Journal of Transportation Technologies, 2021, 11, 471-503
https://www.scirp.org/journal/jtts
ISSN Online: 2160-0481
ISSN Print: 2160-0473

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, Andrew C. Nix, Mario G. Perhinschi, W. Scott Wayne, Jared A. Diethorn, Aaron R. Mull

West Virginia University, Morgantown, USA
Email: th0036@mix.wvu.edu, Andrew.Nix@mail.wvu.edu, Mario.Perhinschi@mail.wvu.edu, Scott.Wayne@mail.wvu.edu, jadiethorn@mix.wvu.edu, armull@mix.wvu.edu

How to cite this paper: Harris, T.P., Nix, A.C., Perhinschi, M.G., Wayne, W.S., Diethorn, J.A. and Mull, A.R. (2021) Implementation of Radial Basis Function Artificial Neural Network into an Adaptive Equivalent Consumption Minimization Strategy for Optimized Control of a Hybrid Electric Vehicle. *Journal of Transportation Technologies*, **11**, 471-503.
https://doi.org/10.4236/jtts.2021.114031

Received: January 13, 2021
Accepted: August 6, 2021
Published: August 9, 2021

Copyright © 2021 by author(s) and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/
Open Access

Abstract

Continued increases in the emission of greenhouse gases by passenger vehicles have 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. 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. The goal of this work is to use a previously defined strategy that 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), which uses an equivalence factor to define the control strategy and the power train component torque split. An equivalence factor that is optimal 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 (drive cycles) are used to train the RBF-ANN. 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 in 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.

Keywords

Hybrid Electric Vehicle, Artificial Neural Network, Equivalent Consumption Minimization Strategy (ECMS), Optimal Control Strategy

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, in everyday driving scenarios, this information is not available. To overcome this lack of knowledge, the best-performing control strategies are augmented with predictive and/or learning capabilities. These are often called adaptive control strategies. The implementation of predictive and learning capabilities results in incomparable efficiency results to those obtained when the future conditions are known *a priori* [1]. The predictive and/or learning capabilities allow control parameters to be adapted during real-time driving.

Adaptive control strategies commonly adjust control parameters based on one of the following two methodologies: predictions made on what the future driving conditions will be, or simply assuming that the future driving conditions will be similar to what the past conditions have been. The work in this paper is based on the latter methodology. Similar work has been performed, but the work described here uses a unique radial basis function (RBF) artificial neural network (ANN) with a unique set of input parameters to dynamically adapt a globally optimal control strategy. The use of the RBF in this work allows the entire ANN to be trained quickly with a single exposure to the set of training data. This is unlike other ANN's, which can require multiple exposures and take substantially more time to train.

This work details and explains the implementation of the RBF-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 are 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.

2. Vehicle Architecture

The vehicle being modeled in this work was a P4 architecture that was selected for use in the West Virginia University (WVU) EcoCAR Mobility Challenge Advanced Vehicle Technology Competition (Figure 1).

Engine: GM 2.5L LCV.

- Peak power: 148 kW.
- Peak Torque: 255 Nm.

Transmission: GM M3D (9T50) 9 Speed Automatic.

Fuel: E10.

Energy Storage System (ESS): GM HEV4.

- Peak Output Power: 50 kW.
- Energy Output: 1.5 kWh.

Motor: Magna Powertrain eAWD.

- Peak Power: 50.
- Peak Torque: 200 Nm.
- Integrated Gear Ratio: 9.17.

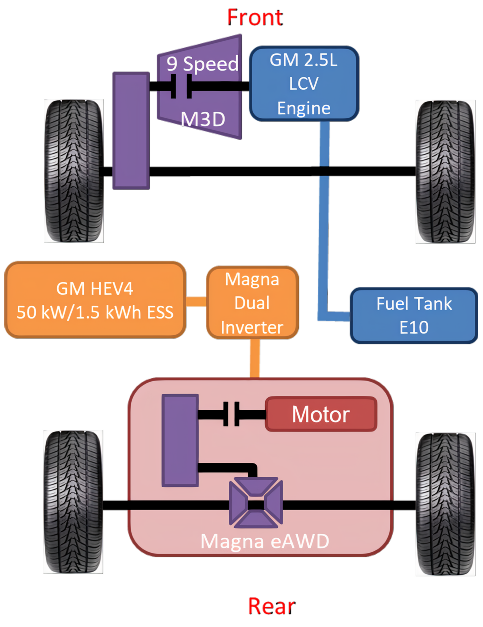

Figure 1. WVU architecture.

Inverter: Magna Powertrain Dual Inverter.

The P4 architecture has three primary modes of power flow: frontwheel drive (FWD) with opportunity charging, FWD with regenerative braking, and all-wheel drive (AWD). 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 high-voltage (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.

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), leds 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 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.

3. Overview of HEV Control Strategies

The design of an optimal control strategy for an HEV is a complex problem. The goal of the work described in this paper is to create a strategy which optimally splits the driver commanded torque between the ICE and motor in a way which solely maximizes fuel economy. An optimal control strategy 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 control strategies may be categorized as either rules based or optimal based strategies. Rules based strategies are effective for real-time implementation as the control is 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 ideally equivalent to the globally optimal cost function. The cost function is then minimized throughout the operation of the vehicle. However, the true global cost function is only known if the drive cycle is known *a priori*. Strategies employing appropriate cost functions have been shown to closely approximate the global optimal solution [1].

3.1. Dynamic Programming

Dynamic programming (DP) is a numeric method of solving the optimal energy management problem over the course of an entire drive cycle. It is limited to the simulation environment because of the need for *a priori* knowledge of the drive cycle. 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 making implementation not practical in a real-time controller during normal driving [1]. However, many different research groups have used fuel economy results obtained from DP as a baseline against which they can compare their own control strategies [2] [3] [4] [5].

3.2. Equivalent Consumption Minimization Strategy (ECMS)

The method of the ECMS is used to convert the global optimal solution into a series of instantaneous minimization problems making it is less computationally intensive and exhibiting results that closely approximate the global optimal solution. Onori *et al.* [1] has presented a global optimal energy management formula as follows:

$$J = \int_{t_0}^{t_f} \dot{m}_{f,eqv}\left(u(t),t\right)\mathrm{d}t \tag{1}$$

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 hybrid electric vehicles, a typical constraint is that the starting and ending SOC be within a certain threshold and prevention of the vehicle speed deviating too far from the drive cycle speed trace that is being followed.

The ECMS is formulated based on the premise that the battery is essentially an energy buffer: if electrical energy is used, it will eventually need to be replenished. 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, engine and motor torque commands, that minimize the equivalent fuel consumption. 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:

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

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:

$$\dot{m}_f(t) = \frac{P_{eng}(t)}{\eta_{eng}(t) * Q_{lhv}} \tag{3}$$

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:

$$\dot{m}_{elec}(t) = \frac{s(t)}{Q_{lhv}} * P_{batt}(t) \tag{4}$$

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 2), and has the following equation:

$$p(SOC) = 1 - \left(\frac{SOC(t) - SOC_{target}}{0.5 * (SOC_{\max} - SOC_{\min})} \right)^a \tag{5}$$

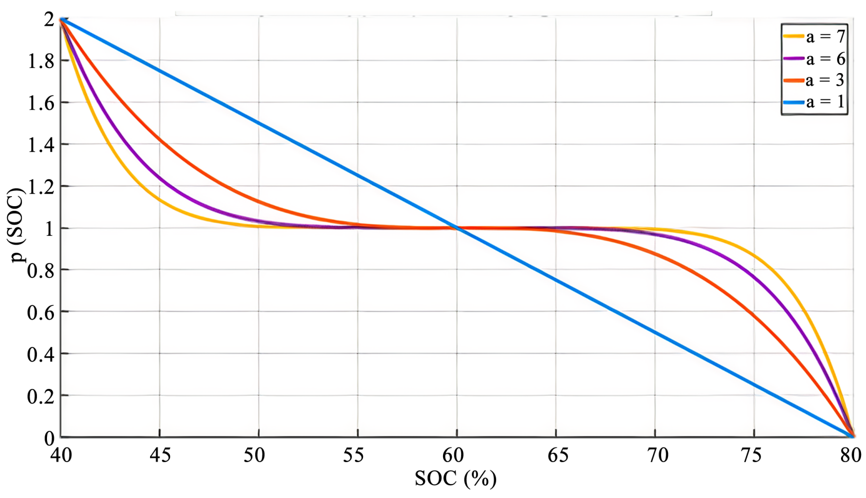

Figure 2. SOC penalty function with varying SOC penalty factors.

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 3, $SOC_{target} = 60\%$, $SOC_{\max} = 80\%$, and $SOC_{\min} = 40\%$.

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:

$$\dot{m}_{f,eqv}(t) = \frac{P_{eng}(t)}{\eta_{eng}(t) * Q_{lhv} * p(SOC)} + \frac{s(t)}{Q_{lhv}} * P_{batt}(t) \tag{6}$$

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

$$P_{f,eqv}(t) = \frac{P_f(t)}{\eta_{eng}(t)} + s(t) * P_{batt}(t) * p(SOC) \tag{7}$$

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 (7). This additional term takes the absolute value of the difference between the last engine power and the current engine power. Added to Equation (7), this term acts as a cost, making it more expensive to select an engine power that differs greatly from the last commanded engine power, as shown below:

$$P_{eng,rate\ limit} = \left| P_{eng}^{t} - P_{eng}^{t-1} \right| \tag{8}$$

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 Equation (9).

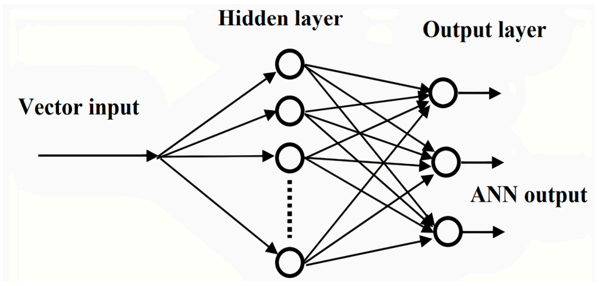

Figure 3. Single hidden layer ANN diagram [17].

$$P_{f,eqv}(t) = \frac{P_f(t)}{\eta_{eng}(t)} + s(t) * P_{batt}(t) * p(SOC) + P_{eng,rate\ limit} \tag{9}$$

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.

3.3. Adaptive-ECMS (A-ECMS)

Research has been ongoing to create an 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 by implementing a dynamically varying equivalence factor. 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].

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 exhibited characteristics of the optimal solution [1] [6] [7] [8] [9]. Work done in driving pattern recognition has also been performed in an effort to improve fuel economy [10] [11] [12] [13].

4. 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 [14] [15] [16]. Of particular interest is the work done by Furquan *et al.* [14], 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 [14].

Additionally, work done by M. Perhinschi *et al.* [15] 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 [15].

5. 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.

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. 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. Many of the component models used were initially created by MathWorks using the Powertrain Block set.

5.1. Artificial Neural Network (ANN) Description and 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 3**.

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

$$f(\bar{x}) \approx \sum_k w_k \varphi_k(\bar{x}) \tag{10}$$

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 [17]. A diagram of a hidden layer neuron of the RBF-ANN is shown in Figure 4.

Where x_j^* is the distance between the input vector $\bar{x}$ and the center $\bar{c}_j$ for the j^{th} 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:

$$\varphi(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-x^2}{2\sigma^2}} \tag{11}$$

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 5).

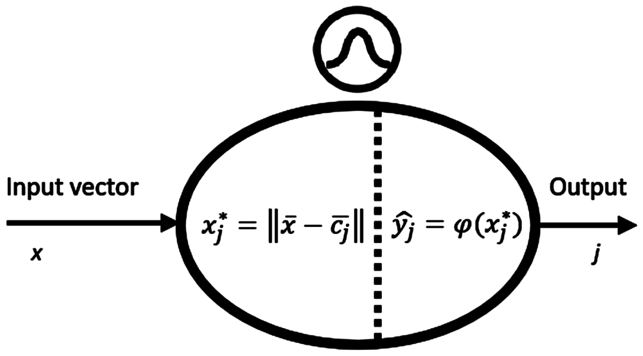
Input vector
x
$x_j^* = \|\bar{x} - \bar{c}_j\|$
$\hat{y}_j = \varphi(x_j^*)$
Output
j

Figure 4. RBF neuron in the hidden layer [17].

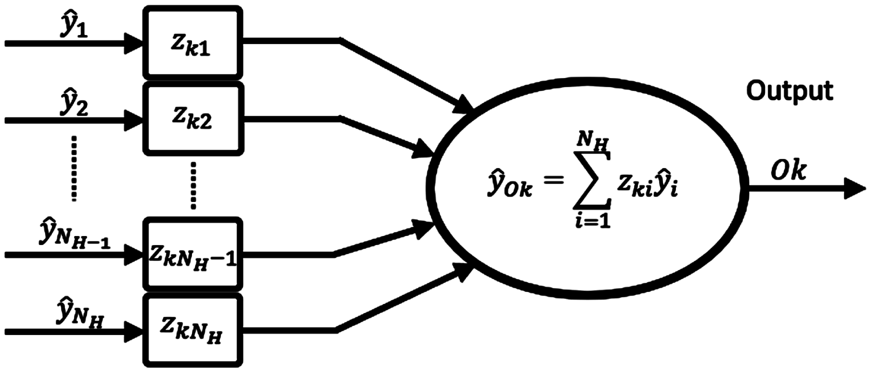

Figure 5. Output neuron in RBF-ANN [17].

Where $z_{1\cdots N_h}$ 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 6 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 numbers of center vectors as there are inputs ($\overline{c}_2 \cdots \overline{c}_n$).

Figure 6 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 ($\overline{x}$) 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.

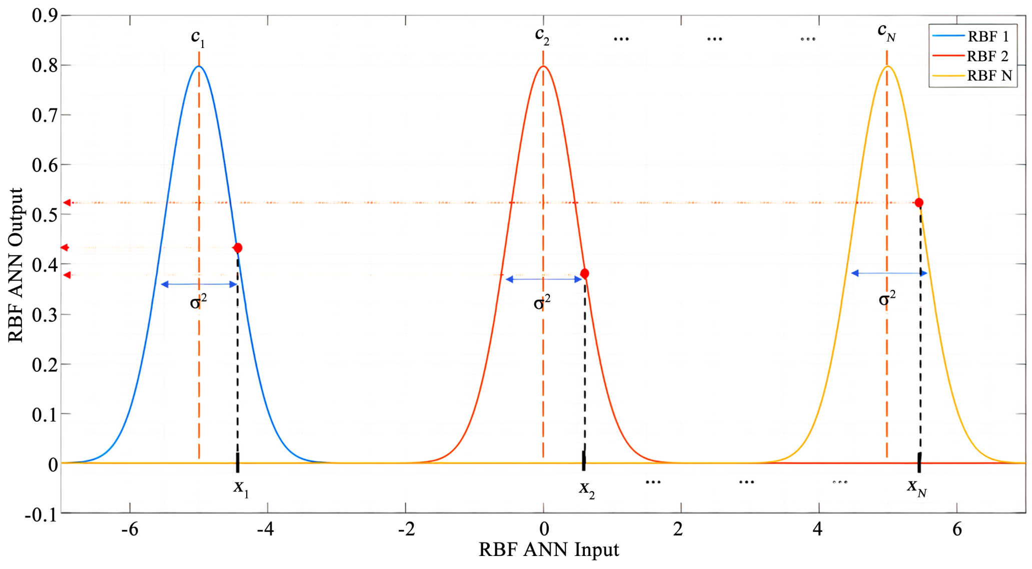

Figure 6. Well-bounded set of input data to RBF-ANN.

As described in Section 3.2, the ECMS is formulated using an equivalence factor that relates the consumption of fuel energy to electrical energy. Traditionally, the equivalence factor is optimized offline for specific drive cycles in order to maximize fuel economy for the given cycle. The objective of the ANN-ECMS is to dynamically change the equivalence factor based on a past set of driving conditions—which are determined over a sliding time window. This is able to be done while driving real-time. Since the equivalence factor is the only parameter being updated in this work, only one neuron is needed in the output layer of the ANN.

5.2. 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 [ga/s];
- Average Negative Jerk [ga/s];
- Total Distance [mile];
- Idle Time (sec);
- Average Speed [m/s];
- Maximum Speed [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 7 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 8 shows a linear regression plot of the normal driver over the entire HUDDS drive cycle.

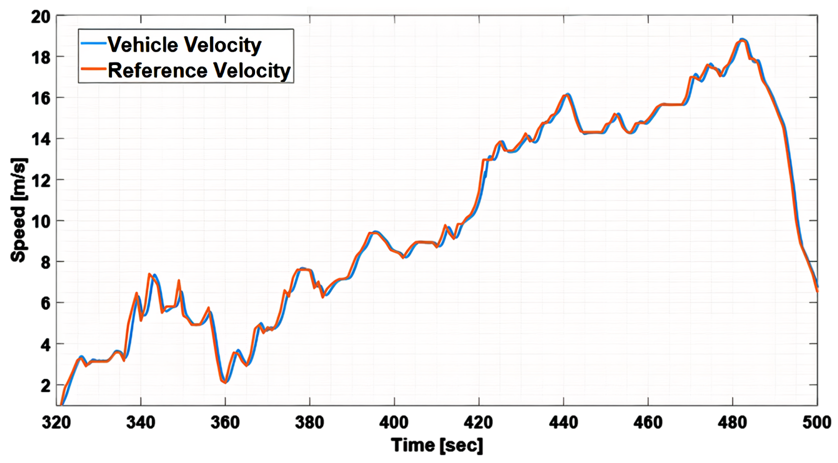

Figure 7. Normal driver speed trace.

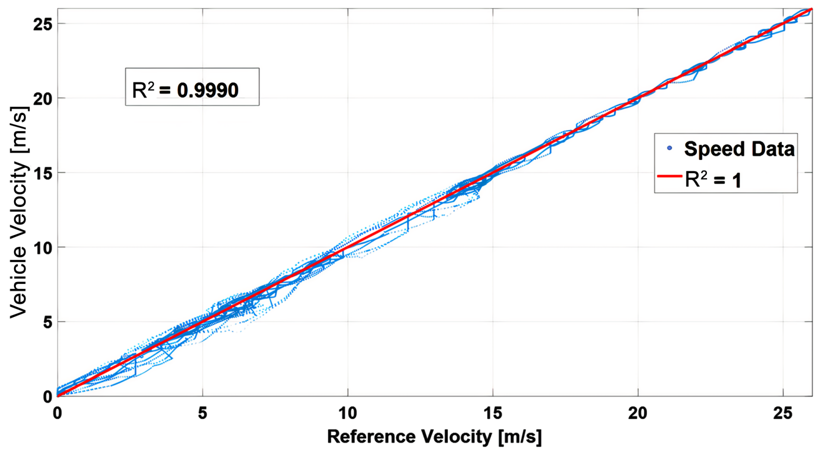

Figure 8. Linear regression of normal driver.

Figure 9 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 10** shows a linear regression plot of the smooth driver over the entire HUDDS drive cycle.

Figure 10 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. 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

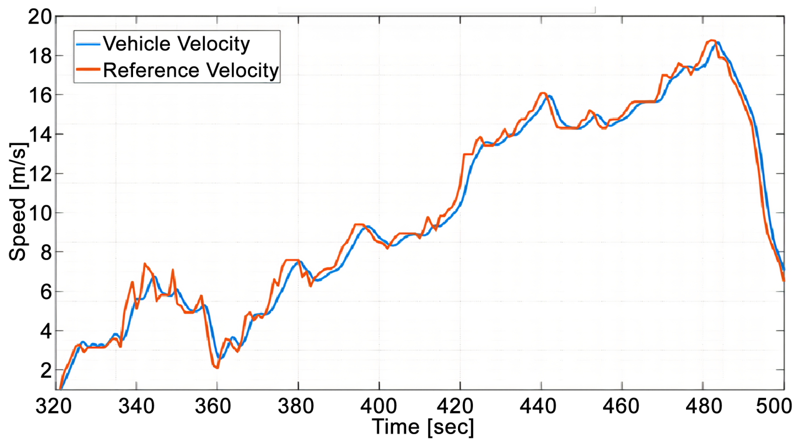

Figure 9. Smooth driver speed trace.

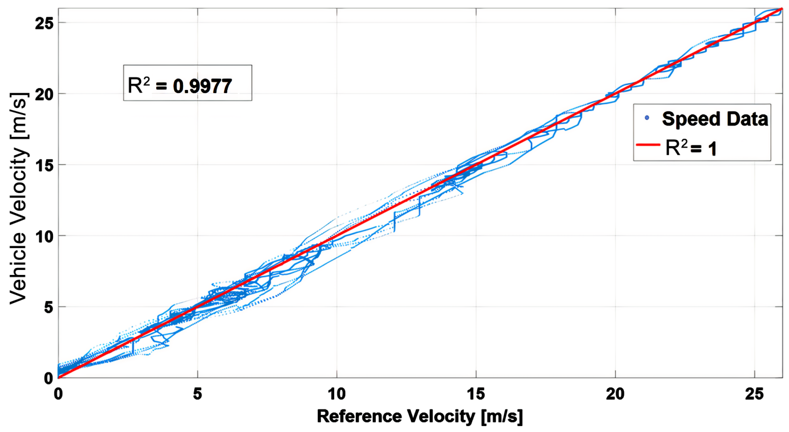

Figure 10. Linear regression of smooth driver.

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.

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.

5.3. 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.

5.4. 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 $i = 1,2,3,\cdots,N_K$ and $j = 1,2,3,\cdots,N_H * N_K$ 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:

$$D = \{d_{ji}\} = \begin{bmatrix} \|\bar{x}_1 - \bar{c}_1\| & \|\bar{x}_2 - \bar{c}_1\| & \cdots & \|\bar{x}_{N_K} - \bar{c}_1\| \\ \|\bar{x}_2 - \bar{c}_1\| & \|\bar{x}_2 - \bar{c}_2\| & \cdots & \|\bar{x}_{N_K} - \bar{c}_2\| \\ \vdots & \vdots & \ddots & \vdots \\ \|\bar{x}_1 - \bar{c}_{N_H}\| & \|\bar{x}_2 - \bar{c}_{N_H}\| & \cdots & \|\bar{x}_{N_K} - \bar{c}_{N_H}\| \end{bmatrix} \quad (12)$$

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:

$$\varphi(D) = \frac{1}{\sqrt{2\pi\sigma^2}} \mathrm{e}^{\frac{-D\otimes D}{2\sigma^2}} \tag{13}$$

where $D\otimes D$ is the element wise product and $\varphi(D)$ will be a $N_H \times N_K$ sized matrix. The product of the output of the hidden layer and the weights are supposed to approximate the training data, so the weights (d) are determined as follows:

$$Z = \left[\varphi(D)^{\mathrm{T}}\right]^{-1} * \bar{y} \tag{14}$$

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

5.5. 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 11).

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.

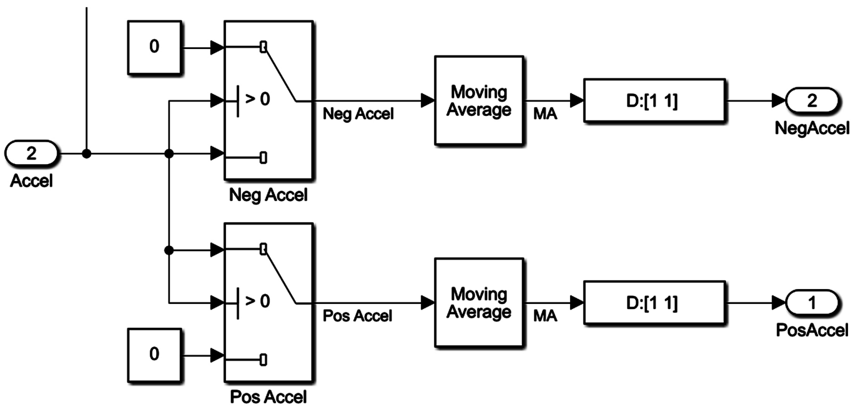

Figure 11. Acceleration/deceleration logic.

The positive jerk (Pos Jerk) and negative jerk (Neg Jerk) values are determined (Figure 12) 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.

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 13). 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 13).

The distance is determined using a discrete-time integrator (Figure 14), which receives the vehicle speed (VehSpd), and a trigger (gen).

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 15).

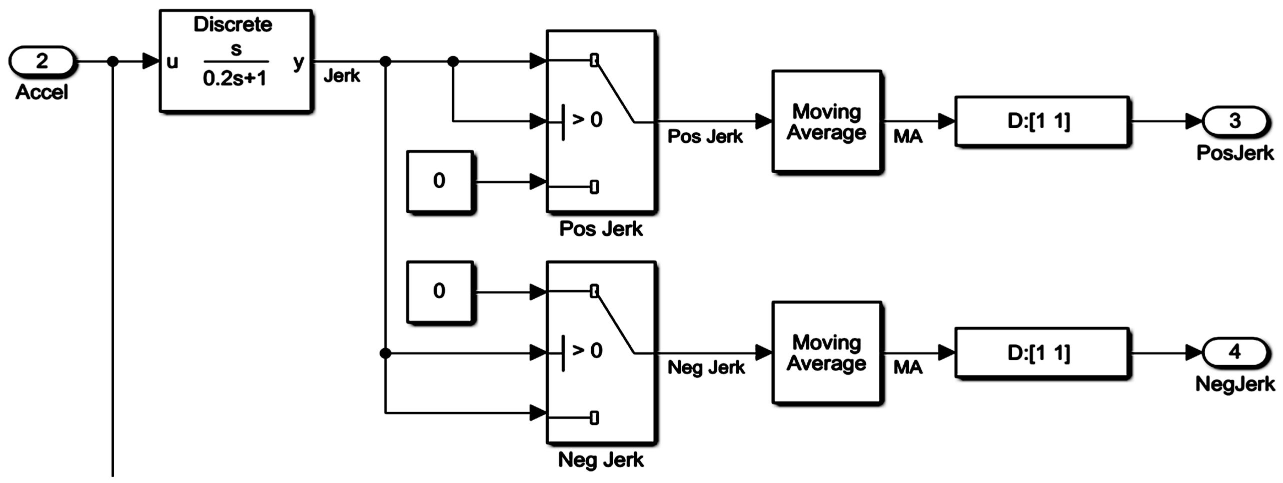

Figure 12. Positive and negative jerk calculation.

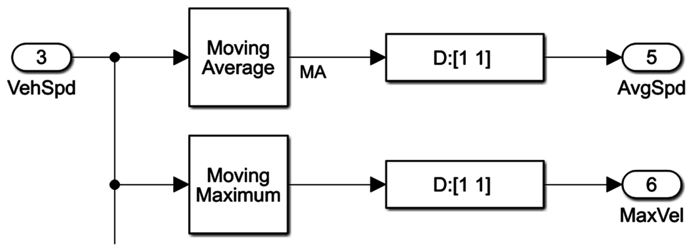

Figure 13. Average vehicle speed and maximum vehicle speed.

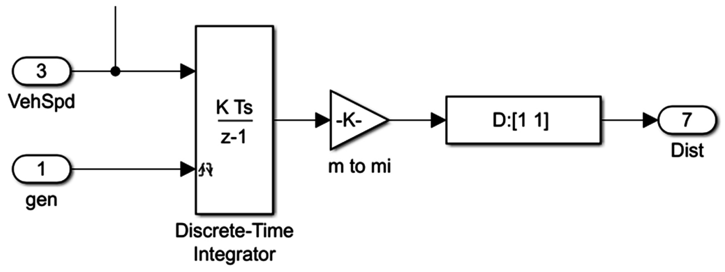

Figure 14. Distance calculation.

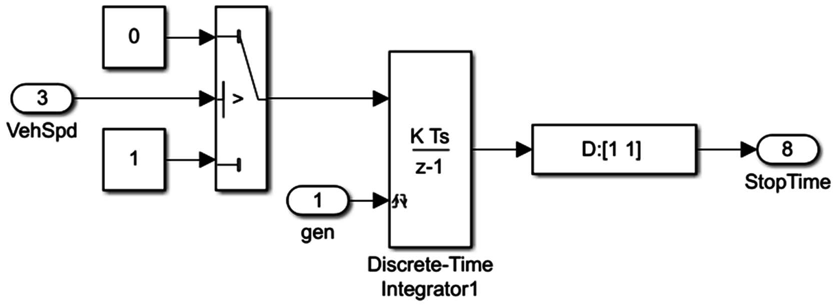

Figure 15. Idle time calculation.

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 15. 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 16). The sample and hold block is triggered using the rising and falling edge of the "gen" signal described from Figure 15.

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 17 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.

The inputs are fed to a function block that implements the RBF-ANN (Figure 18). The output of the function block is the equivalence factor (equiv_factor).

Lastly, before being passed to the ECMS cost function block, 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

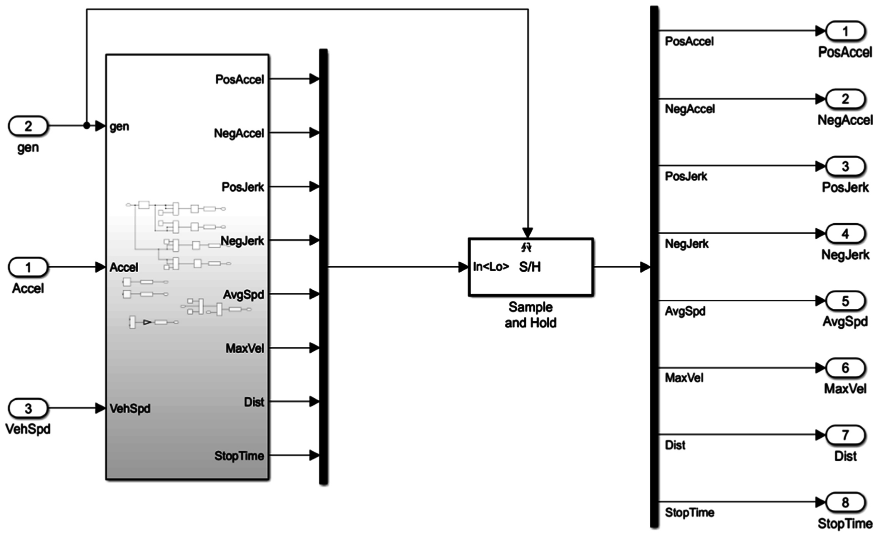

Figure 16. Sample and hold of input signals.

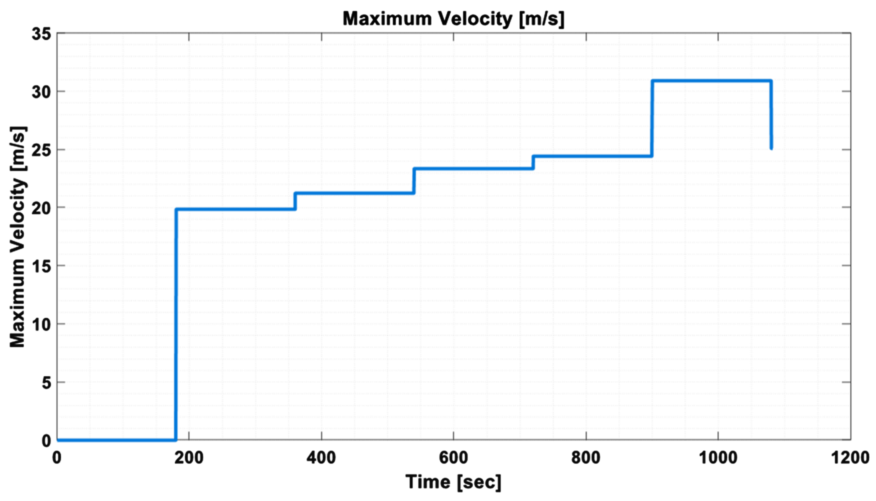

Figure 17. RBF-ANN maximum velocity input.

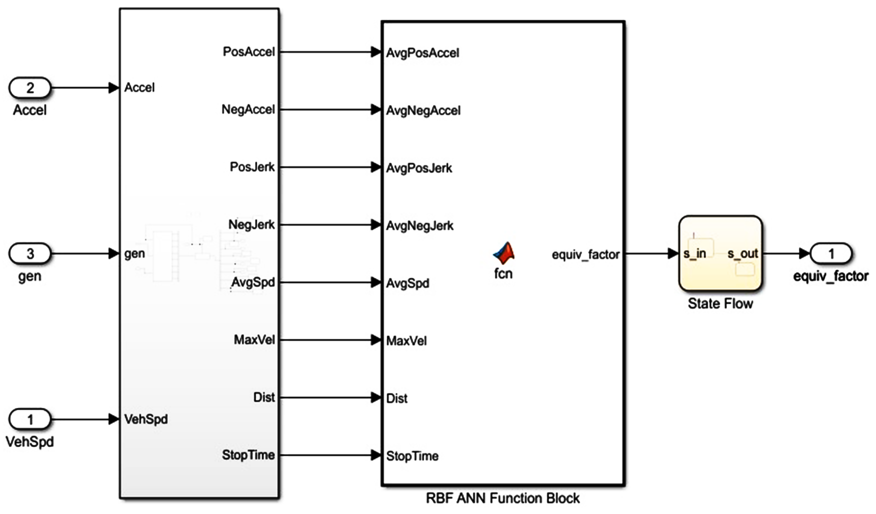

Figure 18. RBF-ANN function block.

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 the Appendix.

6. 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 ANNECMS that were not used to train the ANN. Then, the results of ANNECMS will be compared to optimal ECMS.

6.1. 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. 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 comes 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.

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.* [11]. 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 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 EUDC cycle contains moderately low speeds with very low acceleration and no idle time. The NYCC cycle is characterized by low speeds, very aggressive acceleration, and frequent instances of idle time. The ECEExtra cycle is similar to the EUDC cycle in aggression level and idle time. However, the ECEExtra cycle does not reach as high of speeds as the EUDC cycle. The Jap1015 cycle, is characterized by low speeds with a few areas of aggressive acceleration. The cycle also contains frequent instances of extended idle time. 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 (iC) 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 each 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 Figures 19-21 respectively.

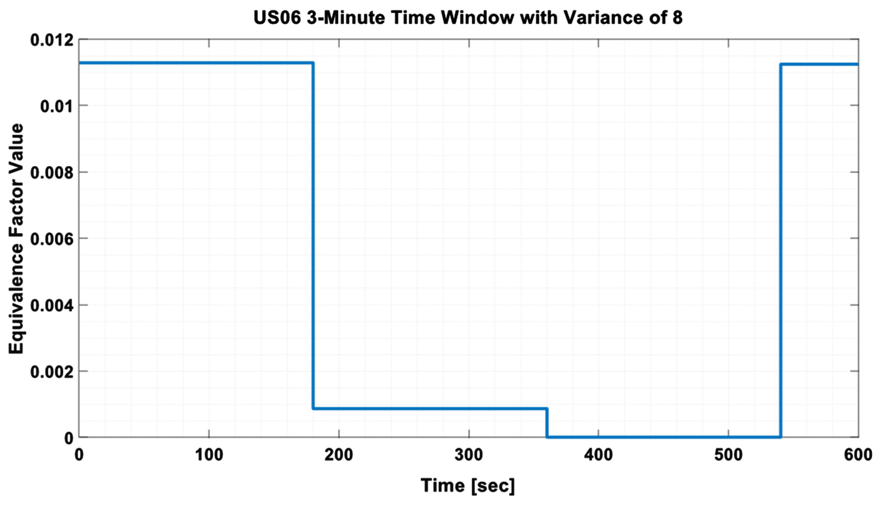

Figure 19. Varying equivalence factor over the course of us06 drive cycle with a variance of 8.

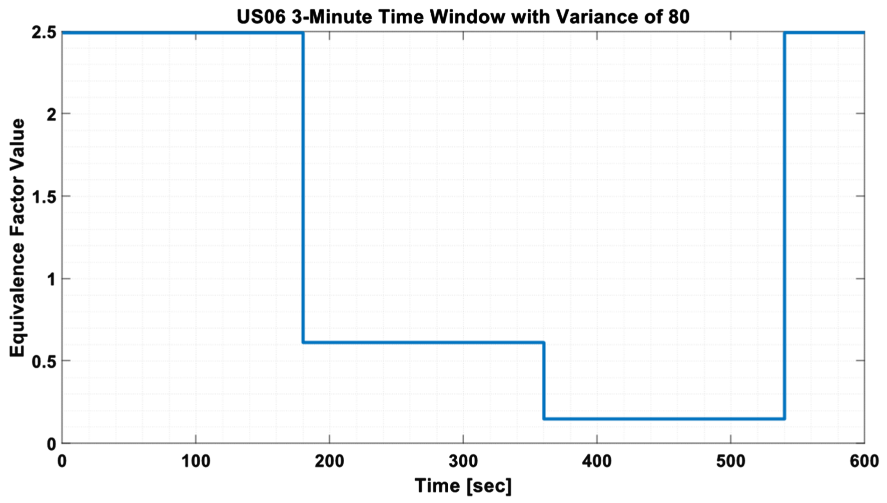

Figure 20. Varying equivalence factor over the course of us06 drive cycle with a variance of 80.

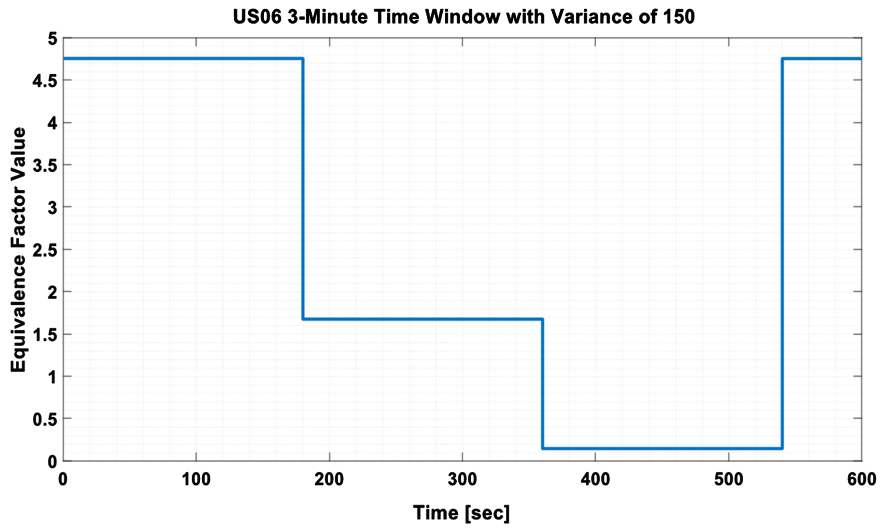

Figure 21. Varying equivalence factor over the course of us06 drive cycle with a variance of 150.

These figures show that as the variance increases, the magnitude of the equivalence factor (y-axis) also increases. As discussed earlier, with respect to Figure 4, a low variance value tends to push the output to zero. This is apparent when viewing Figure 19. The low variance of 8, results in equivalence factor values near zero.

With respect to this work, Figures 19-21 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 Figures 19-21 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.

Figures A1-A3 (In Appendix) 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. An examination of Figures A1-A3 (in Appendix), 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:

$$\%Error = \frac{ECSM_{FE} - ANN_{FE}}{ECMS_{FE}} * 100 \tag{15}$$

where $ECSM_{FE}$ 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. Figures A4-A6 (In Appendix) show the percent error for each of the 3 time windows and the variance values of 8, 80, and 15. 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 1. 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 were FTP-72, Artemis Urban, and HUDDS. The Addition of these cycles is shown in Table 2.

Table 1. % 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

Table 2. 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 2, there is now a clearer best performer. A time window of 3 minutes and a variance of 80 result 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.

6.2. 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. 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 relatively consistent 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.

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 2** was a 4-minute time window. If the individual drive cycles of **Table 2** 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 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 A7** shows the fuel economy comparison between the ANN-ECMS and optimal ECMS. **Figure A8** shows the percent error between the ANN-ECMS and optimal ECMS fuel economy.

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.

In the work of Gu *et al.* [11], 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 [11]. 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 A9**, 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.

The comparison in **Figure A9** 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.* [11] and Jeon *et al.* [9], 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.

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 validation cycles were: SC03, IM240, NEDC, JC08, and RTS95. Comparison of the ANN-ECMS and Optimal ECMS fuel economy results are shown in **Figure A10**. **Figure A11** shows the percent error between the optimal ECMS fuel economy and the fuel economy obtained using ANN-ECMS.

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%.

7. 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 were 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 Gu *et al.* was 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 ANNECMS 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 AECMS.

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 could 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.

Acknowledgements

This work was performed under partial funding of graduate students by the EcoCAR Mobility Challenge, Advanced Vehicle Technology Competition, sponsored by US Department of Energy, General Motors (GM) and the Mathworks, and managed by Argonne National Laboratory. The authors would like to thank their GM EcoCAR mentor, Dr. William Cawthorne, for his assistance with the vehicle controls and torque split modeling effort.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Onori, S., Serrao, L. and Rizzoni, G. (2016) Hybrid Electric Vehicles Energy Management Strategies. Springer, London. https://doi.org/10.1007/978-1-4471-6781-5

[2] Lin, C.C., Kang, J.M., Grizzle, J.W. and Peng, H. (2001) Energy Management Strategy for a Parallel Hybrid Electric Truck. *Proceedings of the* 2001 *American Control Conference*, Arlington, 25-27 June 2001, 2878-2883.

[3] Lin, C.C., Zoran, F., Wang, Y., Loucas, L., Peng, H., Assanis, D. and Stein, J. (2001) Integrated, Feed-Forward Hybrid Electric Vehicle Simulation in SIMULINK and its Use for Power Management Studies. SAE Technical Papers, 15 p.

[4] Sciarretta, A., Back, M. and Guzzella, L. (2004) Optimal Control of Parallel Hybrid Electric Vehicles. *IEEE Transactions on Control Systems Technology*, **12**, 352-3633. https://doi.org/10.1109/TCST.2004.824312

[5] Yang, Y., Hu, X., Pei, H. and Peng, Z. (2016) Comparison of Power-Split and Parallel Hybrid Powertrain Architectures with a Single Electric Machine: Dynamic Programming Approach. *Applied Energy*, **168**, 683-690. https://doi.org/10.1016/j.apenergy.2016.02.023

[6] Lee, D., Cha, S.W., Rousseau, A. and Kim, N. (2012) Optimal Control Strategy for PHEVs Using Prediction of Future Driving Schedule. *World Electric Vehicle Journal*, **5**, 149-158. https://doi.org/10.3390/wevj5010149

[7] Chen, Z., Yang, C. and Fang, S. (2020) A Convolutional Neural Network-Based Driving Cycle Prediction Method for Plug-in Hybrid Electric Vehicles with Bus Route. *IEEE Access*, **8**, 3255-3264. https://doi.org/10.1109/ACCESS.2019.2960771

[8] Chasse, A., Sciarretta, A. and Chauvin, J. (2010) Online Optimal Control of a Parallel Hybrid with Costate Adaptation Rule. *IFAC Proceedings Volumes*, **43**, 99-104. https://doi.org/10.3182/20100712-3-DE-2013.00134

[9] Jeon, S., Jo, S. and Lee, J. (2002) Multi-Mode Driving Control of a Parallel Hybrid Electric Vehicle Using Driving Pattern Recognition. *Journal of Dynamic Systems Measurement and Control*, **124**, 141-149. https://doi.org/10.1115/1.1434264

[10] Lin, C.C., Jeon, S., Peng, H. and Lee, J.M. (2010) Driving Pattern Recognition for Control of Hybrid Electric Trucks. *Vehicle System Dynamics*, **42**, 41-58. https://doi.org/10.1080/00423110412331291553

[11] Gu, B. and Rizzoni, G. (2006) An Adaptive Algorithm for Hybrid Electric Vehicle Energy Management Based on Driving Pattern Recognition. *Proceedings of the ASME* 2006 *International Mechanical Engineering Congress and Exposition*, *Dynamic Systems and Control*, *Parts A and B*, Chicago, 5-10 November 2006, 249-258.

[12] Kumar, S. (2015) Fuzzy Logic Based Driving Pattern Recognition for Hybrid Electric Vehicle Energy Management. Arizona State University, Tempe. https://repository.asu.edu/attachments/164107/content/K

[13] Zhou, Y., Ravey, A. and Péra, M.C. (2020) Multi-Mode Predictive Energy Management for Fuel Cell Hybrid Electric Vehicles Using Markov Driving Pattern Recognizer. *Applied Energy*, **258**, Article ID: 114057.
https://doi.org/10.1016/j.apenergy.2019.114057

[14] Furqan, Iqbal, J., Malik, A.N. and Haider, W. (2010) Neural Network Based Aircraft Control. 2010 *IEEE Student Conference on Research and Development* (*SCOReD*), Kuala Lumpur, 13-14 December 2010, 262-266.
https://doi.org/10.1109/SCORED.2010.5704013

[15] Perhinschi, M., Napolitano, M., Campa, G., Seanor, B. and Gururajan, S. (2004) Design of Intelligent Flight Control Laws for the WVU YF-22 Model Aircraft. *AIAA 1st Intelligent Systems Technical Conference*, Chicago, 20-22 September 2004, AIAA 2004-6282.

[16] Troudet, T., Garg, S. and Merrill, W. (1991) Neural Network Application to Aircraft Control System Design. AIAA Paper, 91-2715-CP.

[17] Perhinschi, M. (2019) Artificial Neural Networks. Department of Mechanical and Aerospace Engineering, West Virginia University, Morgantown.

Appendix

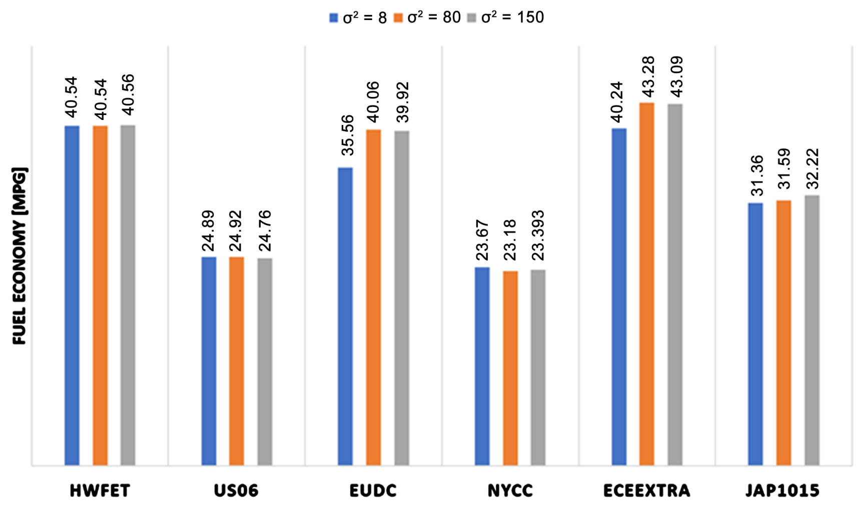

Figure A1. Effect of variance [8, 80, 150] on 2-minute time window.

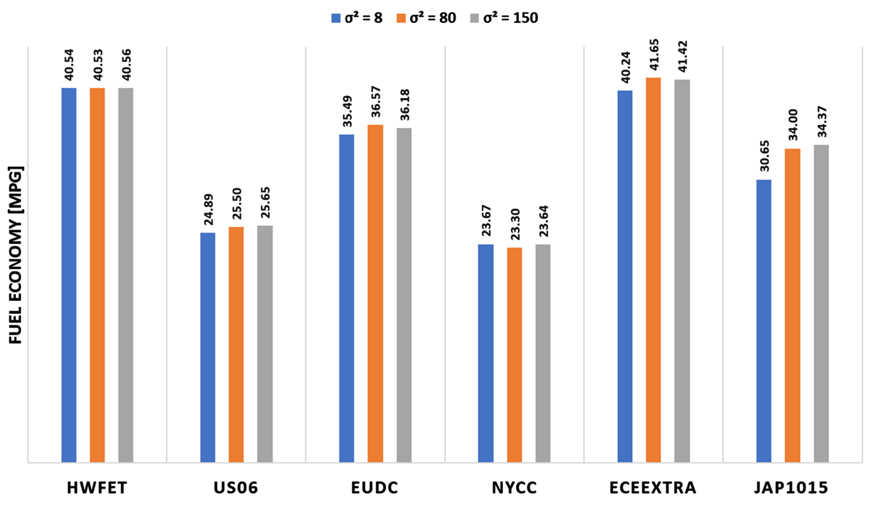

Figure A2. Effect of Variance [8, 80, 150] on 3-minute time window.

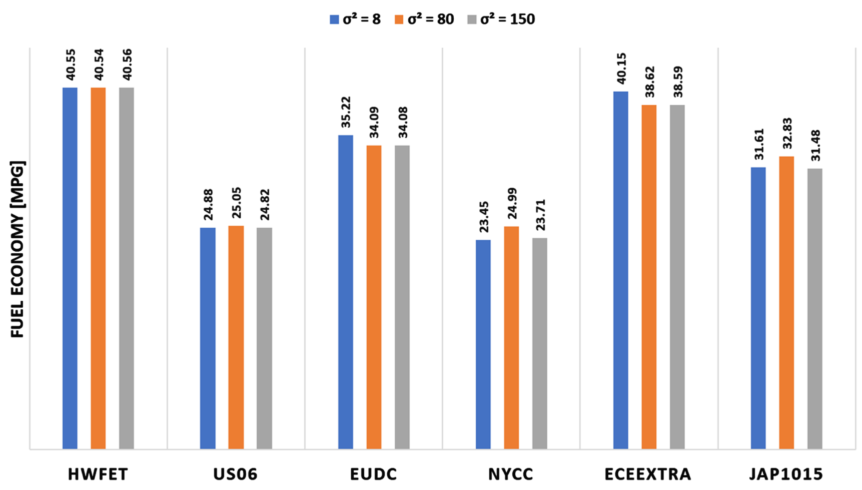

Figure A3. Effect of Variance [8, 80, 150] on 4-minute time window

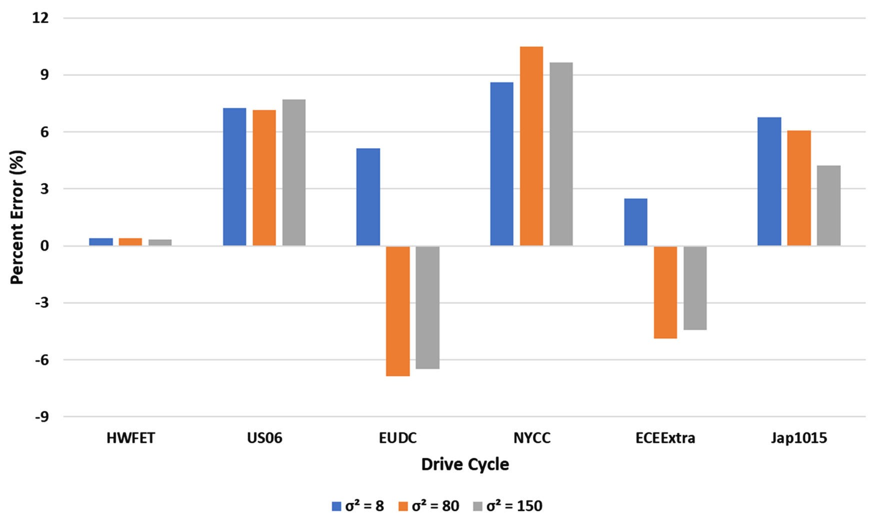

Figure A4. % Error vs σ^2 for the 2-minute time window.

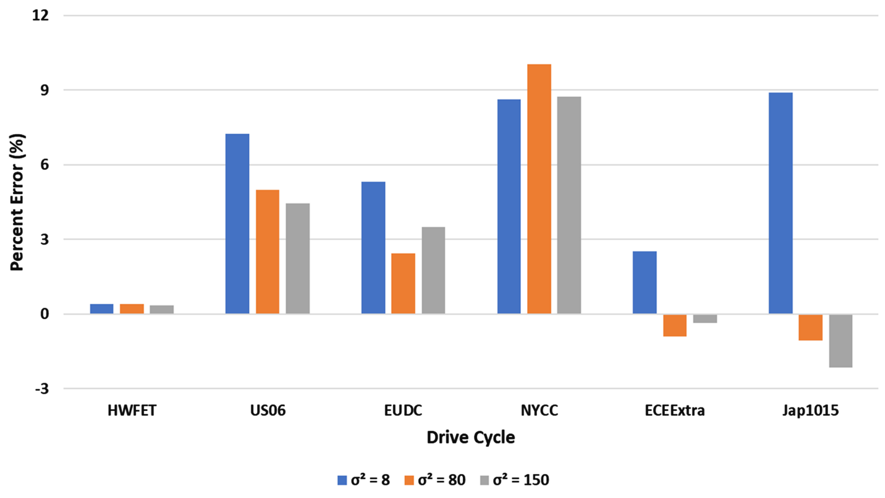

Figure A5. % Error vs σ^2 for the 3-minute time window.

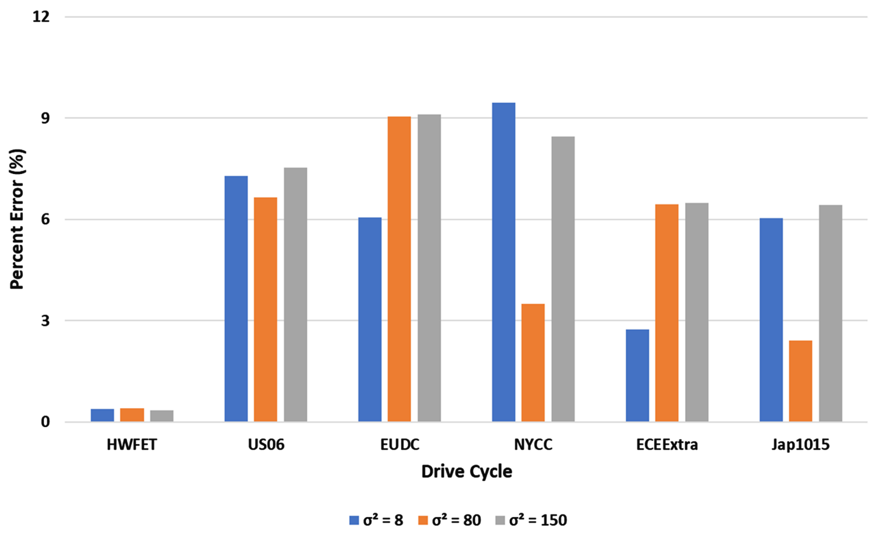

Figure A6. % Error vs σ^2 for the 4-minute time window.

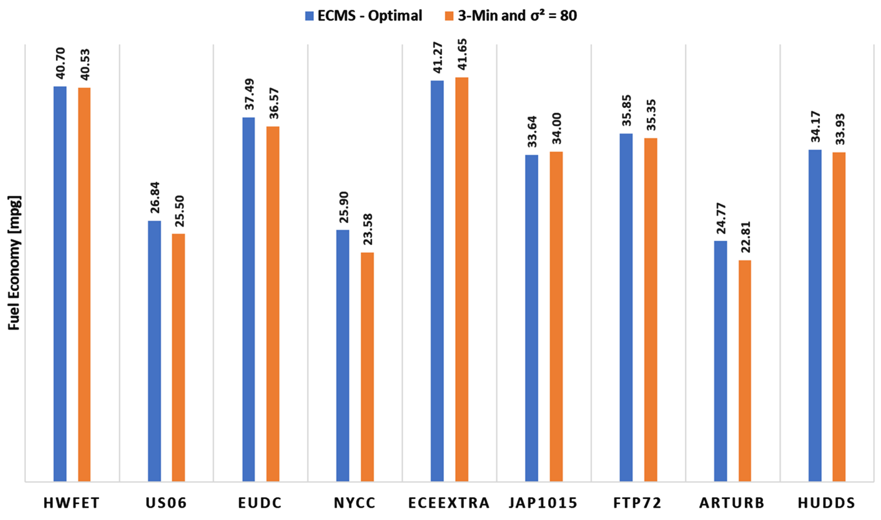

Figure A7. Fuel economy comparison of verification drive cycles.

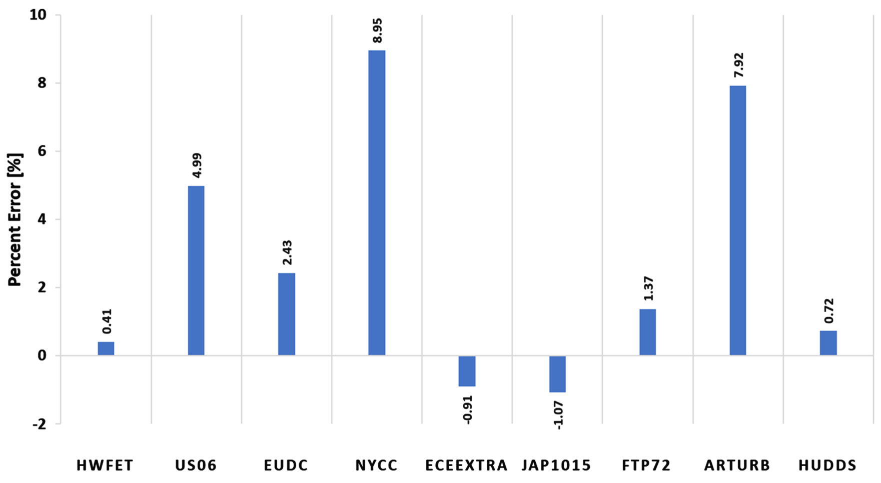

Figure A8. Percent error of comparison of verification drive cycles.

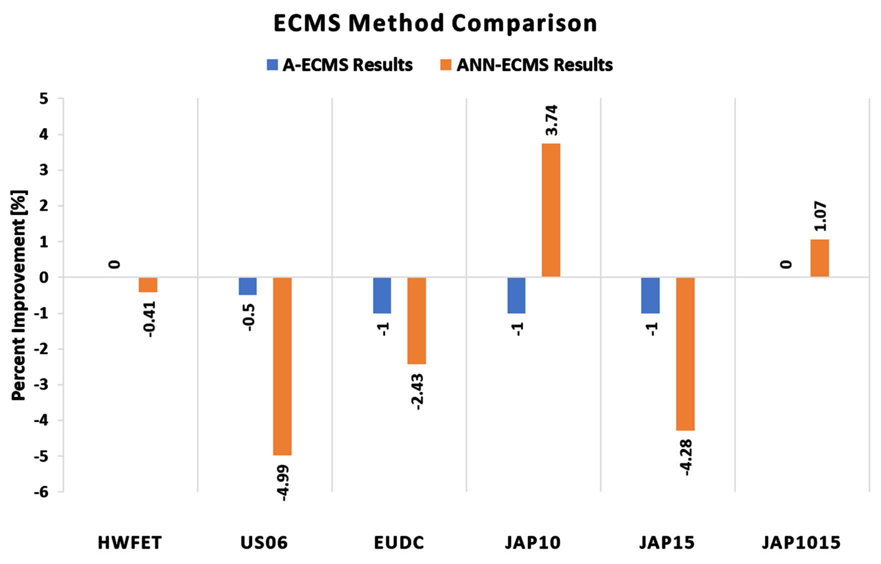

Figure A9. Percent improvements comparison of A-ECMS and ANN-ECMS.

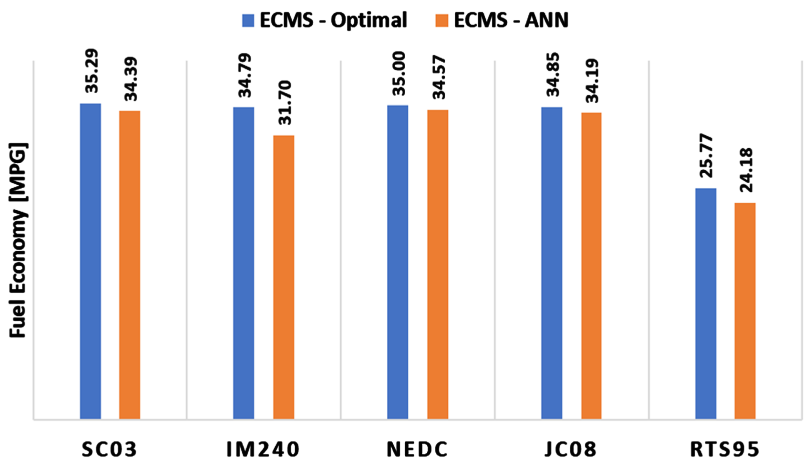

Figure A10. Comparison of fuel economy results of validation drive cycles between optimal.

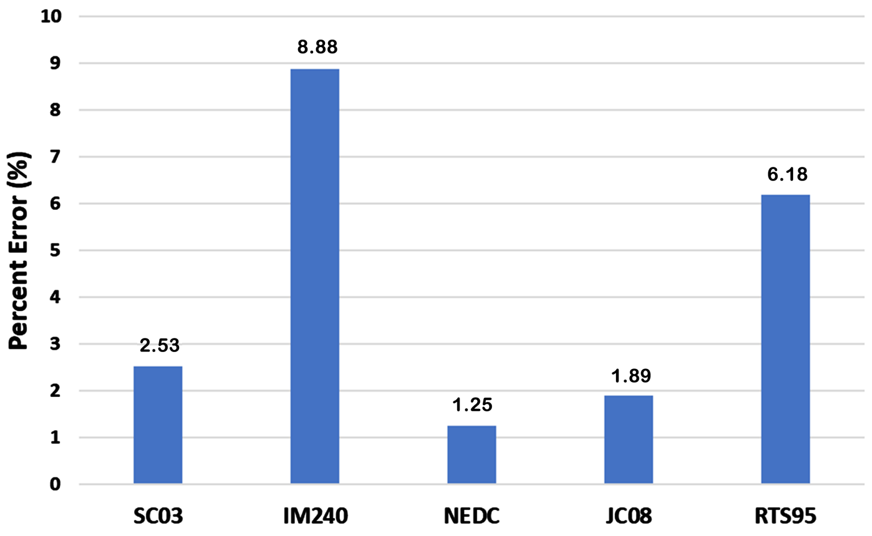

Figure A11. Percent error of validation drive cycles between optimal ECMS and ANN-ECMS Fuel.