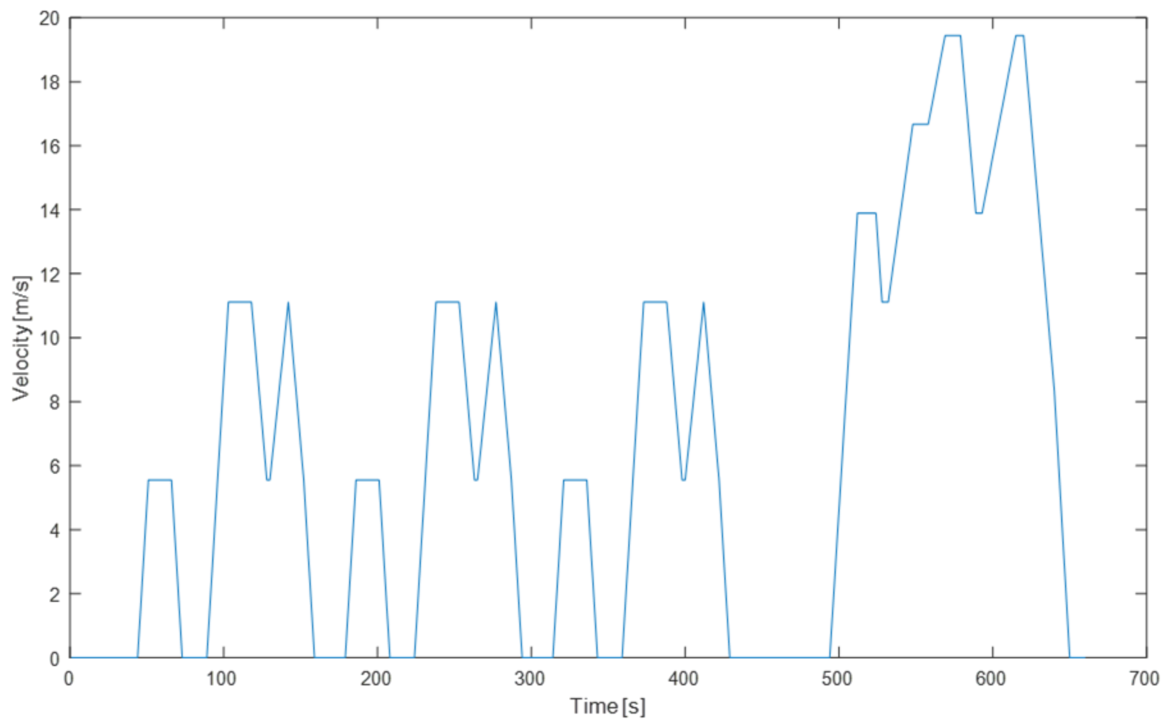


Figure 63: Japanese 10.15 Mode Driving Cycle

Since these 6 cycles offer a wide range of driving conditions, they should provide a thorough verification of the ANN-ECMS performance.

Using these cycles, a value of variance (σ^2) and a time window (T_w) must be selected. Three different time windows of 2, 3, and 4 minutes were evaluated over a range of variances. Initially, a range of variances around 50 was selected, because it was observed that a variance of 50 produced equivalence factors (the RBF ANN output) that were similar in magnitude to those observed to be optimal equivalence factors in the training data. The training data showed a range of equivalence factors between 0.5 and 1.1. Because of the observed similarity with the training data, variance factor values of 50, 60, 70, and 80 were examined over 2 and 3-minute time windows. However, before the 4-minute time-window was fully examined, it was observed that these variance values were having little to no effect on fuel economy. Over all of the verification drive cycles, the fuel economy difference between the variance values was negligible. Therefore, to get an understanding of the effect of variance, the variance range was broadened. Values of 8, 80, and 150 were tested using the 3 different time windows.

The variance value ultimately affects the magnitude of the equivalence factor over the course of a drive cycle. The equivalence factors for variance values of 8, 80, and 150 are shown in Figure 64, Figure 65, and Figure 66 respectively.

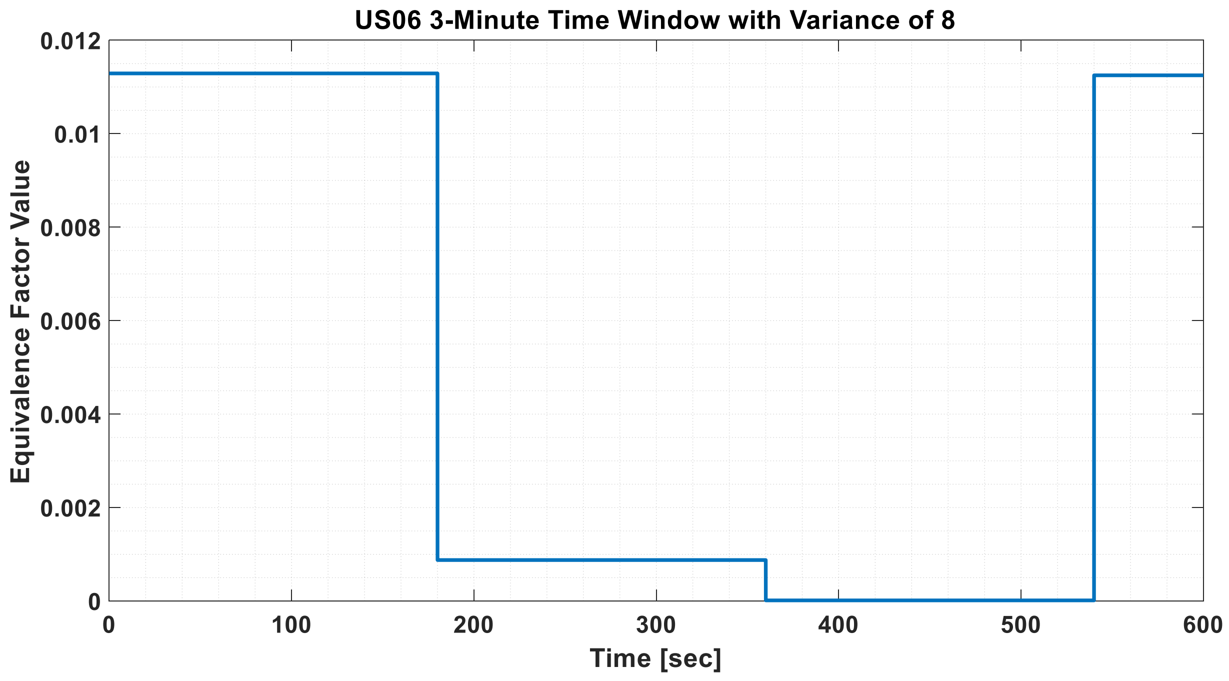


Figure 64: Varying Equivalence Factor Over the Course of US06 Drive Cycle with a Variance of 8

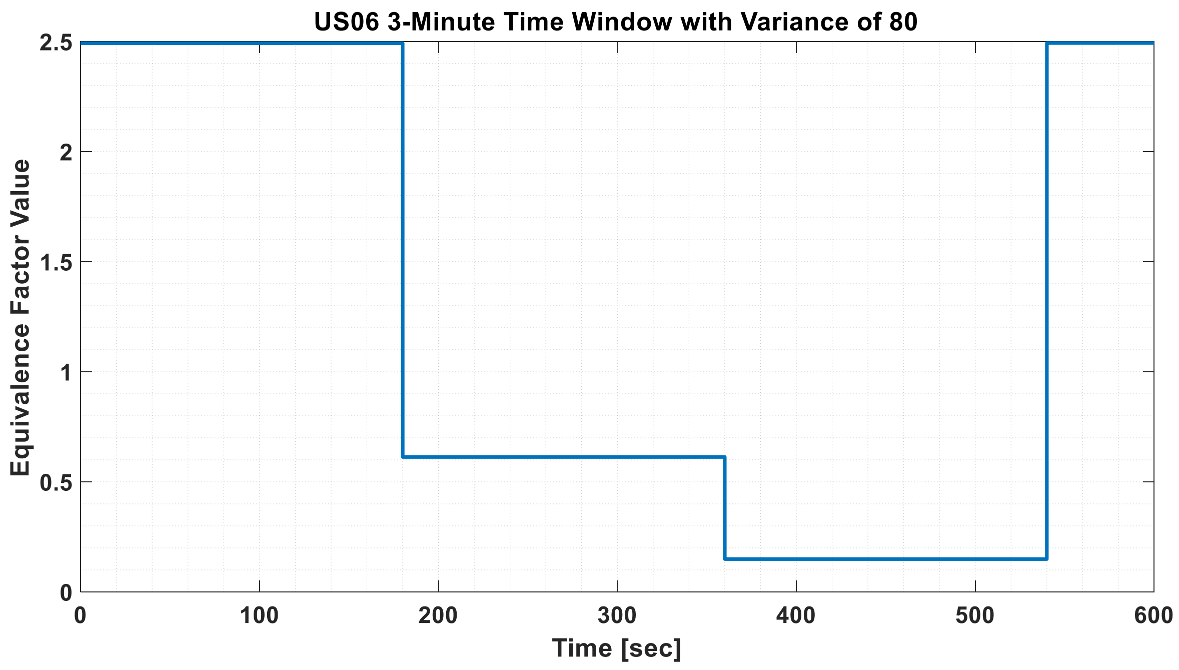


Figure 65: Varying Equivalence Factor Over the Course of US06 Drive Cycle with a Variance of 80

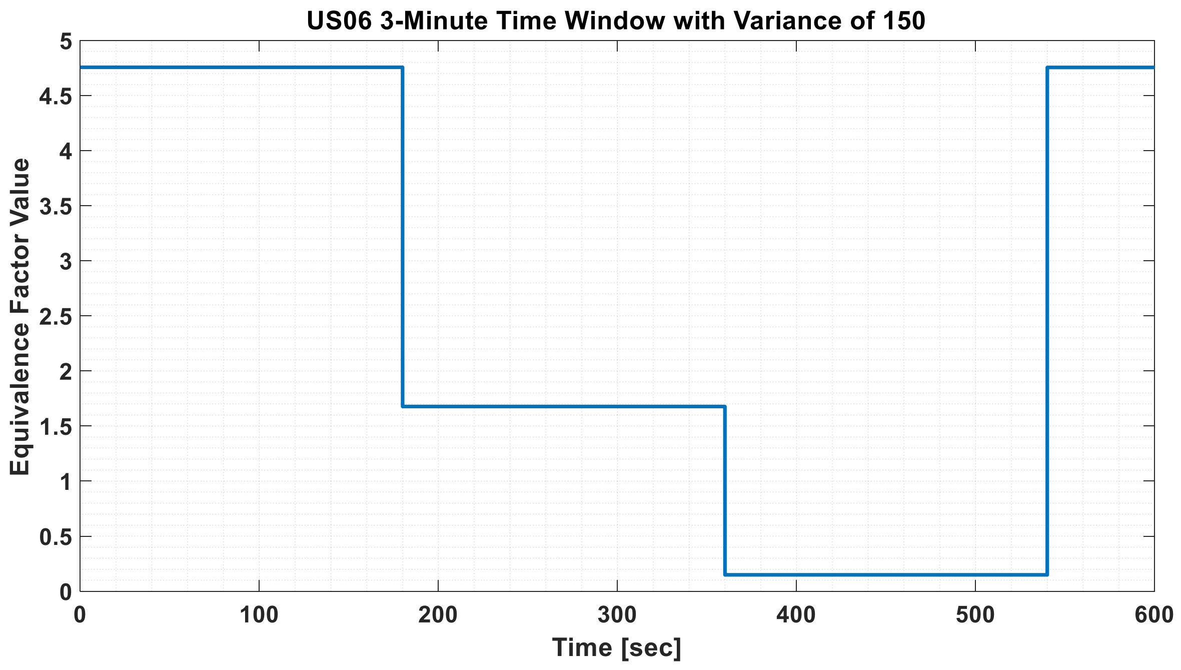


Figure 66: Varying Equivalence Factor Over the Course of US06 Drive Cycle with a Variance of 150

These figures show that as the variance increases, the magnitude of the equivalence factor (y-axis) also increases. As discussed in section 4-Artificial Neural Network, with respect to Figure 43, a low variance value tends to push the output to zero. This is apparent when viewing Figure 64. The low variance of 8, results in equivalence factor values near zero.

With respect to this work, Figure 64, Figure 65, and Figure 66 give an idea of what variance value should be used. From the training data, it is known that the equivalence factor values which produce the maximum fuel economy vary from 0.5 to 1.1. Therefore, a variance value should be selected which yields equivalence factors roughly within that range. Considering the equivalence factor range from the training data, the figures indicate that an equivalence value of 80 or 150 is more likely to produce better results than a value of 8.

Of course, the equivalence factors shown in Figure 64, Figure 65, and Figure 66 are wholly dependent on the inputs, which depend on the characteristics of the drive cycle. Therefore, it is necessary to consider the variance values of 8, 80, and 150 over all of the verification drive cycles.

Figure 67 – Figure 69 show the fuel economy vs. variance for the variance values of 8, 80, and 150 for time windows of 2, 3, and 4 minutes respectively. This comparison is made over all of the verification drive cycles. An analysis of the fuel economy comparisons shown in the following figures will give direction on what variance and time window should be selected for use in the RBF ANN. Up to this point, a comparison to the fuel economy obtained from optimal ECMS has not been made. The following figures only show a comparison between variance values.

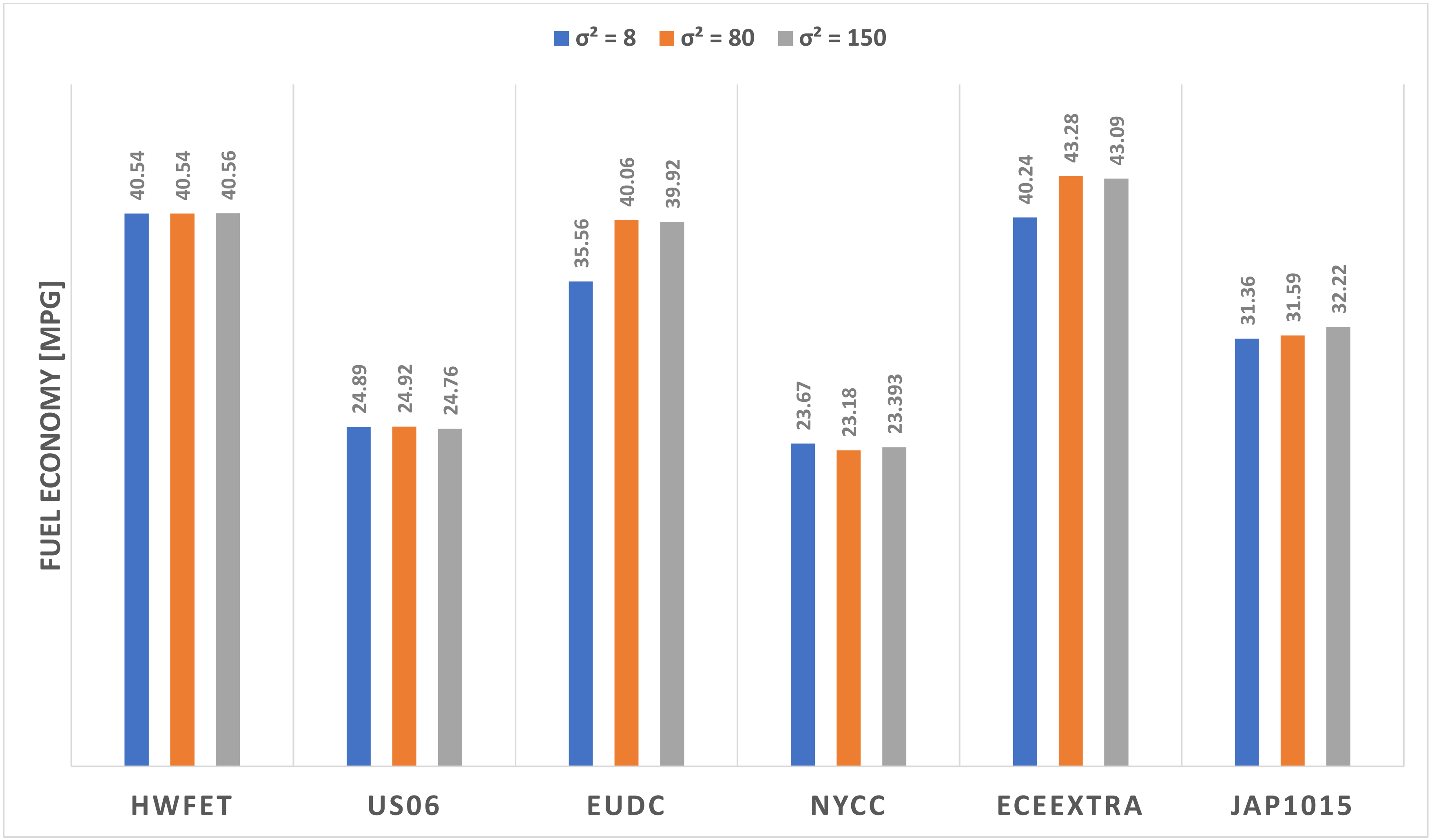


Figure 67: Effect of Variance [8,80,150] on 2-Minute Time Window

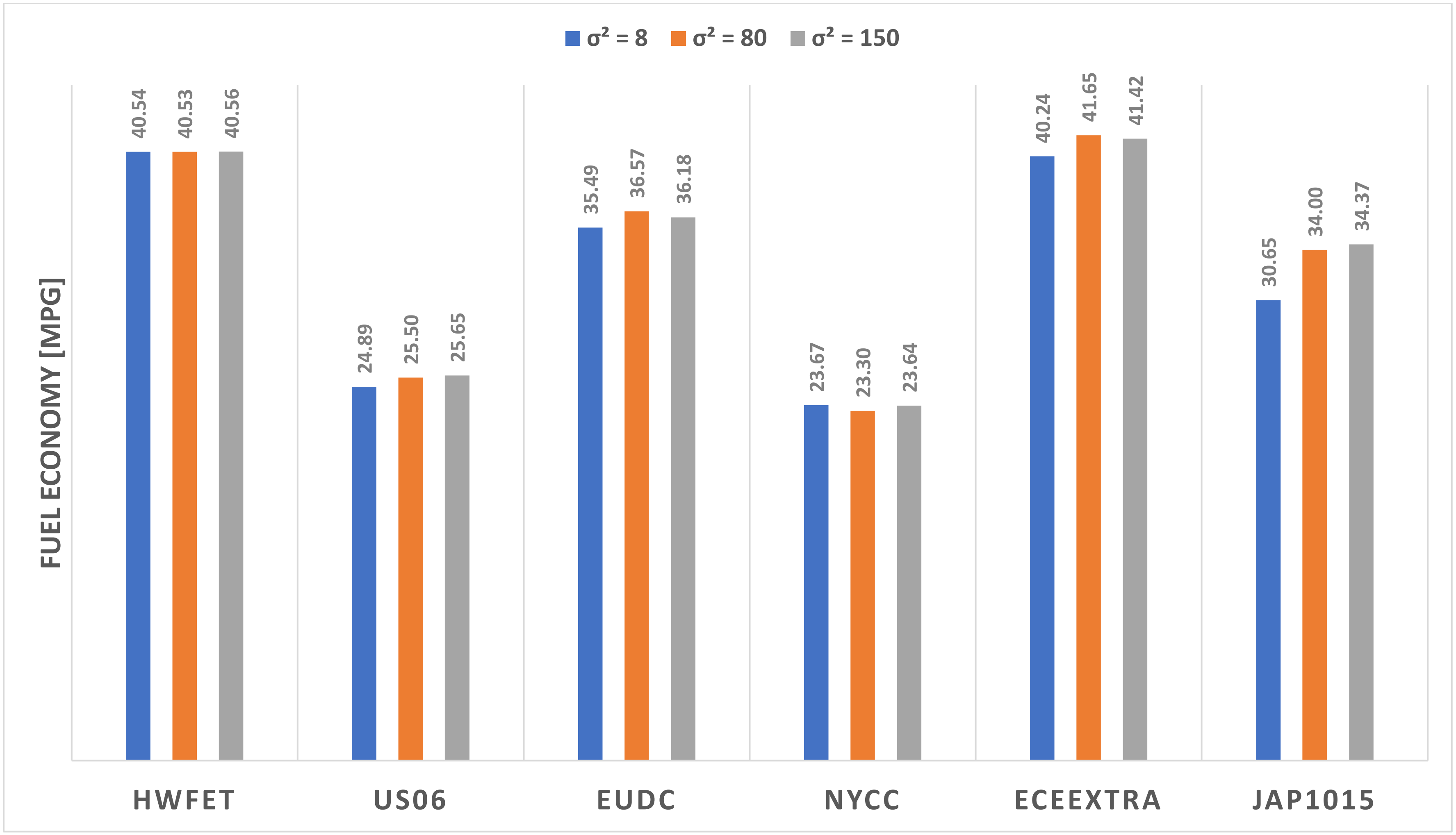


Figure 68: Effect of Variance [8,80,150] on 3-Minute Time Window

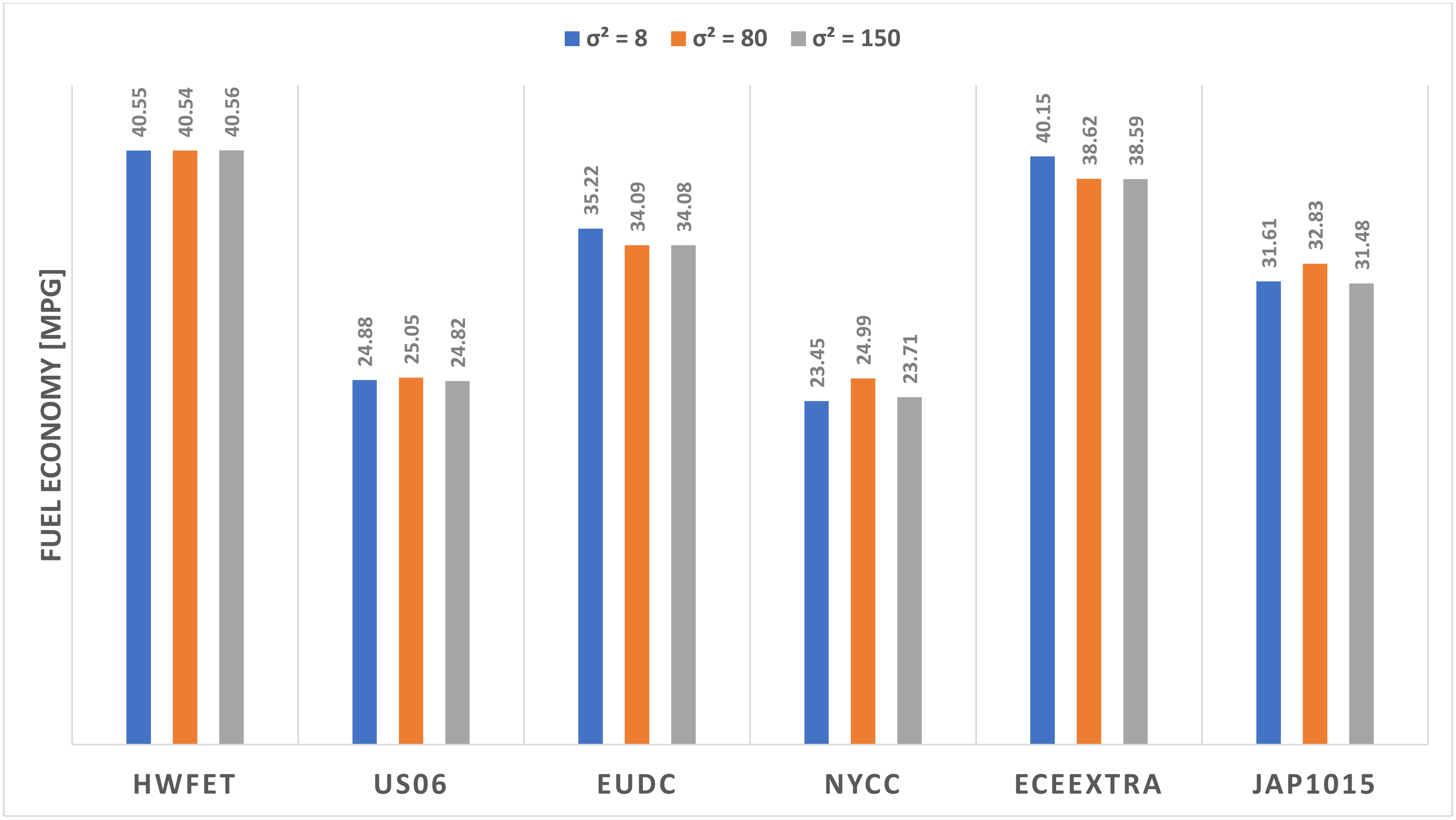


Figure 69: Effect of Variance [8,80,150] on 4-Minute Time Window

An examination of Figure 67,Figure 68, and Figure 69, does not show a clear winner in terms of performance. Ultimately, single values for σ^2 and T_w need to be selected. These values need to be selected such that they maximize performance over the entire range of verification drive cycles. If a variation of percent error is considered, the overall picture becomes clearer. The equation used is as follows:

$$\% \, Error = \frac{ECMS_{FE} - ANN_{FE}}{ECMS_{FE}} * 100 \tag{5-1}$$

Where $ECMS_{FE}$ is the fuel economy determined using optimal ECMS and ANN_{FE} is the fuel economy obtained using ANN-ECMS. Using this equation, positive values indicate the fuel economy obtained using ANN-ECMS is less than that of the optimal from ECMS. Negative values indicate that the ANN-ECMS outperformed the optimal ECMS. Figure 70, Figure 71, andFigure 72 show the percent error for each of the 3 time windows and the variance values of 8, 80, and 15.

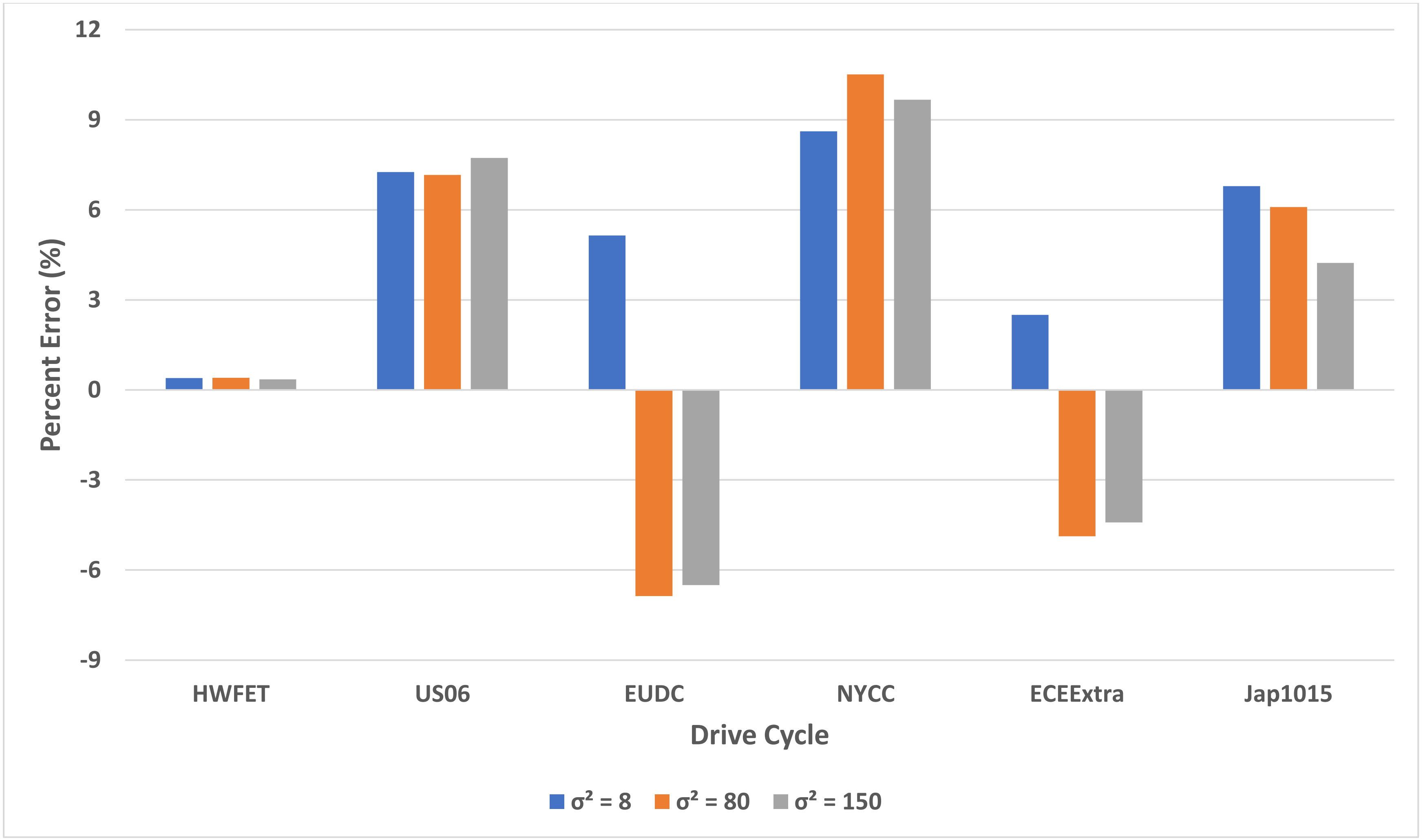


Figure 70: % Error vs σ².for the 2-Minute Time Window

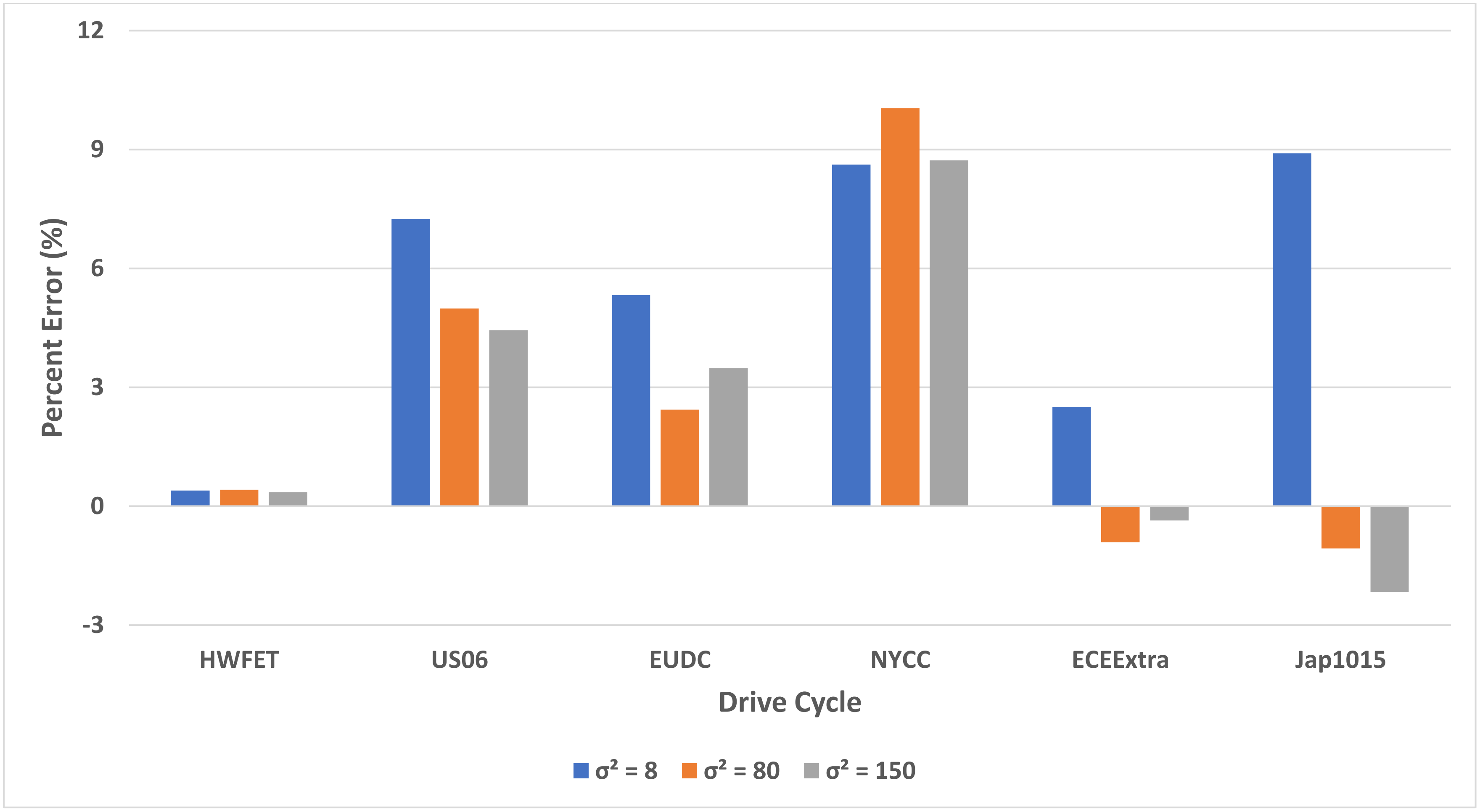


Figure 71:% Error vs σ2.for the 3-Minute Time Window

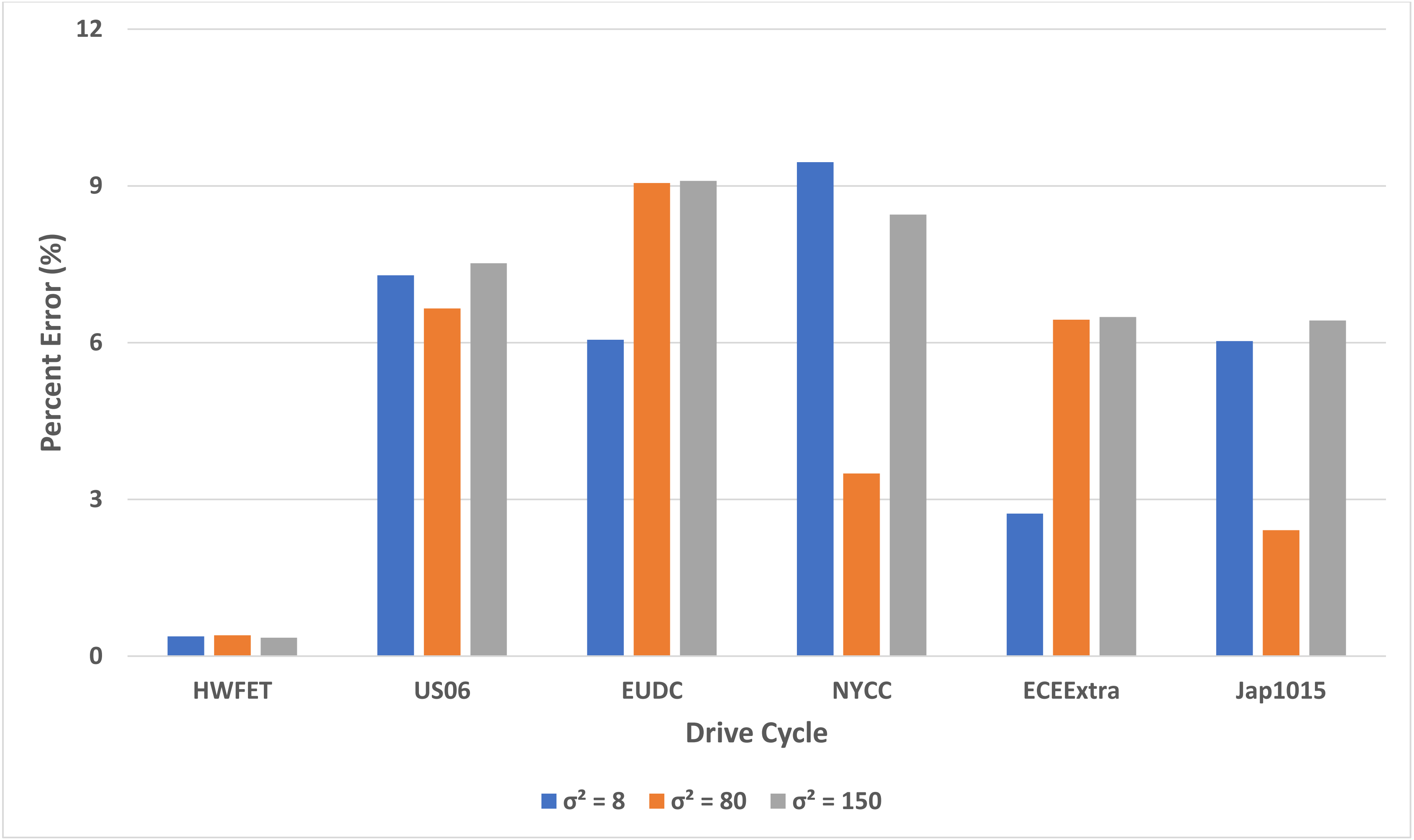


Figure 72: % Error vs σ2.for the 4-Minute Time Window

Despite the metric of percent error, it is still not readily apparent which parameter set of σ^2 and T_w yield the best performance. To arrive at a conclusion of the best performing values, the cumulative performance is determined by adding up the percent error for each set of variance and time window parameters. The best performing set of parameters will be those which yield the lowest cumulative percent error. The addition is shown in Table 8.

Table 8: % Error Comparisons of Validation Drive Cycles

	2-Minute % Error			**3-Minute % Error**			**4-Minute % Error**		
	$\sigma^2 = 8$	$\sigma^2 = 80$	$\sigma^2 = 150$	$\sigma^2 = 8$	$\sigma^2 = 80$	$\sigma^2 = 150$	$\sigma^2 = 8$	$\sigma^2 = 80$	$\sigma^2 = 150$
HWFET	0.40	0.41	0.35	0.39	0.41	0.35	0.38	0.40	0.35
US06	7.26	7.16	7.73	7.25	4.99	4.44	7.29	6.66	7.52
EUDC	5.14	-6.87	-6.50	5.32	2.43	3.48	6.05	9.05	9.10
NYCC	8.62	10.51	9.67	8.62	10.04	8.73	9.45	3.50	8.45
ECEExtra	2.50	-4.87	-4.42	2.50	-0.91	-0.36	2.73	6.44	6.49
Jap1015	6.79	6.09	4.23	8.90	-1.07	-2.16	6.03	2.41	6.42
Sum	30.71	12.42	11.06	32.98	15.90	14.49	31.94	28.46	38.33

Despite this new metric of cumulative performance, it is still not readily apparent which parameter set of variance and time window yield the best results. The best parameter set is only 1.4% away from the second-best performing set. The best performing parameter set corresponds to a time window of 2 minutes and variance of 150, with a cumulative percent error of 11.06%. The second-best performing set has a cumulative percent error of 12.42%. There is not an outstanding set of σ^2 and T_w that is far above the rest.

To more confidently claim the best performing set of parameters, an additional 3 drive cycles from the training data are added to the set of verification drive cycles. These cycles are shown below:

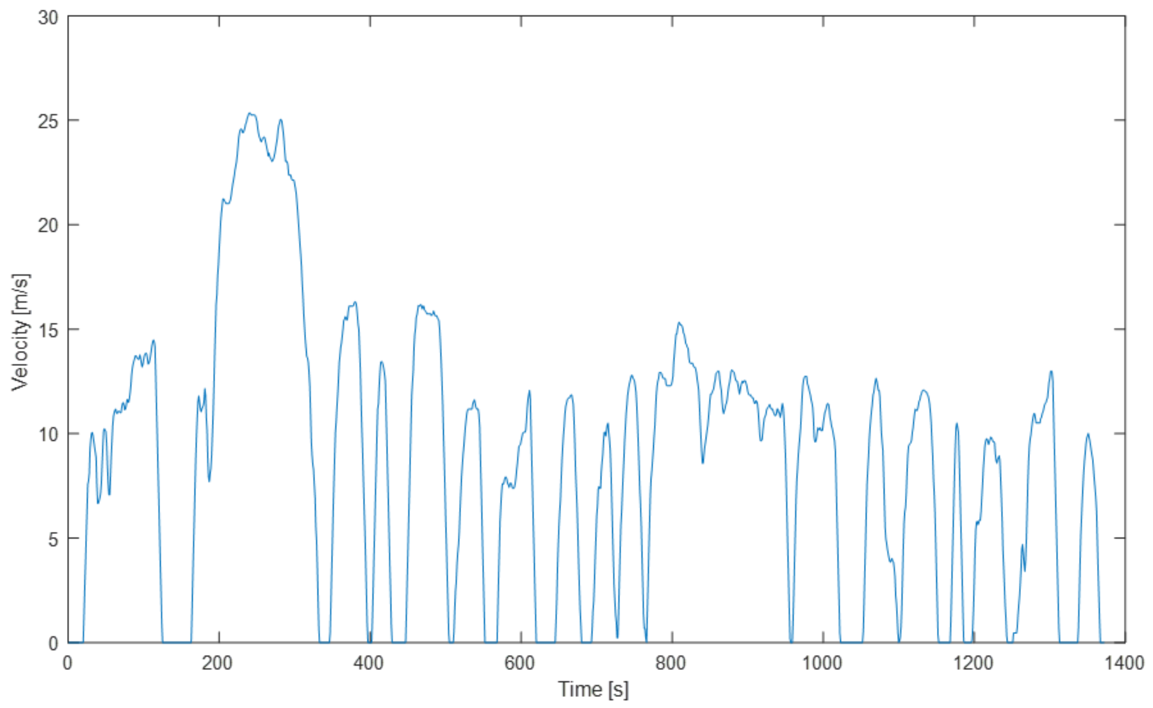


Figure 73: EPA Urban Dynamometer Driving Cycle (FTP-72)

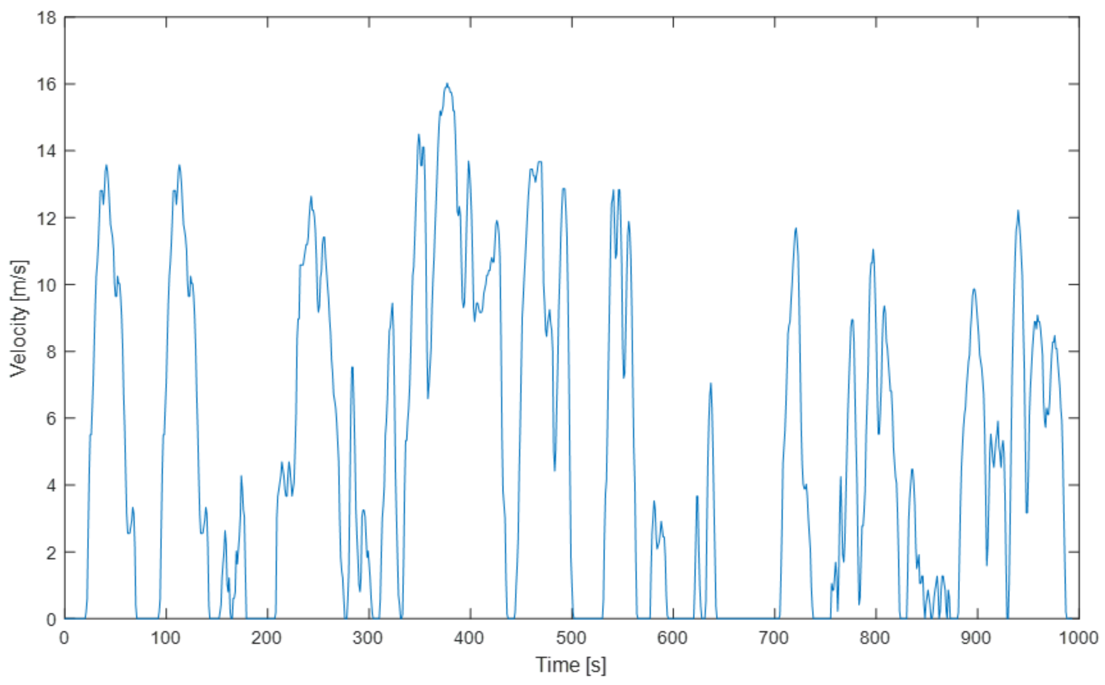


Figure 74: Artemis Urban Velocity

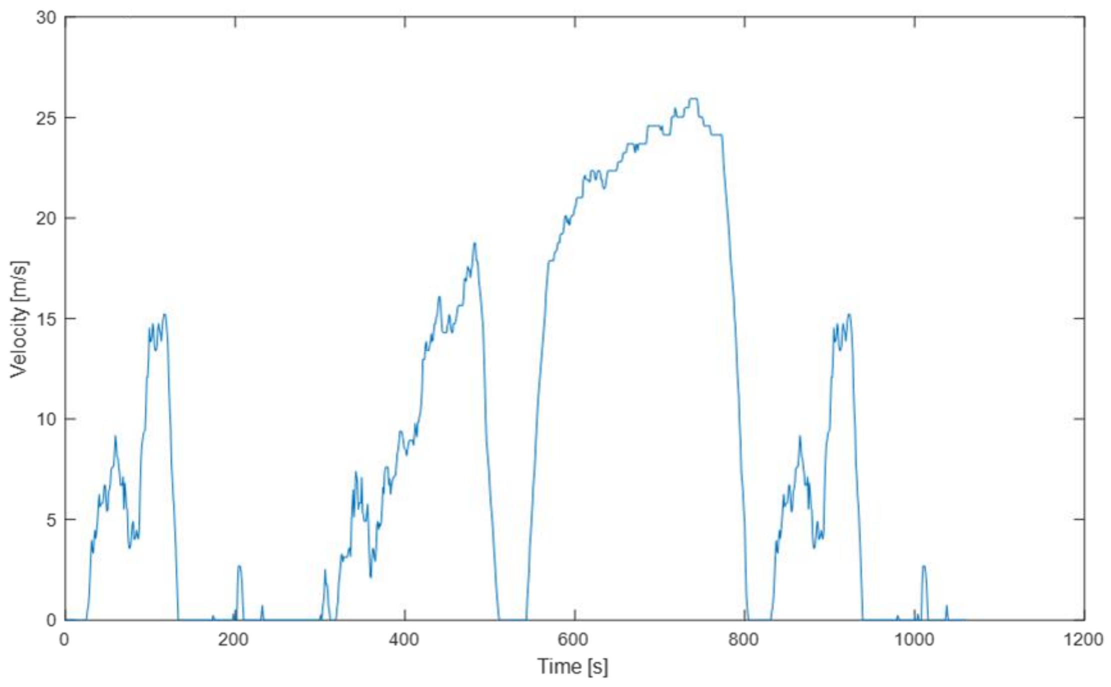


Figure 75: EPA Heavy Urban Dynamometer Driving Cycle

With the addition of these cycles, the table of percent errors can be updated as shown in Table 9.

Table 9: Updated % Error Comparisons

	2-Minute % Error			3-Minute % Error			4-Minute % Error		
	$\sigma^2 = 8$	$\sigma^2 = 80$	$\sigma^2 = 150$	$\sigma^2 = 8$	$\sigma^2 = 80$	$\sigma^2 = 150$	$\sigma^2 = 8$	$\sigma^2 = 80$	$\sigma^2 = 150$
HWFET	0.40	0.41	0.35	0.39	0.41	0.35	0.38	0.40	0.35
US06	7.26	7.16	7.73	7.25	4.99	4.44	7.29	6.66	7.52
EUDC	5.14	-6.87	-6.50	5.32	2.43	3.48	6.05	9.05	9.10
NYCC	8.62	10.51	9.67	8.62	10.04	8.73	9.45	3.50	8.45
ECEExtra	2.50	-4.87	-4.42	2.50	-0.91	-0.36	2.73	6.44	6.49
Jap1015	6.79	6.09	4.23	8.90	-1.07	-2.16	6.03	2.41	6.42
FTP72	8.64	3.99	6.35	8.57	1.37	5.70	9.72	-5.48	-0.41
ArtUrb	6.76	7.67	5.59	7.20	7.92	7.33	7.05	-1.25	1.94
HUDDS	8.18	8.58	8.98	8.15	0.72	2.38	8.12	7.81	1.77
Sum	54.29	32.65	31.98	56.90	25.92	29.89	56.83	29.55	41.63

Based on the comparisons in Table 9, there is now a clearer best performer. A time window of 3 minutes and a variance of 80 results in the lowest cumulative percent error. This is 3.6% above the next best performing set of variance and time window – as compared to 1.4% before the 3 additional drive cycles were added to the set of verification drive cycles. This gives a greater level of confidence that this variance and time window is indeed the best performing set of parameters.

5.2. Effect of Time Window

Why did the 3-minute time window produce better results than the other two time windows? To understand why the time window of 3 minutes yields the best performance, an examination of the effect of time window on the inputs is presented.

The differences between the 2, 3, and 4-minute time windows are the manifest in the input values. The inputs of acceleration, deceleration, positive and negative jerk, average speed, and maximum velocity are all relatively consistent across the time windows. These inputs fall within the hyperspace of the training data most of the time. Figure 76 through Figure 81 show these inputs. If an input falls in between the maximum and minimum input value from the training data, it is within the hyperspace of the training data.

The inputs of acceleration, deceleration, positive and negative jerk, average speed, and maximum velocity for the 2, 3, and 4-minute time windows fall within the hyperspace of the training data. The following figures show these inputs. They are all from the FTP72 drive cycle – which is a cycle used in the training data. The other drive cycles used in the training data show similar trends. Note that the inputs are zero until a full time window has passed, at which point the inputs are updated.

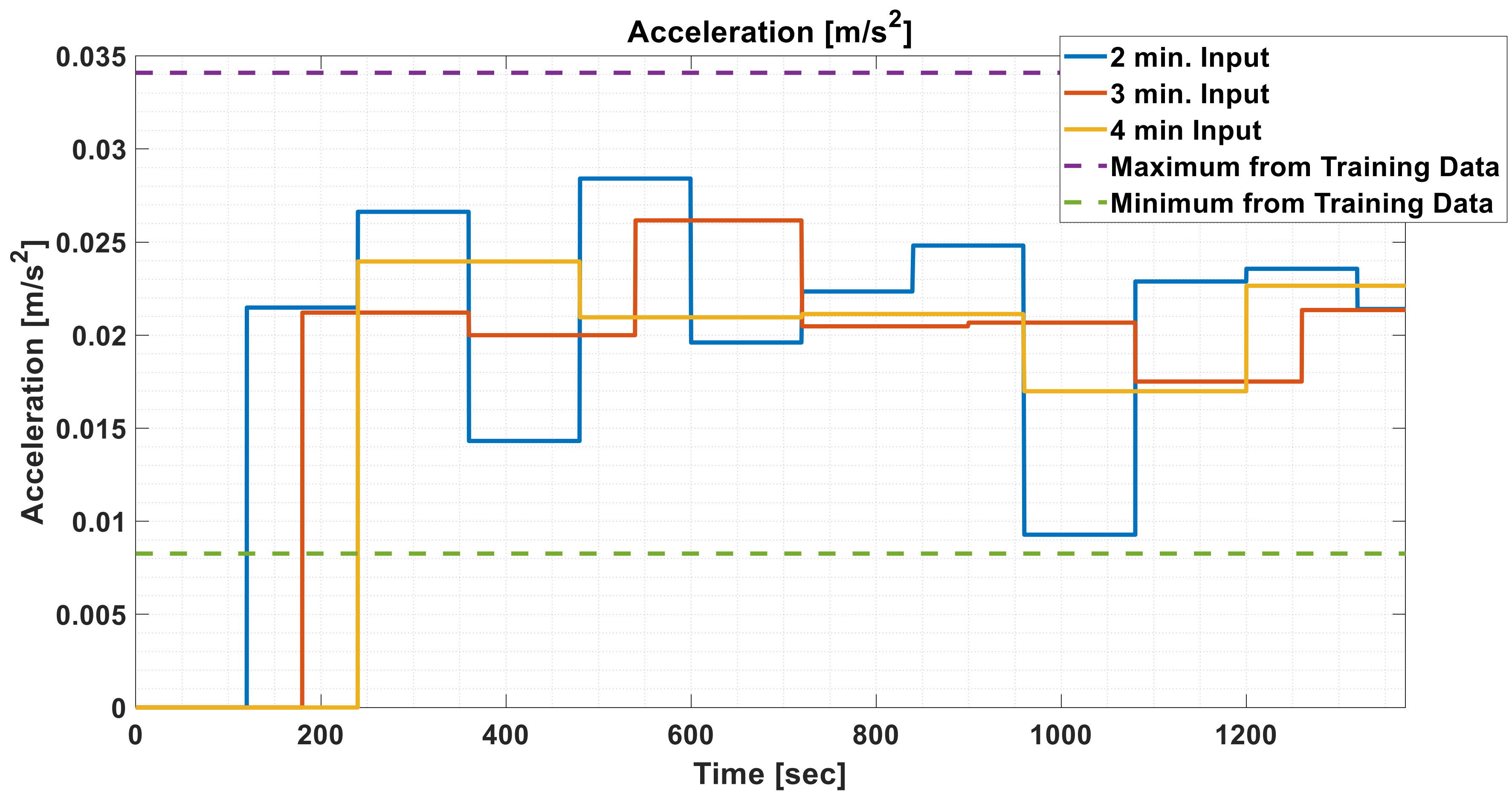


Figure 76: FTP72 Drive Cycle– ANN Input of Acceleration

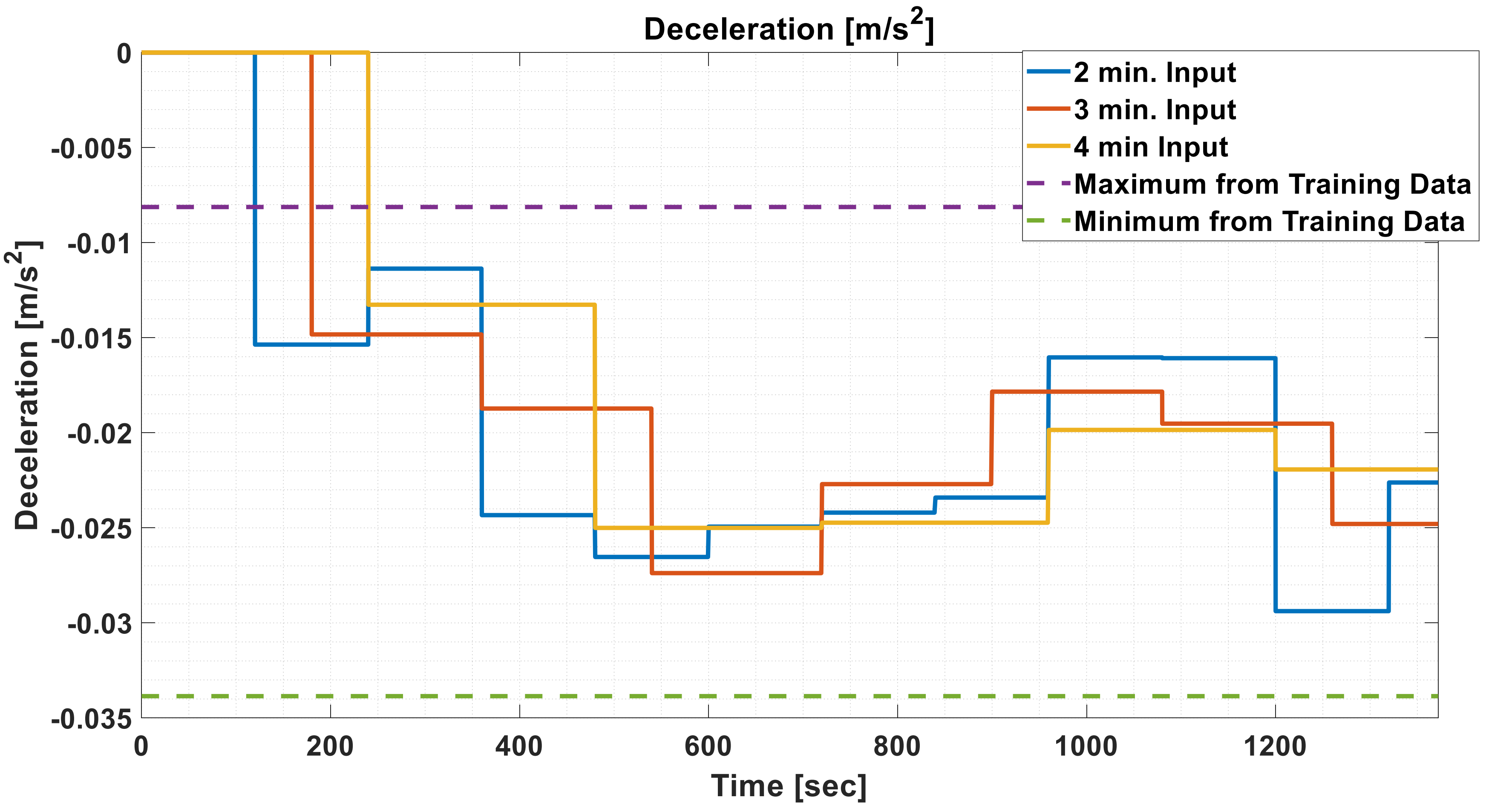


Figure 77: FTP72 Drive Cycle – ANN Input of Deceleration

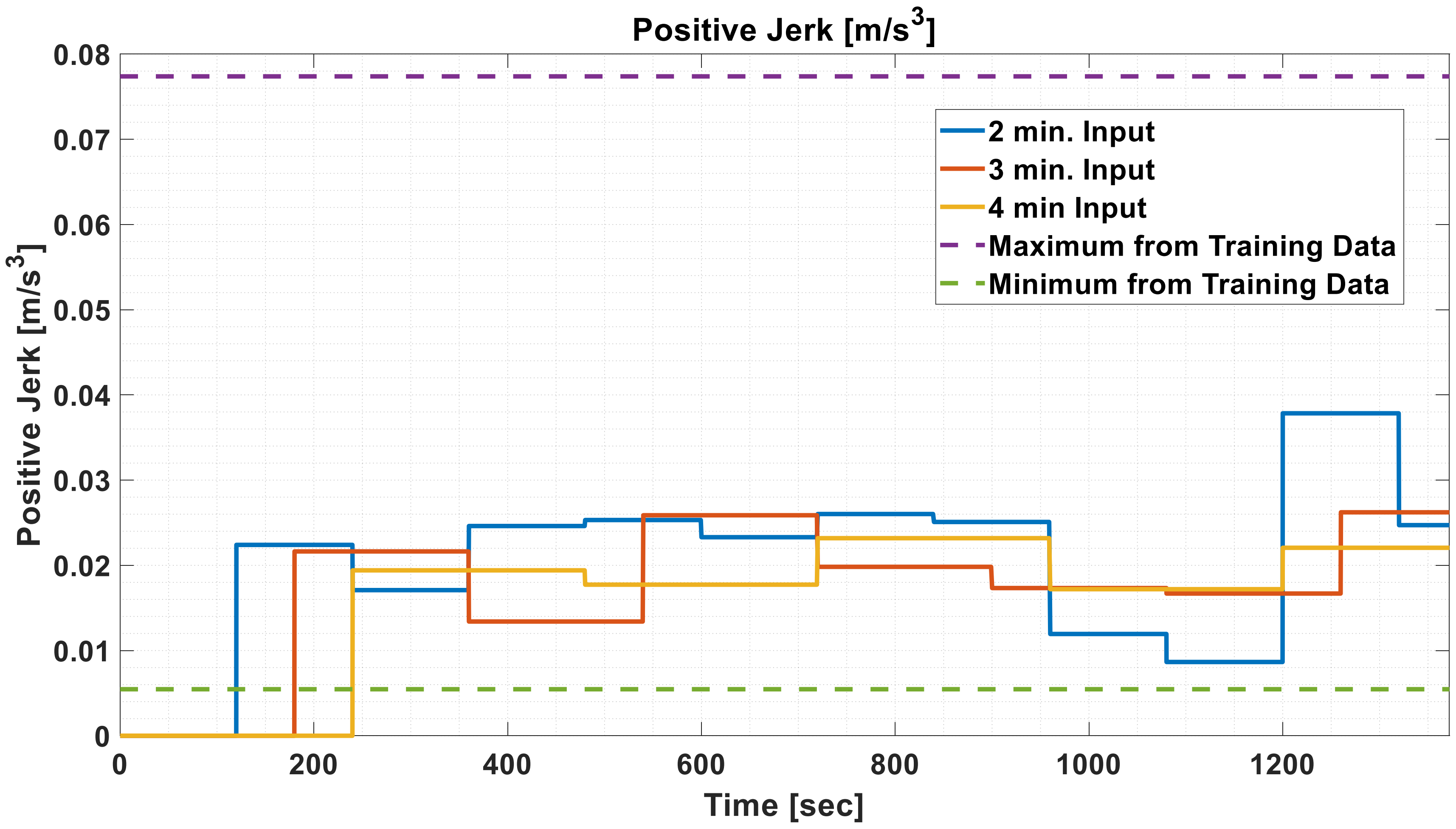


Figure 78: FTP72 Drive Cycle – ANN Input of Positive Jerk

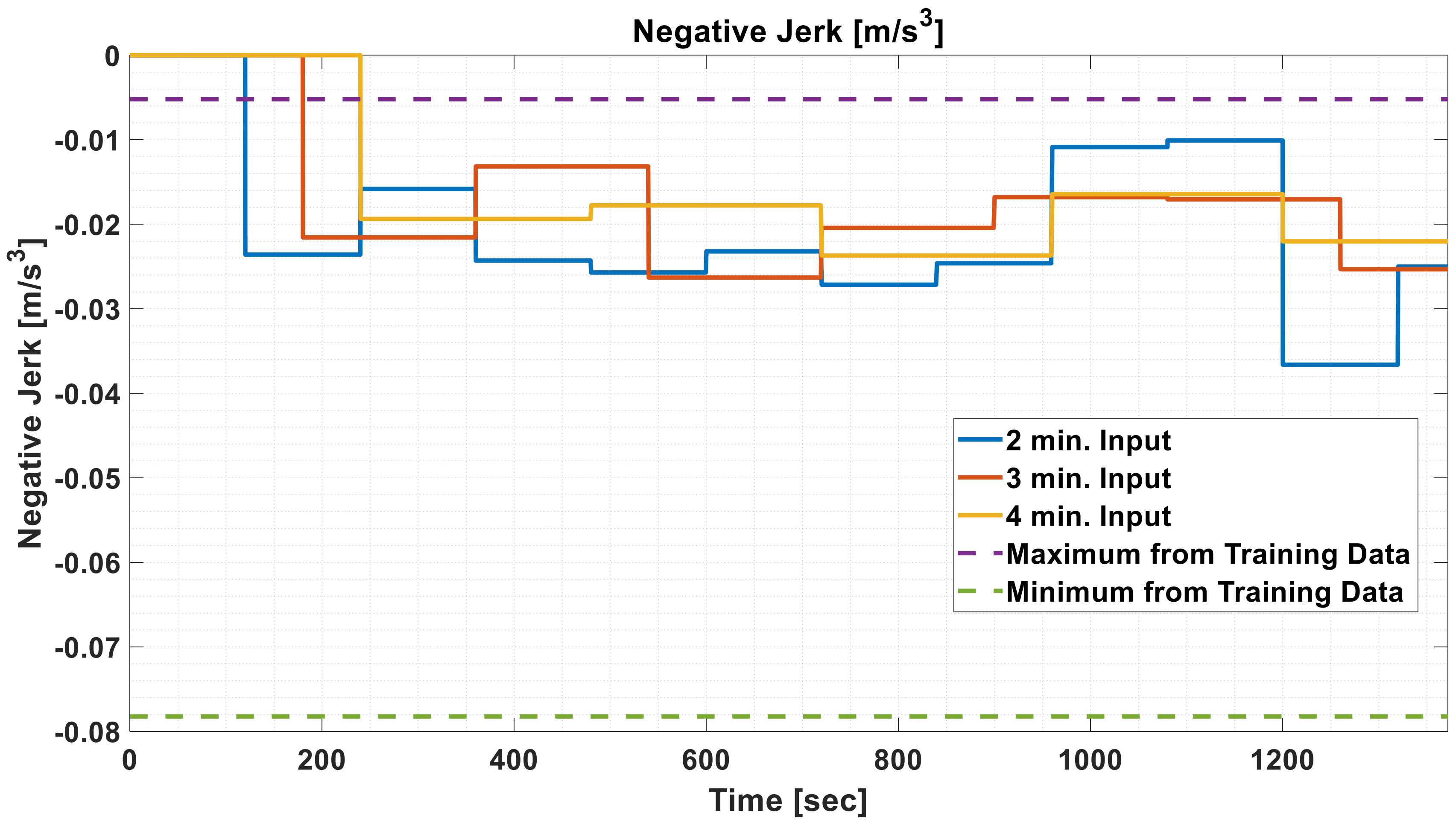


Figure 79: FTP72 Drive Cycle – ANN Input of Negative Jerk

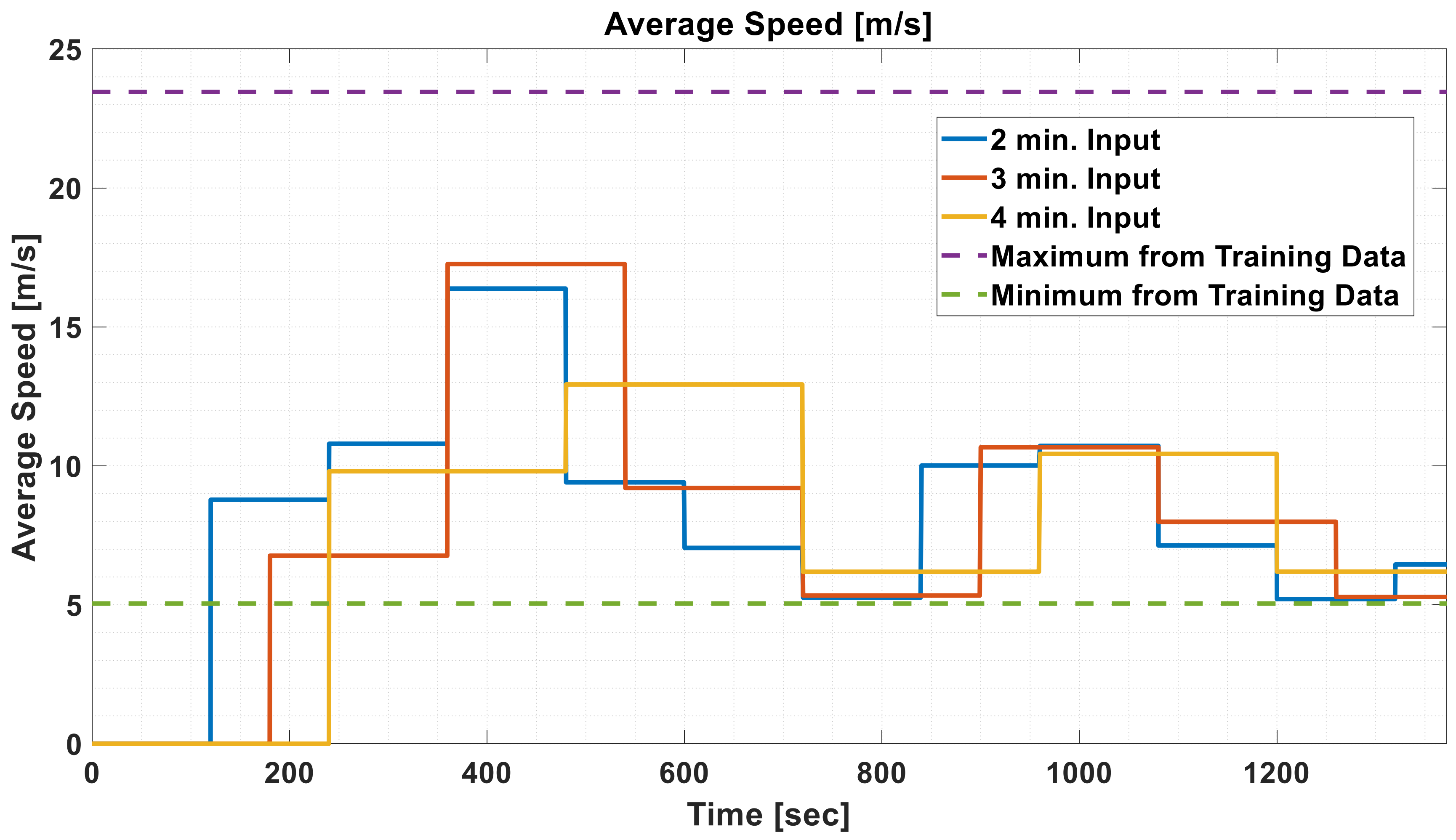


Figure 80: FTP72 Drive Cycle – ANN Input of Average Speed

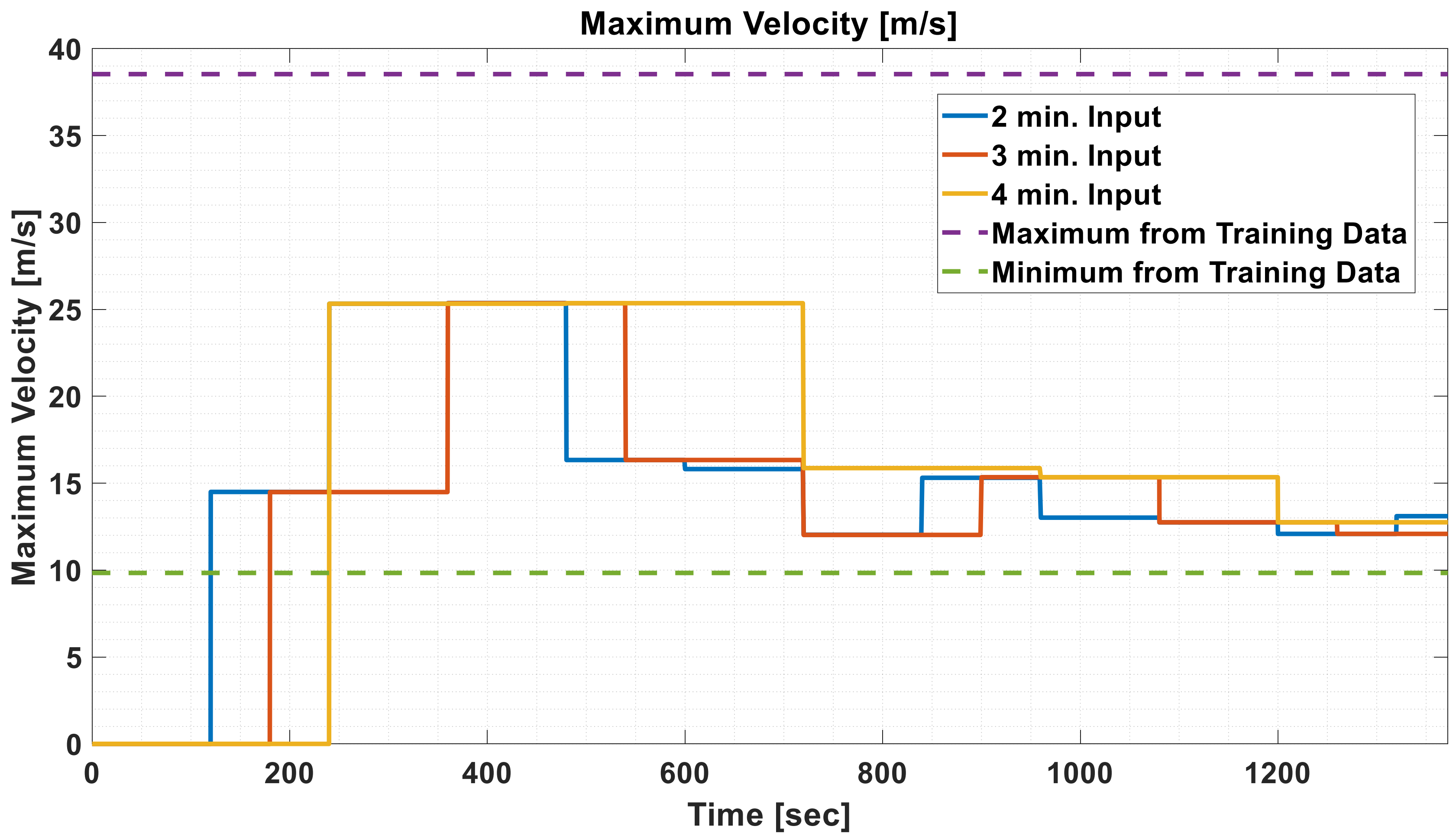


Figure 81: FTP72 Drive Cycle – ANN Input of Maximum Velocity

figuresFigure 76 through Figure 81 show that the inputs of acceleration, deceleration, positive and negative jerk, average speed, and maximum velocity fall within the hyperspace of the training data. These inputs are largely dependent on the characteristics of the drive cycle. Drive cycles with different characteristics could potentially result in inputs that are outside of the bounds of the training data. However, this is why 30 drive cycles with a wide range of characteristics were used in the training data – to ensure that the inputs from any type of driving conditions fell within the hyperspace of training data.

The inputs of distance and idle time behave differently across the 3 time windows. The training data of distance and idle was gathered across entire drive cycles. Consequently, the minimum values of the hyperspace for idle time and distance are relatively large. Over the time window of 2-minutes the inputs of distance and idle time often do not land in the hyperspace of the training data. Conversely, the 3 and 4-minute time windows are long enough to often put the distance and idle time in the hyperspace of the training data. This is shown in figures Figure 82 through Figure 85.

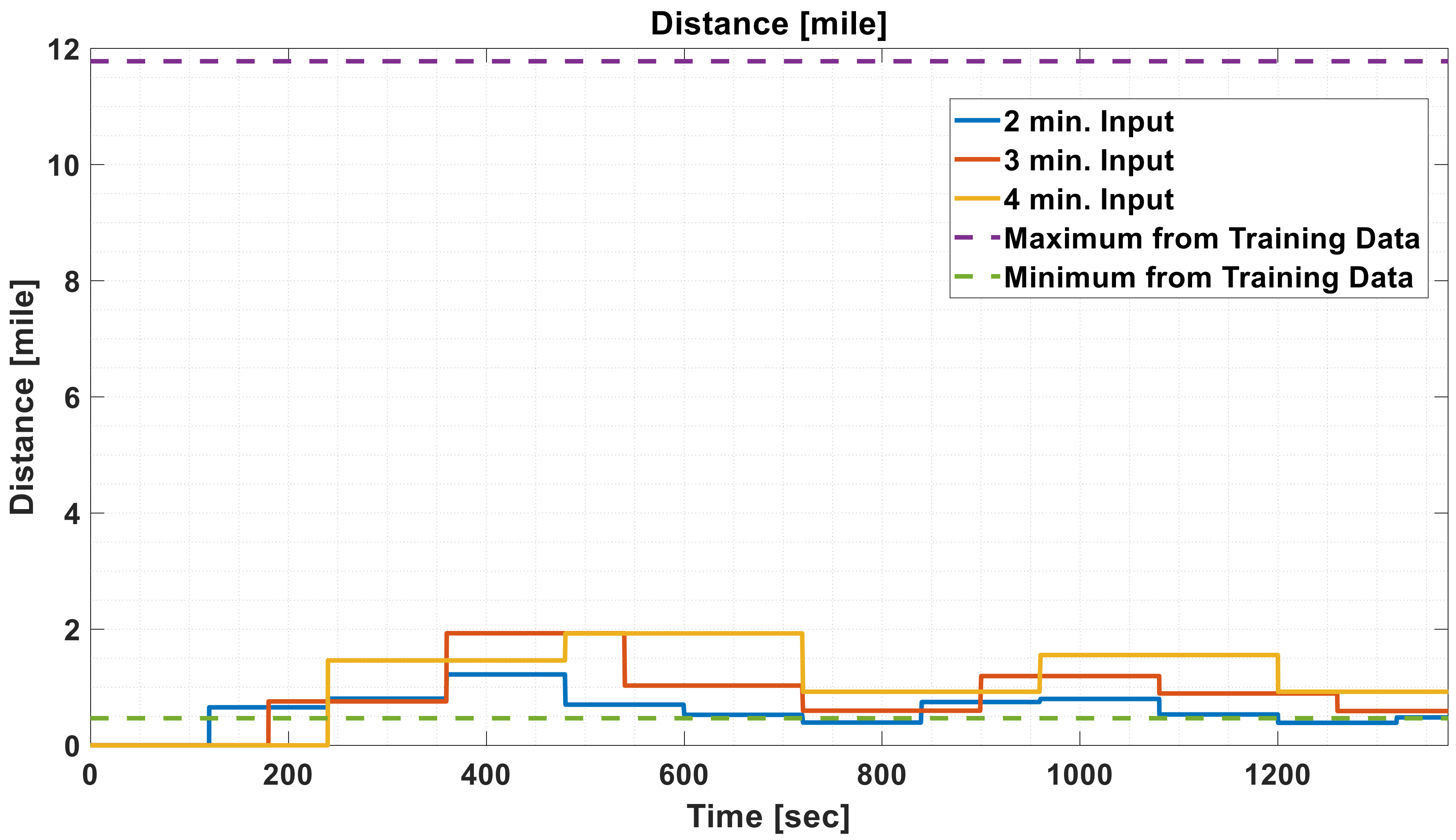


Figure 82: FTP72 Drive Cycle – ANN Input of Distance

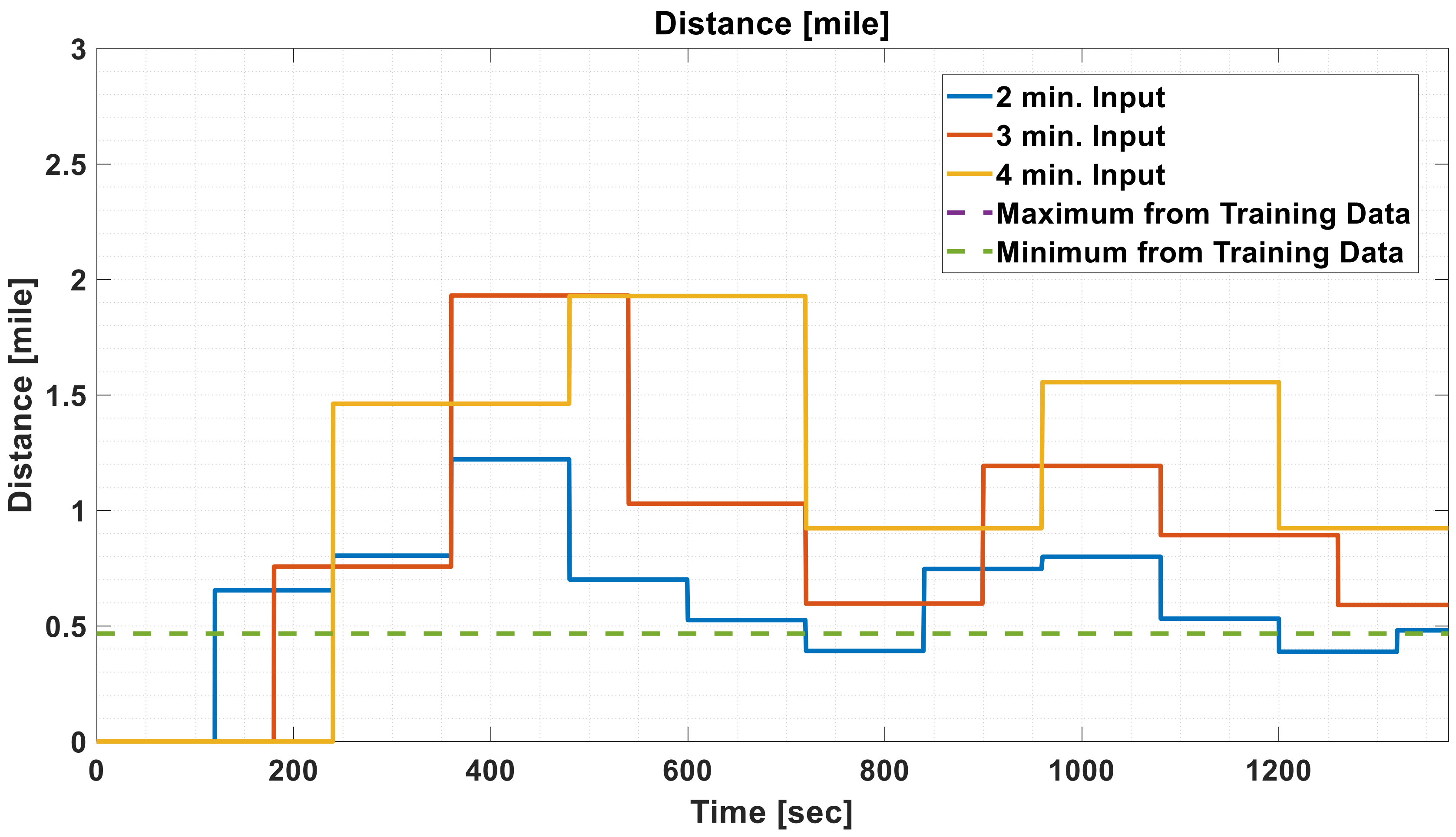


Figure 83: Zoomed-In View of FTP72 Drive Cycle – ANN Input of Distance

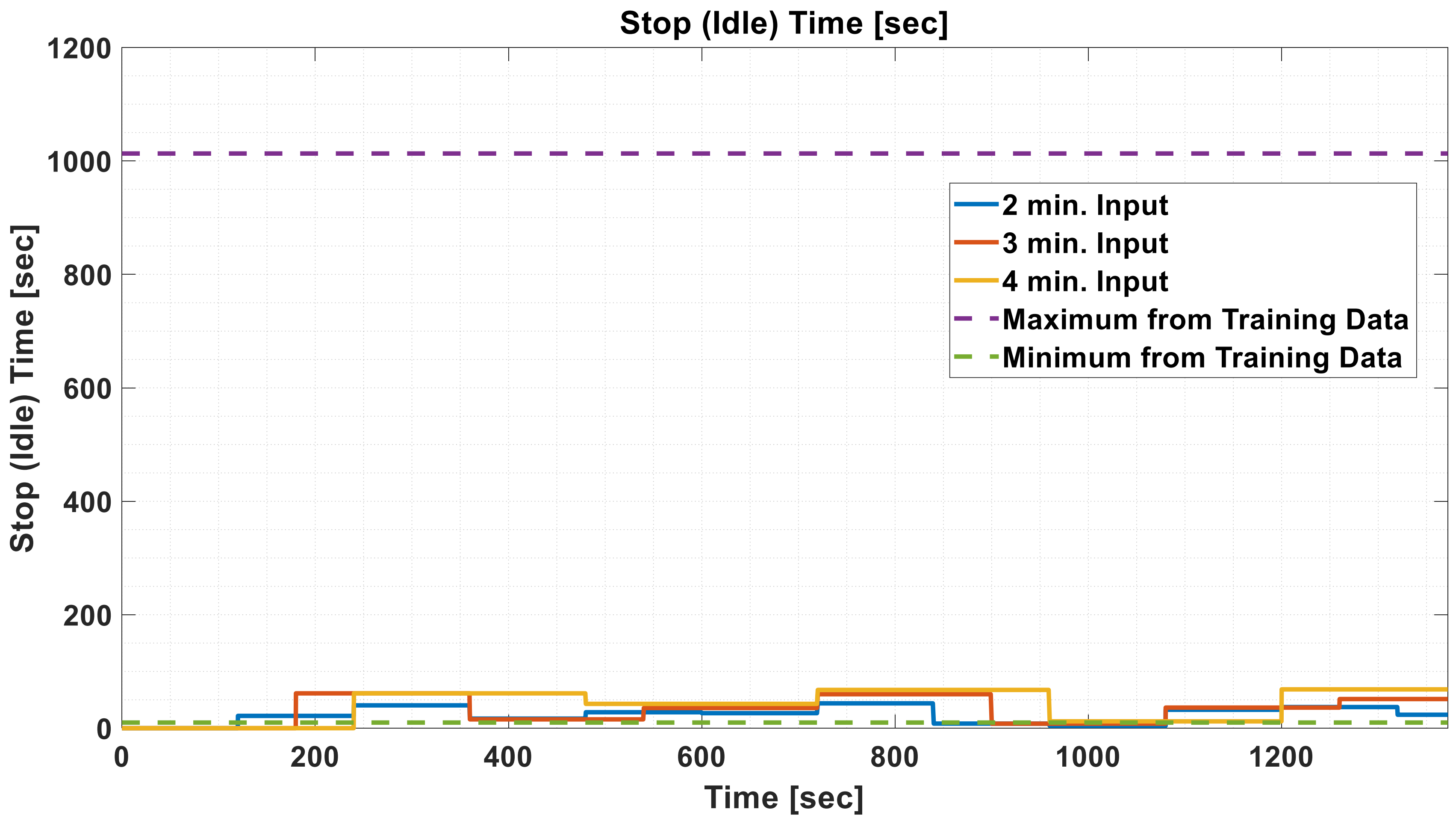


Figure 84: FTP72 Drive Cycle – ANN Input of Idle Time

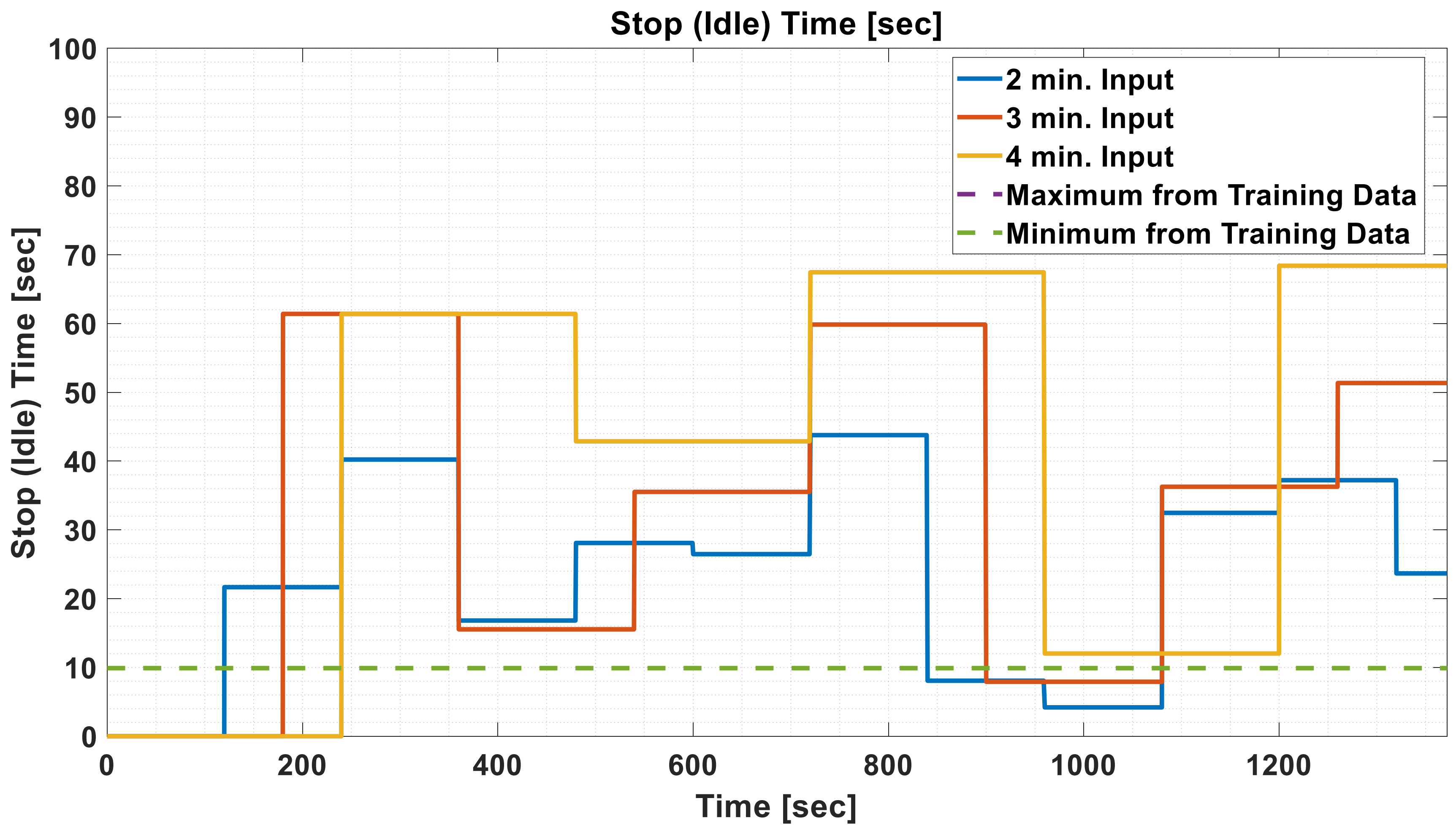


Figure 85: Zoomed-In View of FTP72 Drive Cycle – ANN Input of Idle Time

The 3 and 4-minute time windows relatively consistently put all the inputs within the hyperspace of the training data. This may contribute to the increased performance of the longer time windows. Indeed, the second-best performing set of variance and time window from Table 9 was a 4-minute time window. If the individual drive cycles of Table 9 are examined, it can be seen that the time windows of 3 and 4 minutes often outperform the 2-minute time window.

In summary, a time window (T_w) of 3-minutes and a variance (σ^2) of 80 yield the best ANN-ECMS results over the verification drive cycles when compared to the optimal ECMS results. Figure 86 shows the fuel economy comparison between the ANN-ECMS and optimal ECMS. Figure 87 shows the percent error between the ANN-ECMS and optimal ECMS fuel economy.

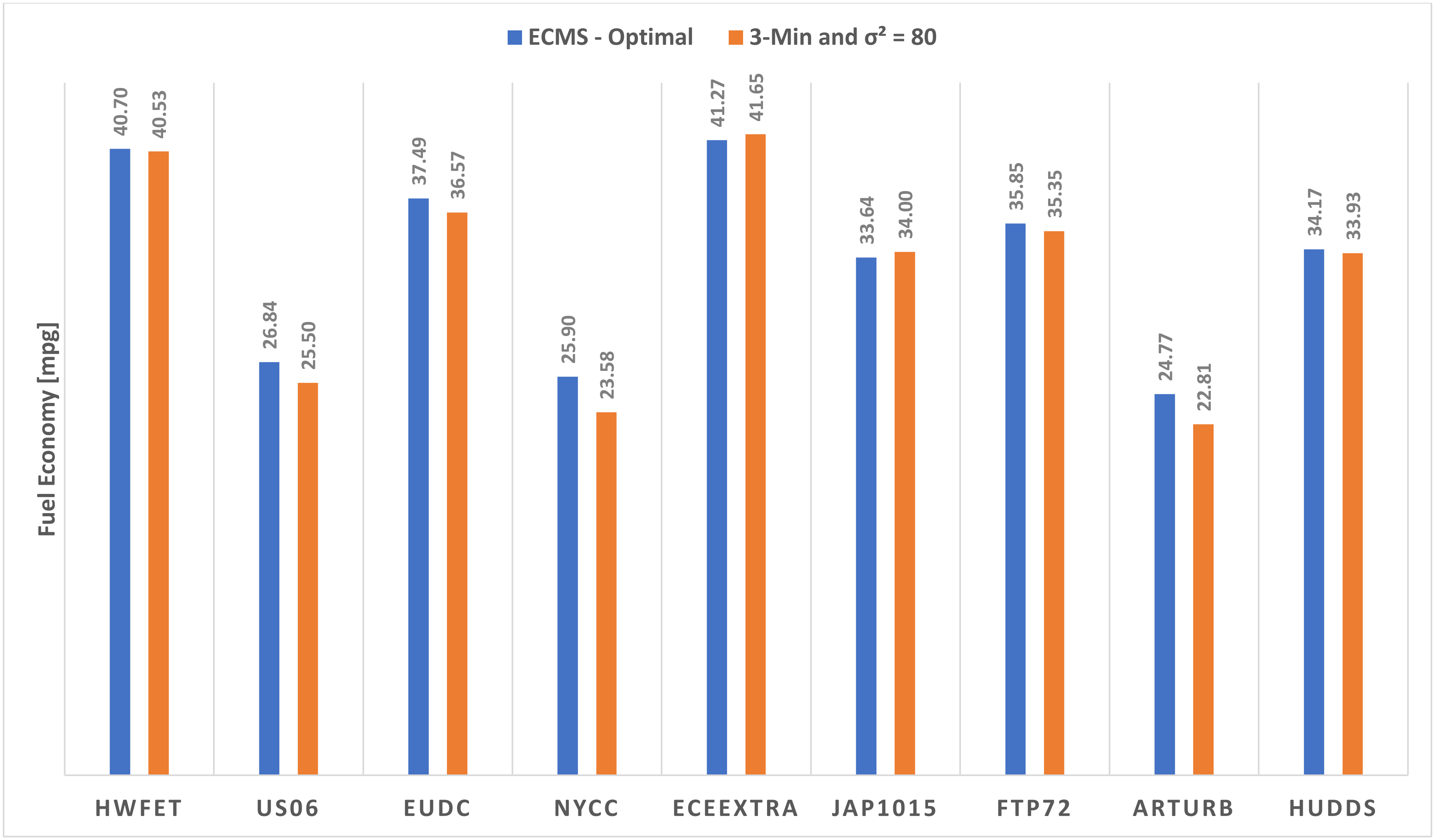


Figure 86: Fuel Economy Comparison of Verification Drive Cycles

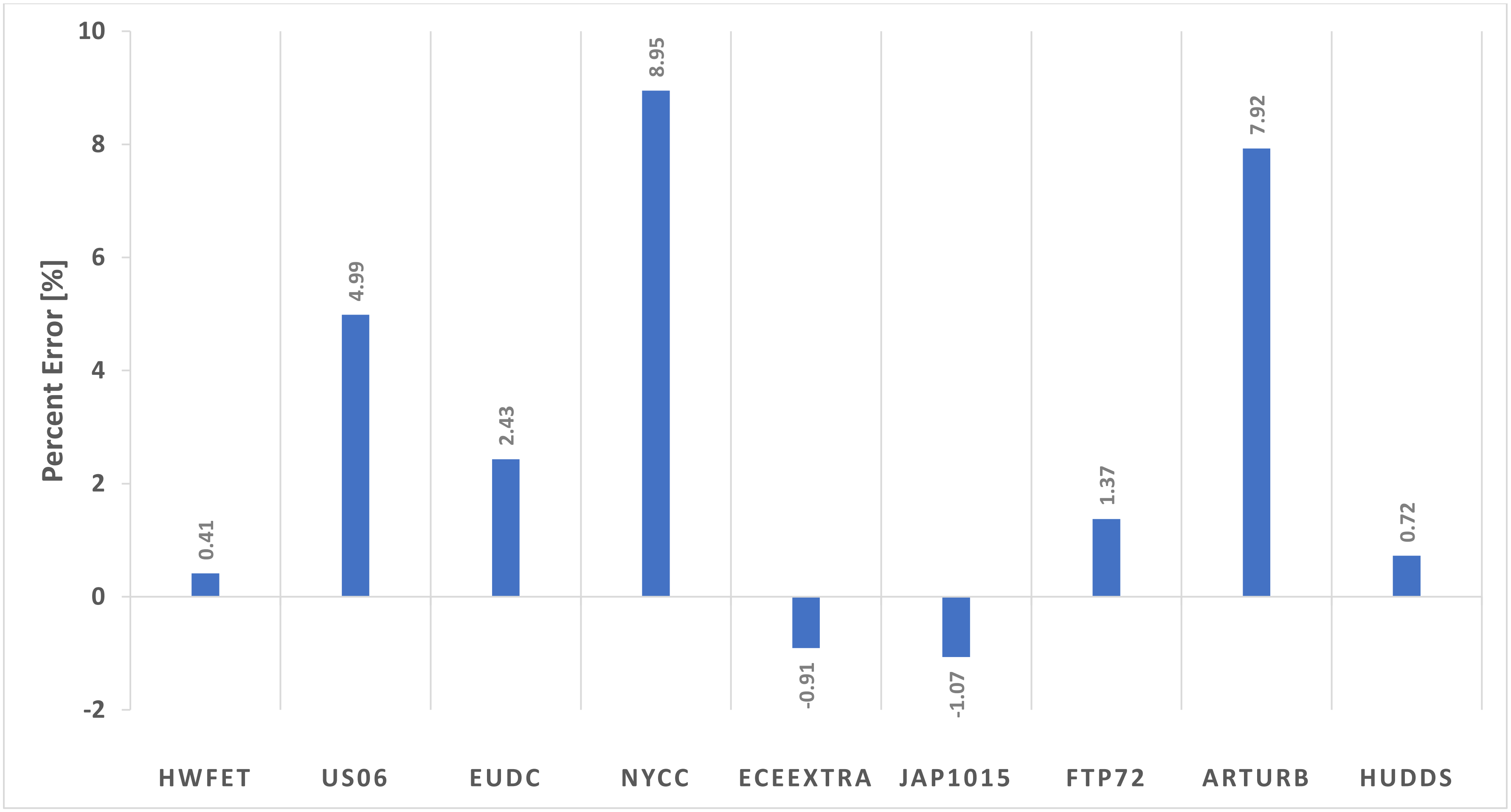


Figure 87: Percent Error of Comparison of Verification Drive Cycles

Of the 9 verification drive cycles, 6 were within +/- 2.43% of the optimal-ECMS. The US06, New York Composite, and Artemis Urban cycles fell outside of this range. The poorer performance of these 3 cycles is attributed to the inputs violating the hyperspace of the training data. The inputs can be seen in Appendix B: RBF ANN Hyperspace Violations.

In the work of Gu et. al [18], a number of drive cycles were evaluated for fuel economy using an A-ECMS and compared to the optimal ECMS using percent improvement. This work also examined a past time window of driving conditions. Based on the past driving conditions, one of four predefined equivalence factors were used [18]. Of the drive cycles analyzed by Gu, 6 were also used in the training data of the ANN used in this work. A comparison can be made between the percent improvements seen by Gu using the A-ECMS and the improvements of the ANN-ECMS. The percent improvement comparison is shown in Figure 88, where positive percentages correspond to performance results that exceed the optimal ECMS and negative percentages correspond to performance results the fall short of the optimal ECMS.

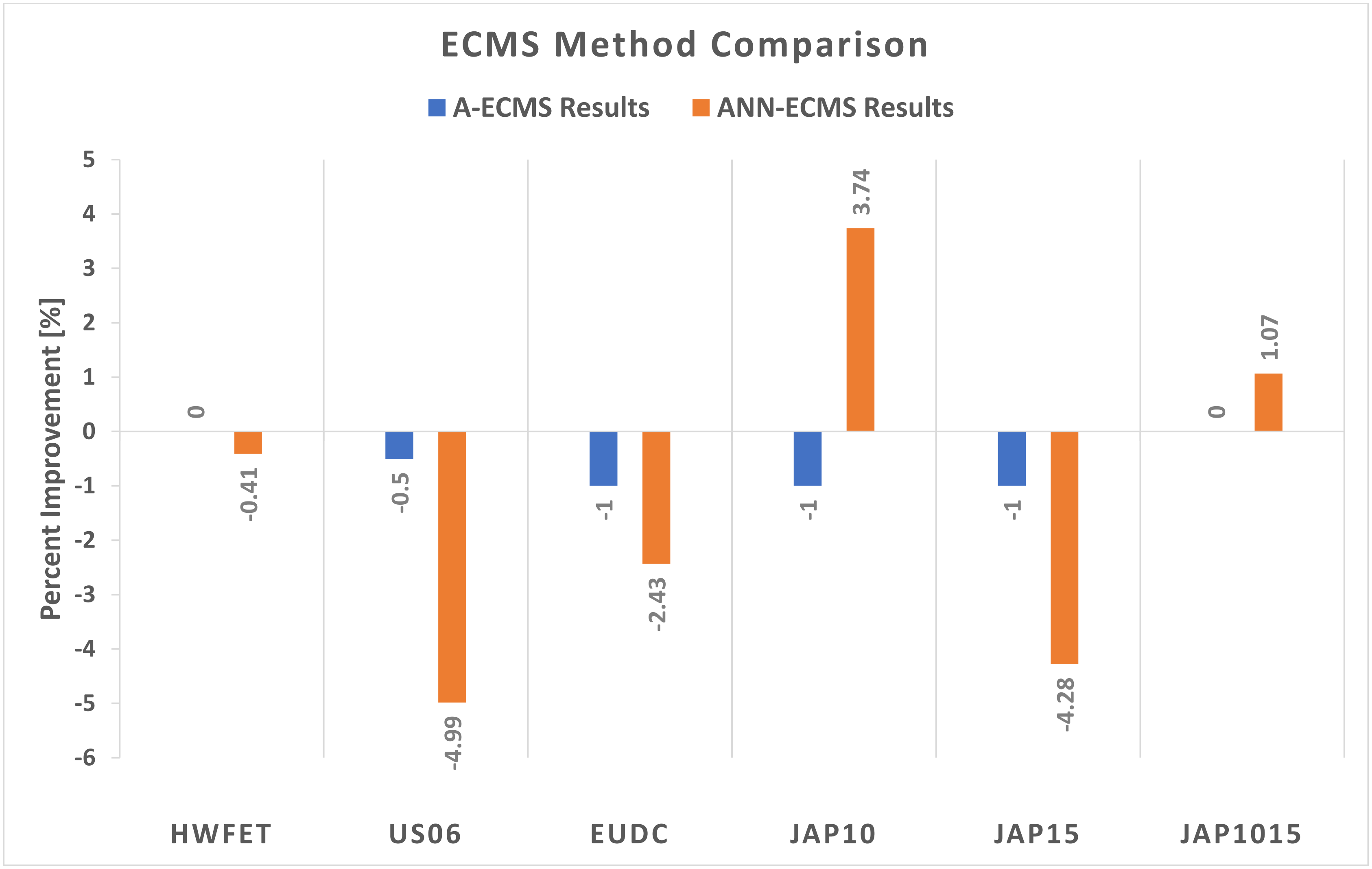


Figure 88: Percent Improvements Comparison of A-ECMS and ANN-ECMS

The comparison in Figure 88 shows that the A-ECMS outperformed the ANN-ECMS in 4 out of the 6 cycles that were compared. The ANN-ECMS outperformed the A-ECMS by 1.07% and 4.74% in the other two drive cycles. This indicates that the ANN-ECMS has great potential. In a later section, recommendations are made on how the ANN-ECMS could be improved to increase its performance.

The reason for better performance of the A-ECMS over 4 of the 6 drive cycles could be attributed to the additional parameters used by Gu et.al to characterize the drive cycles. A total of 21 parameters were used, as opposed to 9 used in this work.

In the work of Gu et. al [18] and Jeon et. al [13], time windows were used to examine past driving conditions and update control parameters. Gu et. al and Jeon et. al used 21 and 24 characterization parameters, respectively, to define the driving conditions. From a computational perspective, both Gu et. al and Jeon et. al claimed that their methods of examining the past time window were simple enough to be implemented with a real-time controller. Based on the number of characterization parameters, the work described in this thesis should be less computationally intensive – only 9 parameters are needed to update the ECMS control parameter.

5.3. Validation Data.

To validate the performance of the ANN, 5 drive cycles were evaluated which had not been used to train the RBF ANN. The cycles chosen offer a broad range of characteristics – acceleration levels, speeds and idle times. The cycles are shown below:

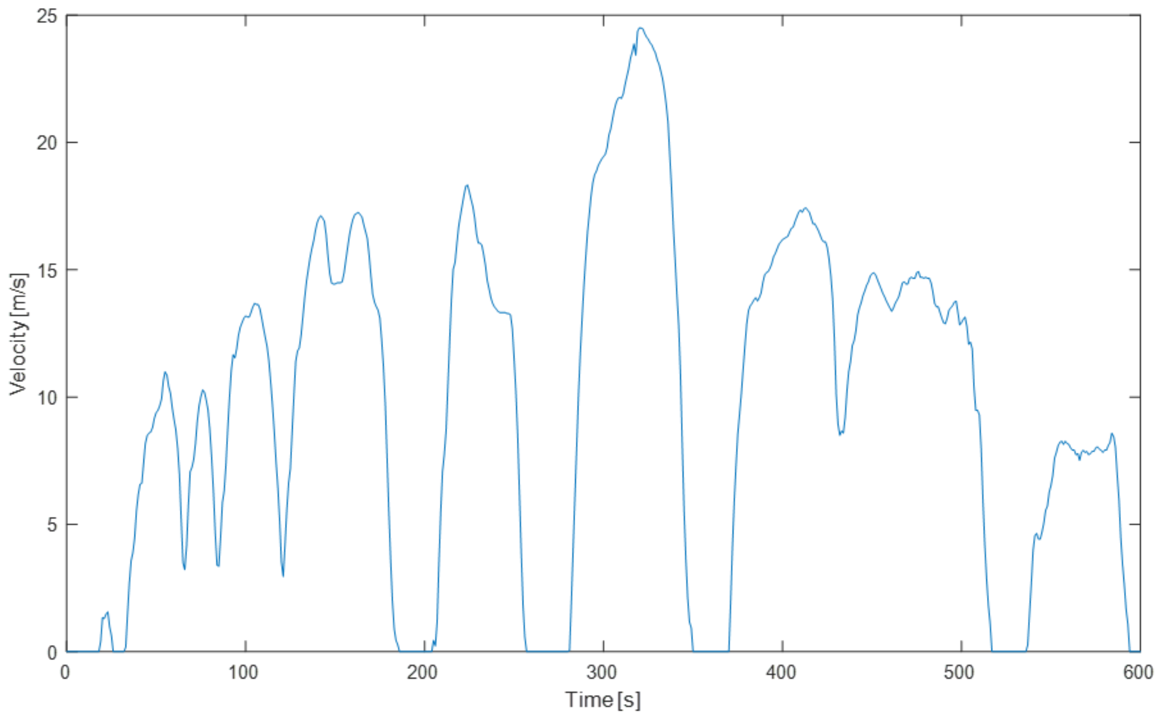


Figure 89: Supplemental FTP Driving Cycle (SC03)

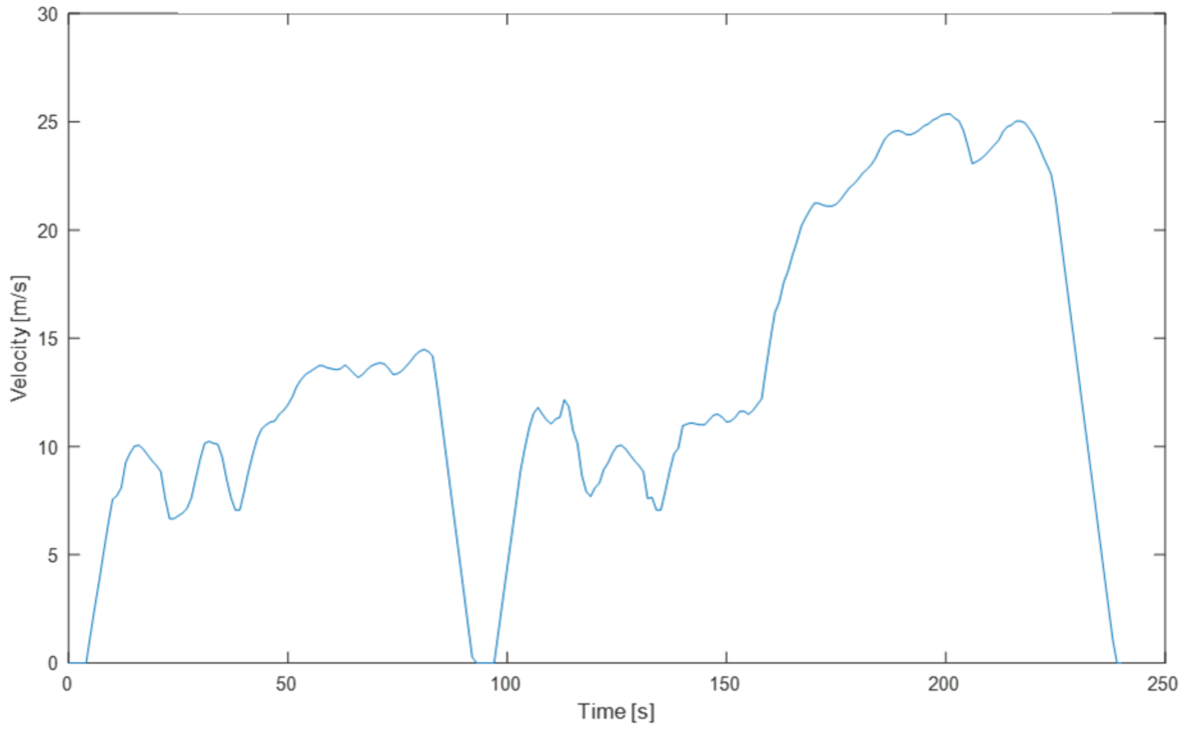


Figure 90: IM240 Inspection and Maintenance Driving Cycle

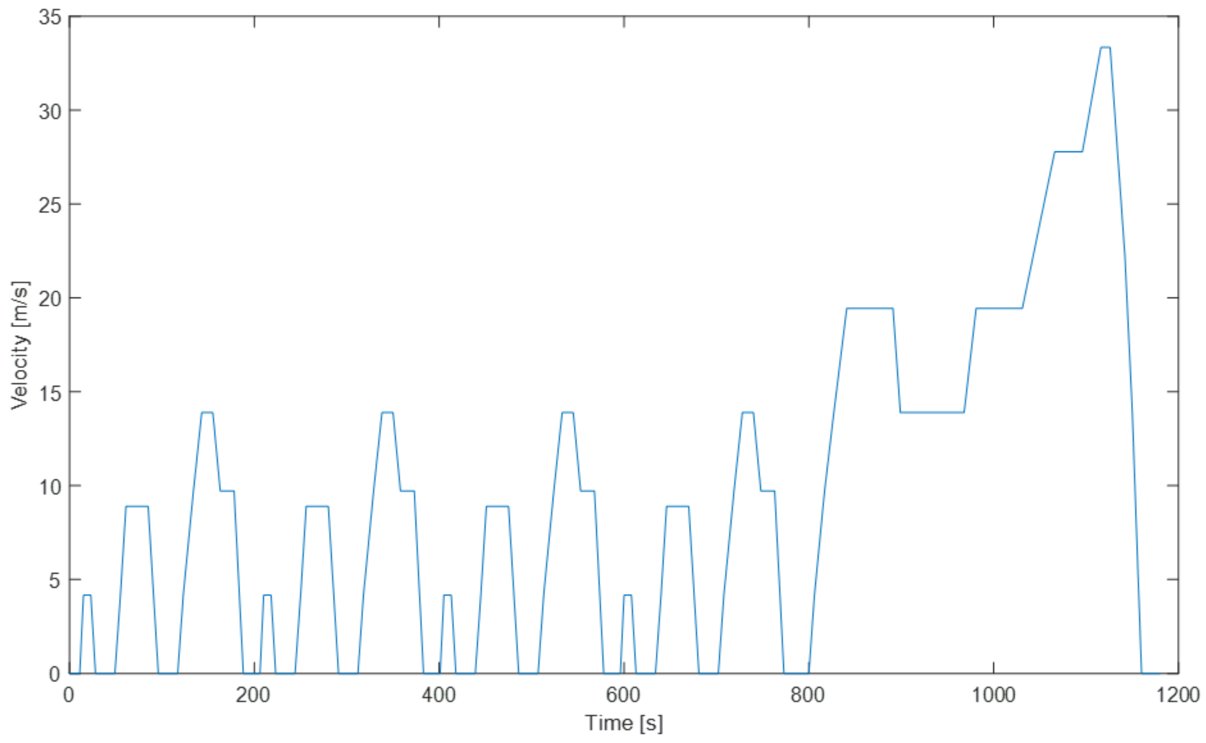


Figure 91: New European Driving Cycle (NEDC)

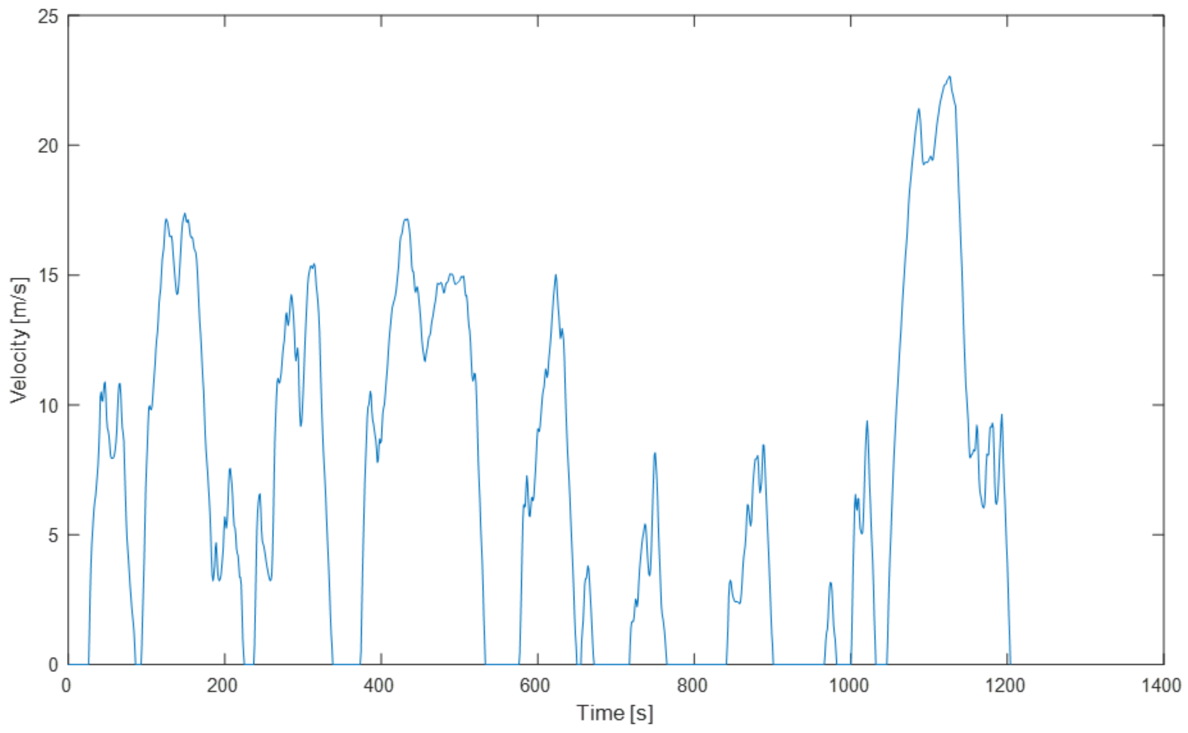


Figure 92: Japanese chassis dynamometer test cycle (JC08)

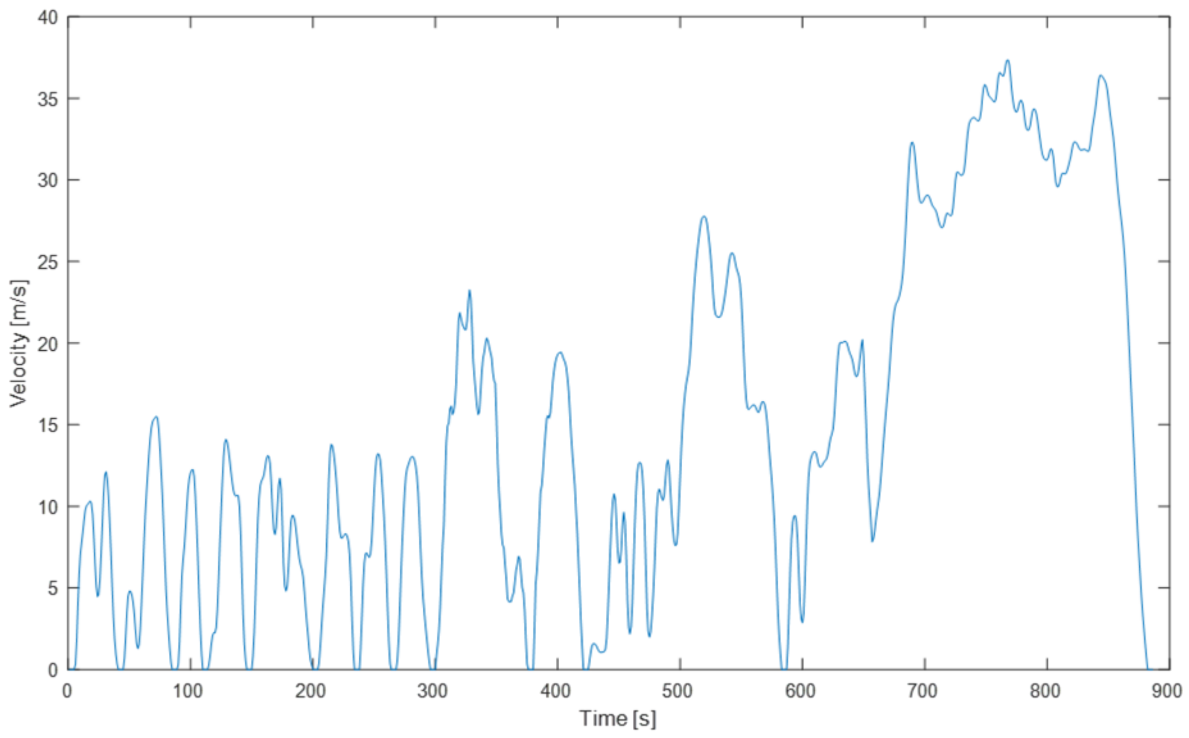


Figure 93: RTS 95 Drive Cycle

Comparison of the ANN-ECMS and Optimal ECMS fuel economy results are shown in Figure 94. Figure 95 shows the percent error between the optimal ECMS fuel economy and the fuel economy obtained using ANN-ECMS.

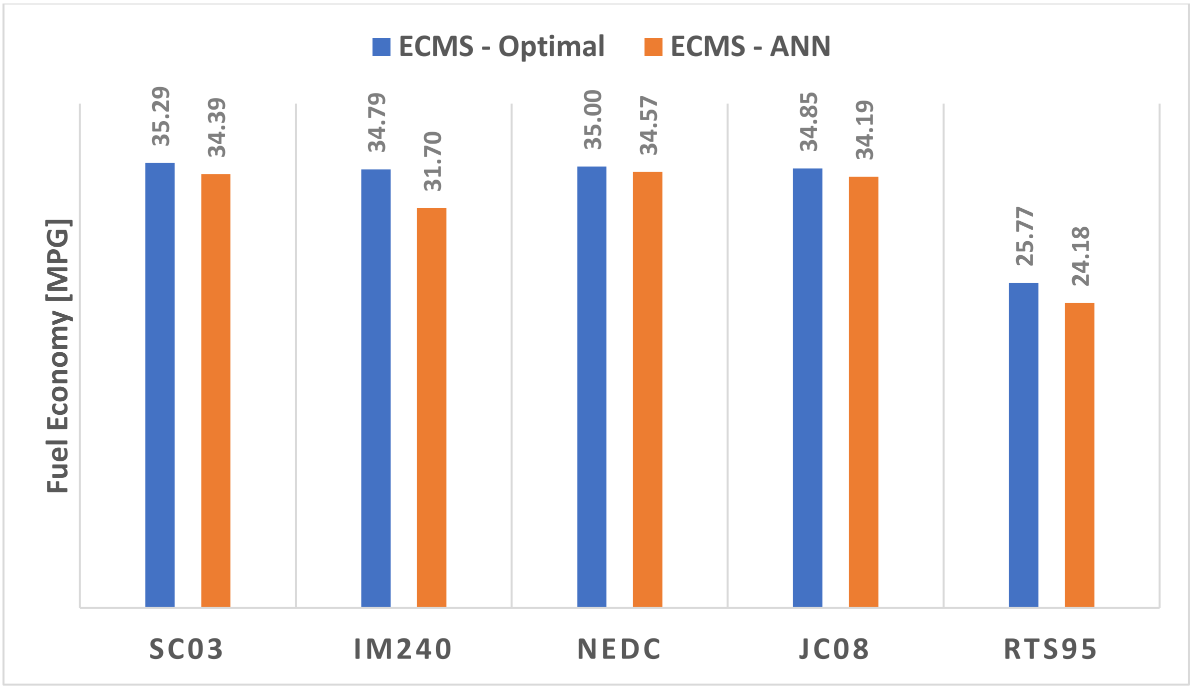


Figure 94: Comparison of Fuel Economy Results of Validation Drive Cycles Between Optimal ECMS and ECMS-ANN

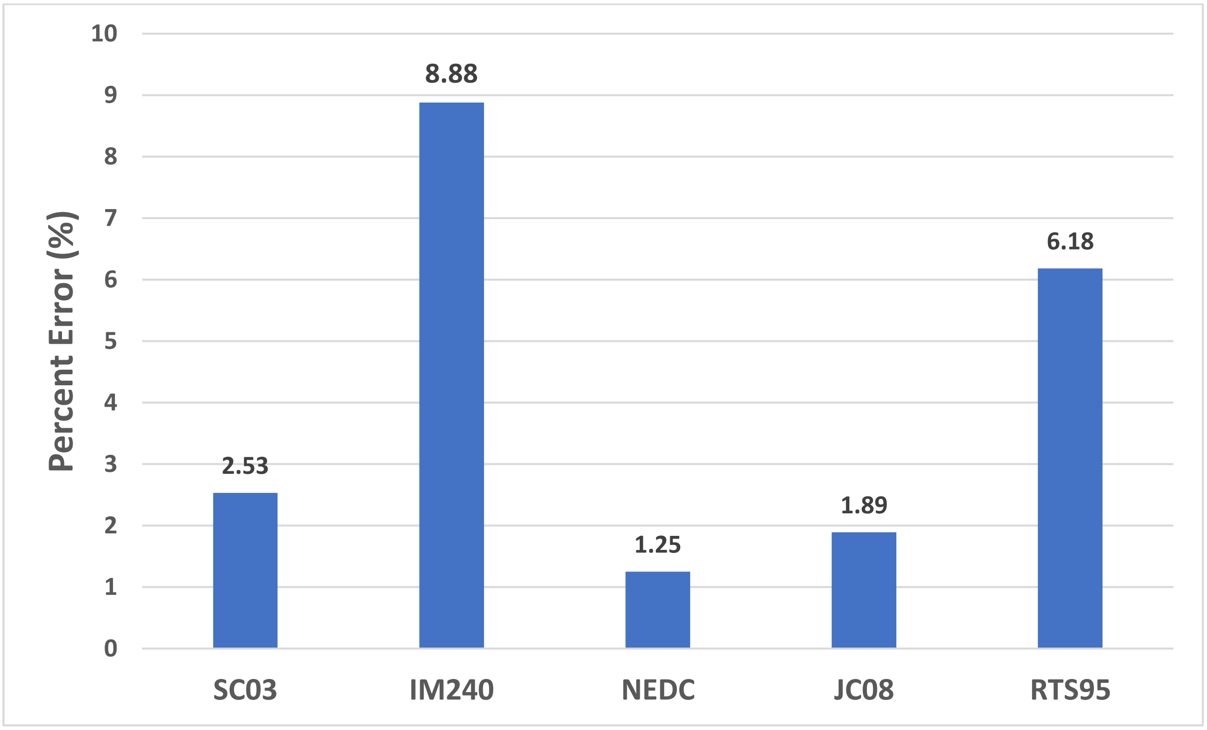


Figure 95: Percent Error of Validation Drive Cycles Between Optimal ECMS and ANN-ECMS Fuel Economy

These results show that the ANN-ECMS performed well in 3 out of the 5 validation drive cycles. With the worst performing showing a percent error of 8.88% and the best performing with a percent error of only 1.25%.

Of the 5 validation drive cycles, the IM240 and RTS95 cycles showed the poorest performance. The poor performance of the RTS95 drive cycle is attributed to violations of the hyperspace. The inputs of this cycle are shown in Appendix B: RBF ANN Hyperspace Violations. The inputs of the IM240 cycle never violate the hyperspace. The poor performance is attributed to the shortness of this cycle. A single time window covers more than half of the drive cycle. Consequently, there is only one chance for the inputs to update. Suggestions for improvements are discussed in the following section.

6. Conclusions and Recommendations

In conclusion, the objective of this work was to develop an ANN to implement with ECMS. An RBF ANN was selected due to the quick training capabilities of the RBF. The end goal was to achieve fuel economy results close to the optimal baseline achievable with ordinary ECMS. The performance of ECMS is dependent on an equivalence factor that must be determined offline with *a priori* knowledge of the drive cycle in order to achieve optimal results. Different driving conditions require different equivalence factors to achieve maximum fuel economy. The RBF ANN examines a past time window of driving conditions to make decisions on how to update the equivalence factor without having future knowledge of the upcoming driving conditions.

A total of 30 different drive cycles were characterized and the optimal fuel economy, using ECMS, was found for each cycle. A total of 9 characteristics from each drive cycle was used to train the RBF ANN. A sensitivity analysis was performed over the internal RBF parameter of variance, which affects how aggressively the ANN interpolates and extrapolates. Additionally, an analysis of the length of the time windows was performed. Time windows of 2, 3, and 4 minutes were tested to determine the effect on fuel economy. Ultimately, it was observed that a variance of 80, and a 3-minue time window resulted in the best performance.

A total of 9 drive cycles from the training data were used to verify the performance of the ANN-ECMS. These drive cycles encompassed a broad range of the characteristics that were used to parameterize each cycle in the training data. The optimal fuel economy was achieved within +/- 2.43% for 6 of the 9 verification drive cycles. The worst performing drive cycle was 8.95% below the optimal, and the best performing was 1.07% above the optimal.

The performance of the ANN-ECMS over the verification drive cycles was compared to an A-ECMS developed by Gu et. al, who also updated the equivalence factor based on a time window of past driving conditions. The method developed by Bo Gu selected from a predefined list of 4 equivalence factors. A comparison over 6 drive cycles showed that the results of the A-ECMS outperformed the ANN-ECMS for 4 of the 6 cycles. In the other two cycles, the ANN-ECMS outperformed the A-ECMS by 1.07% and 4.74%. The better performance of the A-ECMS over the 4 drive cycles is attributed to the greater amount of drive cycle characteristics used to update the A-ECMS.

The ANN-ECMS performance was validated using 5 drive cycles that were not included in the training data of the RBF ANN. Of the 5 drive cycles used for validation, 3 of the 5 achieved a percent error within 2.53% of the results from the optimal ECMS. The poorer performance of the remaining two drive cycles is attributed to the inputs of these cycles being outside of the hyperspace of training data used to train the RBF ANN.

These results could be improved upon, and therefore, merit future work. For future work, it is recommended that different drive cycles be characterized into a few different classes. The ANN could then be trained with drive cycles from the different classes and optimal variances and time windows could be determined for each class. For example, if 3 classes of drive cycles were defined, then the ANN could be trained with drive cycle sets from each class and the best variance and time window could be determined for each class. In real-time operation, the corresponding time-window and variance value would be applied when the ANN determines which drive cycle class the current driving conditions reflect.

Appendix A: MATLAB Code

MATLAB code used to train the RBF ANN:

```
% This script loads in training data and conditions it for use in the
% MATLAB function block that recieves the ANN inputs.

% Load in either rough data set or smooth data set
load TD_Combined_3.mat

% Columns of TD_Combined_3 Matrix
% TD(:,1) = P_Jerk;
% TD(:,2) = N_Jerk;
% TD(:,3) = AAccel;
% TD(:,4) = DDeccel;
% TD(:,5) = AvgSpd;
% TD(:,6) = AvgRunSpd;
% TD(:,7) = Distance;
% TD(:,8) = Max_Vel;
% TD(:,9) = StopTime;
% TD(:,10) = max_mpg_vals;
% TD(:,11) = max_s_vals;

x = TD_Combined_3;
y = TD_Combined_3(:,11); % The s-values

% taking out the, uneeded values in training data
x(:,11) = [];   % s_val is gone
x(:,10) = [];   % MPG is gone
x(:,6) = [];    % AvgRunSpd is gone

% Defining placeholder matrix for use in MATLAB Function block
Kk = zeros(1,30);
Yy = zeros(1,30);
sz = size(x);
y0 = zeros(1,30);

var = 80;   % Variance parameter
const = 1/(sqrt(2*pi*(var)));   % Constant calculated using Variance

% For loop defining center vectors
for i = 1:sz(1)
    c = x(i,:);
    Cc(i,:) = c;
    for j = 1:sz(1)
        V = x(j,:) - c;
        D(i,j) = sqrt(V*V');
    end
end

% Gaussian activation function
```

```
activation = const*(exp(((-D.*D)/(2*(var)))));

% Final matrix of weights
Zz_Combined = activation\y;
```

MATLAB code in RBF ANN Function Block.

```
function y = fcn(AvgPosAccel, AvgNegAccel, AvgPosJerk, AvgNegJerk, AvgSpd, MaxVel, Cc,
var, const, Kk, Yy, y0, Zz_Combined, Dist, StopTime)

% x is the [1X8] input vector
x = [AvgPosJerk AvgNegJerk AvgPosAccel AvgNegAccel AvgSpd Dist MaxVel StopTime];

% i goes from 1 to the number of hidden layers
for i = 1:30

    % This is the implementation of the ANN. Cc is the matrix of center
    % vectors in the hidden neuron.

    % x_j = ||x_bar - cj_bar||

    % V is the difference between the input and the first center vector.
    V = x - Cc(i,:);   %x -> [1X7] C(:,i)' -> [1X7]

    % Yy is norm of V.
    Yy(i) = sqrt(V*V');

    % Kk is activation function.
    Kk(i) = const*exp(-(Yy(i))^2/(2*var));

    % y0 is the output of the output neuron.
    y0(i) = Zz_Combined(i)*Kk(i);
end

% y is the summation of y0
y = sum(y0);
```

Published with MATLAB® R2018b

Appendix B: RBF ANN Hyperspace Violations

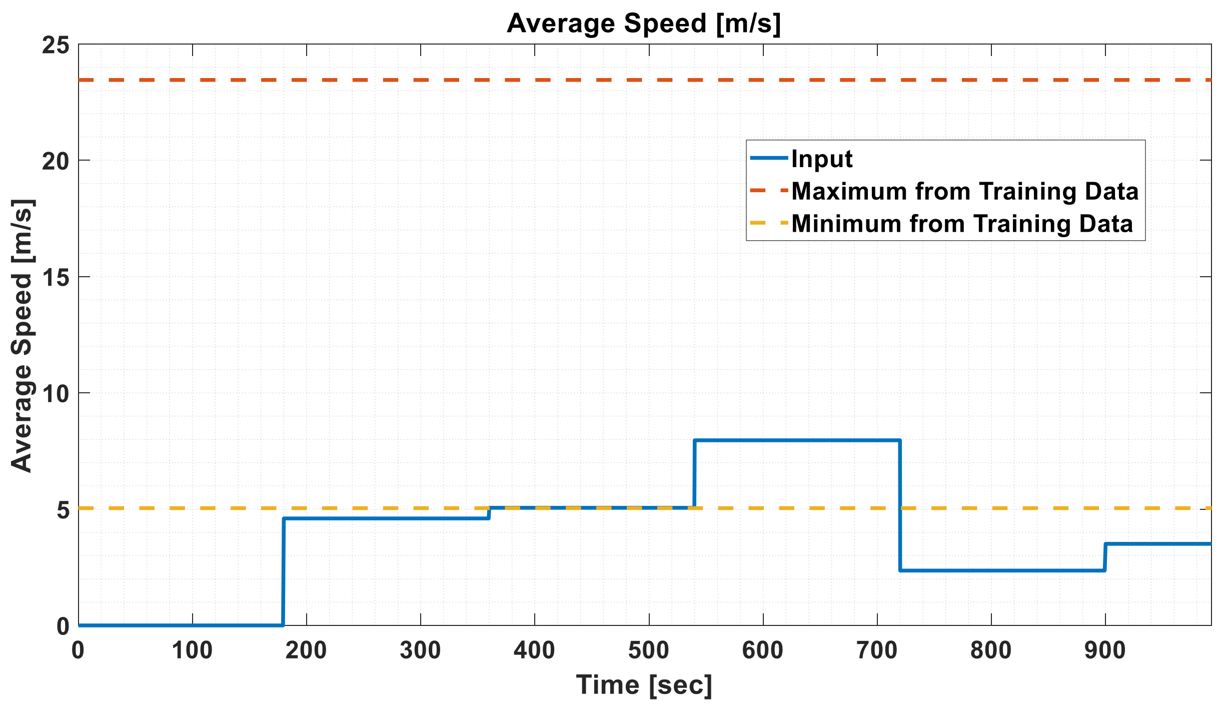


Figure 96: Artemis Urban Cycle Average Speed Input

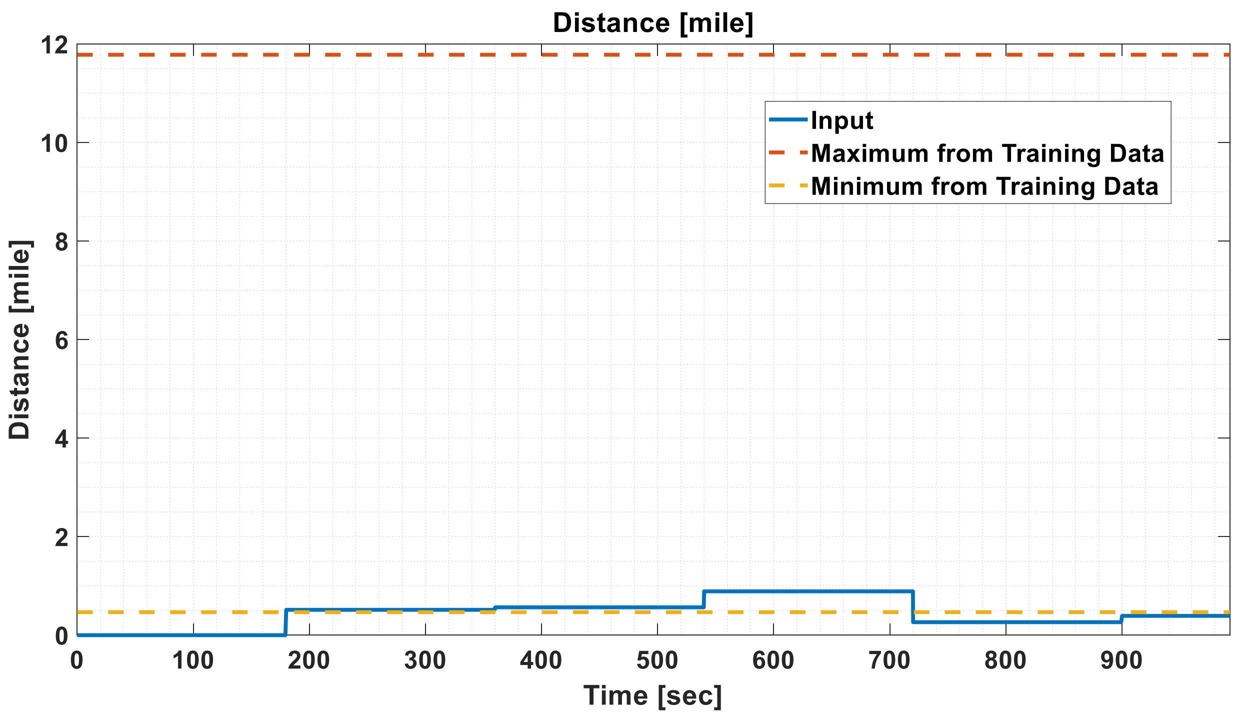


Figure 97: Artemis Urban Cycle Distance Input

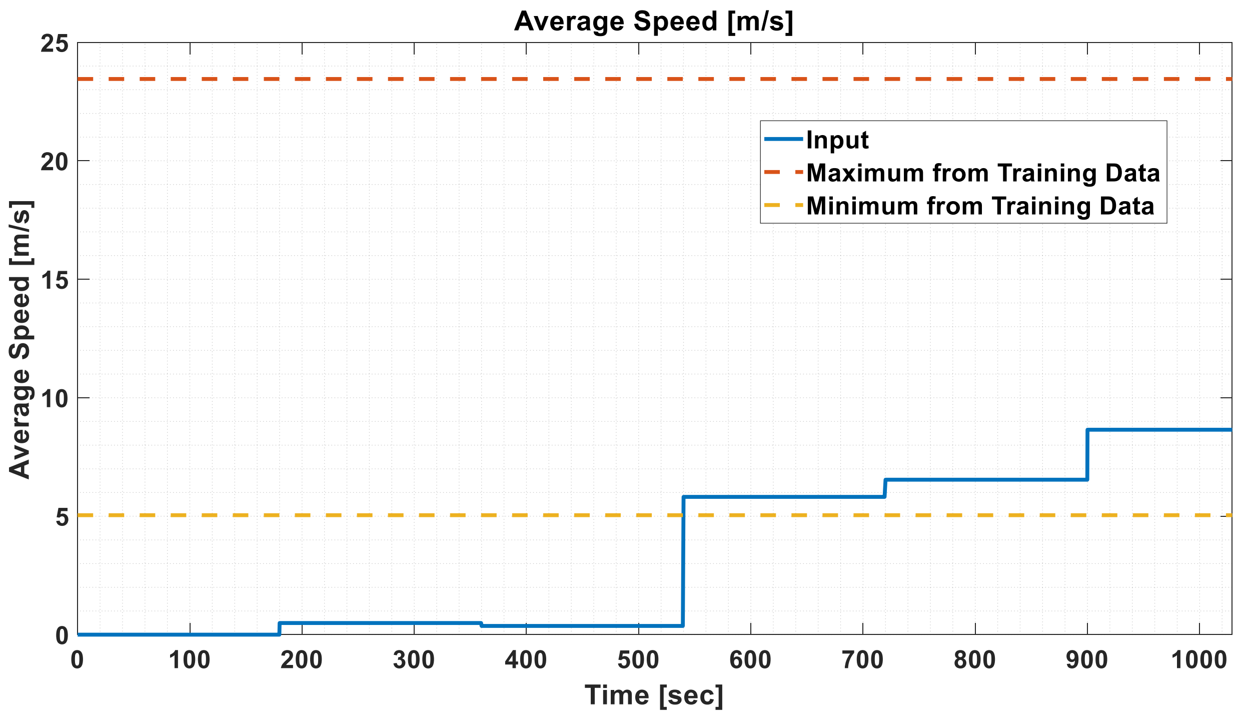


Figure 98: New York Composite Cycle Average Speed Input

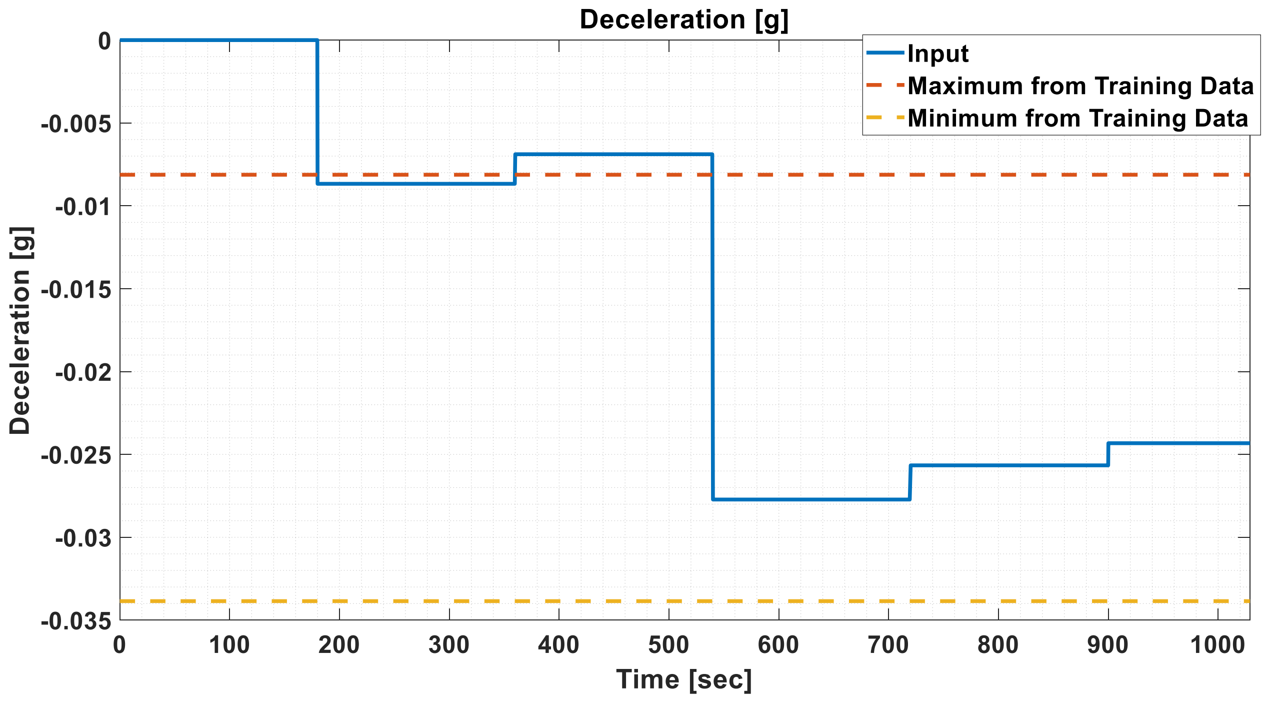


Figure 99: New York Composite Cycle Deceleration Input

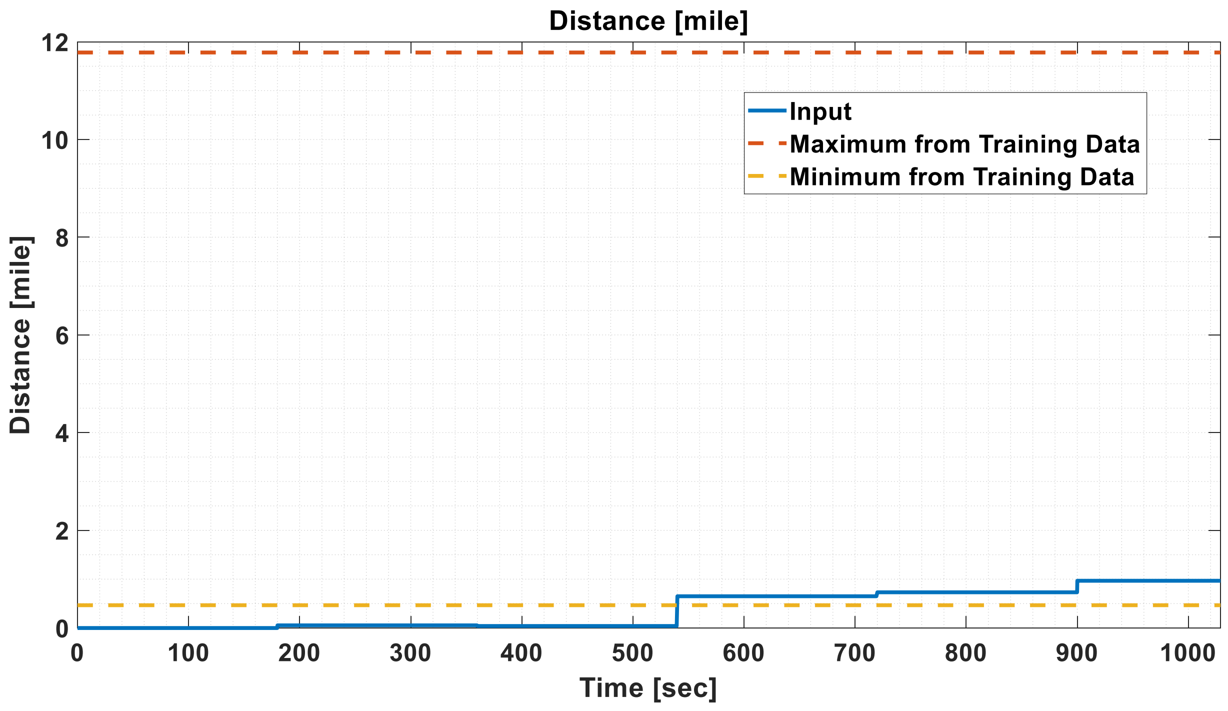


Figure 100: New York Composite Cycle Distance Input

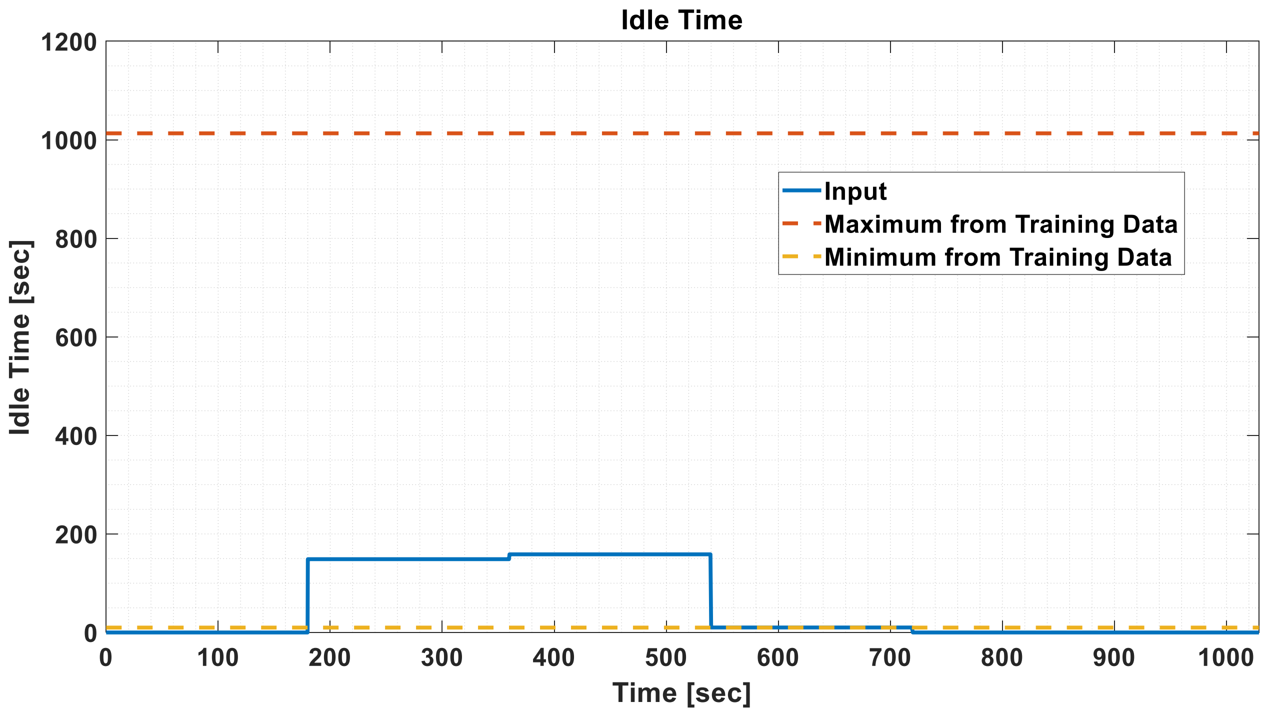


Figure 101: New York Composite Cycle Idle Time Input

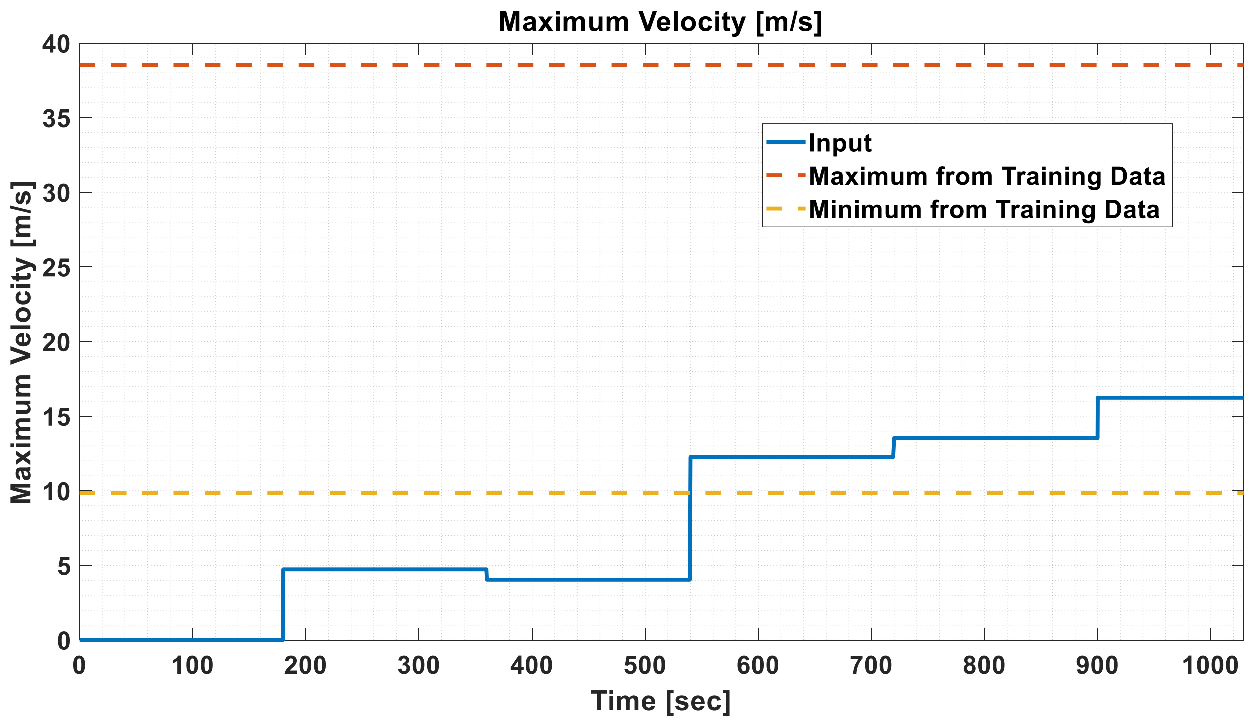


Figure 102: New York Composite Cycle Maximum Velocity Input

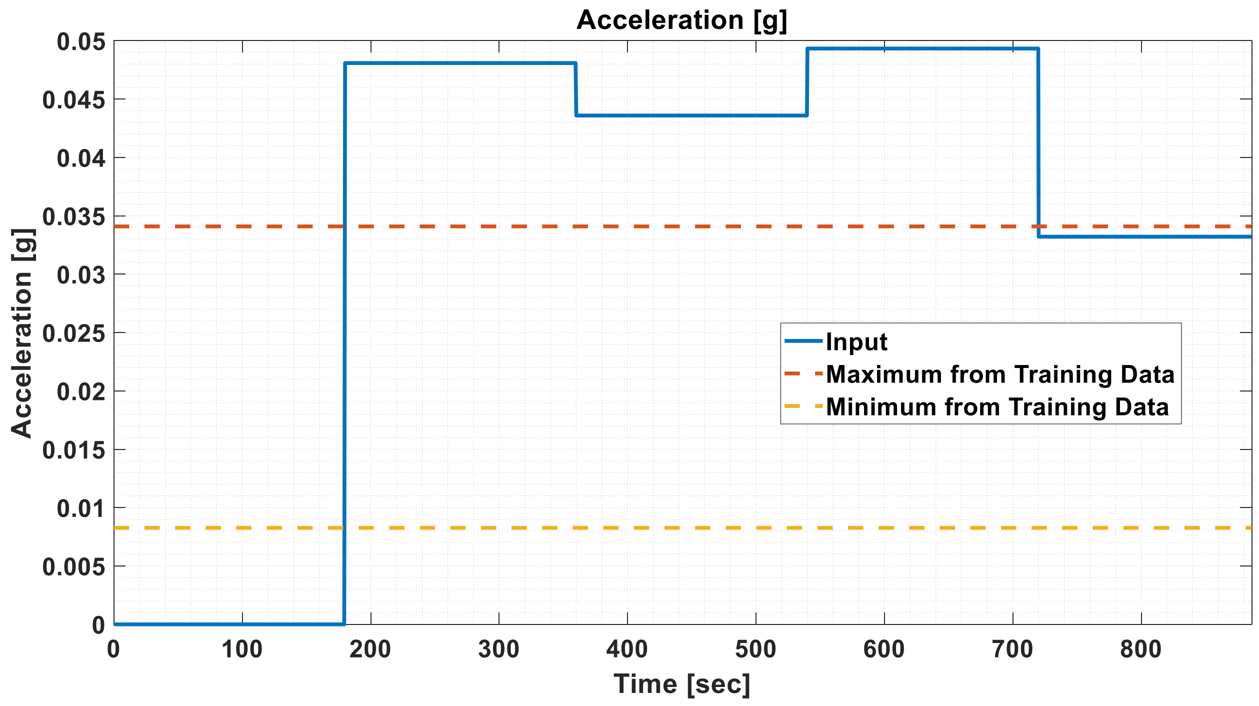


Figure 103: RTS95 Cycle Acceleration Input

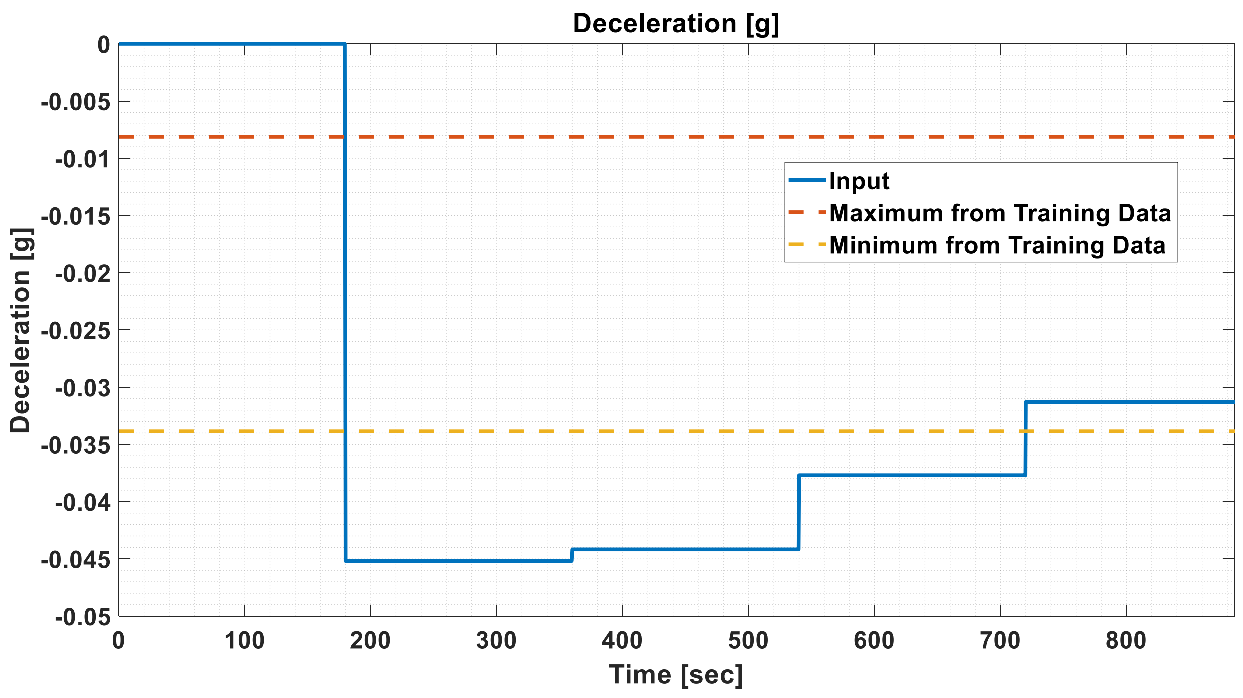


Figure 104: RTS95 Cycle Deceleration Input

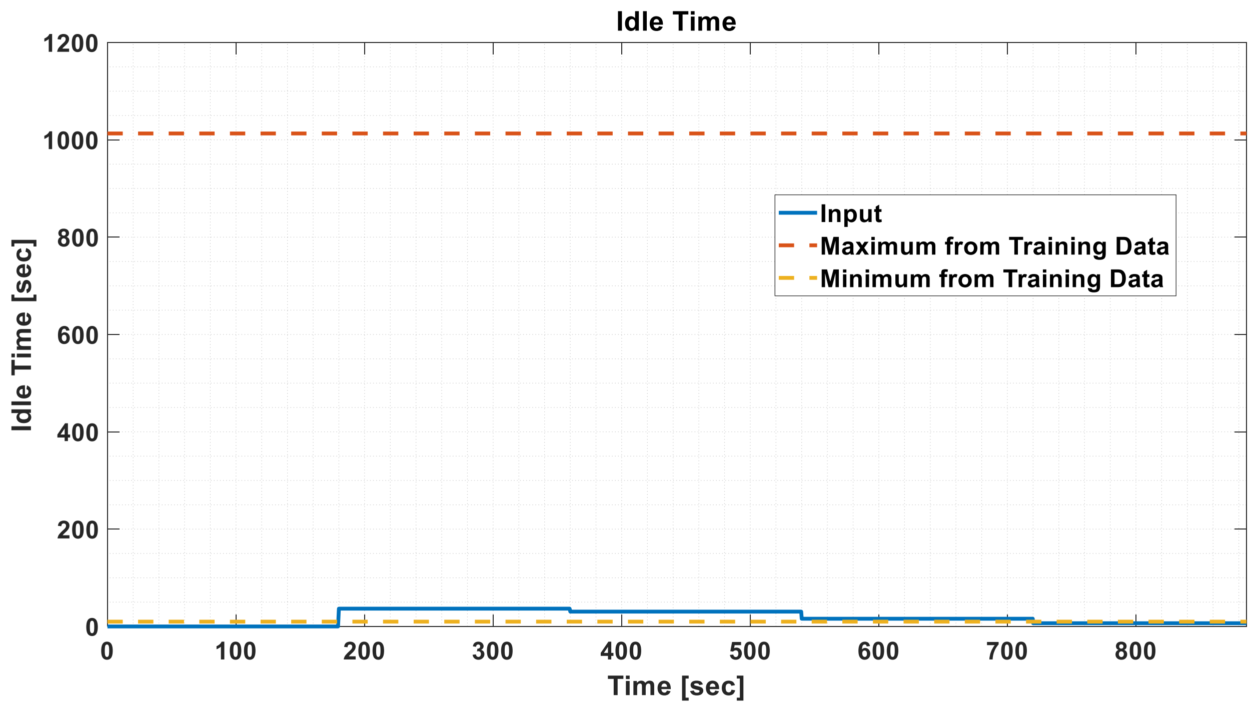


Figure 105: RTS95 Cycle Idle Time Input

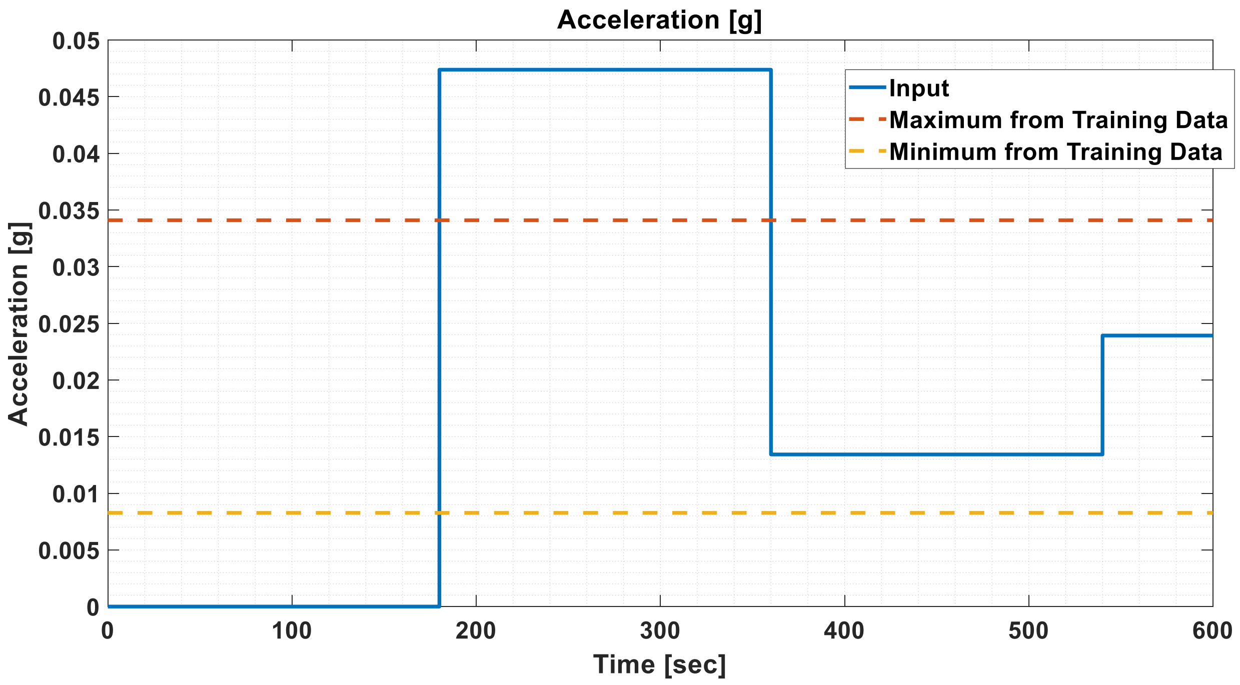


Figure 106: US06 Cycle Acceleration Input

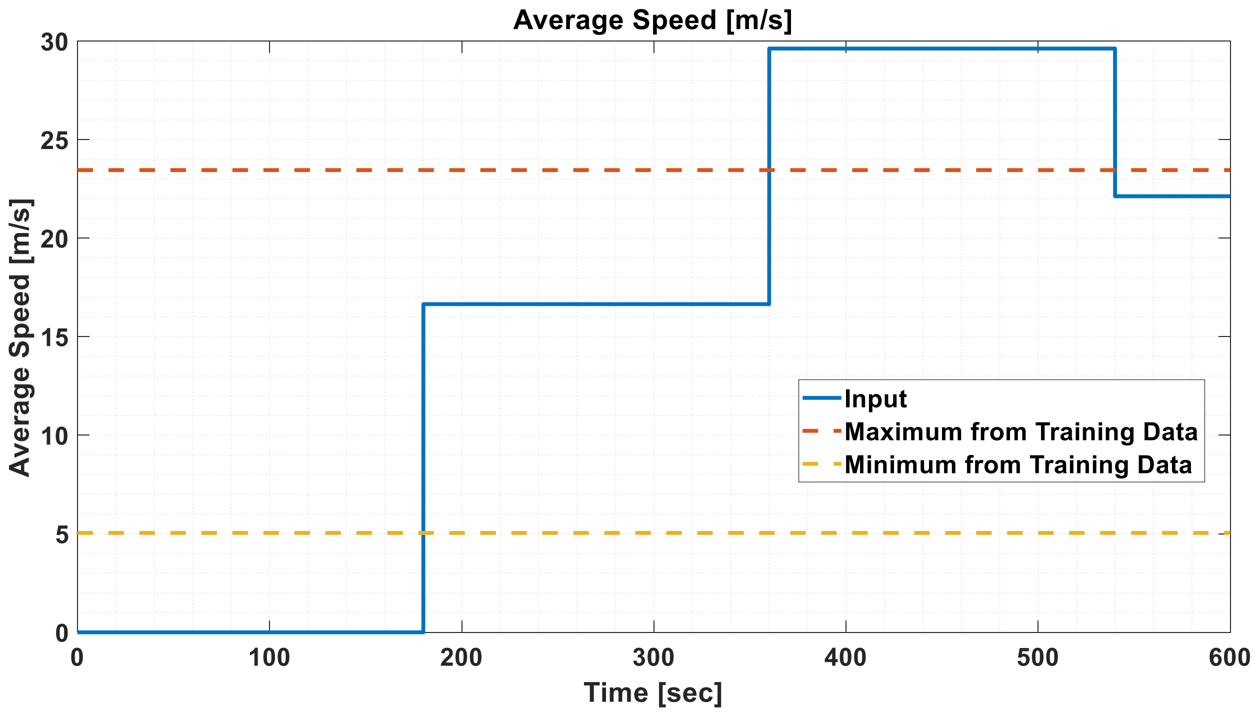


Figure 107: US06 Cycle Average Speed Input

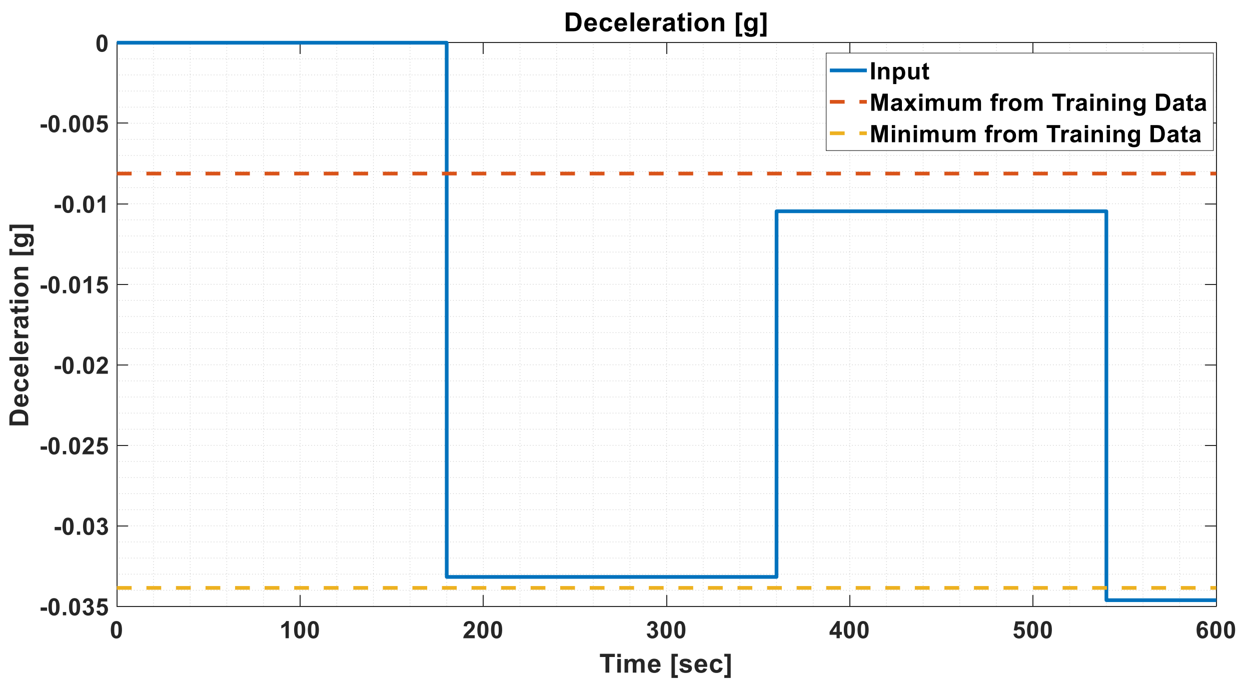


Figure 108: US06 Cycle Deceleration Input

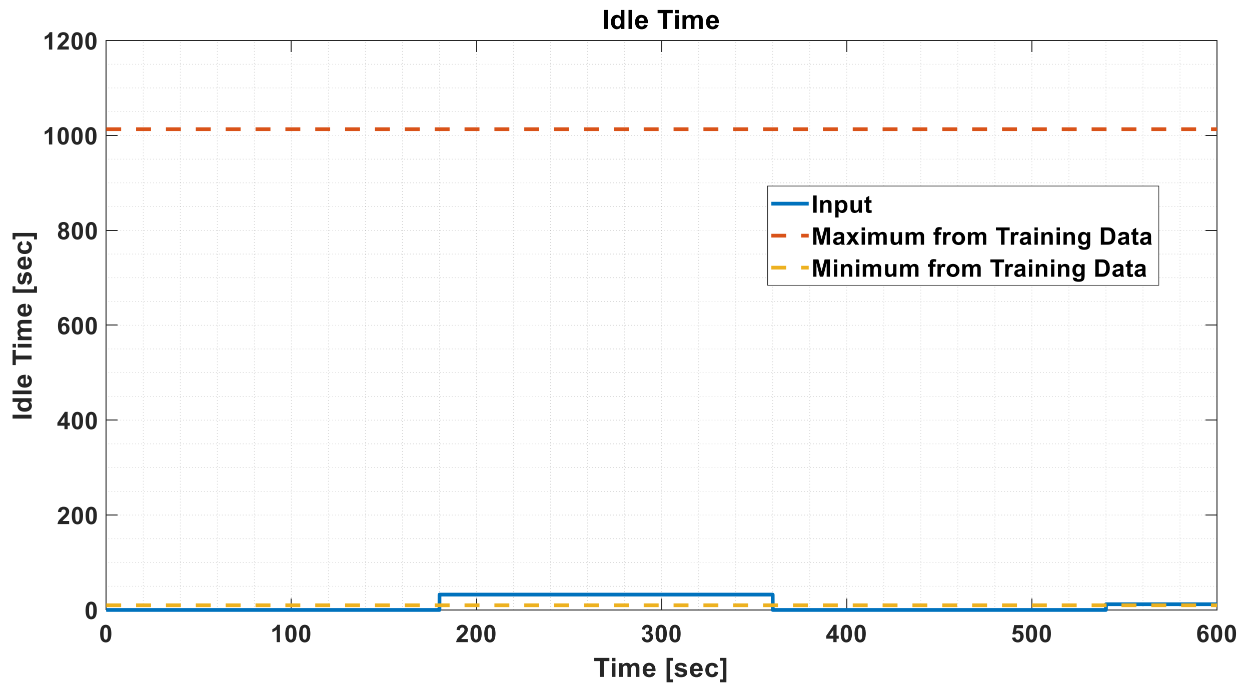


Figure 109: US06 Cycle Idle Time Input

References

[1] G. Rizzoni, "Hybrid Electric Vehicles Energy Management Strategies," Columbus, 2016.

[2] EPA, "Greenhouse Gas Emissions," [Online]. Available: https://www.epa.gov/ghgemissions/sources-greenhouse-gas-emissions.

[3] H. P. Jinming Liu, "Modeling and Control of a Power-Split Hybrid Vehicle," *IEEE Transactions on Control Systems Technology,* 2008.

[4] Argonne National Laboratory, U.S. Department of Energy, "AVTC History," 2020. [Online]. Available: https://avtcseries.org/avtc-history/. [Accessed 14 November 2020].

[5] C.-C. Lin, H. Peng and J.-M. K. Grizzle, "Energy management strategy for a parallel hybrid electric truck," *Proceedings of the 2001 American Control Conference,* vol. 4, pp. 2878-2883, 2001.

[6] C. Lin, Z. Filipi, Y. Wang, L. Louca, D. Peng and J. S. Assanis, "Integrated, fee-forward hybrid electric vehicle Simulation in simulink and its use for power management studies.," 2001.

[7] A. Sciarretta, M. Back and L. Guzzella, "Optimal control of parallel hybrid electric vehicles.," *IEEE Trans. Control Syst. Tecnol,* vol. 12, no. 3, pp. 352-363, 2004.

[8] H. Peng and J. Kang, "Power management strategy for a parallel hybrid electric truck," *IEEE Transactions on Control Systems Technology,* vol. 10, no. 6, pp. 839-849, 2003.

[9] D. Lee, S. W. Cha, A. Rousseau and N. Kim, "Optimal control strategy for PHEVs using prediciton of future driving schedule," *World Electric Vehicle Journal,* vol. 5, pp. 149-158, 2012.

[10] Z. Chen, C. Yang and S. Fang, "A convolutional neural network-based driving cycle prediction method for plug-in hybrid electric vehicles with bus route," *IEEE Access,* vol. 8, pp. 3522-3264, 2020.

[11] A. Chasse, A. Sciarretta and J. Chauvin, "Online optimal control of a parallel hybrid with costate adaption rule," *IFAC Symposium Advances In Automotive Control,* 2010.

[12] I. Arsie, M. Graziosi, C. Pianese, G. Rizzo and M. Sorrentino, "Optimization of Supervisory Control Strategy for Parallel Hybrid Vehicle with Provisional Load Estimate," Department of Mechanical Engineering - University of Salerno, 2004.

[13] S. Jeon, S. Jo, Y. Park and J. Lee, "Multi-Mode Driving Control of a Parallel Hybrid electric Vehicle Using Driving Pattern Recognition," *Journal of Dynamic Systems, Measurement, and Control,* 2002.

[14] H. Kazemi, Y. P. Fallah, A. Nix and S. Wayne, "Predictive AECMS by Utilization of Intelligent Transportation Systems for Hybrid Electric Vehicle Powertrain Control," *IEEE Transactions on Intelligent Vehicles,* vol. 2, no. 2, pp. 75-84, 2017.

[15] H. Kazemi, "Intelligent Transportation Systems, Hybrid Electric Vehicles, Powertrain Control, Cooperative Adaptive Cruise Control, Model Predictive Control," Graduate Theses, Dissertations, and Problem Reports. 3832, 2019. [Online].

[16] W. Vaz, R. Landers and U. Koylu, "Neural network strategy for driving behaviour and driving cycle classification," *International Journal of Electric and Hybrid Vehicles,* vol. 6, no. 3, 2014.

[17] C. Lin, S. Jeon, H. Peng and J. M. Lee, "Driving pattern recognition for control of hybrid electric trucks," *International Journal of Vehicle mechanics and mobility,* vol. 42, no. 1-2, 2010.

[18] B. Gu and G. Rizzoni, "An Adaptive Algorithm for Hybrid Electric Vehicle Energy Management Based on Driving Pattern Recognition," in *ASME International Mechanical Engineering Congross and Exposition*, Chicago, 2006.

[19] S. Kumar, "Fuzzy logic based driving pattern recognition for hybrid electric vehicle energy management," Arizona State University, December 2015. [Online]. Available: https://repository.asu.edu/attachments/164107/content/Kumar_asu_0010N_15593.pdf. [Accessed 15 November 2020].

[20] J. Kessels, M. Koot, P. Cosch and D. Kok, "Online Energy Management for hybrid Electric Vehicles," *IEE Transactions on Vehicular Tecnology,* 2008.

[21] L. Han, X. Jiao and Z. Zhang, "Recurrent Neural Network-Based Adaptive Energy," *Energies,* vol. 13, no. 202, 2020.

[22] N. Connelly, D. George, A. Nix and S. Wayne, "Generation and Analysis of Hybrid-Electric Vehicle Transmission Shift Schedules with a Torque Split Algorithm," *Journal of Transportation Technologies,* vol. 10, no. 1, 2020.

[23] N. Connelly, "Generation and Sensitivity Analysis of Transmissin Shift Schedule For Hybrid-Electric Vehicle.," Graduate Theses, Dissertations, and Problem Reports. 7169, 2018. [Online]. Available: https://researchrepository.wvu.edu/etd/7169 .

[24] D. George, "Hybrid Electric Vehicle Torque Split Algorithm For Reduction of Engine Torque Transients," Graduate Theses, Dissertations, and Problem Reports. 7179., 2018. [Online]. Available: https://researchrepository.wvu.edu/etd/7179.

[25] Furqan, J. Iqbal, A. Malik and W. Haider, "Neural Network Based Aircraft Control," in *IEEE Student Conference on Research and Developement,* Putrajaya, 2010.

[26] M. Perhinschi, M. Napolitano, G. Campa, B. Seanor and S. Gururajan, "Design of intelligent flight control laws for the WVU YF-22 model aircraft," in *AIAA 1st Intelligent Systems Technical Conference,* Chicago, 2021.

[27] T. Troudet, S. Garg and W. Merrill, "Neural network application to aircraft control system design," in *AIAA*, New Orleans, 1991.

[28] The MathWorks, Inc, "Powertrain Blockset Toolbox, Datasheet Battery," The MathWorks, Inc, Product Version 2018b. [Online].

[29] I. The MathWorks, "Powertrain Blockset Toolbox, Mapped Motor," The Mathworks, Inc, Product Version 2018b. [Online].

[30] I. The MathWorks, "Powertrain Blockset Toolbox, Mapped SI Engine," The MathWorks, Inc, Product Version 2018b. [Online].

[31] I. The Mathworks, "Powertrain Blockset Toolbox, Ideal Fixed Gear Transmission," The MathWorks, Inc, Product Version 2018b. [Online].

[32] I. The MathWorks, "Powertrain Blockset Toolbox, Longitudinal Wheel," The MathWorks, Inc, Product Version 2018b. [Online].

[33] CacAdam, "An Optimal Preview Control for Linear Systems," *Journal of Dynamic Systems, Measrement and Control*, 1980.

[34] The MathWorks, Inc, "Powertrain Blockset Toolbox, Vehicle Body 1DOF Longitudinal," The MathWorks, Inc, Product Version 2018b. [Online].

[35] M. Perhinschi, "Artificial Nerual Networks," Department of Mechanical and Aerospace Engineering, West Virginia University, Morgantown, 2019.