ARTIFICIAL NEURAL NETWORKS CONTROL STRATEGY OF A PARALLEL THROUGH-THE-ROAD PLUG-IN HYBRID VEHICLE

by

Mingyu Sun

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science in Mechanical Engineering

School of Mechanical Engineering

West Lafayette, Indiana

December 2018

THE PURDUE UNIVERSITY GRADUATE SCHOOL

STATEMENT OF COMMITTEE APPROVAL

Dr. Peter H. Meckl, Chair

School of Mechanical Engineering

Dr. Gregory M. Shaver

School of Mechanical Engineering

Dr. Oleg Wasynczuk

School of Electrical and Computer Engineering

Approved by:

Dr. Jay P. Gore

Head of the Graduate Program

Dedicated to my parents and grandparents.

ACKNOWLEDGMENTS

I would like to express my deep sense of thanks and gratitude to my advisor, Dr. Peter H. Meckl for his consistent guidance, ample time spent and support throughout the research project. This work would not have been possible without his encouragement, patience and immense knowledge. I would also like to thank the committee members Dr. Oleg Wasynczuk and Dr. Gregory Shaver, for their insightful comments and encouragement.

I would also like to extend my gratitude to Rohinish Gupta, who laid the foundation of this thesis and helped me become familiar with the system in the very beginning. Then I would like to thank Abhilash Reddy for the sleepless nights spent on the program. I would also like to thank Chengxi Li for helping me learn Neural Networks and Pytorch. I would like to thank Kaushal Jain and Harshil Angre for working together in the lab.

I am also grateful to my friends Qinchi Li, Ke Luo, Yinglong Chen, Ting Zhang, Botian Zhang, Dingxiang Meng, Siqing Wei, Jiaming Liu, Erkang Zhou, Peng Zhang, Xia Lou, Hongni Xiang, Zhongkai Yu, Yiheng Liu, Hanyang Zhang and those who helped me at Purdue.

Last but not least, I am grateful to my parents and grandparents for their support and love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

C_t	Cell State
E_{engine}	Engine energy consumption
E_{motor}	Motor energy consumption
E_{total}	Total energy consumption
f_t	Forget Gate
$F_{inertia}$	Inertia force
F_{load}	Force due to road load
F_{tract}	Tractive force required at the wheels
I_{batt}	Battery current
I_{m}	Motor current
J_{π}	Cost function, dynamic programming
J_t	Cost function, ECMS
L_k	Instantaneous cost function, dynamic programming
M_{veh}	Mass of vehicle
N_g	Gear number
P_e	Engine power
τ_e	Engine torque
τ_{ev}	Engine torque at wheels
τ_m	Motor torque
τ_{mv}	Motor torque at wheels
τ_v	Torque required by the vehicle

v	Vehicle speed
ω_e	Angular velocity of engine
ω_{ed}	Angular velocity of engine differential
ω_m	Angular velocity of electric motor
ω_{md}	Angular velocity of motor differential
ω_w	Angular velocity of wheel
x_i	State, dynamic programming
X_{ed}	Engine differential ratio
X_g	Gear ratio

LIST OF ABBREVIATIONS

ANN	Artificial Neural Networks
NNM	Neural Networks Model
APP	Accelerator Pedal Position
ASM	Automotive Simulation Model
CAN	Controller Area Network
CNN	Convolutional Neural Network (CNN)
DEF	Diesel Exhaust Fluid
DOC	Diesel Oxidation Catalyst
DP	Dynamic Programming
DPF	Diesel Particulate Filter
DPM	Dynamic Programming Model
ECU	Electronic Control Units
ECMS	Equivalent Energy Consumption Minimization Strategy
EMI	Electro-Magnetic Induction
EIA	Energy Information Administration
ESS	Energy Storage System
FC	Fully-connected Layer
HIL	Hardware in Loop
HEV	Hybrid Electric Vehicle
HWFET	Highway Fuel Economy Test
ICE	Internal Combustion Engine

LSTM	Long Short-term Memory
PHEV	Plug-in Hybrid Electric Vehicle
PMP	Pontryagin's Minimum Principle
RNN	Recurrent Neural Network
pSOC	Proportional State of Charge
SAE	Society of Automotive Engineers
SCR	Selective Catalytic Reduction
SIL	Software in Loop
SOC	State of Charge
UDDS	Urban Dynamometer Driving Schedule
ZIF	Zero insertion force

ABSTRACT

Author: Sun, Mingyu, M.S.M.E.
Institution: Purdue University
Degree Received: December 2018
Title: Artificial Neural Networks Control Strategy of a Parallel Through-the-Road Plug-In Hybrid Vehicle
Committee Chair: Peter H. Meckl

The increasing amounts of vehicle emissions and vehicle energy consumption are major problems for the environment and energy conservation. Hybrid vehicles, which have less emissions and energy consumption, play more and more important roles in energy efficiency and sustainable development.

The power management strategies of a parallel-through-the-road hybrid architecture vehicle are different from traditional hybrid electric vehicles since one additional dimension is added. To study power management strategies, a simplified model of the vehicle is developed. Four types of power management strategies have been discovered previously based on the simplified model, including dynamic programming model, equivalent consumption minimization strategy, proportional state-of-charge algorithm, and regression model. A new power management strategy, which is artificial neural network model, is developed. All these five power management strategies are compared, and the artificial neural network model is proven to have the best results among the implementable strategies.

1. INTRODUCTION

1.1 Introduction

The transportation sector is playing a vital role in current society and has become an essential part of the United States economy. There is a significant impact on people's daily lives. For example, people use various methods of transportation, including, but not limited to, daily basis travel, business meetings, medical emergency, etc. Due to urbanization, industrialization, modernization, and globalization, people are encouraged to use transportation. The United States Energy Information Administration (EIA) announced that the transportation sector delivered energy consumption increases at an annual average rate of 1.4 percent. Therefore, the increased transportation demand has led fuel consumption to grow tremendously. EIA has published the United States energy consumption by different sectors in 2017, and the distribution is shown in Figure 1.1.

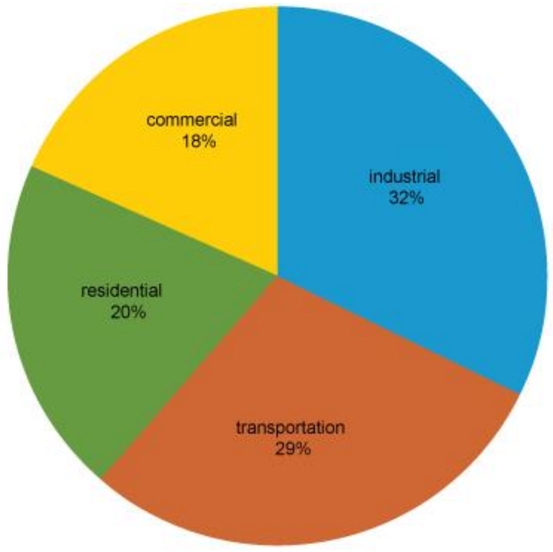


Figure 1.1.1. The United States energy consumption by different sectors in 2017 [1].

Consequently, the transportation sector has become one of the largest contributors to the greenhouse gas emissions in the United States. From the data shared by the United States Environmental Protection Agency, the overview of greenhouse gas emissions from different sectors is shown in Figure 1.2.

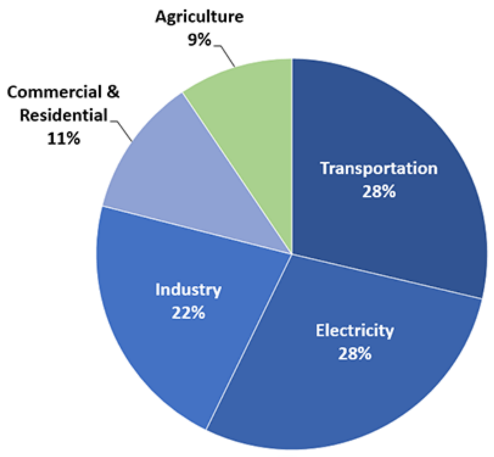


Figure 1.1.2. The total United States greenhouse gas emissions by different economic sectors in 2016 [2].

As of today, fossil fuels, which include gasoline, diesel, and some other liquid fuels, are the primary source of transportation energy worldwide. According to the data from the U.S. Department of Transportation, there are about 183 million light duty vehicles in the United States as of 2016. The yearly number of new registered vehicles in the United states from 1990 to 2016 is shown in the following graph in Figure 1.3.

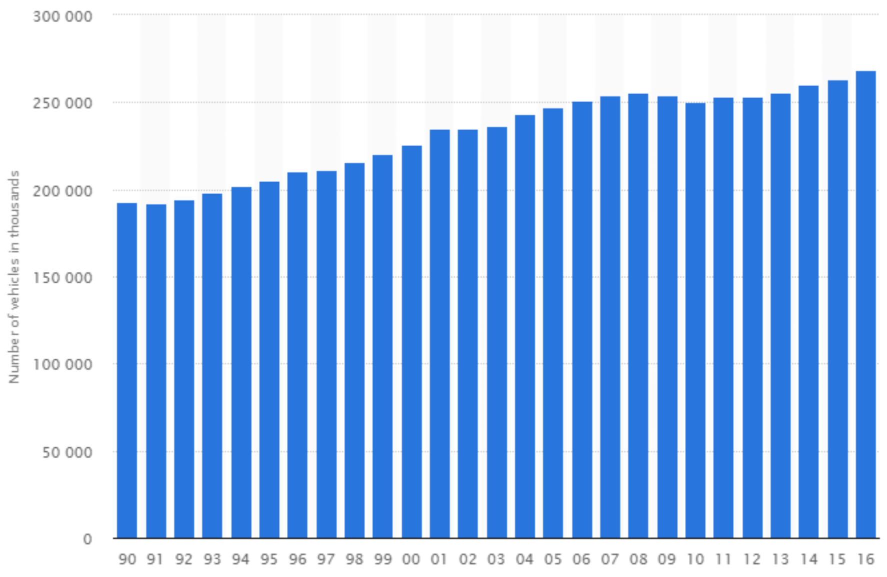


Figure 1.1.3. The number of registered vehicles in the United States from 1990 to 2016 [3]

In the graph, it clearly indicates that the demand for vehicles has increased dramatically in the past two decades. This implies that people will still need to consume an incredible amount of fossil fuel in the future. Even though fossil fuels are globally developed, inexpensive, reliable and easy to assess, there are two major disadvantages using fossil fuels. First of all, since fossil fuels are non-renewable energy, they cannot be replenished once they are harvested. There is only a finite amount of fossil fuel available in our lifetime. Furthermore, the combustion of fossil fuel can damage the environment. Burning fossil fuels results in the production of carbon dioxide, nitrogen dioxide, and other greenhouse gases, which can cause global warming.

The gases staying within the atmosphere are known as greenhouse gases. The principal greenhouse gases and the content percentages are shown in Table 1.1.

Table 1.1. The main types of greenhouse gases and content percentages [4]

Greenhouse gases	**Chemical structure**	**Percentage**
Carbon dioxide	CO_2	81%
Methane	CH_4	10%
Nitrous oxide	N_2O	6%
Hydrofluorocarbon	HFC	3%

The average global temperature record from Urban Milwaukee illustrates that the planet as a whole has warmed up from 1980 to 2015. The graph is shown in Figure 1.4. The y-axis of the graph is the value of the average temperature of the year subtracting the temperature over a long period. As shown in the graph, the average temperature appears to be cooler in between the years of 1860 to 1980. However, the temperature starts to grow hotter when moving toward the 1980s. The highest global average temperature occurred in 2014.

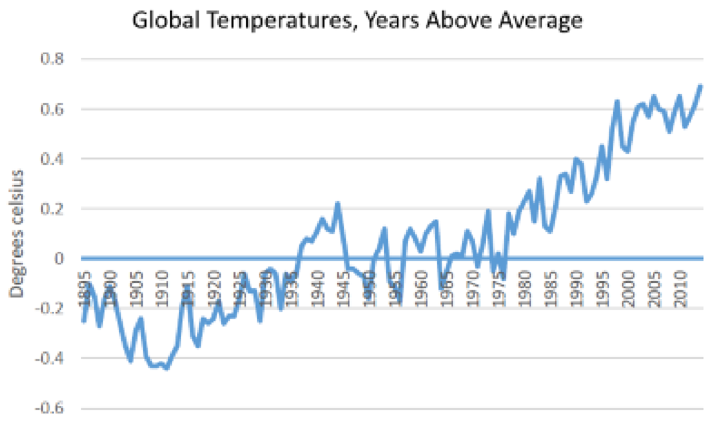


Figure 1.1.4. The average global temperature in Celsius from 1895 to 2015 [5]

In the United States, as shown in Figure 1.5, the temperature has more variability than the global temperature. The United States temperature has risen more quickly since the 1970s. The year of 2012 was the warmest year.

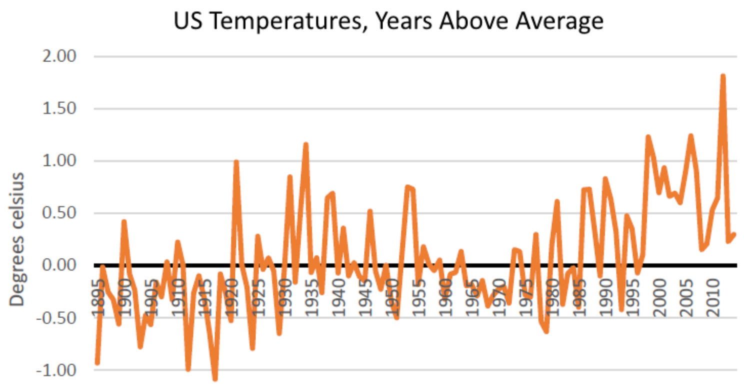


Figure 1.1.5. The United States average temperature in Celsius from 1895 to 2015 [5]

The rapid temperature change is mainly caused by greenhouse gas emissions. Since burning fossil fuels can release carbon, the main source of greenhouse gases is primarily from the combustion of fossil fuels. Figure 1.6 illustrates that carbon dioxide has been increased by 133 percent since1970. In addition, methane and nitrous oxide have grown by 17 and 7 percent, respectively. From Figure 1.2, it is clear that the main contributors of greenhouse gases are the transportation sector and the electricity sector.

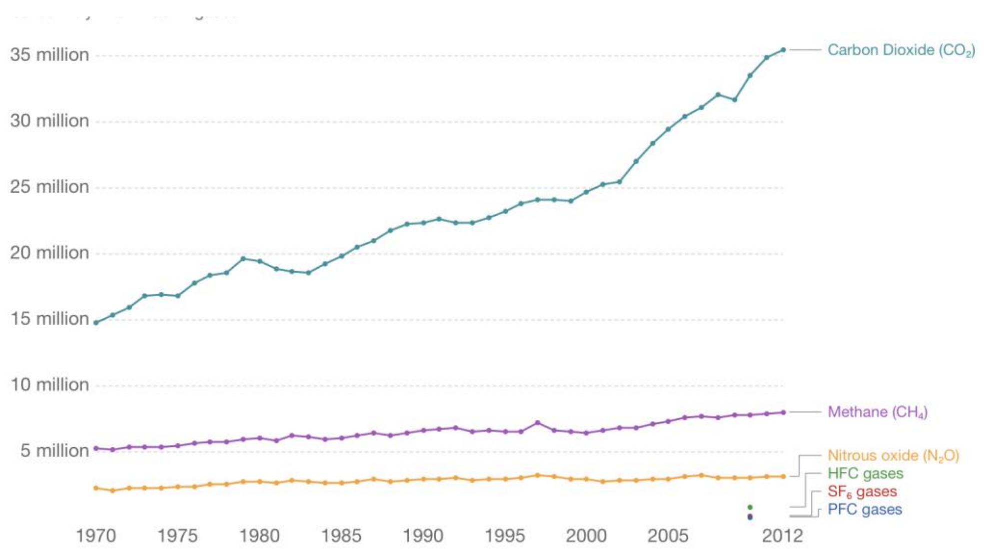


Figure 1.1.6. The global greenhouse gas emissions from 1970 to 2012 [6]

Nowadays, there is a great deal of information and enthusiasm regarding reducing greenhouse gases from alternative energy sources. Until now, diesel engines have been one of the most efficient ways of transporting freight on highways. In order to protect the environment, biodiesel has become popular, since it has the identical functionality to regular petroleum diesel. There are many benefits to using biodiesel. For instance, it can be produced and used at home; it has a minimal environmental impact and can improve air quality due to low exhaust emissions. Consequently, it is a renewable substitute for petroleum diesel.

On the other hand, electric vehicles are becoming more mainstream to reduce greenhouse gas emissions. There are some advantages of using electric vehicles. First, an electric car is powered exclusively by electricity with not only less carbon production, but also fewer toxic gases and smoke emissions. Secondly, less amount of energy is needed to receive the same performance during operation, which makes it more efficient. Thirdly, it can save people a lot of money. Not only the electricity is less expensive than fossil fuels, but also no engine maintenance is needed.

However, due to the capacity limitation and high cost of batteries, electric vehicles have limited range, which leads to a well-known phenomenon: range anxiety. Therefore, hybrid vehicles have more advantages.

Since hybrid vehicles use two different main energy sources, engine and motor, it is necessary to have a systematic study to find the best real-time optimization-based power management strategy. In this research, the power split configuration has been studied through the software-in-loop and hardware-in-loop simulations. A variety of power management strategies will be studied, which includes the following:

1. Dynamic programming
2. Equivalent Consumption Minimization Strategy
3. Proportional State-of-Charge Algorithm
4. Regression Modeling
5. Artificial Neural Network Modeling

For this project, the hybrid vehicle used for testing is a 2013 Chevrolet Malibu, which was developed in the EcoCAR2 competition in 2014. This vehicle features a 1.7 L turbo diesel engine, which is used in the Opel Astra. Both biodiesel and regular petroleum diesel fuel are compatible in the engine. In addition, a 16.2 kW-h Li-ion battery pack was added to power the vehicle. Furthermore, a 100-kW electric motor from Magna was installed in the rear of the vehicle. This testing vehicle will be tested on a dynamometer to achieve the highest accurate results.

1.2 Hybrid Vehicle Architectures

The hybrid system is a combination of a conventional engine and an electric motor. Therefore, a hybrid vehicle can have different modes:

- Mode I – Starting stage or low speed:

Only the electric motor provides power. In this stage, there is no fuel consumption and no exhaust emissions.

- Mode II – Normal driving condition:

The conventional engine starts to play an important role not only to drive the wheels, but also to send power to a generator and recharge the electric motor batteries. In return, the electric motor can provide the vehicle more power and torque while minimizing the fuel consumption.

- Mode III – Deceleration or braking stage:

The system can improve the efficiency by capturing the energy from the turning wheels and the brakes. Then, the kinetic energy will be converted into electrical energy to charge the battery pack for future use. Compared to an electric vehicle, which only has one 1-speed architecture, a Hybrid Electric Vehicle has three different architectures:

1.2.1 Series Architecture

In series configuration, the electric motor is the most important means to provide power to the wheels. The block diagram of such configuration is shown in Figure 1.7.

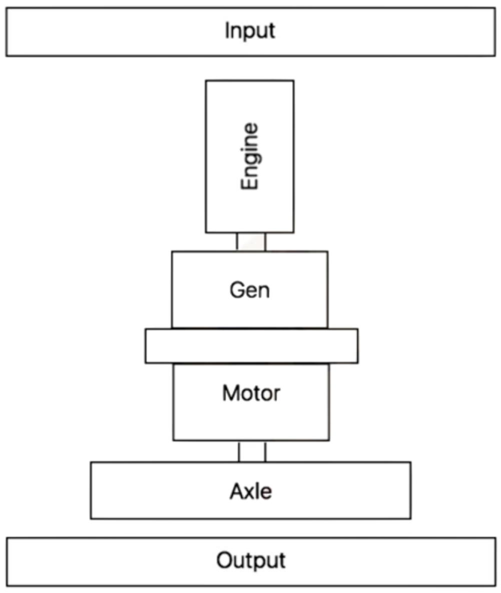


Figure 1.1.7. The block diagram of series hybrid architecture

The conventional engine is the main power source to drive the generator. Then, the generator converts the mechanical energy to charge the battery. Once there is electricity stored in the battery, it will drive the motor and provide energy to the axle and two wheels. Series configuration is not the most optimum configuration. Since the energy is being converted twice from mechanical to electrical and back to mechanical energy, there is a lot of energy loss during the conversion all the way from the engine to the wheels. Moreover, since the electric motor is the only source to power the vehicle, the engine, generator and motor have to be powerful enough. If a vehicle has a high-power demand, this configuration may underperform.

1.2.2 Parallel Architecture

The parallel hybrid system consists of both a conventional combustion engine and an electric motor that power the vehicle individually. The block diagram of such a configuration is shown in Figure 1.8.

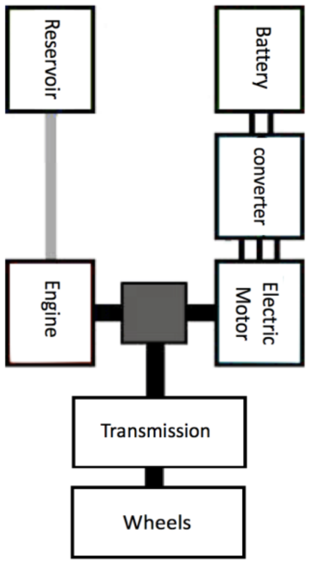


Figure 1.1.8. The block diagram of parallel hybrid architecture

When power demand is low, such as during braking, parallel hybrids utilize the motor as a generator for supplemental recharging to recover kinetic energy. The engine and motor can act much like an alternator in conventional vehicles. Compared to the series configuration, there are many benefits using the parallel configuration. The electrical system in the parallel configuration tends to be smaller than that in the series configuration. In addition, it can meet the instantaneous needed power to power the wheels. On the other hand, since the parallel configuration is more complex than the series configuration, there are some disadvantages of using the parallel configuration. For example, it requires a transmission for the engine.

1.2.3 Parallel Through-the-Road Architecture

The parallel through-the-road configuration has the ability to power each axle with a different driveline model. The block diagram of such a configuration is shown in Figure 1.9.

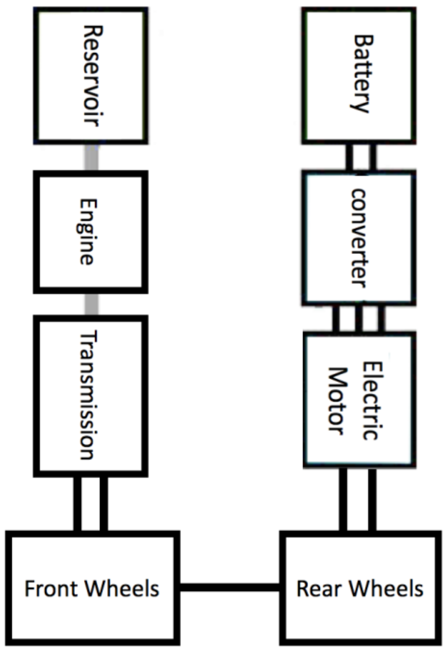


Figure 1.1.9. The block diagram of parallel through-the-road configuration

As shown, there is no coupling between the electric motor and the conventional combustion engine. The electric portion of the powertrain contains a battery and an electric motor, which connects to the rear wheels. On the other hand, the mechanical portion contains a reservoir fuel storage system, a combustion engine and a multispeed transmission, which can power the front wheels. Even though these two different powertrains power each axle independently, they are connected in parallel through the road so that they can rotate at the same speed.

1.2.4 Other Architecture and Approaches to Hybridization

There are some other more complex hybrid vehicle drivetrain architectures. Some are designed to use planetary gears and multiple electric motors. For example, the series-parallel hybrid configuration, which is also known as power-split hybrid configuration, allows the vehicle to operate in either series mode or parallel mode. The block diagram of such a configuration is shown in Figure 1.10.

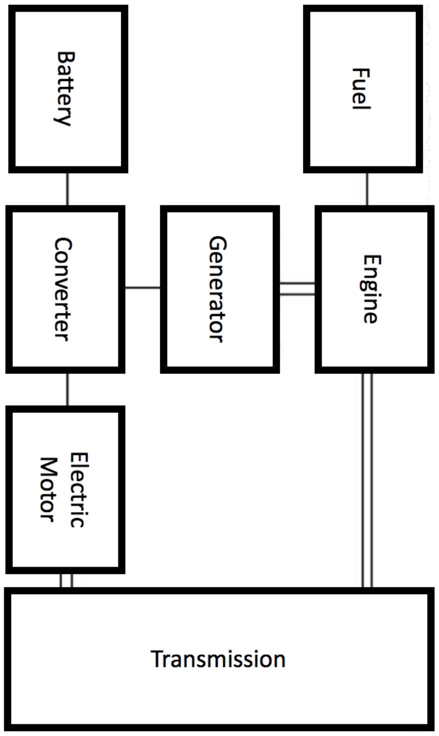


Figure 1.1.10. The block diagram of series parallel configuration

In such a hybrid drivetrain, the power generated from two power sources, an electric motor and combustion engine, can be shared to drive the wheels through a power splitter. It is simply known as a planetary gear set. The combustion engine also acts as a generator, which can charge the batteries. Based on different vehicle state and driver requirements, the ratios of the combustion engine power and the electric motor power can be varied from zero percent to a hundred percent. By using the series parallel configuration, the fuel economy and drivability can be optimized.

In hybrid vehicles, other types of power sources can be used other than a gasoline engine or a diesel engine coupled with an electric motor. For example, some vehicles utilize a combustion engine coupled with a flywheel, compressed air, hydraulic systems, and so on. Hydraulic systems are usually installed in heavy-duty vehicles, because they can provide a large amount of torque.

1.3 Power Management Strategies

In designing hybrid electric vehicle control systems, since two different power sources can drive the vehicle through either mechanical path, electrical path, or the combination of the two paths, it is necessary to develop an energy optimization technique. The power management strategies are different for the plug-in hybrid electric vehicles, because they can be plugged into an electrical outlet to charge the battery. Therefore, in designing plug-in hybrid electric vehicle control systems, one additional dimension must be added compared to regular hybrid electric vehicles. This is because the battery of plug-in hybrid electric vehicles has a larger capacity, since the battery can be charged from some external sources, such as the power grid. In terms of hybrid electric vehicles, because there is no external power source to charge the battery, it is important to optimize the usage of energy from the battery in different driving scenarios. For example, the control system has the ability determine when to deplete the battery, and when to maintain the charge-sustaining mode for the different drive cycles.

1.4 Current Research

The objectives of the project contain two different phases:

Phase 1. Development of models:

This involved three steps. First, choose the correct type of testing vehicle for the research. The individual components, such as engine, motor, battery and powertrain should be calibrated first, and then integrated together to build the full model. Secondly, test the chosen vehicle on a dynamometer and collect data from the dynamometer test. Finally, develop different models for all components of the vehicle mentioned earlier. This portion of the project was completed by Gupta [7].

Phase 2. Power management strategy:

Based on the simplified model, this involved development of various further improved power management strategies, such as dynamic programming, equivalent consumption minimization strategy, proportional state-of-charge algorithm, regression model and artificial neural network model. By using these different control strategies, a further implementation will be employed. These selected strategies will be tested in simulation, then will be compared with the results obtained from the previous phase.

1.5 Distribution of Thesis Content

This thesis consists of five chapters:

Chapter 1 is the introduction to hybrid vehicles. It contains the description of the environmental problems when driving traditional vehicles and the reason why hybrid vehicles are playing an important role nowadays. In addition, it introduces four different hybrid vehicle configurations. After that, it describes various types of power management strategies for hybrid vehicles. Then, it has an overview of the current research.

Chapter 2 reviews the previous work done on the model development and the power management strategies for hybrid vehicles. Furthermore, it introduces the dynamic programming strategies, artificial neural networks and long short-term memory unit.

Chapter 3 has the detailed description of the testing vehicle, the type of controller, and the testing equipment that have been used in the current project. In addition, it provides information how to operate vehicles on the dynamometer, and how to acquire the data. It also introduces the drive cycles and the hardware-in-loop simulation.

Chapter 4 presents various types of power management strategies that have been discovered previously, such as dynamic programming, equivalent consumption minimization strategy, proportional state-of-charge algorithm, and regression model. In this chapter, it will introduce a new power management strategy, which is known as artificial neural network model. At the end of the chapter, there is a comparison among these different power management strategies.

Chapter 5 has the conclusion of the thesis, shows the key contributions, and provides recommendations for future work.

2. LITERATURE REVIEW

This chapter provides the basic knowledge of some important literature, which is related to my work in this thesis. The first section introduces the different categories of control strategies. The second section gives a brief mathematical principle of Dynamic Programming (DP). The third section introduces the basic concept of Artificial Neural Networks (ANN). And the fourth section discusses the Long Short-Term Memory Unit (LSTM), which is an algorithm of ANN.

2.1 Control Strategies

The control logic for the conventional ICE (internal combustion engine) vehicles is simple. For most ICE vehicles, if the ABS, the traction control, and automatic cruise control are not involved, the accelerator pedal and brake pedal are the only inputs, thus there is no need of a control strategy. However, the control strategies are significant for hybrid electric vehicle (HEV). The energy consumption, emission, and driving performance [8] are the variables that need to be optimized and all of them rely on the control strategy.

- Numerical optimization methods:

  If the entire drive cycle is known in advance, this solution can be implemented. Dynamic programming (DP) [9] [10] [11] [12] is widely used for optimization in this category. Since the drive cycle for a real vehicle is unpredictable, this method is obviously not implementable. However, since this method collects and calculates the entire information of the drive cycle, and is computationally intensive, even though it is not directly implementable, it can be used as a perfect benchmark for other strategies.

- Analytical optimization methods:

  Since the numerical optimization for HEV is complex and requires lots of computational time, analytical methods help simplify computation. The entire drive cycle is still required to be known for this method. Pontryagrin's minimum principle (PMP) [13] [14] [15] is an example of this method.

- Instantaneous minimization methods:

  This method calculates the problem as a sequence of local problems and minimizes at each time step instead of approaching the global optimization. It requires the definition of local minimization instead of the entire drive cycle. Equivalent energy consumption minimization strategy (ECMS) is widely used in real-time control as a simple example of this method. ECMS was first introduced in reference [16] and developed by Giorgio Rizzoni and other authors at Ohio University [17] [18] [19] [20]. This method is achieved by considering the battery as an auxiliary and restorable fuel tank and adding the equivalent fuel consumption to minimize the instantaneous energy consumption. This method can be used in real time since it outputs the instant optimization. However, the optimization has to be tuned in advance for every drive cycle to achieve the ideal result. Thus, this method is still not practical for real vehicles since drive cycles are unpredictable.

- Heuristic methods:

  Heuristic methods use rules and algorithms instead of minimization or optimization. Based on the parameters of the vehicle and engineering rules, the computation is relatively simple and robust. Fuzzy-logic-based control [21] [22] is one example of this method. Since it does not use an optimization method, the result may not be optimal, and it depends on how well the rules are tuned and trained.

- Blended methods:

  Obviously blending two or more methods can combine the advantages of each method. For example, a method introduced in [23] blends DP (Numerical optimization methods) and Fuzzy-logic-based controls (Heuristic methods).

In this thesis, the DP (Numerical optimization method), ECMS (Instantaneous minimization method), proportional SOC (Heuristic method), regression approaching DP and neural networks approaching DP are discussed to optimize the problem. Among these five methods, proportional SOC, regression approaching DP and Neural networks approaching DP are the only methods that can be implemented in real vehicles.

2.2 Dynamic Programming

In 1940, Richard Bellman first invented the term ‘Dynamic Programming’ to define a problem-solving process where we have to make the best decisions one after another. Then he re-clarified the definition of DP which is ‘specifically to nesting smaller decision problems inside larger decisions’ [24]. Dynamic Programming (DP) is an approach for efficiently solving optimization problems that require a broad range of searching and consist of overlapping subproblems and optimal substructure. DP solves global optimization problems by finding the optimal solution for sub-problems then, step by step, reaching the final optimal solution. This approach is based on Bellman's Principle of Optimality, which he wrote as below:

"An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision." [25]

In general, this statement concludes that 'optimal policies have optimal sub-policies'. And it can be understood in a straightforward way. Suppose there is a problem having a sub-problem with a non-optimal sub-policy. If we substitute the sub-policy with the optimal one, it is easy to find out that the optimal sub-policy will improve the previous original policy. Problems that can be solved by DP have the 'optimal substructure' property, which is that the globally-optimal solution can be constructed from locally-optimal solutions to subproblems.

To explain DP mathematically, we consider the following discrete-time system

$$x_{k+1} = f_k(x_k, u_k) \tag{2.1}$$

where *k*=0,1…*N*-1 and $u_k \in U(x_k)$. And we have the input of control given by

$$\pi = \{u_0, u_1, \dots u_{N-1}\}, \tag{2.2}$$

with the control inputs such that $u_k \in U(x_k)$ for all x_k. Then we have the cost of π at the beginning time step x_0 in the following equation

$$J_\pi(x_0) = L_N(x_N) + \sum_{k=0}^{N-1} L_k(x_k, u_k) \tag{2.3}$$

where L_k is the instantaneous function.

Since the optimal cost function is defined to minimize the total cost, it can be mathematically stated in the following way:

$$J^*(x_0) = \min\ J_\pi(x_0) \tag{2.4}$$

Also, the optimal policy will be

$$\pi^* = \{u_0^*, u_1^*, \dots u_{N-1}^*\} \tag{2.5}$$

According to equations 2.4 and 2.5，it is straightforward to see that

$$J_{\pi^*}(x_0) = J^*(x_0) \tag{2.6}$$

Now, if we want to minimize the cost from time *i* to time *N*,

$$V_i = L_N(x_N) + \sum_{k=i}^{N-1} L_k(x_k, u_k) \tag{2.7}$$

and consider the “tail policy”$\{u_i^*, u_{i+1}^*, \dots u_{N-1}^*\}$, i.e., due to Bellman’s optimality theory, the tail policy is the optimal solution to the tail sub-problem. Thus, the dynamic algorithm with this principle begins at the final step *N*, and backwardly reaches the start step by using the sequence of policies

$$u_k^* = \arg \min_{u_k \in U(x_k)} (L_k\,(x_k, u_k) + J_{k+1}(x_{k+1})) \qquad \text{（2.8）}$$

where *k*= *N*-1, *N*-2, ….,1, 0. By this procedure, $J_0\ (x_0)$ is obtained at the end, which is just the optimal cost.

2.3 Artificial Neural Networks

Artificial Neural Networks (ANN) are a very powerful computing system that is designed to perform certain specific tasks like clustering, classification, and pattern recognition by simulating the animal’s neural networks. This biologically-inspired system resembles the human brain in two ways. Firstly, ANN acquire knowledge through learning. Secondly, ANN’s knowledge is stored within the inner-neuron connection strengths (called hidden-layers) as synaptic weights.

Most ANN structures consist of connected units and/or nodes that mimic the biological neurons in the brain. A typical ANN contains a huge number of neurons (units) built in a series of layers including input layers, hidden layers and output layers. Those units that receive signals from the outside world are located in the input layer. With these input signals, the network will learn and recognize. Then, input signals are conveyed from one neuron to another by the links between them and those units that connect inputs and outputs are hidden layers. In real ANN implementation, the signal at each neuron is represented as a value and the links between neurons are assigned with weights. By some simple mathematic computation, the signal changes its state in a forward pass. The weights of connections and bias will be adjusted to achieve the desired value at the output according to some specific loss function (or objective function). Eventually, the signals reach the final stage called the output layer, which consists of units that respond to the previous signal about how it’s learnt any task.

As mentioned above, the weights and bias among links need to be adjusted and we call them free parameters. The process of adjusting free parameters is just the learning process in ANN. For learning to take place, ANN requires training in the first place. While training, many learning algorithms have been introduced:

(1) Gradient Descent is the most basic learning algorithm used in supervised training cases. When the predicted output is different from the target output, there will be a difference or error and we call it loss. By finding the gradient, the weights will be changed in order to minimize the loss.

(2) Back Propagation is an extension of gradient-based delta learning rule. If there exists error between the prediction and target, the error will be propagated backward from the last layer (output) through intermediate layers (hidden layers) to the first layer (input). Mathematically, the chain rule is applied during back propagation.

Due to ANN's capability of learning, it is widely used in many applications. For example, ANN is applied to tag spam emails among all the emails by doing a classification. In addition, financial analysts use some prediction models of ANN to predict the stock market trend. Also, in this thesis (Chapter 4), an ANN structure named Long Short-Term Memory Unit (LSTM) helps us to predict the vehicle's fuel and electricity split given the velocity, acceleration and other information.

2.4 Long Short-Term Memory Unit

As ANNs have been developed, researchers have proposed various neural network structures in order to solve tasks with different goals. The most common structures are Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). RNN is a special neural network architecture that is able to process the sequential signal. With a feedback link, RNN memorizes the temporal property of the input signal and further makes a prediction for the current output. Thus, RNN is especially useful for tasks based on time sequence signal, such as unsegmented, connected handwriting recognition or speech recognition.

Although, theoretically, RNNs are capable of handling long-term information, in practice, RNNs have difficulties in learning 'long-term dependencies' due to the 'gradient vanishing' problem discovered by Hochreiter [26] and Begio, et al.[27]. Fortunately, Hochreiter and Schmidhuber [28] designed a unit called Long Short-Term Memory unit (LSTM) which solves this problem.

LSTM is a special kind of RNN that is able to learn long-term dependencies. A common LSTM structure includes a cell state, input gate, output gate and a forget gate (f_t) which is shown in Figure 2.1.

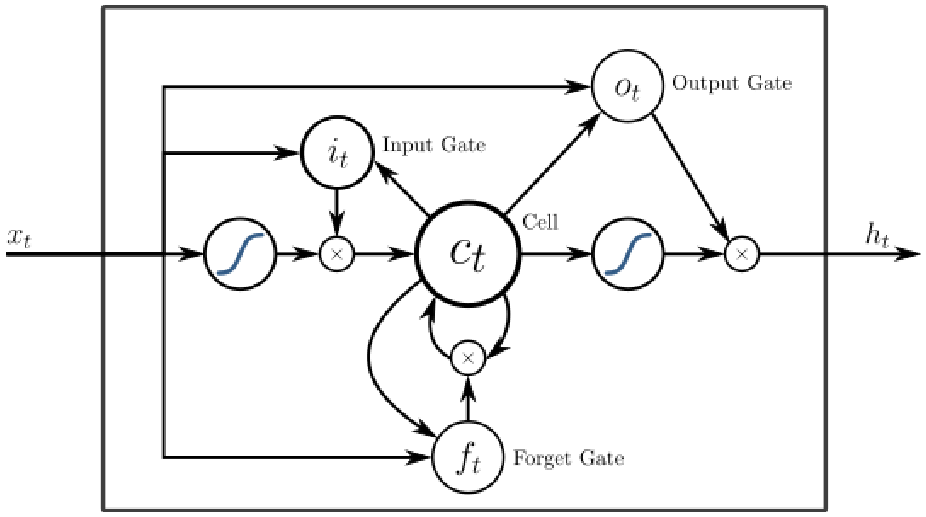


Figure 2.1. Long short-term memory unit structure [29]

1. The cell state (C_t) is the most pivotal part of LSTM and is usually understood as the long-term memory.

2. Forget gate is the first step for the signal to go through. LSTM uses a sigmoid layer called ‘forget gate layer’ to decide what information to throw away from the cell state.

3. Then LSTM decides which new information to be stored in the C_t through a *tanh* layer after a sigmoid layer. The sigmoid layer is called ‘input gate layer’ which determines which element in the C_t is going to be updated. And the *tanh* layer generates the updated values to replace the old states.

4. To update the old C_t is easy. The previous steps already find which value to forget and which value to update.

5. Finally, LSTM computes the output. The output is a filtered version of the C_t. A sigmoid layer is applied to find which part of the C_t is going to be selected as output. Then the C_t is fed into a *tanh* layer in order for the value to be between -1 and 1. Multiplying the previous output with the result from *tanh* will be the final output.

3. SYSTEM ARCHITECTURE

3.1 Vehicle Platform

The current vehicle platform for the project is a 2013 Chevrolet Malibu. This testing vehicle was provided by EcoCAR2, which is a college-level advanced vehicle technology engineering competition formed by the United States Department of Energy and General Motors, in June 2012. Chevrolet Malibu is a mid-size car manufactured since 1964. The first available hybrid version of Malibu in the market was manufactured in 2016.

The context of the EcoCAR2 competition is to build a parallel-through-the-road plug-in hybrid vehicle (PHEV), which still maintains high performance while minimizing energy consumption and greenhouse gas emissions. The Purdue team displaced the Malibu 2.4L gasoline engine with a General Motors 1.7L CIDI diesel engine, which utilizes 20 percent biodiesel and 80 percent diesel fuel. Also, this testing vehicle features a turbocharger, exhaust gas recirculation, and charge air cooler. For the electric drive train, a 100kW Magna motor is coupled to the rear wheels. In order to power the motor, the 288V DC Energy Storage System (ESS), which consists of a DC-DC converter and A123 6S15P3 battery pack, is installed in the trunk of the vehicle. The plug-in capability is provided to the vehicle through a BRUSA charger. The schematic diagram of the parallel-through-the-road PHEV design is shown in Figure 3.1.

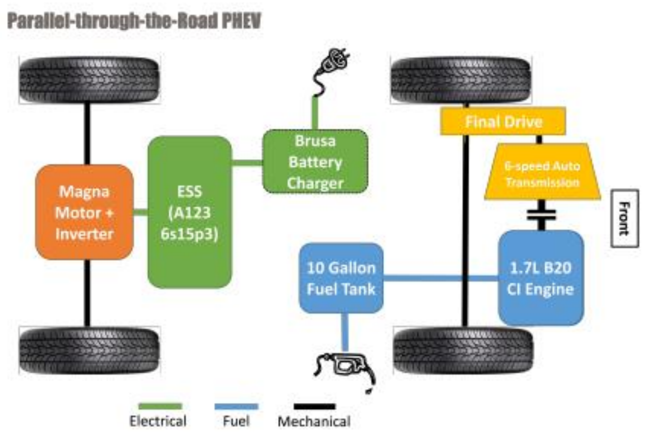


Figure 3.1. The final architecture selection of the Parallel-through-the-road PHEV [7]

For the fuel system, the regular Malibu's fuel tank for the front drivetrain was replaced with a smaller 10-gallon fuel tank. The battery can be charged by using a level-2 charging station. The detailed information for the two drivetrains is shown in Tables 3.1 and 3.2.

Table 3.1. Detailed information of the front drivetrain [7]

Front Drivetrain	
Name of the component	**Detail information**

IC engine	Originally used in Opel Astrea;
	Turbocharged 1.7L diesel engine with EGR;
	4 cylinder in-line. Rated 96 kW at 2500 RPM.

Fuel System	Denso common rail fuel system
Transmission	GM 6T40 6-speed automatic transmission
Fuel Storage	10-gallon-capacity tank

After-treatment	Under-floor Diesel Oxidation Catalyst (DOC);
	Diesel Particulate Filter (DPF);
	Urea Selective Catalytic Reduction (SCR) configuration;
	Onboard Diesel Exhaust Fluid (DEF) storage and delivery

Table 3.2. Detailed information of the rear drivetrain [7]

Rear Drivetrain	
Name of the component	**Detail information**
Electric motor	100 kW Magna motor
Transmission	Fixed gear transmission integral to motor
Energy Storage system	16.2 kWh A123 Li-ion battery with 6S15P3 configuration

3.2 Supervisory Controller

The vehicle controller used in the current work is a MicroAutobox II controller, which was donated by dSPACE. The vehicle controller is used as a real-time system for many different fast control prototyping applications, such as powertrain, chassis control and so on. The features of the controller are the same as the previous work. The technical details of MicroAutobox II controller hardware are shown in Tables 3.3 and 3.4.

Table 3.3. The specifications for dSPACE MicroAutobox II controller [30]

Parameter	**Specification**
Processor	IBM PPC 750 FL, 900 MHz (includes 1 MB level 2 cache)
Memory	16 MB main memory; 6 MB memory exclusively for communication between MicroAutoBox and PC/notebook; 16 MB nonvolatile flash memory containing code section and flight recorder data; Clock/calendar function for time-stamping flight recorder data.
CAN interface	4 channels
Analog Inputs	32 16-bit channels
Analog Outputs	8 16-bit channels
Digital Inputs	24 outputs
Digital Outputs	24 outputs, 5 mA output current
Voltage operation	12V
Signal conditioning	Overvoltage protection; overcurrent and short circuit protection

Regarding the SAE-J113-41 standards, the power under normal operating conditions should be lower than 25 Watts. The power supply, such as power inputs and power outputs, are provided by Zero insertion force (ZIF) connectors. For signal monitoring and flashing the memory, an ethernet connection is used.

3.3 Controller Area Network (CAN) communication

A modern vehicle can have up to 70 electronic control units (ECUs), which are known as nodes. The ECUs include an engine control unit, powertrain control module, transmission control unit, etc. Previously, automotive manufacturers produced vehicle ECUs connected by using increasingly complex point-to-point wiring systems. When the number of electronics in vehicles grew, too many wires were stuffed into a small space, which can be heavy and expensive. Therefore, it became necessary to replace dedicated wires with some in-vehicle networks to reduce cost, complexity, and weight.

This is where the CAN standard comes in handy as it allows ECUs to communicate with each other without complex dedicated wiring in between. CAN is known as a high-integrity serial bus system for intelligent device networking. It emerged as the standard in-vehicle network. The ECUs can be connected via the CAN, which acts as a central networking system. It allows all the ECUs to communicate within the whole system without causing any dangerous overheating and overload to the controller computer [31]. A comparison of a system with or without CAN is shown in Figure 3.2.

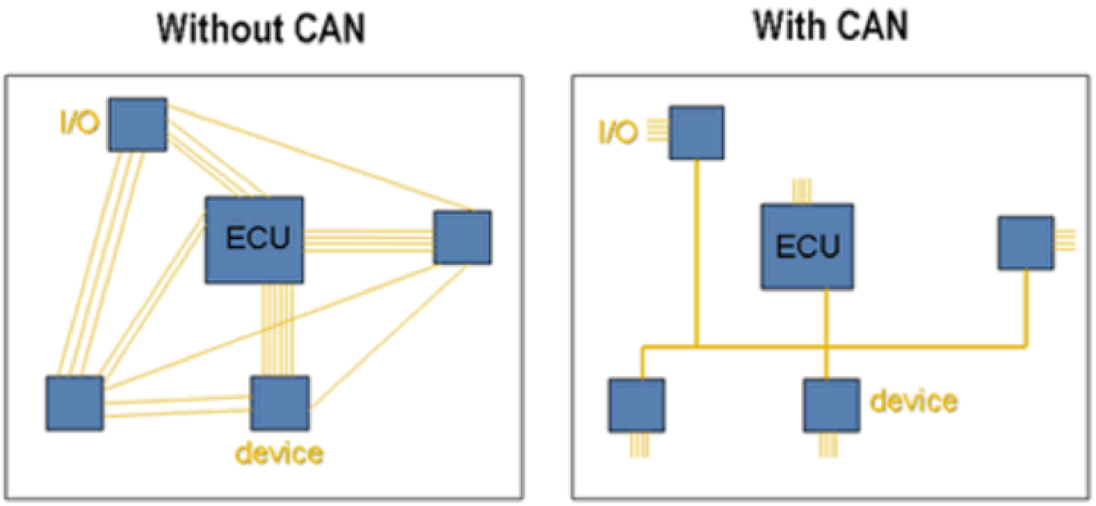


Figure 3.2. The networks in different systems [32].

Here are some advantages from using CAN in vehicles:

1. Low cost: ECUs communicate via a single CAN interface, which is an inexpensive and durable network that assists multiple nodes to communicate with each other. In other words, there are no direct analog signal lines to every device in the vehicle, which can reduce errors, weight and costs.
2. Centralized: The CAN system allows for central error diagnosis and configuration across all ECUs.
3. Robust: CAN offers robust communication among different nodes. The system is robust towards failure of subsystems and electromagnetic interference, making it ideal for vehicles.
4. Efficient: Each ECU contains a CAN controller chip. Some messages are prioritized based on their own IDs, so that the highest priority IDs will not get interrupted.
5. Flexible: Each ECU contains a chip allowing it to receive all transmitted messages. Different nodes can decide the relevance of messages and act accordingly. This allows easy modification with minimum impacts and inclusion of additional nodes such as CAN bus data loggers.

As shown in Table 3.3, the number of CAN interfaces has 4 channels. It includes engine CAN, vehicle CAN, motor CAN and battery CAN. The detailed information for each channel is described in Table 3.4.

Table 3.4. Detailed information of 4 different CAN interfaces [7]

CAN number	**CAN channels**	**Description**
1	Vehicle CAN	Vehicle CAN box is connected to all the systems of the vehicle, such as engine, transmission controller, etc. When the engine is running but the motor is not working, the engine CAN and vehicle CAN buses will be connected together. When the motor is working but the engine is not running, the engine CAN still sends the information to the vehicle CAN. The engine CAN will fake the signals through the MicroAutobox so that it will pretend the engine is still connected to the vehicle. When the vehicle believes that the engine is on, the shift lever will put on Neutral. Since the CAN channel is connected to the after-treatment controller, it will deliver all the information without collecting any data from the after-treatment controller.
2	Engine CAN	The operating engine will collect all the information and send the required signals to the vehicle CAN through the MicroAutobox
3	Motor CAN	The motor CAN, which is similar to the engine CAN, is on a separate CAN bus. This is to accommodate the CAN message protocols from two different manufacturers and prevent CAN conflicts. The motor CAN is mainly for handling the function of the electric motor.
4	Battery CAN	The battery CAN, which is similar to the engine and motor CAN, is also operating on a separate CAN bus. The battery charger from BRUSA is connected to the same CAN channel as the battery control module.

There are three important CAN Bus message components for data logging: the CAN ID, the Control field and Data field. Since CAN only provides the raw data, which is the basis for communication, a standardized protocol is needed to further communicate among different ECUs. The encountered standard used in this project is known as SAE J1939 standard, which is defined by the Society of Automotive engineers (SAE). It provides a set of standardized messages and conversion rules that apply across all the components in the vehicle.

A three-wire J1939 CAN cable is used in the current testing vehicle, which includes a shielding wire, CAN High wire and CAN Low wire. According to the SAE J1939 standards collection, the length is maintained to be less than 40 meters. The CAN is also manufactured with two 121-Ohm termination resistors at each end to prevent overloading and reflections.

3.4 Drive Cycles

A driving cycle is one of the methods to assess the performance of vehicles in various modes, such as fuel consumption and polluting emissions. It represents the relationship between the speed of a vehicle versus time. The current work used the United States Environmental Protection Agency federal test procedures to examine the testing vehicle in two different driving conditions. For instance, Urban Dynamometer Driving Schedule (UDDS) is used to simulate the fuel economy in urban driving conditions; Highway Fuel Economy Test (HWFET) is used to simulate the tailpipe emissions and fuel economy in highway driving conditions. Detailed information is described below:

1. UDDS testing

   The Urban Dynamometer Driving Schedule is also known as FTP-72 test or LA4 test. The cycle simulates urban driving conditions as described in Table 3.5 and shown in Figure 3.3.

Table 3.5. Detail information about UDDS testing [7]

Parameter	**Specification**
Duration	1369 seconds
Distance	12.07 km (7.45 miles)
Average Speed	31.5 km/h (19.59 mph)
Maximum Speed	91.2 km/h (56.7 mph)

The cycle simulates a total 12.07 km urban route with frequent stops. The maximum speed of the testing vehicle should reach to 91.2 km/h and the average speed is 31.5 km/h. A total 1369 s duration includes two different phases in the driving cycle:

a. A "cold start" phase of 505 seconds over an overall distance of 5.78 km at 41.2 km/h average speed

b. A "transient phase" of 864 seconds

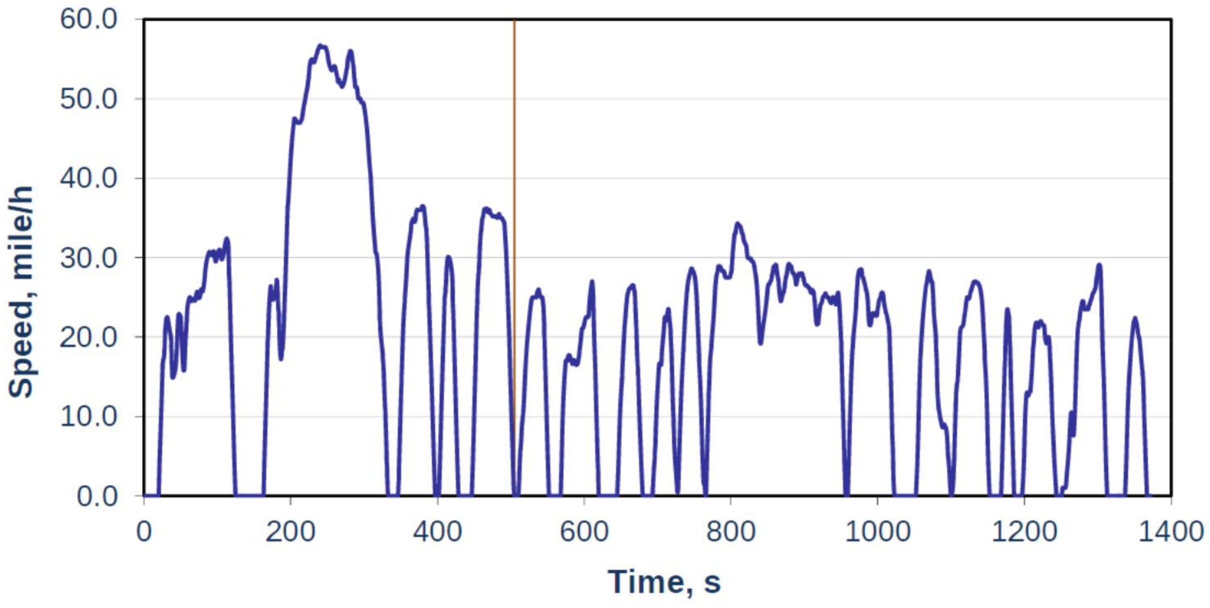


Figure 3.3. The EPA UDDS driving cycle

2. HWFET testing

   The cycle simulates highway driving conditions as described in Table 3.6 and shown in Figure 3.4.

Table 3.6. Detail information about HWFET testing [7]

Parameter	**Specification**
Duration	765 seconds
Distance	16.51 km (10.26 miles)
Average Speed	77.7 km/h (48.3 mph)
Maximum Speed	96.6 km/h (60 mph)

During this test, a warmed-up engine is used to run an overall 16 km distance with no stops. The average speed of the vehicle is 77 km/h with a top speed of 97 km/h.

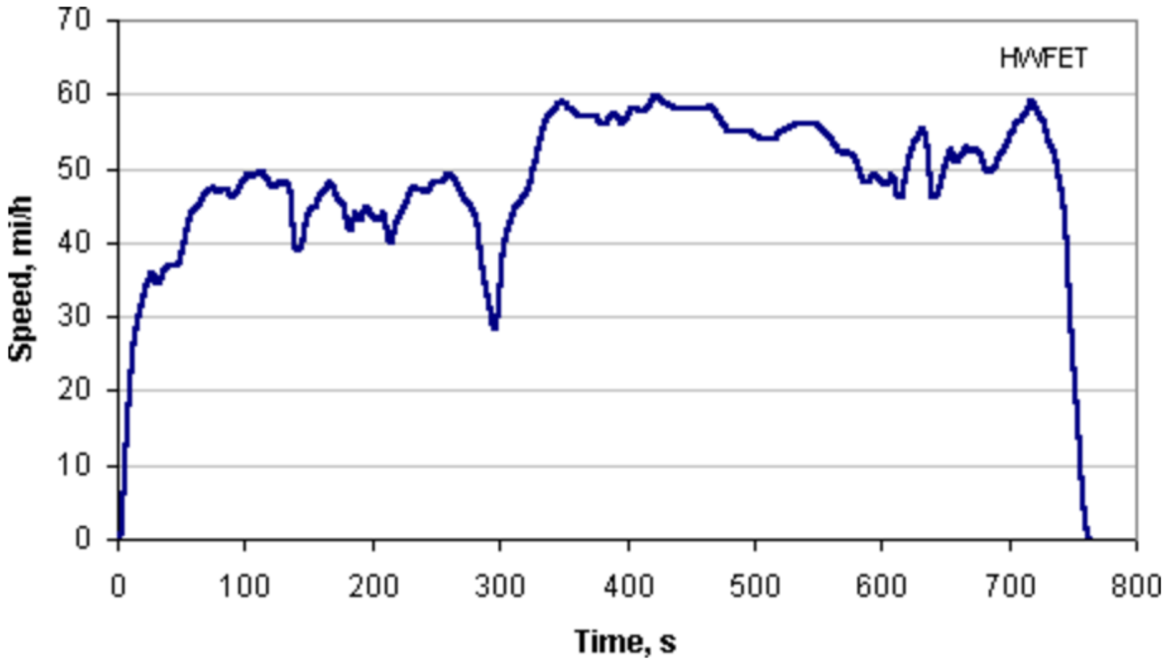


Figure 3.4. The EPA HWFET driving cycle

3.5 Hardware-in-Loop (HIL) Simulation Setup

HIL simulation is a well-known real-time simulation technique used to test many different systems, such as the controller. For a regular simulation loop, a controller, such as PID, can send a signal to the simulated process, then the simulated process will send back a signal to the controller. The controller will then again send a signal to the simulated process. Therefore, the loop is known as the simulation loop. All of this can be done by computer. However, for a HIL simulation, hardware will be used to implement the controller. It can provide an effective platform by adding the complexity of the plant under control to the test platform, such as providing feedback signals and control signals. Therefore, there are some advantages of using HIL simulations, such as safe, low cost, repeatablility, and so on.

The current HIL setup consists of the HIL test-bench from dSPACE. Even though the testing vehicle is implemented in software through Automotive Simulation Models (ASM) from dSPACE, the supervisory controller is implemented in hardware on the MicroAutobox. ASM is a real-time model, which can be used to simulate the combustion engines, electric components, etc, in different traffic scenarios. The schematic of the HIL setup is shown in Figure 3.5.

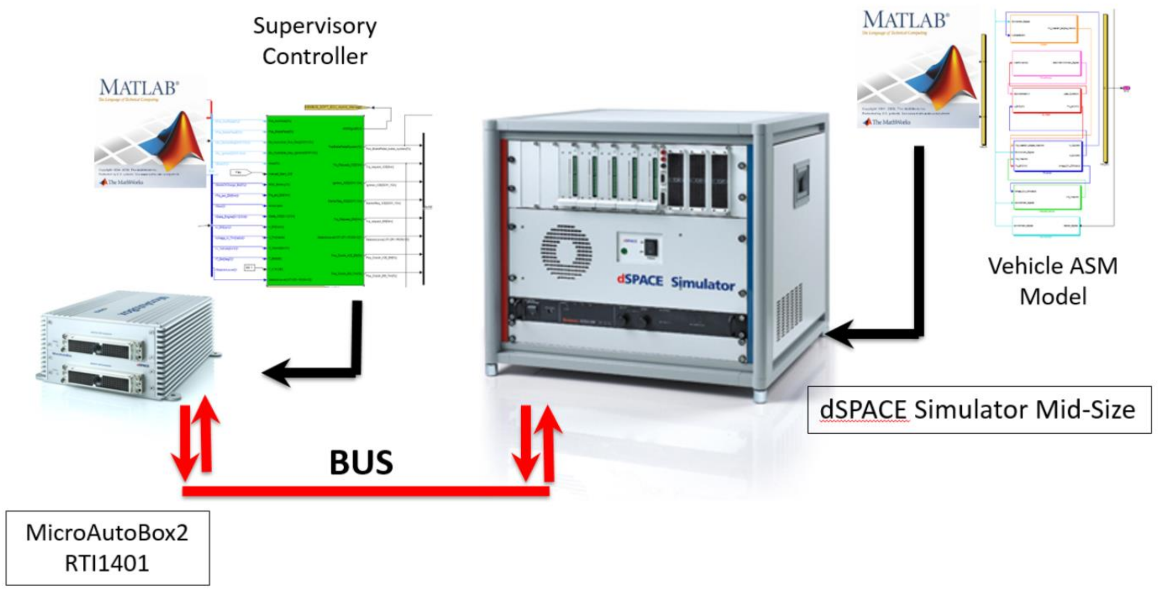


Figure 3.5. The brief schematic of HIL simulation setup [7].

4. POWER MANAGEMENT STRATEGY DEVELOPMENT

4.1 Simplified Model

In chapter 3, the Automotive Simulation Models (ASM) were introduced. A control strategy has to be developed to approach the best energy consumption. There are two types of computational approaches, forward-facing method and backward-facing method. In the forward-facing method, the total powertrain torque is given firstly, and used to calculate the simulating vehicle velocity. The simulating velocity is compared with the actual vehicle velocity by PID control. In the backward-facing model, the known vehicle speed is used to calculate powertrain torque. Since both the motor and engine provide the powertrain torque, the torque split also needs to be calculated.

The forward-facing method, compared to the backward-facing method, is more complicated from a computational aspect.

Since the vehicle speed is given by the drive cycle, the total tractive force F_{tract} required at the wheels can be calculated for the drive cycle. It is the sum of the road load force F_{load} and the inertial force $F_{inertia}$,

$$F_{tract} = F_{load} + F_{inertia} \tag{4.1}$$

where F_{load} is given in equation 3.2 in Chapter 3 and $F_{inertia}$ is given as,

$$F_{inertia} = M_{veh} \times \frac{dv}{dt} \tag{4.2}$$

where M_{veh} is the total mass of the vehicle. Therefore, the inertia forces of wheels and gears are not considered in the simplified model.

The angular velocity of the wheel ω_w is,

$$\omega_w = \frac{v}{r_w} \tag{4.3}$$

where r_w is the wheel radius.

The angular velocity of the engine differential is,

$$\omega_{ed} = \omega_w \times X_{ed} \tag{4.4}$$

where X_{ed} is the engine differential ratio.

The angular velocity of the engine is,

$$\omega_e = \omega_{ed} \times X_g \tag{4.5}$$

where X_g is the transmission gear ratio.

The angular velocity of the motor differential is,

$$\omega_{md} = \omega_w \times X_{md} \tag{4.6}$$

where X_{md} is the motor differential ratio.

The angular velocity of the motor is equal to the motor differential speed,

$$\omega_m = \omega_{md} \tag{4.7}$$

The total wheel torque required τ_v is,

$$\tau_v = \frac{F_{tract}}{r_w} \tag{4.8}$$

The torque split is the ratio between the motor torque at the wheels τ_{mv} and the total torque at the wheels τ_v ,

$$Split = \frac{\tau_{mv}}{\tau_v} \tag{4.9}$$

Therefore, the engine torque at the wheels τ_{ev} is,

$$\tau_{ev} = \tau_v - \tau_{mv} \, . \tag{4.10}$$

The total torque can be calculated from the vehicle speed, which is given by the drive cycle, and the torque $Split$ is the output of the power management strategies. The value for torque $Split$ is between -1 to 1. Table 4.1 shows the different modes.

Table 4.1. Values of the Torque $Split$.

Model	Value
Charging-only	-1
Charging	$(-1,0)$
Engine-only	0
Blended	(0,1)
Motor-only	$1\ ,\ \tau_v > 0$
Regeneration	$1\ ,\ \tau_v < 0$

The charging mode represents the case when the torque provided by the engine is more than that required by the vehicle, which means the motor acts as a generator to charge the battery. This situation happens when the vehicle is decelerating, or the battery State of Charge is too low. When all the torque provided by the engine is used to charge the battery, the torque $Split$ value is -1. The blended mode represents the case when the motor and engine work together to provide the torque. When only the engine provides torque, the torque $Split$ value is 0. When only the motor provides torque, the torque $Split$ value is 1. When the total torque required is negative, and used to charge the battery, the engine is idle and the motor works as a generator, the torque $Split$ value is 1.

The torque provided by the motor τ_m is,

$$\tau_m = \frac{\tau_{mv}}{X_{md}} \tag{4.11}$$

subject to the engine mechanical limitation,

$$\min(\tau_m) \leq \tau_m \leq \max(\tau_m) \tag{4.12}$$

where

$$\max(\tau_m), \min(\tau_m) = \ f(\omega_m) \tag{4.13}$$

The torque provided by the engine τ_e is,

$$\tau_e = \frac{\tau_{ev}}{X_{ed} \times X_g} \tag{4.14}$$

subject to the motor mechanical limitation,

$$0 \leq \tau_e \leq \max(\tau_e) \tag{4.15}$$

where

$$\max(\tau_e), = \ f\left(\omega_e, N_g\right), \tag{4.16}$$

and the gear number (N_g) is a significant parameter. It can be calculated from the transmission model. The transmission shift map of the 6T40 GM transmission is shown in Figure 4.1.

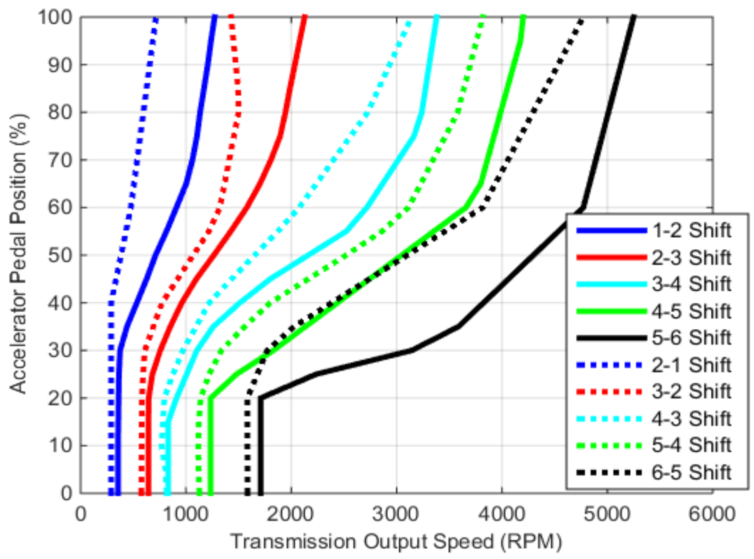


Figure 4.1. Transmission Map [7].

The map shows the accelerator pedal position (APP) as function of transmission output speed. The change of gear number is instantaneous from 0-6. At each time step, the accelerator pedal position is calculated from the engine power, which combined with the gear number locates the transmission output speed in the map. Since the gear number of the next step is unknown, two output speeds, including an up-shift speed ($\omega_{low-high}$) and a down-shift speed ($\omega_{high-low}$) are located at the same time. Engine speed (ω_e) is then compared to the two output speeds. If it is greater than the up-shift speed, the gear number is incremented. Conversely, if engine speed is lower than the downshift speed, the gear is reduced, which is shown in Table 4.2.

Table 4.2. Change of Gear number

If	Then
$\omega_e > \omega_{low-high}$	$N_{g_t} = N_{g_{t-1}} + 1, N_{g_t} \in [1,6]$
$\omega_e < \omega_{high-low}$	$N_{g_t} = N_{g_{t-1}} - 1, N_{g_t} \in [0,5]$

To acquire the gear number, the accelerator pedal position (APP) must be calculated firstly. Pedal position is related to the engine power (P_e),

$$P_e = \tau_e \times \omega_e \tag{4.17}$$

The relationship of the accelerator pedal position (APP) to the engine power (P_e) is shown in Figure 4.2.

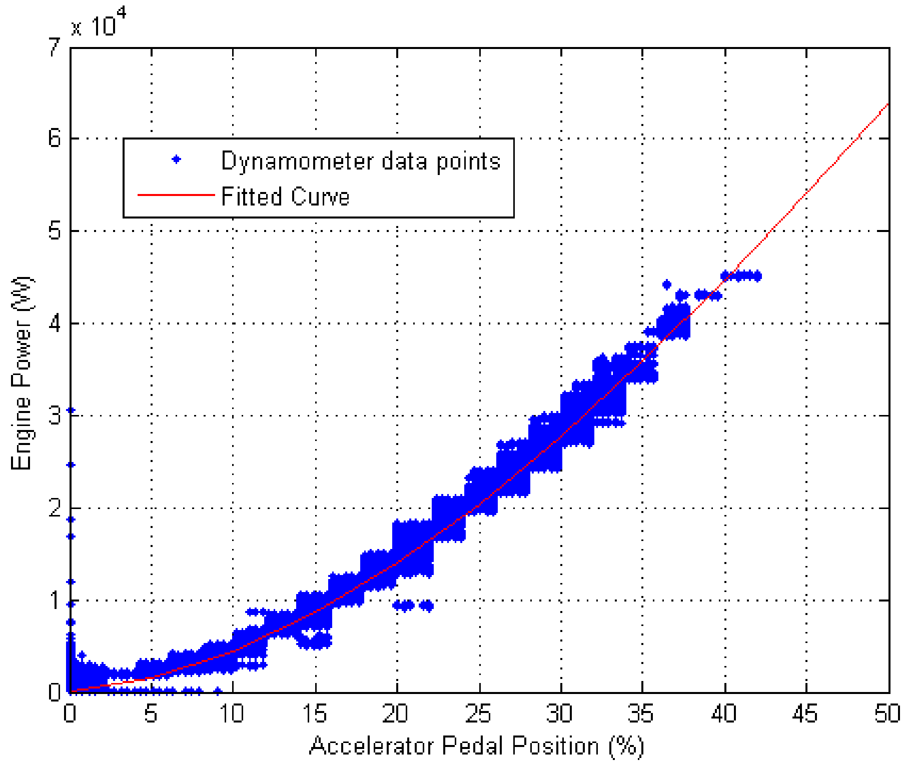


Figure 4.2. Relationship between accelerator pedal position and engine power [7]

For the motor, the current (I_{m}) can be calculated as,

$$I_{\mathrm{m}} = \frac{\tau_m \times \omega_m}{V_m} \tag{4.18}$$

where V_m is the motor voltage, which is assumed to be 300V in the model. A 10% power loss between battery and motor in either direction is also assumed in this model. Therefore, battery current (I_{batt}) can be calculated from motor current (I_{m}) as follows:

$$I_{\mathrm{batt}} = f(I_{\mathrm{m}}) \tag{4.19}$$

The state of charge for the next step (SOC_{t+1}) can be calculated from the present step (SOC_t),

$$SOC_{t+1} = \left(SOC_t \frac{I_{batt}/(N_p \times Q_{cell})}{\Delta t}\right) \tag{4.20}$$

where $N_p \times Q_{cell}$ is the total capacity of the battery. N_p and Q_{cell} are battery parameters that represent the number of cells and the capacity of a single cell, respectively. Time step (Δt) can be assigned as 0.1 second, 1 second, or another value.

The output power produced by the motor (P_{motor}) and the output power produced by the engine (P_{engine}) can be calculated from the output torque and speed of the motor and engine:

$$\begin{gathered} P_{motor} = \tau_m \times \omega_m \\ P_{engine} = \tau_e \times \omega_e \end{gathered} \tag{4.21}$$

The total energy consumption E_{total} ,which is the total output energy, can be calculated as the sum of the motor energy consumption (E_{motor}) and the engine energy consumption (E_{engine}) :

$$\begin{gathered} E_{total} = E_{motor} + E_{engine} \\ E_{motor} = \sum_t P_{motor} \\ E_{engine} = \sum_t P_{engine} \end{gathered} \tag{4.22}$$

In most cases, the input energy is usually compared. However, in this thesis, since the previous strategies have already been proven to be more efficient, the comparison is between previous strategies and a new strategy for the same hardware setup. Thus, the comparison of output energy works in the same way.

The fueling map, which shows the relation between injected fuel and pedal position as well as engine speed, is shown in Figure 4.3.

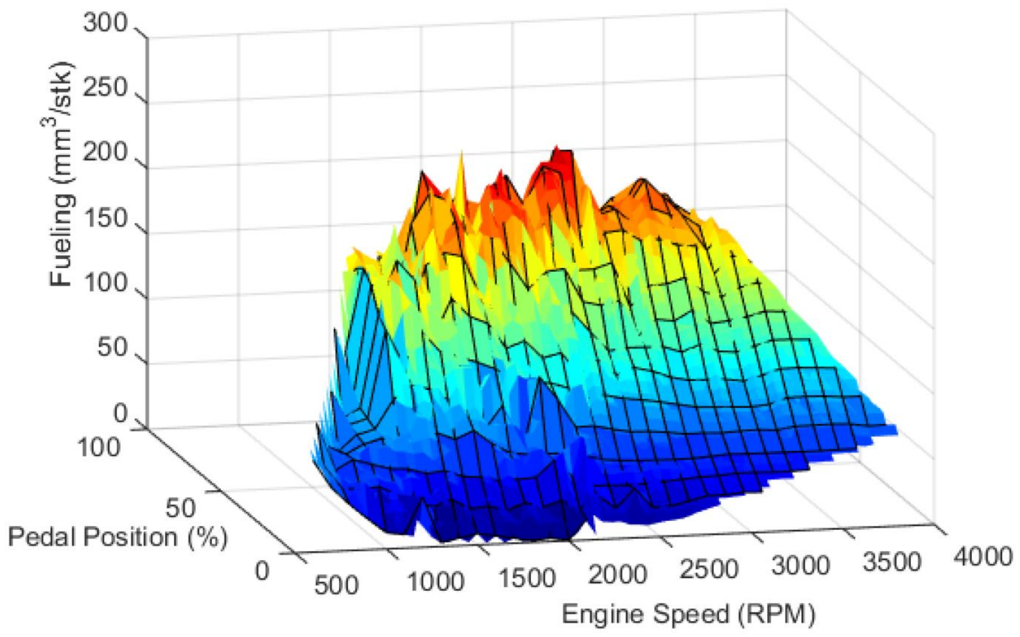


Figure 4.3. Fueling map to predict injected fuel based on engine speed and accelerator pedal position [7]

The fuel consumption can be calculated from the surface fit equation of the fueling map.

4.2 Dynamic Programming

Chapter 2 has explained the basic concept of Dynamic Programming (DP), which is a useful algorithm to solve a complicated optimization problem by breaking it down into a set of easier sub-problems. Normally starting from the final state, DP examines the solutions to the previous sub-problems and generates the optimal solution for the given problem. By thoroughly searching in the solution space, eventually DP will obtain an optimal answer to the problem. This thesis utilizes the open-source DP code available in [33] and compares the DP result with Neural Network result.

Since DP solves the optimization problem backward (from the final state), it also needs the vehicle velocity information at the beginning as a prior. Therefore, DP is not capable of solving the problem in real-time, which is a fatal shortcoming in practical application. However, the model trained by neural networks overcomes this disadvantage and achieves good accuracy, as will be shown in Section 4.6. In this thesis, the performance obtained from the DP algorithm is used as a benchmark to be compared with other approaches.

4.2.1 Implementation of Dynamic Programming

The starting and ending state of charge (SOC) are required for running DP. The reason for this is that the simulation tends to forecast the vehicle always in electric-motor-only mode if there is no constraint on the ending SOC and the cost function only minimizes the net GHG_{WTW} emissions. Because of this, we divide the total GHG_{WTW} emissions into two parts: $GHG_{WTW,electricity}$ and $GHG_{WTW,fuel}$. By doing this, when trying to minimize the GHG emissions from fuel, the net GHG_{WTW} will also be minimized. Hence, we define the cost function as

$$J_v = GHG_{WTW,fuel} \tag{4.23}$$

The split is normalized from -1 to 1 and the max SOC is set to be 0.6, min SOC is 0.4.

The other constraints remain unchanged from the simplified model described in equations 4.13 to 4.16. Besides, the SOC and gear number are given for the starting and ending time steps.

4.2.2 Dynamic Programing Results in UDDS Drive Cycle

In the city UDDS drive cycle, the torque split result is shown in Figure 4.4.

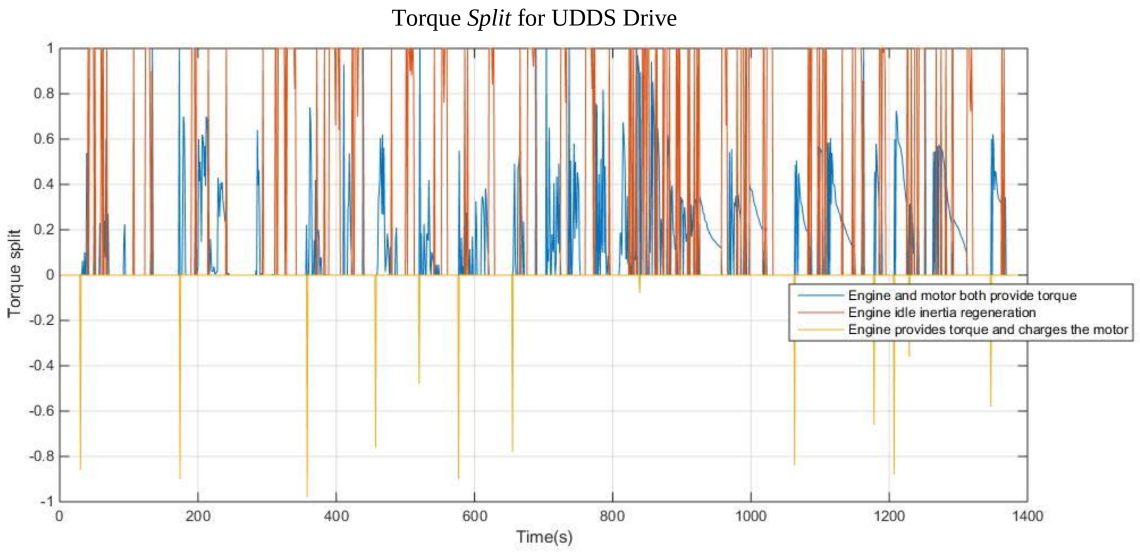


Figure 4.4. Dynamic Programing torque split in UDDS drive cycle

The torque split is a continuous result over time. It is divided into three different colors to distinguish the three situations. The blue line represents the situation when both the engine and motor provide the torque. The orange line represents the situation when the required torque is negative, which charges the battery and the engine is idle. The yellow line represents the case when the engine provides torque to the vehicle and charges the battery at the same time.

The battery state of charge (SOC) is maintained at 55 percent as Figure 4.5 shows.

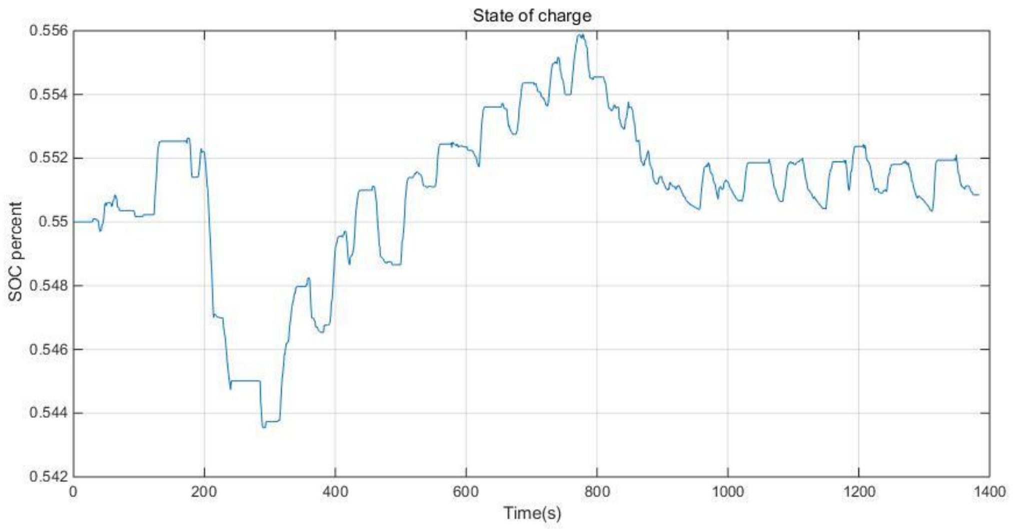


Figure 4.5 Dynamic programming battery SOC for UDDS drive cycle

During the whole drive cycle, the motor energy consumption (E_{motor}), engine energy consumption (E_{engine}) and total energy consumption (E_{total}) can be calculated from formula (4.21) . The results are shown in Table 4.3.

Table 4.3. Dynamic programming energy consumption results for UDDS drive cycle

Motor Energy Consumption (E_{motor})	2.0490×10^6 (J)
Engine Energy Consumption (E_{engine})	7.5592×10^6 (J)
Total Energy Consumption (E_{total})	9.6082×10^6 (J)

The required power at each time step is shown in Figure 4.6.

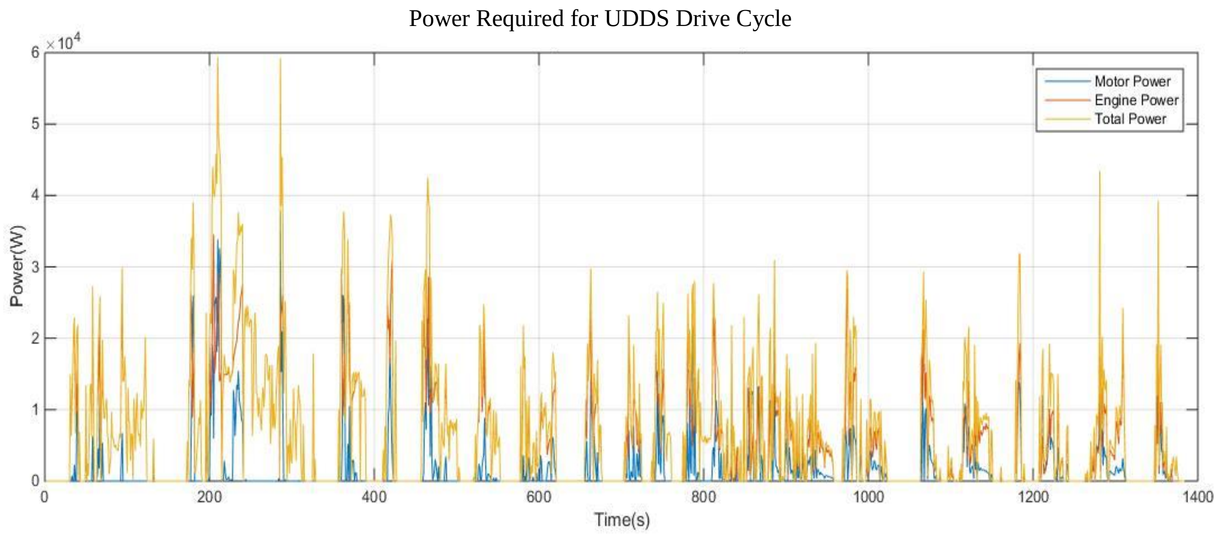


Figure 4.6. Dynamic programming required power for UDDS drive cycle

4.2.3 Dynamic Programing Results in Highway Drive Cycle

In the Highway FET (HWFET) drive cycle, the torque split result is shown in Figure 4.7.

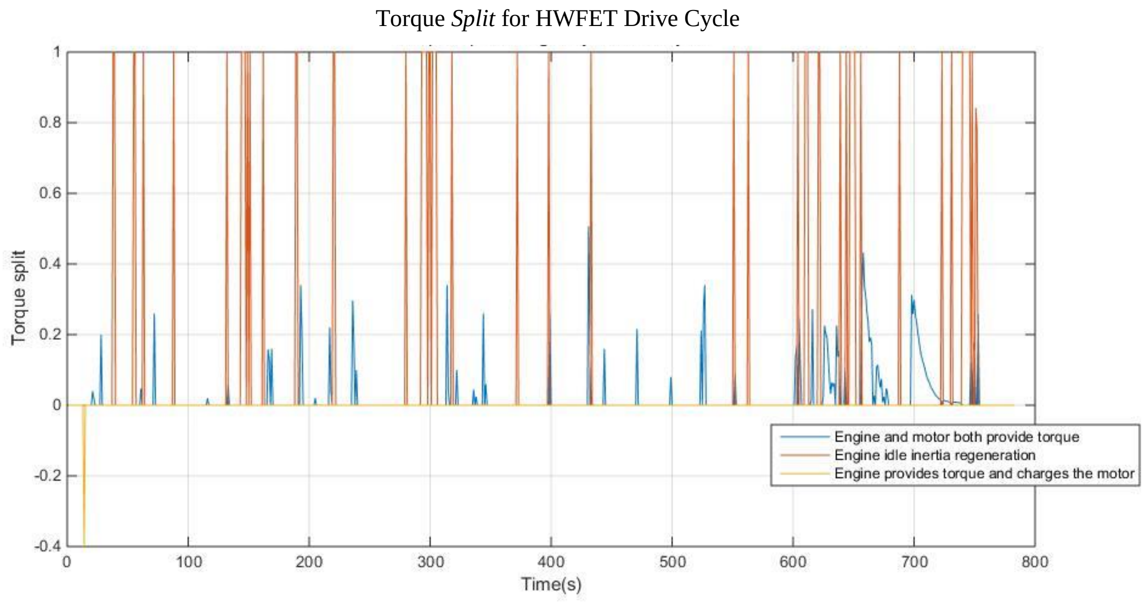


Figure 4.7. Dynamic programing torque split in HWFET drive cycle

The battery state of charge (SOC) is maintained at 55 percent, as Figure 4.8 shows.

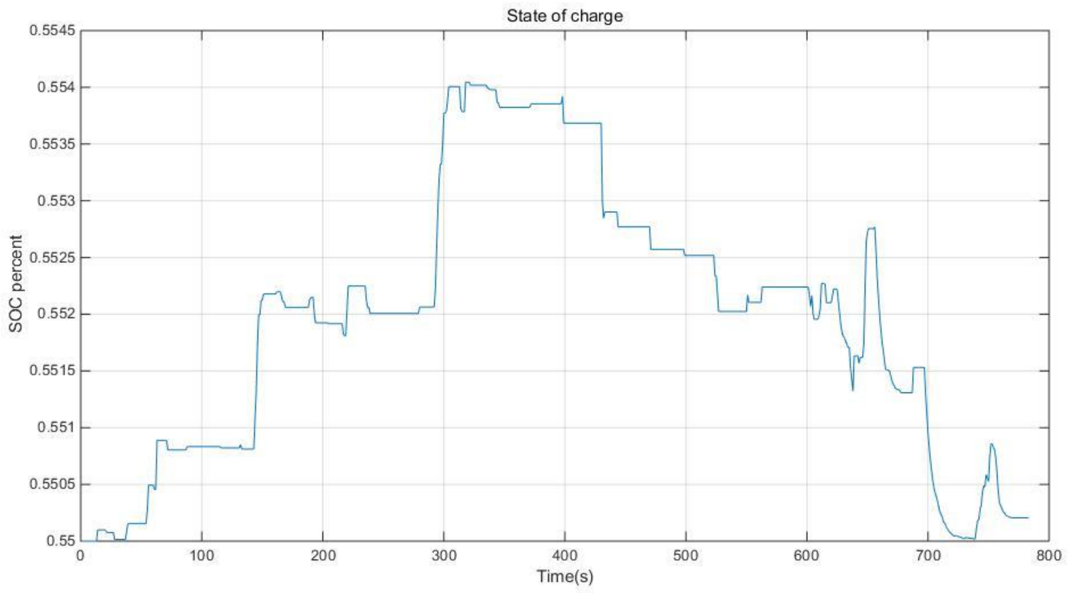


Figure 4.8. Dynamic programming battery SOC for HWFET drive cycle

During the whole drive cycle, the motor energy consumption, engine energy consumption and total energy consumption are shown in Table 4.4.

Table 4.4 Dynamic programming energy consumption results for HWFET drive cycle

Motor Energy Consumption (E_{motor})	4.9428×10^5 (J)
Engine Energy Consumption (E_{engine})	1.1636×10^7 (J)
Total Energy Consumption (E_{total})	1.2130×10^7 (J)

The required power at each time step is shown in Figure 4.9.

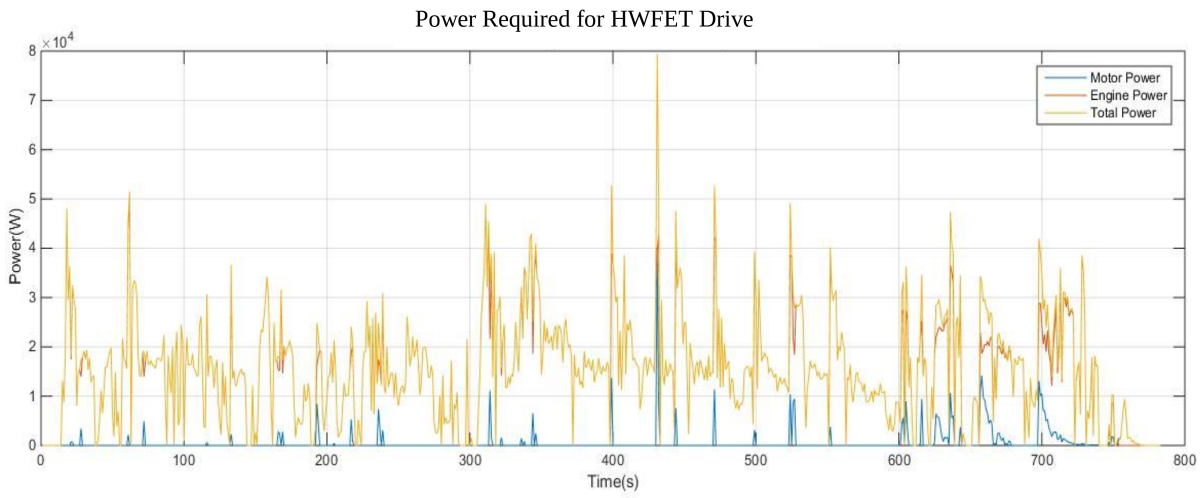


Figure 4.9. Dynamic programming required power for HWFET Drive Cycle

4.3 Equivalent consumption minimization strategy (ECMS)

ECMS is known as a real-time fuel consumption optimization control. It is often used to find the minimum power split profile among different energy sources for a given driving cycle. This concept of the strategy is based on charge-sustaining vehicles. When vehicles are in charge-sustaining mode, the system operation becomes as efficient as possible. Therefore, the initial and final SOC difference is almost negligible. This strategy calculates the optimality locally and instantaneously instead of solving the global minimization problem. The ECMS fuel cost function can be used to calculate the equivalent fuel cost, which is formulated as

$$\arg\min J_t = \tau_e^{opt}, \tau_m^{opt} \tag{4.24}$$

where τ_e^{opt} is the engine torque, τ_m^{opt} is the motor torque, and J_t is the cost function.

The global fuel optimality of a vehicle can be obtained from the instantaneous minimization of the cost function, which only depends on the global system variables at the current time, such as the engine torque and motor torque. The cost function considers multiple factors, such as CO, CO_2, and NO_x emissions and battery state of charge.

4.3.1 Implementation of ECMS

The cost function requires the prior knowledge of the equivalence factor, ζ. It is defined as the following:

$$J_t = GHG_{WTW,fuel} + \zeta * pen * GHG_{WTW,electricity} \tag{4.25}$$

where ζ is the equivalence factor, and pen is the penalty function.

$$GHG_{WTW,fuel} = f(\tau_e, \omega_e) \tag{4.26}$$

$$GHG_{WTW,electricity} = f(\tau_m, \omega_m) \tag{4.27}$$

The value of the equivalence factor strongly corresponds to the fuel consumption. It is defined as a dimensionless ratio of electrical power flow and chemical power flow. If the value of the equivalence factor is higher, it represents that the cost of using electrical energy is expensive, and therefore the controller reduces electrical usage from the battery. Conversely, if the value of the equivalence factor is lower, it implies that the cost of using fuel energy is more expensive than using electrical energy, and therefore the controller encourages battery use. The optimal value of the equivalence factor can be found only if the driving cycle is known.

Since SOC stays almost constant through the whole process, and the energy can only come from the two main fuel sources, engine and battery, the used electricity during battery discharge will need to be replenished by charging the battery using the fuel from the engine. Therefore, two different modes can be operated in vehicles:

(1) In the charging mode, the battery power becomes negative, so the stored electrical energy will recharge the battery to reduce the engine load, which means a fuel saving in vehicles.

(2) In the discharging modes, the battery power becomes positive, so the electricity stored in a battery pack is not enough to power the engine and turn the wheels, which implies higher fuel consumption.

In both cases, an equivalence factor, ζ, can be tuned for both charging and discharging modes. Under different driving conditions, an appropriate equivalence factor must be obtained.

The penalty function, pen, is defined as the following:

$$pen = \begin{cases} \left(1 + \left(\dfrac{SOC_{ref} - SOC(t)}{SOC_{ref} - SOC_{min}}\right)^{2n_{p1}+1}\right) & if\ SOC(\mathrm{t}) < SOC_{ref} \\ \left(1 - \left(\dfrac{SOC(t) - SOC_{ref}}{SOC_{max} - SOC_{ref}}\right)^{2n_{p2}+1}\right) & if\ SOC(\mathrm{t}) \geq SOC_{ref} \end{cases} \tag{4.28}$$

where

$SOC(t) =$ The instantaneous value of state of charge

$SOC_{ref} =$ The desired nominal value

$SOC_{min}, SOC_{max} =$ The minimum and maximum admissible values

$n_{p1}, n_{p2} =$ integer numbers

This function is also represented in Figure 4.10.

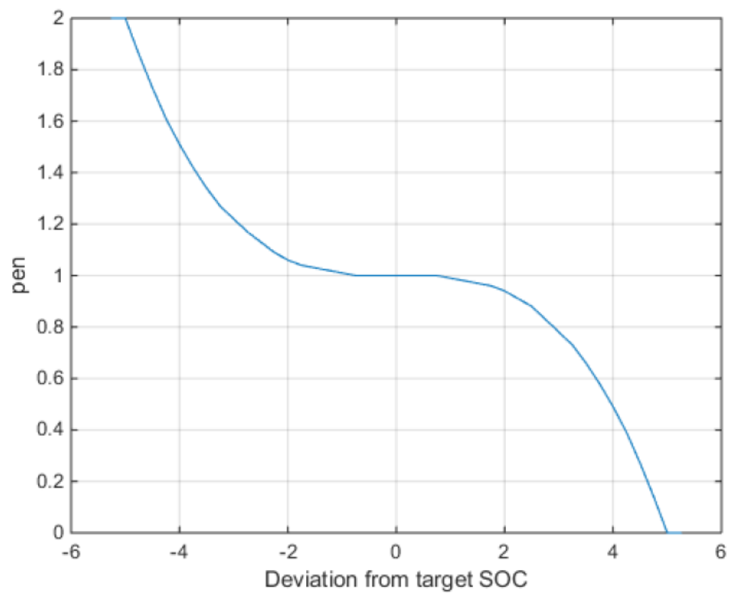


Figure 4.10. The plot of the penalty function vs. deviations from target SOC

In order to find the local minimization power split, the following steps can be used:

(1) Obtain the maximum and minimum motor torques at the current speed

(2) Find the possible motor torques and the corresponding engine torques that should satisfy the following condition: $\tau_{ev} = \tau_v - \tau_{mv}$

(3) Once we find all the combination of motor torques and engine toques, the cost functions can be obtained

(4) Find one combination from before with the minimum cost that must satisfy all the constraints.

4.3.2 ECMS Results in UDDS Drive Cycle

In the city UDDS drive cycle, the torque split result is shown in Figure 4.11.

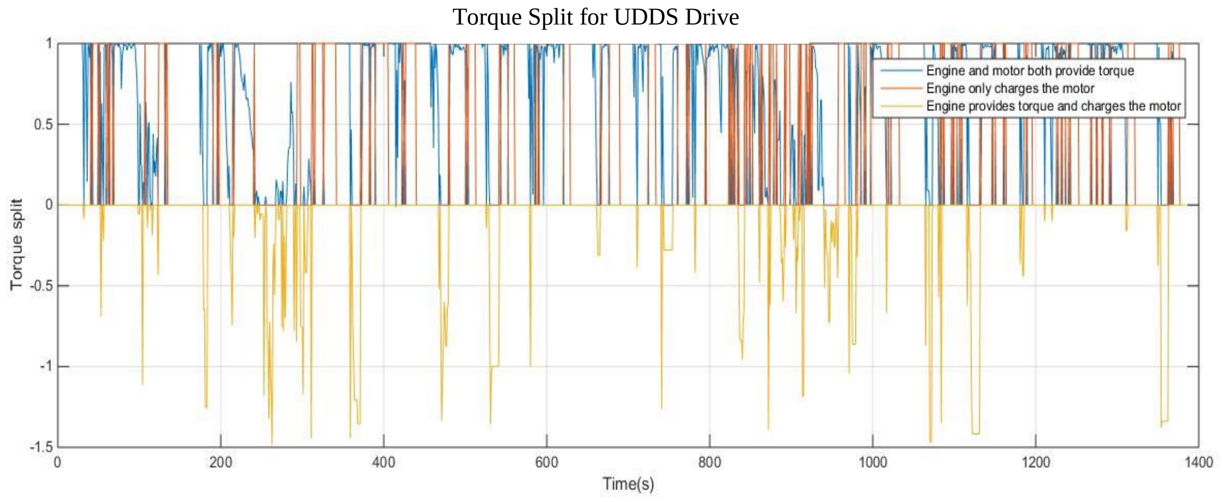


Figure 4.11. ECMS torque split in UDDS drive cycle

The torque split is a continuous result over time. It is divided into three different colors to distinguish the three situations. The blue line represents the situation when both the engine and motor provide the torque. The orange line represents the situation when the required torque is negative, which charges the battery and the engine is idle. The yellow line represents the case when the engine provides torque to the vehicle and charges the battery at the same time.

The battery state of charge (SOC) is maintained at 55 percent, as Figure 4.12 shows.

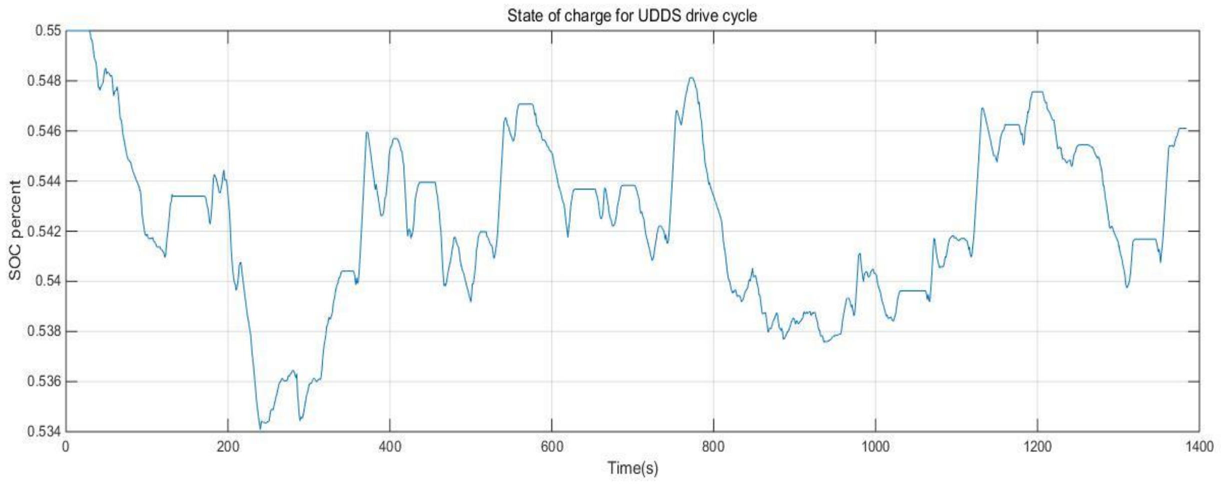


Figure 4.12. ECMS battery SOC for UDDS drive cycle

During the whole drive cycle, the motor energy consumption (E_{motor}), engine energy consumption (E_{engine}) and total energy consumption (E_{total}) can be calculated from formula (4.21) . The results are shown in Table 4.5.

Table 4.5 ECMS energy consumption results for UDDS drive cycle

Motor Energy Consumption (E_{motor})	5.3856×10^6 (J)
Engine Energy Consumption (E_{engine})	7.8520×10^6 (J)
Total Energy Consumption (E_{total})	1.3238×10^7 (J)

The required power at each time step is shown in Figure 4.13,

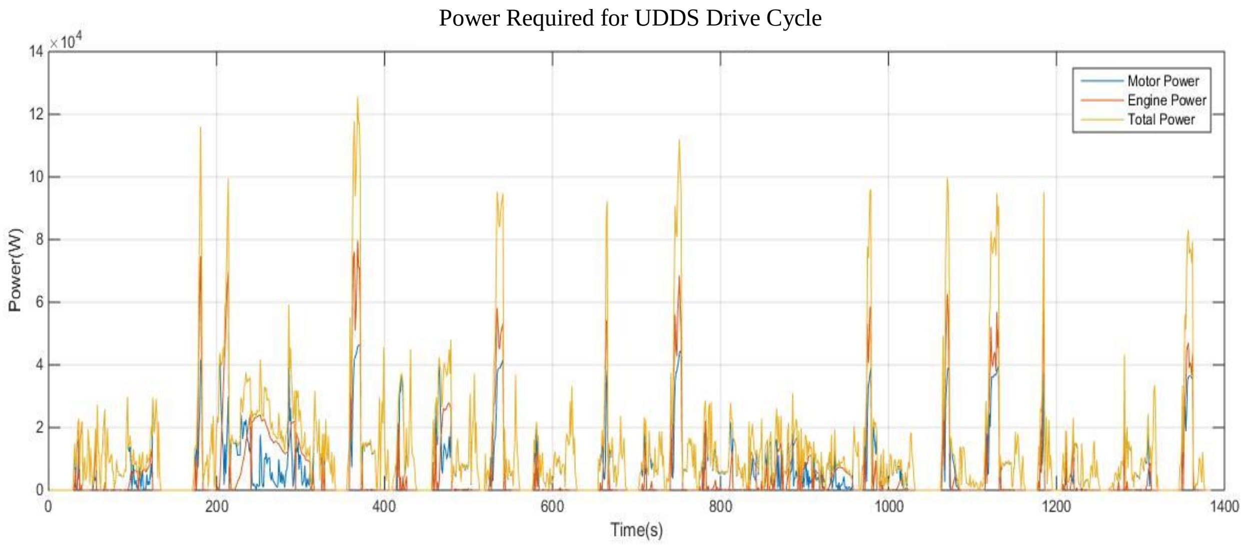


Figure 4.13. ECMS required power for UDDS drive cycle

4.3.3 ECMS Results in Highway Drive Cycle

In the Highway FET (HWFET) drive cycle, the torque split result is shown in Figure 4.14.

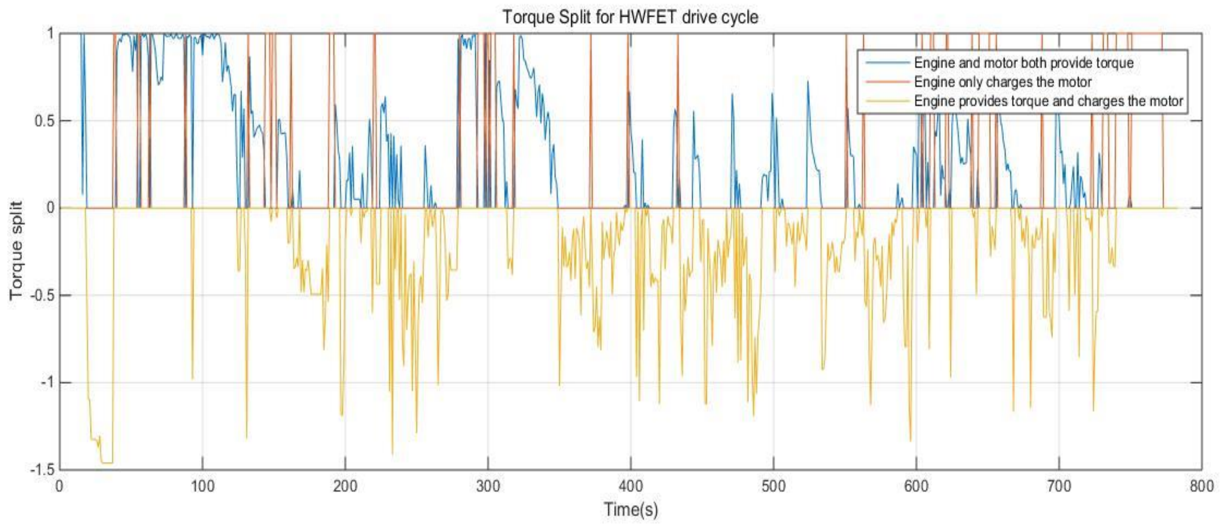


Figure 4.14. ECMS torque split in HWFET drive cycle

The battery state of charge (SOC) is maintained at 55 percent, as Figure 4.15 shows.

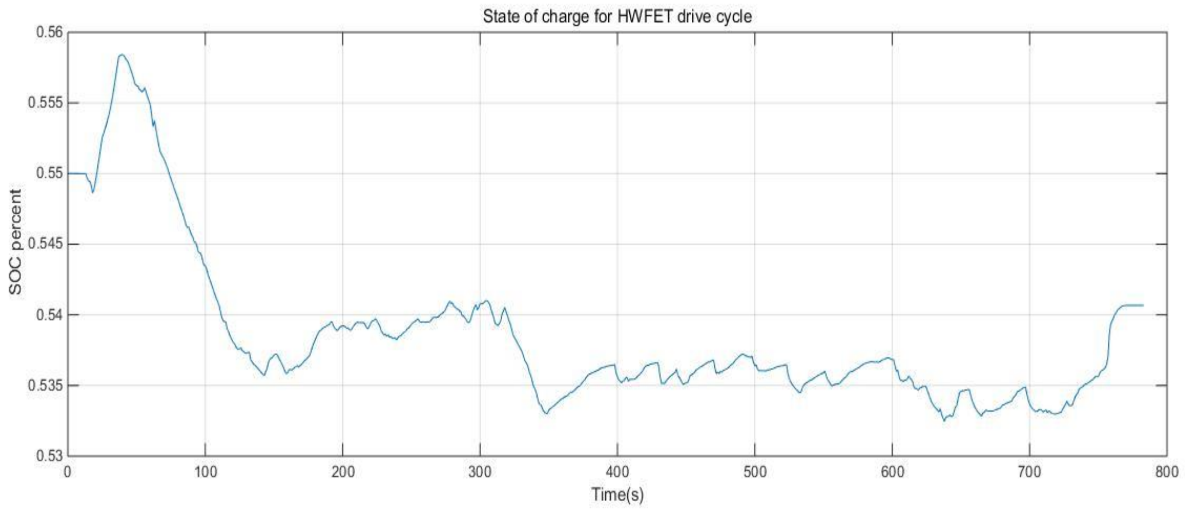


Figure 4.15. ECMS battery SOC for HWFET drive cycle

During the whole drive cycle, the motor energy consumption, engine energy consumption and total energy consumption are shown in Table 4.6.

Table 4.6. ECMS energy consumption results for HWFET drive cycle

Motor Energy Consumption (E_{motor})	3.5721×10^6 (J)
Engine Energy Consumption (E_{engine})	1.1280×10^7 (J)
Total Energy Consumption (E_{total})	1.4852×10^7 (J)

The required power at each time step is shown in Figure 4.16.

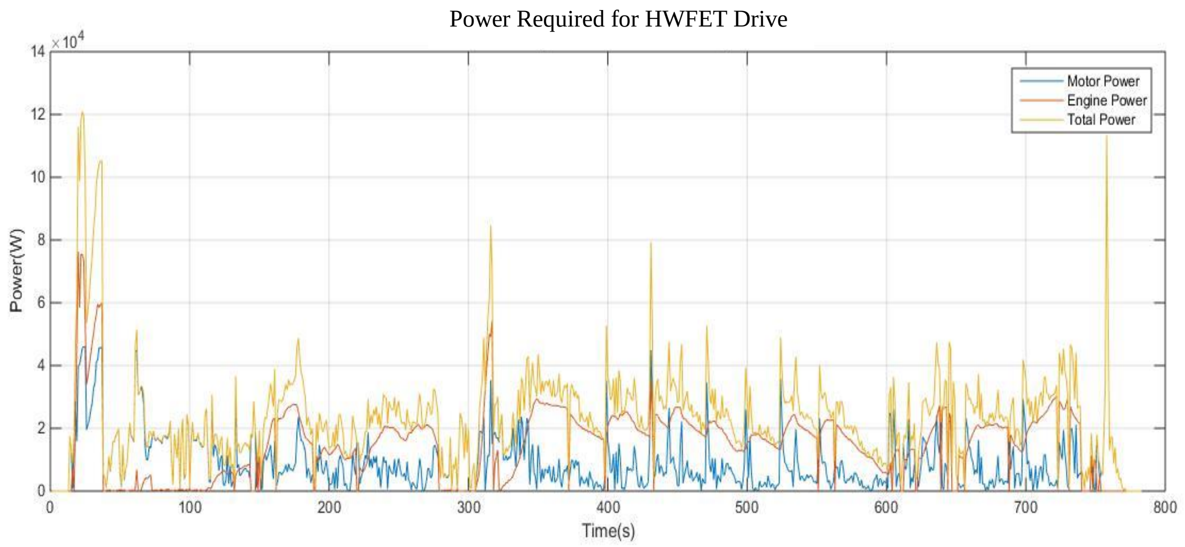


Figure 4.16. ECMS required power for HWFET drive cycle

4.4 Proportional State-of-charge (pSOC) Algorithm

The pSOC algorithm was developed by a previous student from Purdue University, Rohinish Gupta [7]. It is another technique to improve fuel economy. This method works based on the difference between the current SOC and the target SOC. The optimum torque split between the engine and the electric motor can be calculated as the following:

$$Split = K * (SOC_{present} - SOC_{target}) \tag{4.29}$$

where

$$K = \frac{1}{100 * (SOC_{UB} - SOC_{LB})/2}$$

$SOC_{present}$ = The present state of charge

SOC_{target} = The target state of charge

SOC_{UB} = The upper bound of state of charge

SOC_{LB} = The lower bound of state of charge

Split has the highest value of 1, which is associated with SOC_{UB}; it has the lowest value of -1, which is associated with SOC_{LB}. The procedure of the pSOC algorithm is comparable to that of feedback control strategy, which also describes the relationship between the current SOC and the target SOC. When the current SOC value is higher than the target SOC value, the electric motor will be used to power the vehicle. On the other hand, when the current SOC is lower than the target SOC value, the engine will start to work to charge the battery in the vehicle. Similarly, in the case of pSOC, when the current SOC value is higher than the SOC_{UB} value, the vehicle will be driven entirely on electricity. Conversely, when the current SOC value is lower than the SOC_{LB} value, the engine will be enabled.

4.4.1 Implementation of pSOC algorithm

If the initial battery state of charge is set to be 20% for UDDS drive cycle, the results of SOC profile by using proportional state-of-charge algorithm are presented in Figure 4.17.

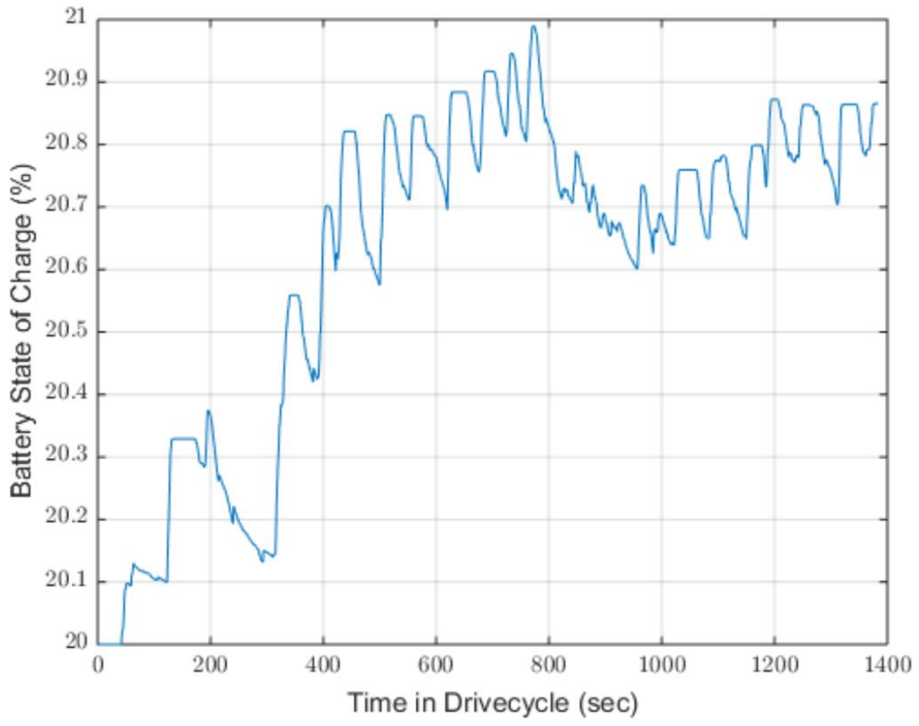


Figure 4.17. The results of battery state of charge by using pSOC algorithm

As shown in the figure, the overall battery state of charge is growing with increasing time during the drive cycle. The final state of charge is about 4.5 percent higher than the initial state of charge when using the proportional state-of-charge algorithm strategy. Figure 4.18 shows the state of charge profile over 10 concatenated drive cycles.

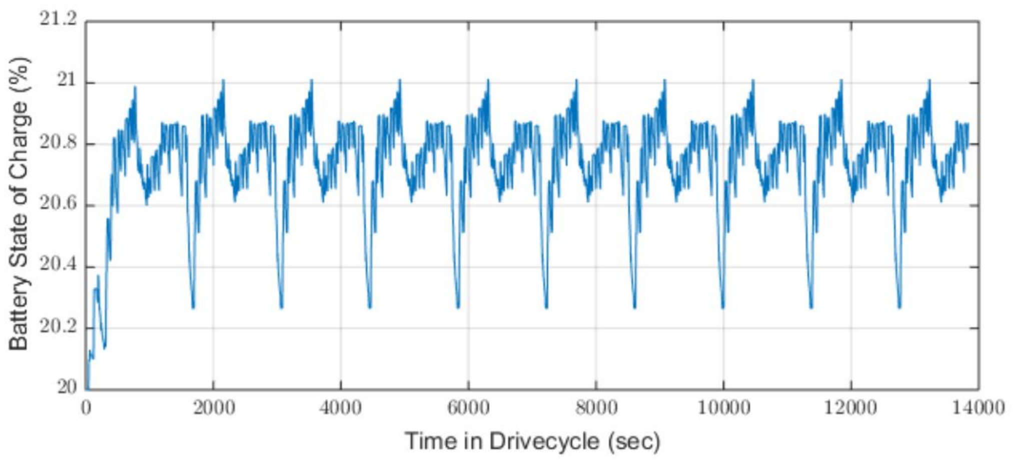


Figure 4.18. The results of battery state of charge for driving schedule drive cycle test with 10 times concatenated by using pSOC algorithm

Similar to the algorithms mentioned earlier, this algorithm can also be used to simulated on a drive-trace with 10 different city drive cycles concatenated. The results are shown in Figure 4.19.

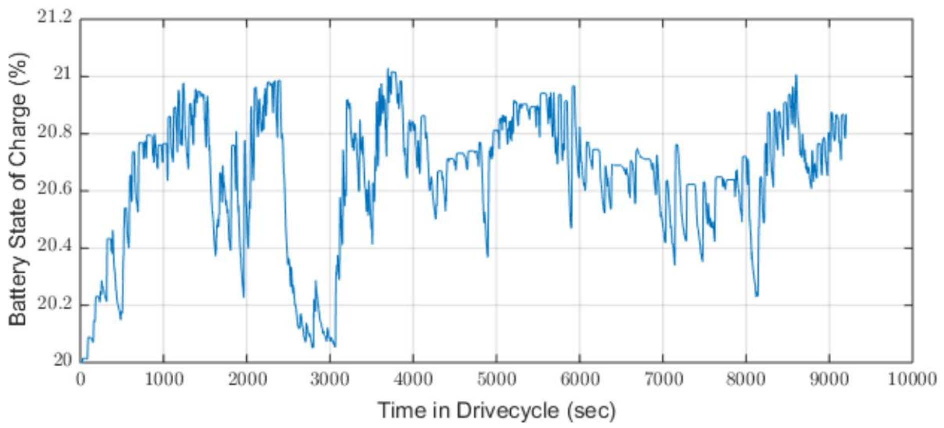


Figure 4.19. The results of battery state of charge for driving schedule drive cycle test with 10 different city drive cycles concatenated using pSOC algorithm

4.4.2 pSOC Results in UDDS Drive Cycle

In the city UDDS drive cycle, the torque split result is shown in Figure 4.20.

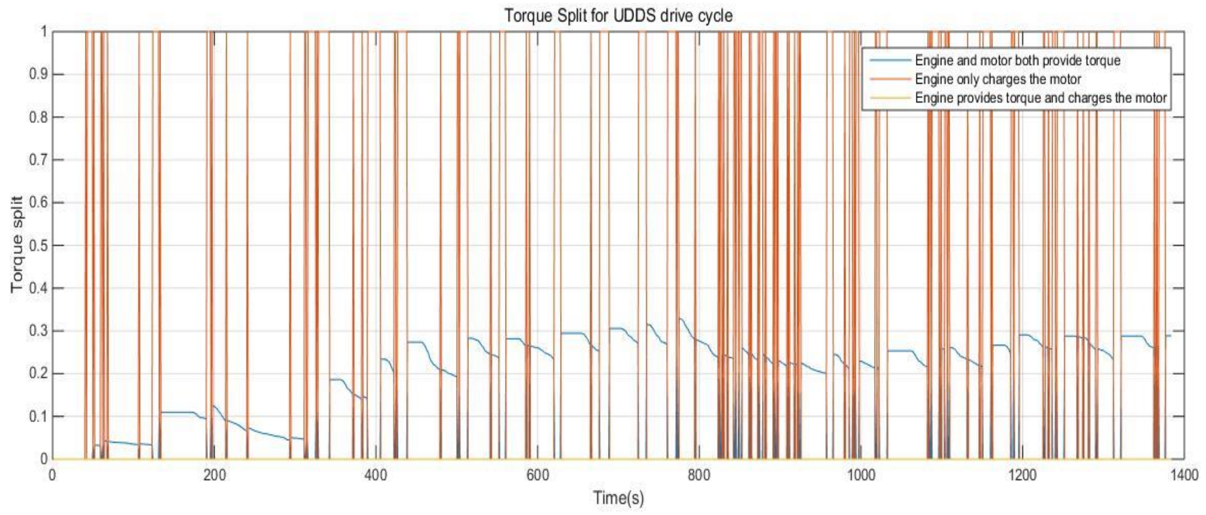


Figure 4.20. pSOC torque split in UDDS drive cycle

The torque split is a continuous result over time. It is divided into three different colors to distinguish the three situations. The blue line represents the situation when both the engine and motor provide the torque. The orange line represents the situation when the required torque is negative, which charges the battery and the engine is idle. The yellow line represents the case when the engine provides torque to the vehicle and charges the battery at the same time.

The battery state of charge (SOC) is maintained at 55 percent, as Figure 4.21 shows.

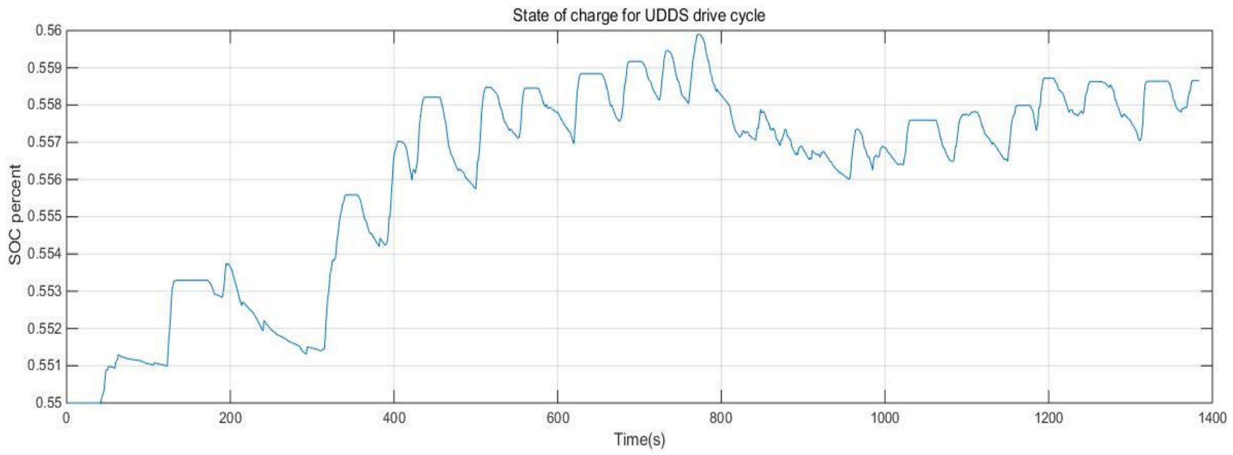


Figure 4.21. pSOC battery SOC for UDDS drive cycle

During the whole drive cycle, the motor energy consumption (E_{motor}) , engine energy consumption (E_{engine}) and total energy consumption (E_{total}) can be calculated from formula (4.21) , the results are shown in Table 4.7.

Table 4.7. pSOC energy consumption results for UDDS drive cycle

Motor Energy Consumption (E_{motor})	1.8019×10^6 (J)
Engine Energy Consumption (E_{engine})	7.9455×10^6 (J)
Total Energy Consumption (E_{total})	9.7474×10^6 (J)

The required power at each time step is shown in Figure 4.22.

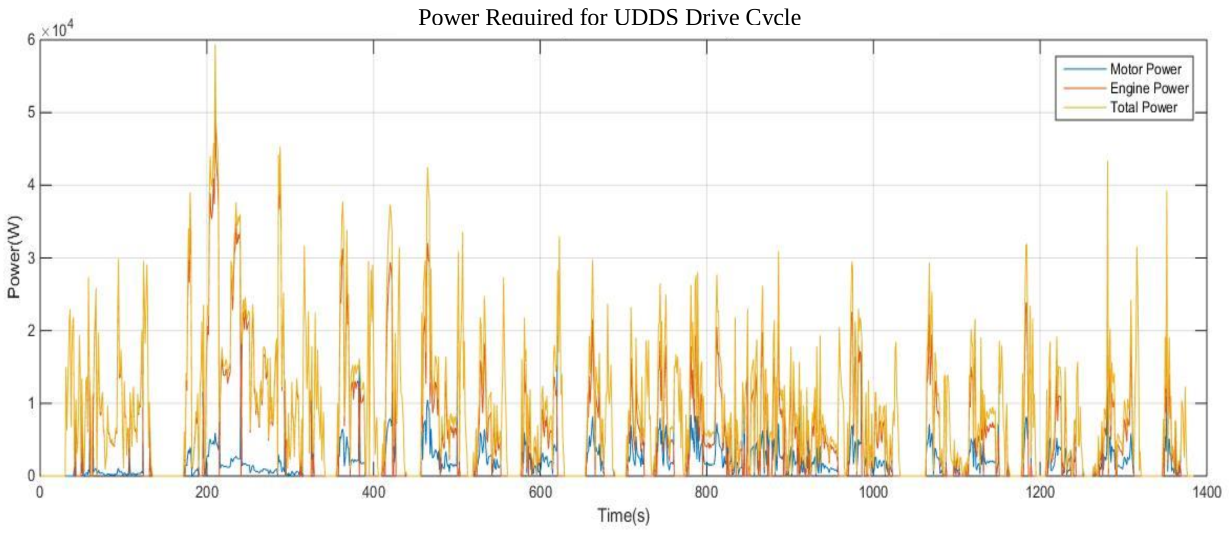


Figure 4.22. pSOC required power for UDDS drive cycle

4.4.3 pSOC Results in Highway Drive Cycle

In the Highway FET (HWFET) drive cycle, the torque Split result is shown in Figure 4.23.

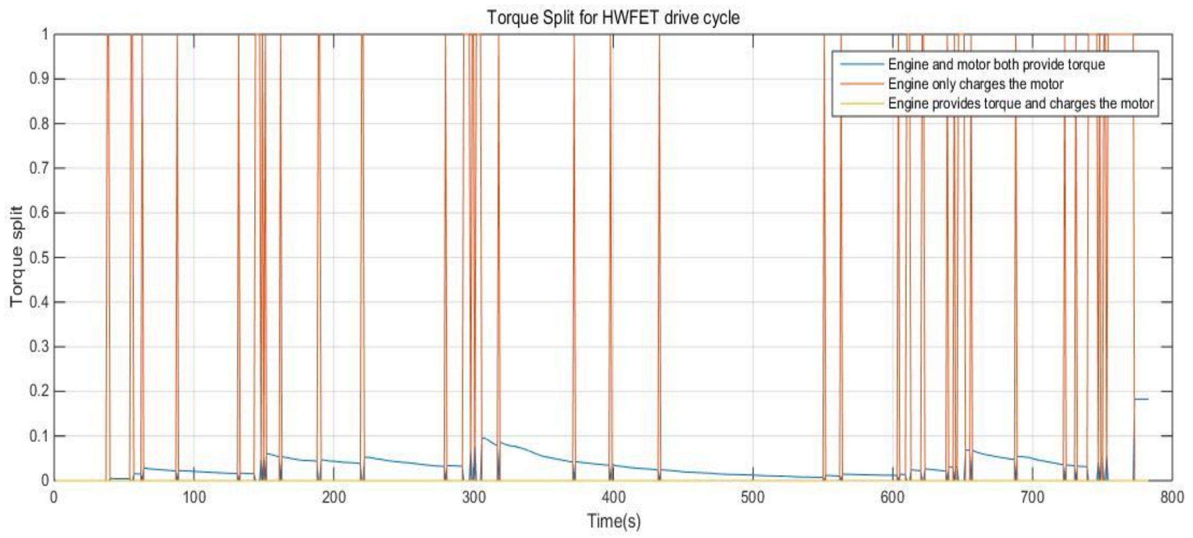


Figure 4.23. pSOC torque split in HWFET drive cycle

The battery state of charge (SOC) is maintained at 55 percent. as Figure 4.24 shows.

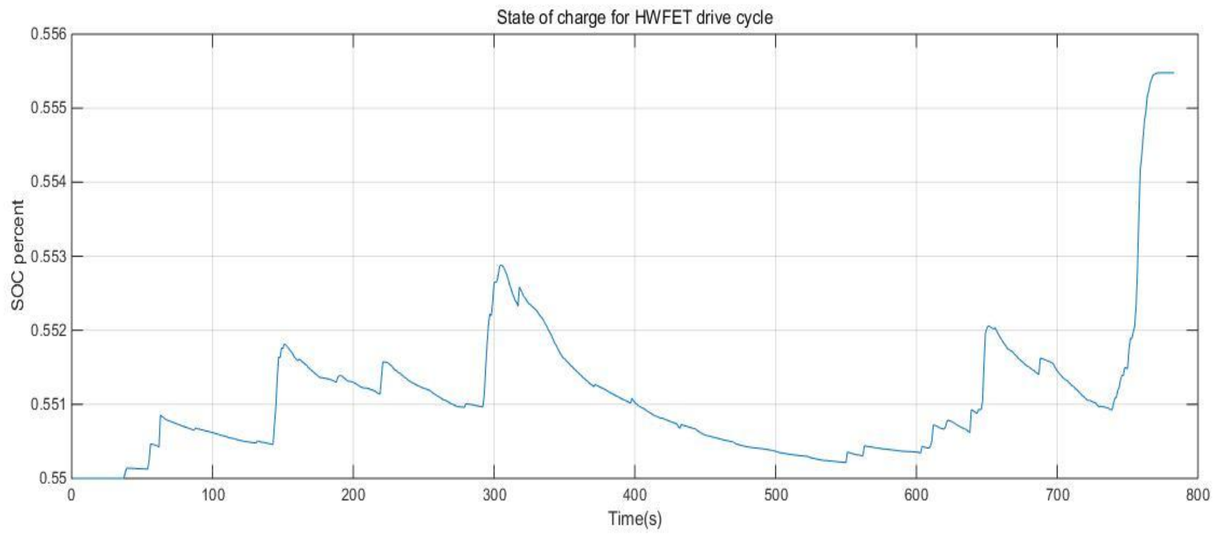


Figure 4.24. pSOC battery SOC for HWFET drive cycle

During the whole drive cycle, the motor energy consumption, engine energy consumption and total energy consumption are shown in Table 4.8.

Table 4.8. pSOC energy consumption results for HWFET drive cycle

Motor Energy Consumption (E_{motor})	4.0145×10^5 (J)
Engine Energy Consumption (E_{engine})	1.1703×10^7 (J)
Total Energy Consumption (E_{total})	1.2104×10^7 (J)

The required power at each time step is shown in Figure 4.25.

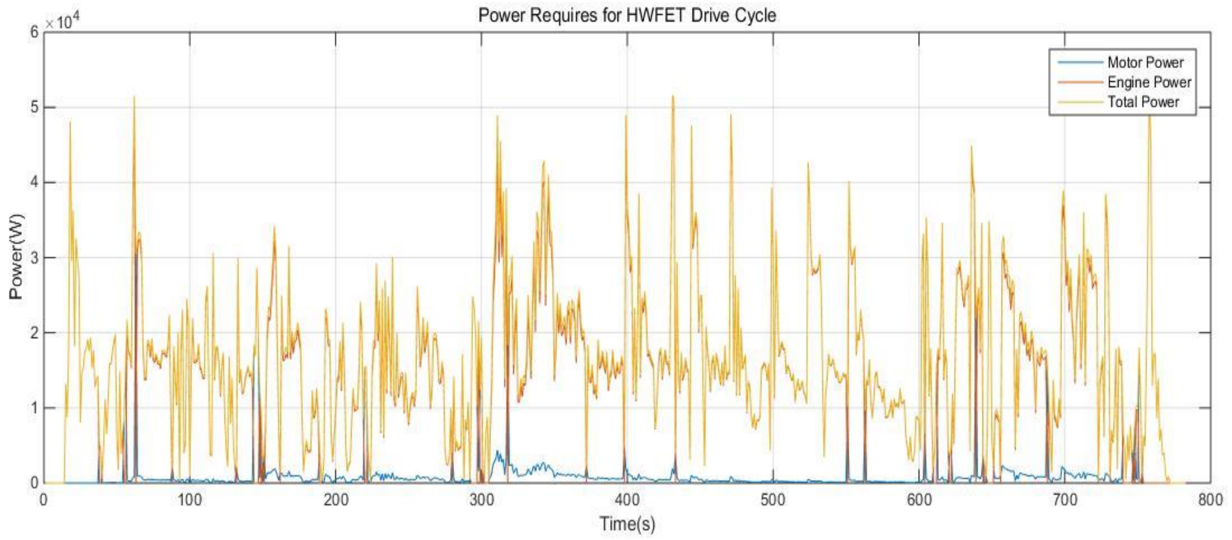


Figure 4.25. pSOC required power for HWFET drive cycle

4.5 Regression Modeling

The Regression Model was developed by a previous student from Purdue University, Rohinish Gupta [7]. The DPM was used as the benchmark to develop the Regression Model. The Regression Model presents another way to represent DPM as a mapping between selected input signals and the output, which is the torque split. By looking at the strongest relationships between the signals and the output, the inputs were selected and signals that correlate strongly with other input signals were eliminated.

By applying the regression technique, the fuel consumption for a hybrid vehicle can be estimated. This control strategy is often used when the initial state of charge is different from the final state of charge. Regression modeling can be used to estimate the fuel consumption among different variables simply by doing interpolation or extrapolation. Compared to other control strategies, the benefit of using regression modeling is that it is more computationally efficient. However, the result of this control strategy is not as optimal as the others. At the same time, the regression modeling is applicable for real-time implementation.

When there is a large amount of dynamic programming data generated, the optimal results can be found for a large number of drive cycles. The analysis is only done for the vehicles that are in charge-sustaining mode and urban driving.

The regression strategy replicates the dynamic programming strategy. When the dynamic programming data points are observed, a regression technique can be used to generate the best trend curves from the bulk of data, where the difference between the measured and estimated values is minimum.

4.5.1 Implementation of Regression Modeling

Since regression analysis is a well-known mathematical way to estimate the relationships among different variables, the observable physical variables of hybrid vehicles in the controller are defined as shown in Table 4.9.

Table 4.9. Different observable physical variables in the controller

Name of the variable	**The symbol of the variable**
Torque required by the vehicles	τ_v
Engine torque	τ_e
Engine speed	ω_e
Motor torque	τ_m
Motor speed	ω_m
Vehicle speed	v
Vehicle acceleration	a
Battery State-of-charge	SOC
Gear Number	N_g

The degree of relationship among different observable physical variables is measured through the correlation analysis. The measured correlation coefficient of the linear regression is in the range of -1 and 1. A positive correlation coefficient value indicates that the two variables change in the same direction. On the other hand, a negative correlation coefficient value indicates that the two variables change in the opposite direction. Six independent variables are chosen out of these 9 variables, which are $\tau_v, \tau_e, \omega_e, \tau_m, v$ and SOC.

The correlation among those different variables is shown in Table 4.10.

Table 4.10. The relationship between different observable physical variables in the controller

	τ_v	τ_e	ω_e	τ_m	ω_m	v	a	SOC	N_g
τ_v	--	0.34	0.37	0.26	0.13	0.13	0.99	-0.01	0.07
τ_e	0.34	--	0.67	0.20	0.73	0.73	0.20	-0.10	0.66
ω_e	0.37	0.67	--	0.27	0.74	0.74	0.39	-0.08	0.72
τ_m	0.26	0.20	0.27	--	0.25	0.25	0.24	-0.02	0.27
ω_m	0.13	0.73	0.74	0.25	--	1	0.02	-0.17	0.96
v	0.13	0.73	0.74	0.25	1	--	0.02	-0.17	0.96
a	0.99	0.20	0.39	0.24	0.02	0.02	--	0.01	-0.02
SOC	-0.01	-0.10	-0.08	-0.02	-0.17	-0.17	0.01	--	-0.17
N_g	0.07	0.66	0.72	0.27	0.96	0.96	-0.02	-0.17	--

When fitting the regression models, two possible strategies can be chosen:

1. Linear regression model
2. Nonlinear regression model

Linear regression model is commonly used to fit data, since it is convenient to use for interpolation and extrapolation. The nonlinear regression models have added flexibility when the data are nonlinear. Different forms of nonlinear models are shown in Table 4.11.

Table 4.11. Different types of regression models

Name of different term	**The symbol of the term**
Linear	X_i
Quadratic	X_i^2
Cross-quadratic	$X_i * X_j$
Exponential terms	$e^{X_i}\ and\ \frac{1}{1+e^{-X_i}}$
Square root terms	$\sqrt{\lvert X_i \rvert}$

The linear regression correlation coefficients among different variables are shown in Table 4.12. The variables and equations have been defined earlier.

Table 4.12. Linear regression model with different terms

Linear regression terms	**Correlation Coefficient**
ω_e	-4007.67
SOC	68.97
τ_v	480.74
v	-1032.45
τ_e	-0.3854
τ_m	-443.66

The quadratic regression correlation coefficients among different variables are shown in Table 4.13. The variables and equations have been defined earlier.

Table 4.13. Linear regression model with different terms

Quadratic regression terms	**Correlation Coefficient**
$SOC * SOC$	31.68
$\tau_v * \tau_v$	32.67
$v * v$	-37.66
$\tau_e * \tau_e$	-0.7911
$\tau_m * \tau_m$	-68.38

The cross-quadratic regression correlation coefficients among different variables are shown in Table 4.14. The variables and equations have been defined earlier.

Table 4.14. cross-quadratic regression model with different terms

Cross-quadratic regression terms	**Correlation Coefficient**
$\omega_e * SOC$	-2.80
$\omega_e * \tau_v$	-0.489
$\omega_e * \tau_m$	5.63
$SOC * \tau_v$	3.60
$SOC * v$	0.3273
$SOC * \tau_m$	0.8816
$\tau_v * v$	2.10
$\tau_v * \tau_e$	1.36
$\tau_v * \tau_m$	-1.93
$v * \tau_e$	1.25
$v * \tau_m$	1.84
$\tau_e * \tau_m$	-1.77
$\omega_e * \omega_m$	100.59

The exponential regression correlation coefficients among different variables are shown in Table 4.15. The variables and equations have been defined earlier.

Table 4.15. Exponential regression model with different terms

Exponential regression terms (1)	**Correlation Coefficient**
e^{ω_e}	428.42
e^{SOC}	-68.62
e^{τ_v}	-92.33
e^{v}	163.71
e^{τ_m}	132.59

The exponential regression correlation coefficients among different variables are shown in Table 4.16. The variables and equations have been defined earlier.

Table 4.16 exponential regression model with different terms

Exponential regression terms (2)	**Correlation Coefficient**
$\frac{1}{1+e^{-\omega_e}}$	12936.26
$\frac{1}{1+e^{-\tau_e}}$	-1532.453
$\frac{1}{1+e^{-v}}$	3376.72
$\frac{1}{1+e^{-\tau_m}}$	1241.87

The square root correlation coefficients among different variables are shown in Table 4.17. The variables and equations have been defined earlier.

Table 4.17. Square root regression model with different terms

Square root terms	**Correlation Coefficient**
$\sqrt{\omega_e}$	226.83
$\sqrt{\tau_v}$	-1.69
$\sqrt{v}$	9.6
$\sqrt{\tau_e}$	-0.7679
$\sqrt{\tau_m}$	0.5784

4.5.2 Regression Model Results in UDDS Drive Cycle

In the city UDDS drive cycle, the torque split result is shown in Figure 4.26.

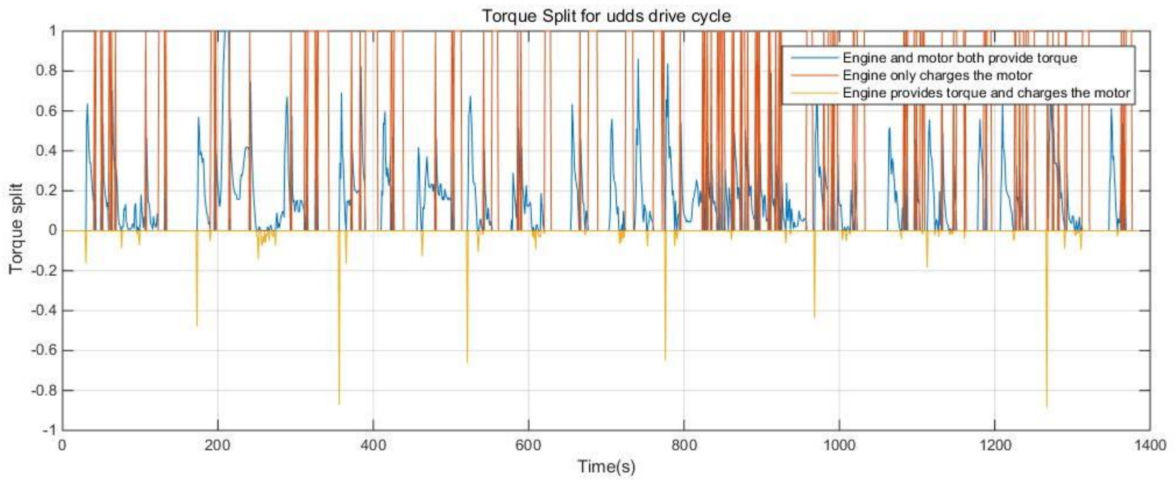


Figure 4.26. Regression model torque split in UDDS drive cycle

The torque split is a continuous result over time step. It is divided into three different colors to distinguish the three situations. The blue line represents the situation when both the engine and motor provide the torque. The orange line represents the situation when the required torque is negative, which charges the battery and the engine is idle. The yellow line represents the case when the engine provides torque to the vehicle and charges the battery at the same time.

The battery state of charge (SOC) is maintained at 55 percent, as Figure 4.27 shows.

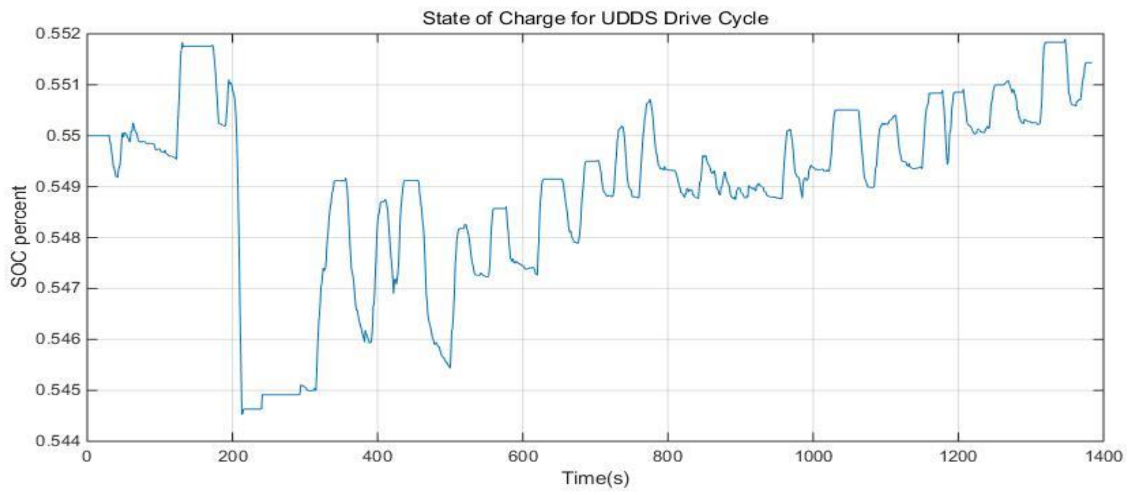


Figure 4.27. Regression model battery SOC for UDDS drive cycle

During the whole drive cycle, the motor energy consumption (E_{motor}), engine energy consumption (E_{engine}) and total energy consumption (E_{total}) can be calculated from formula (4.21) . The results are shown in Table 4.18.

Table 4.18. Regression model energy consumption results for UDDS drive cycle

Motor Energy Consumption (E_{motor})	2.2770×10^6 (J)
Engine Energy Consumption (E_{engine})	7.4058×10^6 (J)
Total Energy Consumption (E_{total})	9.6828×10^6 (J)

The required power at each time step is shown in Figure 4.28.

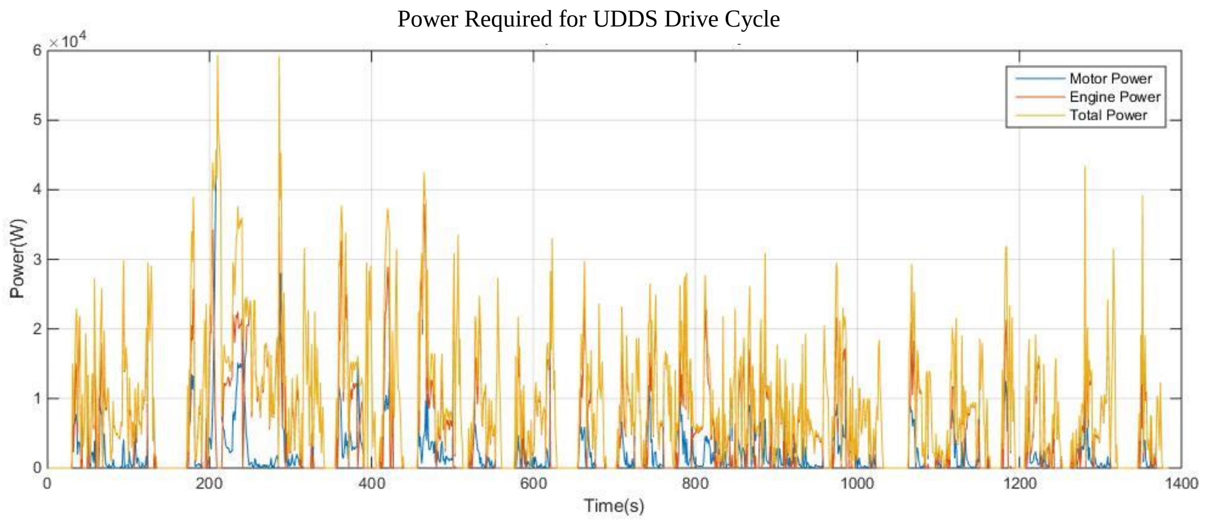


Figure 4.28. Regression model required power for UDDS drive cycle

4.5.3 Regression Model Results in Highway FET Drive Cycle

In the Highway FET (HWFET) drive cycle, the torque split result is shown in Figure 4.29.

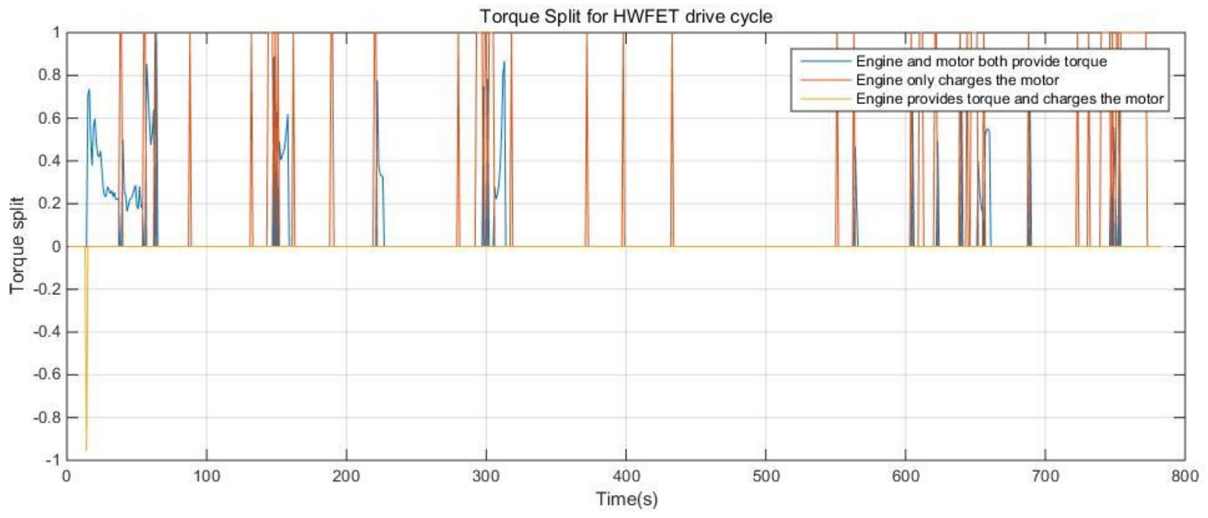


Figure 4.29. Dynamic programing torque split in HWFET drive cycle

The battery state of charge (SOC) is maintained at 55 percent, as Figure 4.30 shows.

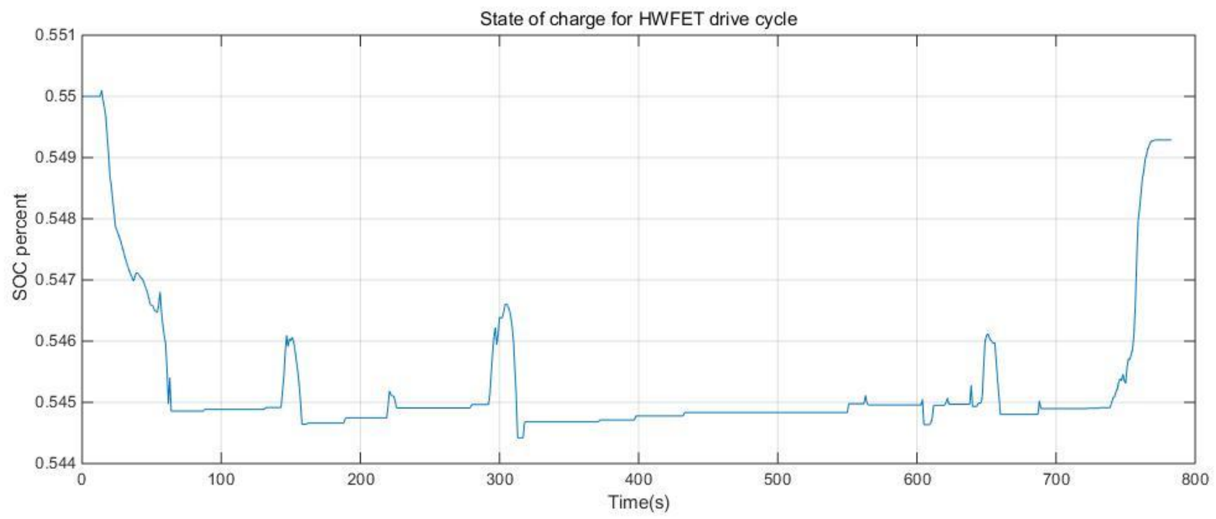


Figure 4.30. Regression model battery SOC for HWFET drive cycle

During the whole drive cycle, the motor energy consumption, engine energy consumption and total energy consumption are shown in Table 4.19.

Table 4.19. Regression model energy consumption results for HWFET drive cycle

Motor Energy Consumption (E_{motor})	7.5568×10^7 (J)
Engine Energy Consumption (E_{engine})	1.1247×10^7 (J)
Total Energy Consumption (E_{total})	1.2003×10^7 (J)

The required power at each time step is shown in Figure 4.31.

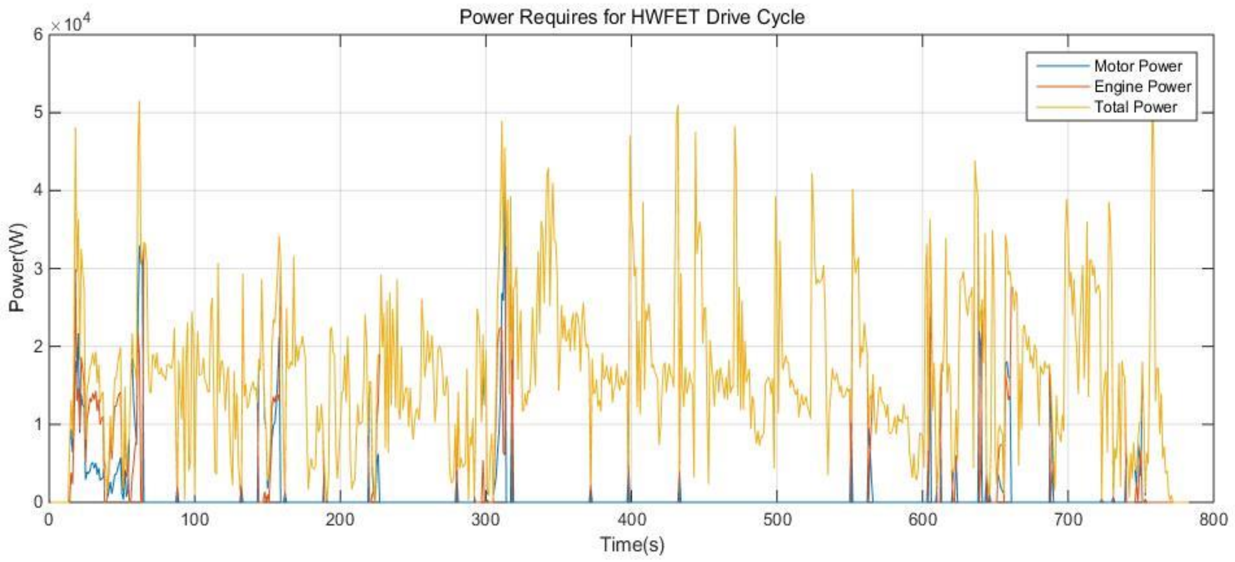


Figure 4.31. Regression model required power for HWFET drive cycle

4.6 Artificial Neural Network Modeling

4.6.1 Input/Output of Neural Networks

Similar to 4.5 Regression Modeling, we use the observable physical variables available in the controller as the inputs to a neural network. According to 4.5, 6 variables of the 9 listed have high P-value. Thus, we only select 6 variables to predict the torque split. The 6 variables are listed as follows:

1. Torque required by the vehicle (τ_v)
2. Engine torque (τ_e)
3. Engine speed (ω_e)
4. Motor torque (τ_m)
5. Vehicle speed (v)
6. Battery State-of-Charge (SOC)

The output of this neural network is a scalar of torque split with range [0,1]. When split equals 0, it means 0% of total torque is provided by the motor. The higher the split is, the higher the torque is from the motor.

4.6.2 Neural Network Architecture

Our neural network structure is simple compared to most neural network architectures. Most neural networks dealing with sequential data have the structure of convolutional layers appended with Long short-term memory (LSTM) and followed by fully-connected layers at the end. Convolutional layers, known as a set of learnable filters, are used to learn the features of the input data. For a 2-D convolutional networks, normal feature maps will be generated after each convolutional layer. However, in this thesis, our input at each time step is a vector with 6 entries of physical variables, and these 6 variables are obtained from a mathematical formula with physical meanings. Therefore, we can consider the input vector as the feature itself and we no longer need convolutional layers to learn the feature of the data. In light of this, we simplify our neural network structure by removing the convolutional layers and keeping only LSTM and fully-connected layers. We call this network 'LSTM Torque'. The visualized network architecture is shown in Figures 4.32 and 4.33. The FC is fully-connected layer.

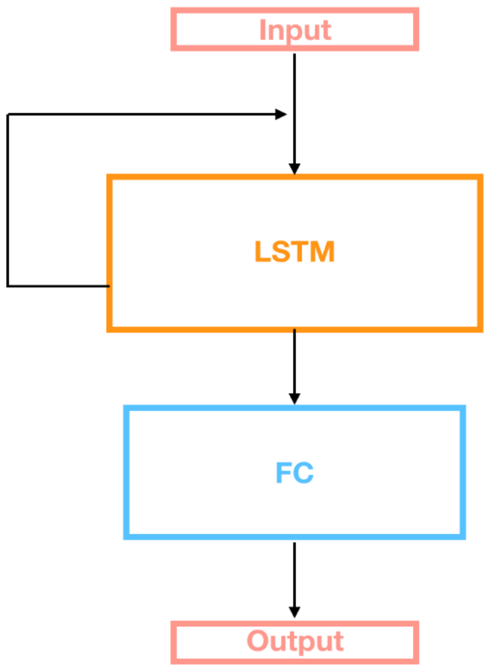


Figure 4.32. Rolled structure of LSTM torque

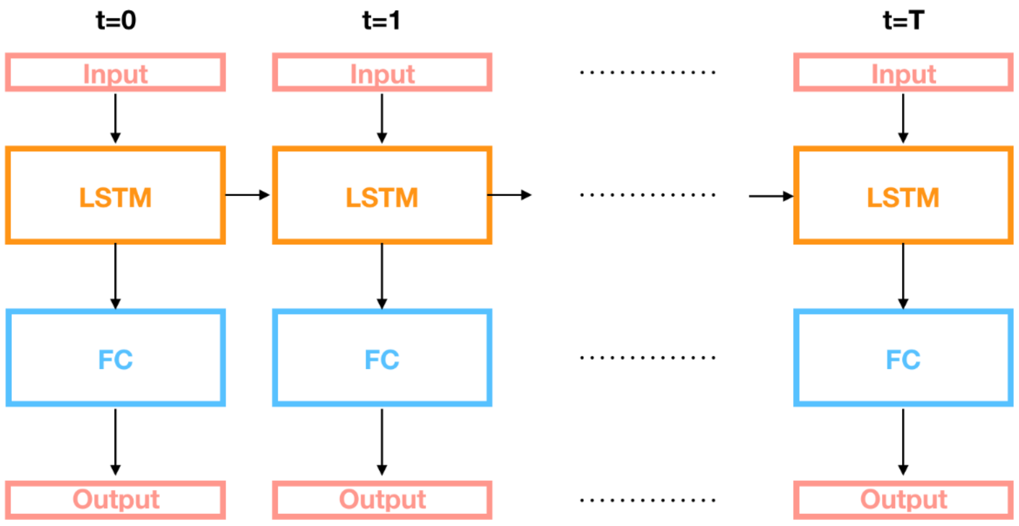


Figure 4.33. Unrolled structure of LSTM torque

4.6.3 Introduction of Dataset

- Training and validation:

Since a neural network is known as a supervised learning method, we need to provide data with annotations for neural networks to train. And in order to train a model that has good performance, a large amount of data is required as a training dataset. Besides, to estimate how well the model has been trained and to analyze the model properties, a validation phase is needed to help choose the best-performing model.

Thirty-six city drive cycles, which are listed in Appendix A, are collected to train and validate the dataset. The thirty-six drive cycles are named as 'City1', 'City2',…, 'City 36'. Among them, there are 3 drive cycles ('City27', 'City28', 'City29') which have too few data points to run the Dynamic Programing Model (DPM) on them. Hence, we discard these 3 drive cycles and only keep the remaining 33 drive cycles as our training/validation dataset. The torque split label for each drive cycle at each time step is obtained by DPM results. Figure 4.34 shows the statistics of number of data points in each drive cycle.

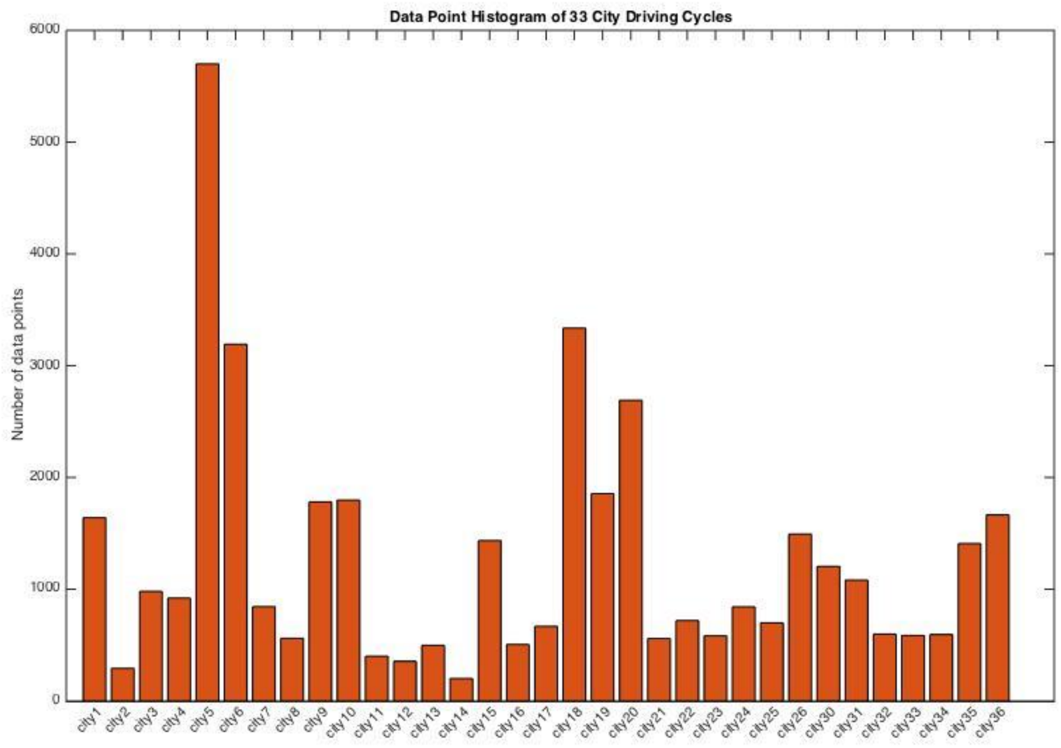


Figure 4.34. Data point histogram of 33 city drive cycles

Table 4.20 shows thirty-six drive cycles and the number of data points.

Table 4.20. Thirty-Six city drive cycles

Data File	Number of Data Points	Data File	Number of Data Points
City 1	1640	City 19	1855
City 2	292	City 20	2690
City 3	981	City 21	560
City 4	921	City 22	720
City 5	5700	City 23	583
City 6	3192	City 24	843
City 7	844	City 25	700
City 8	561	City 26	1494
City 9	1781	City 27	Invalid
City 10	1797	City 28	Invalid
City 11	401	City 29	Invalid
City 12	356	City 30	1204
City 13	498	City 31	1082
City 14	201	City 32	599
City 15	1436	City 33	586
City 16	506	City 34	595
City 17	668	City 35	1409
City 18	3337	City 36	1666

Two drive cycles ('City14', 'City36') in Table 4.20 are randomly selected as the validation data. Eventually, there are 39,831 data points for training and 1,867 data points for validation.

- Testing:

  In the testing phase, the UDDS and Highway FET drive cycles are used to evaluate the performance of the trained model. UDDS has 1,384 data points and Highway FET has 783 data points.

4.6.4 Training Hyper-parameters

Although a neural network is able to learn the parameters itself by back-propagation, many experiments are required to search for the best setting in the hyper-parameter space. In this thesis, we show our experiment results in the following part with different (1) hidden dimension, (2) loss function, (3) learning rate, (4) momentum and (5) epoch number.

- Choice of hidden dimension:

  In the implementation, the hidden dimension is set equal to the cell state dimension. The hidden state and output are the same in LSTM. But there is an additional cell state. The previous cell state, previous hidden state and the current input are used to update the current cell state and the current hidden state. We explored more by experimenting with different hidden dimensions in the following.

  We set hidden dimension size to be 16, 32, 64 and 128 and trained each model for 1,000 epochs. To control other factors, the MSE loss type is chosen; the learning rate is 0.01 and the momentum is 0.9; these parameters will be chosen and explained in the next parts. The results of training loss and validation loss with different epochs in the case of 16, 32, 64 and 128 hidden dimensions are shown in Figures 4.35, 4.36, 4.37, and 4.38.

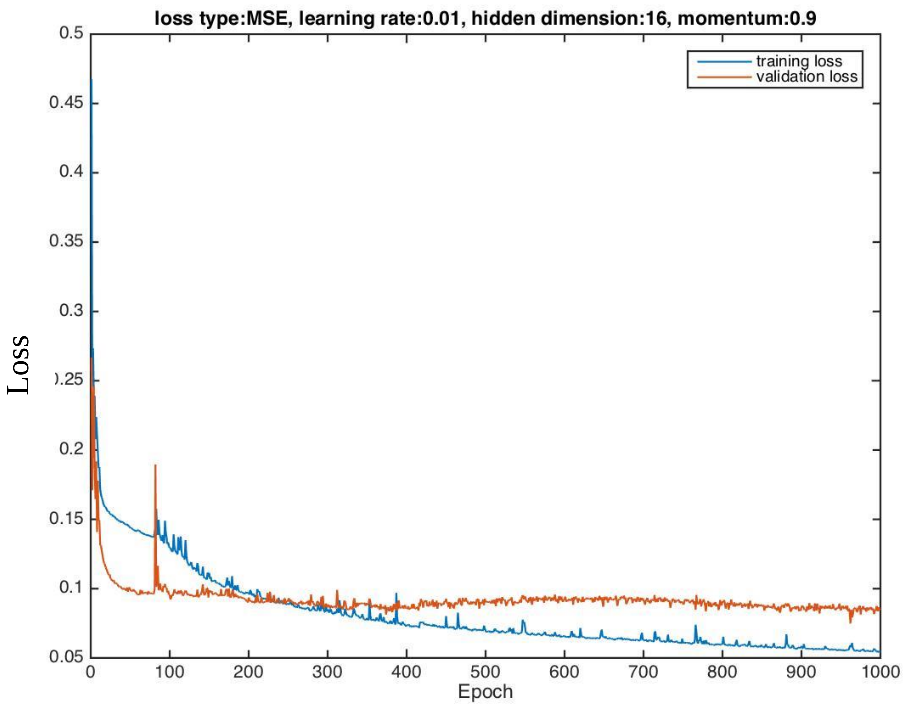


Figure 4.35. Training loss and validation loss for 16 hidden dimensions

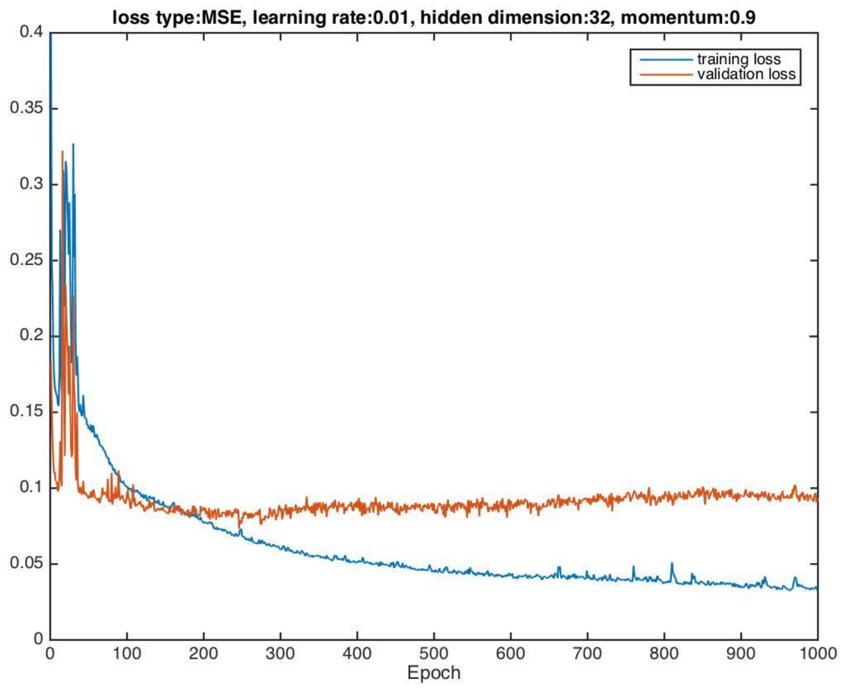


Figure 4.36. Training loss and validation loss for 32 hidden dimensions

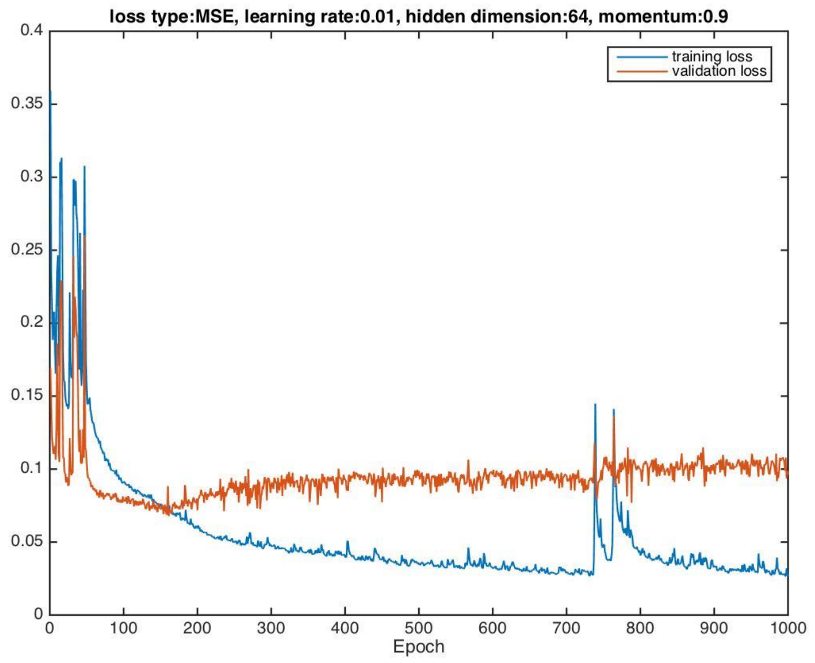


Figure 4.37. Training loss and validation loss for 64 hidden dimensions

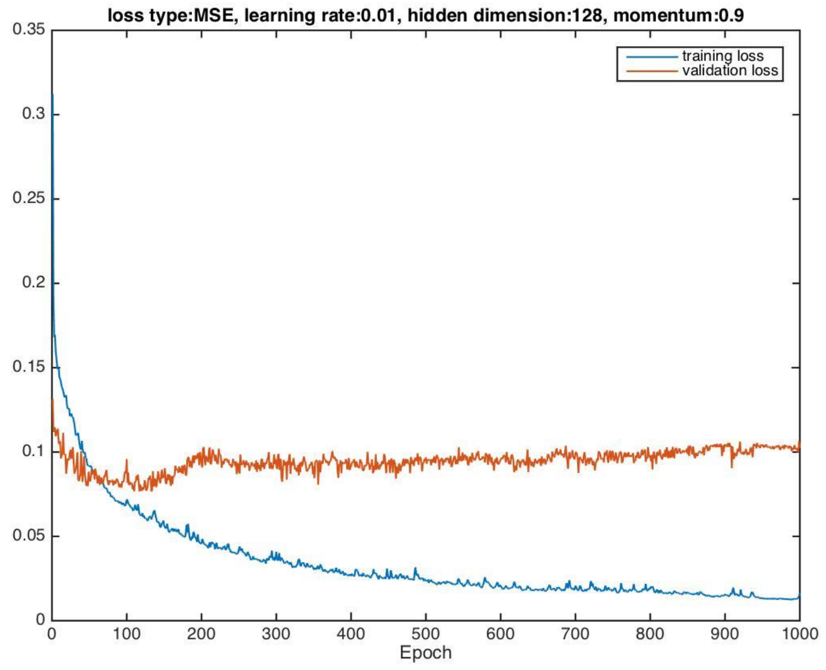


Figure 4.38. Training loss and validation loss for 128 hidden dimensions

From these figures, it can be noticed that the training loss decreases with epoch but validation loss first decreases and then increases due to over-fitting (we will discuss this in 'Choice of iteration number'). Thus, the minimum validation loss can be taken from each training and shows the tendency in Figure 4.39.

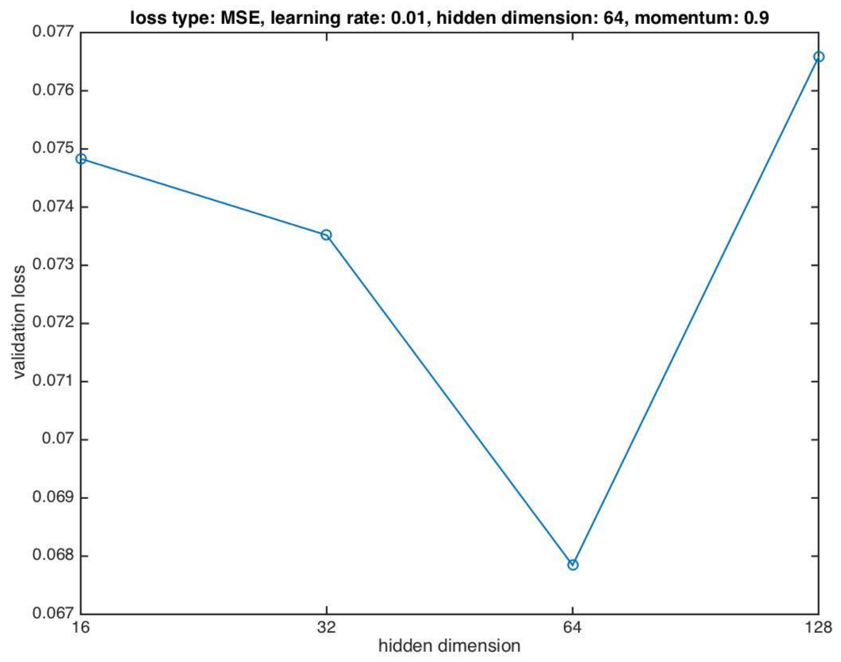


Figure 4.39. Compare validation loss of different hidden dimensions

It can be concluded that when the number of hidden dimensions is 64, the model achieves the best validation performance. Hence, the hidden dimension of 64 was chosen in the final model.

- Choice of loss function:

The loss function in neural networks measures the difference between model output $(\hat{y})$ and target output(y). It is a non-negative value, where the robustness of the model increases along with the decrease in value of loss function. Since we expect our model to precisely predict the desirable result, we always want the loss in the trained model to be as small as possible. Also, while optimizing the model, the gradient is derived from loss. Therefore, it is critical to use a suitable loss function. For our purpose which is to output a scalar at every time step, there are two commonly used loss functions - Mean Square Error (MSE) loss and L1 loss. We show our comparison of these two options in the following part. To control other factors, the hidden dimension is set to 64, the learning rate is 0.01 and the momentum is 0.9.

MSE loss is widely used in linear regression. The standard form is defined as

$$L = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 \tag{4.30}$$

where $\left(y^{(i)} - \hat{y}^{(i)}\right)$ is named as the residual, and the target of the MSE loss function is to minimize the residual sum of squares. Figure 4.40 shows the training and validation losses with MSE loss function.

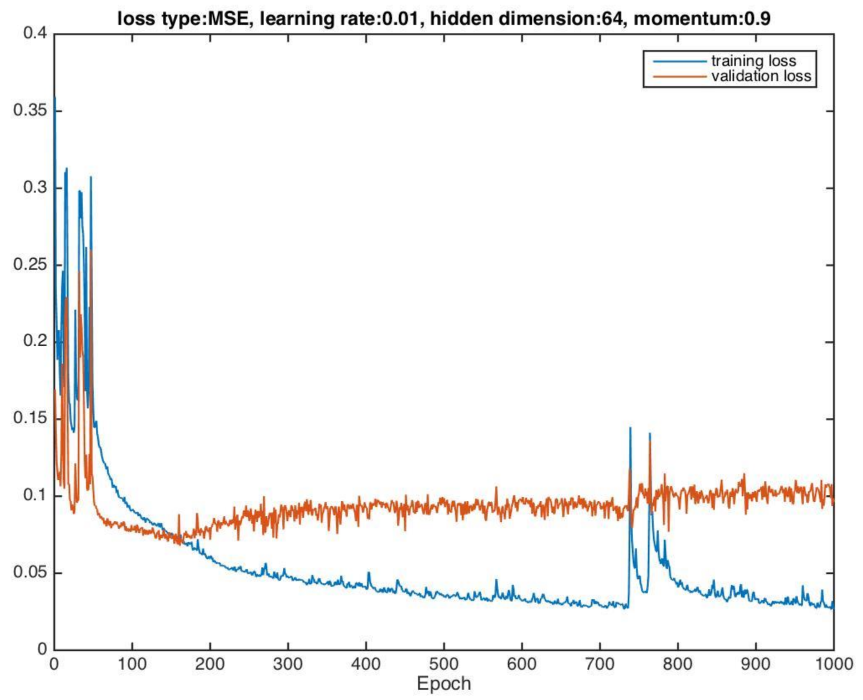


Figure 4.40. Training loss and validation loss for MSE loss type

L1 loss function minimizes the absolute differences between the estimated values and the existing target values. The standard form is defined as

$$L = \frac{1}{n}\sum_{i=1}^{n} |y^{(i)} - \hat{y}^{(i)}| \tag{4.31}$$

Figure 4.41 shows the training and validation losses with L1 loss function.

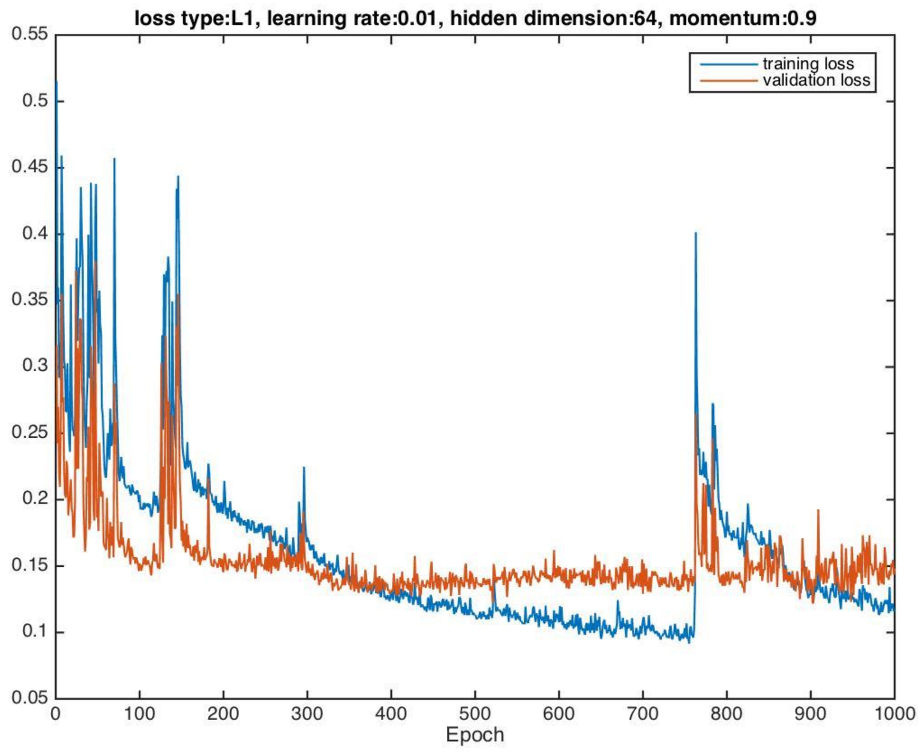


Figure 4.41. Training loss and validation loss for L1 loss type

Loss function only affects the training process and when it comes to determine which model is better, we propose another testing criterion which is the absolute difference of predictions and targets for test data. We discover that MSE loss gives absolute difference error around 0.05 and L1 loss gives absolute difference error around 0.08, indicating MSE loss is superior to L1 loss in our experiment. This seems contradictory to what we expect. However, we should notice that the loss function is only designed for minimizing the loss in training dataset during optimization, but does not guarantee to achieve the minimal error on test data because the way we compute this absolute difference for test data is different from computing either MSE loss or L1 loss. Therefore, we choose to use MSE loss in our final training model.

- Choice of learning rate:

Once the neural network backpropagates to calculate the derivative of the loss function with respect to each weight, learning rate is needed to decide by how much the derivative should be subtracted from the weights. There is a trade-off between selecting a large or small learning rate. With large learning rate, the neural network converges fast at the beginning of training. But when the model reaches closer and closer to the global minimizer, large learning rate makes the loss function overshoot the minimum and never reach it. Small learning rate is safer but if the learning rate is too conservative, the minimization function will take an extremely long time to reach the minimum. Also, if a local minimum exists, which is quite common in the parameter space, using a small learning rate usually causes the loss function to be trapped in a local minimum. Figure 4.42 illustrates these with a 2-D example.

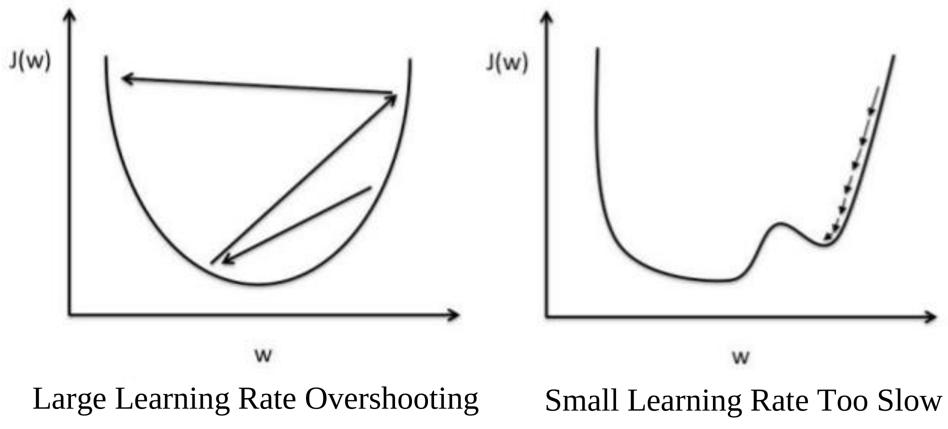

J(w)
w
Large Learning Rate Overshooting
J(w)
w
Small Learning Rate Too Slow

Figure 4.42. Examples of large and small learning rate [34].

To make a wise choice of learning rate requires many experiments. We tested learning rate for 0.1, 0.05, 0.005 and 0.001. To control other factors, the hidden dimension is set to 64, the MSE loss type is chosen, and the momentum is set to 0.9.

Figures 4.43, 4.44, 4.45, 4.46 and 4.47 show the training and validation losses for 0.1, 0.01, 0.05, 0.005 and 0.001 learning rate, respectively.

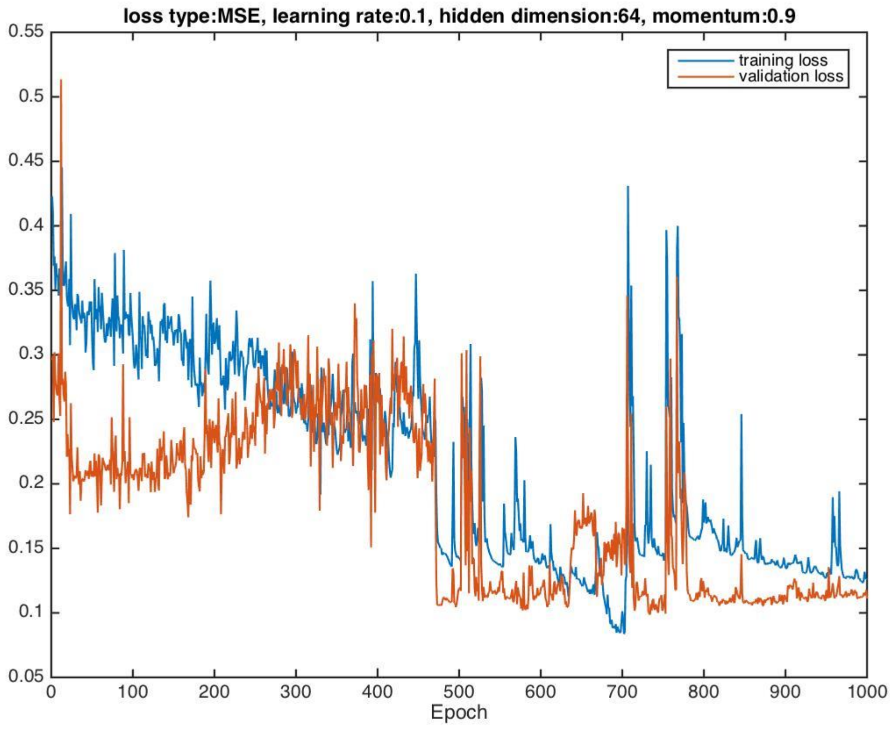


Figure 4.43. Training loss and validation loss for 0.1 learning rate

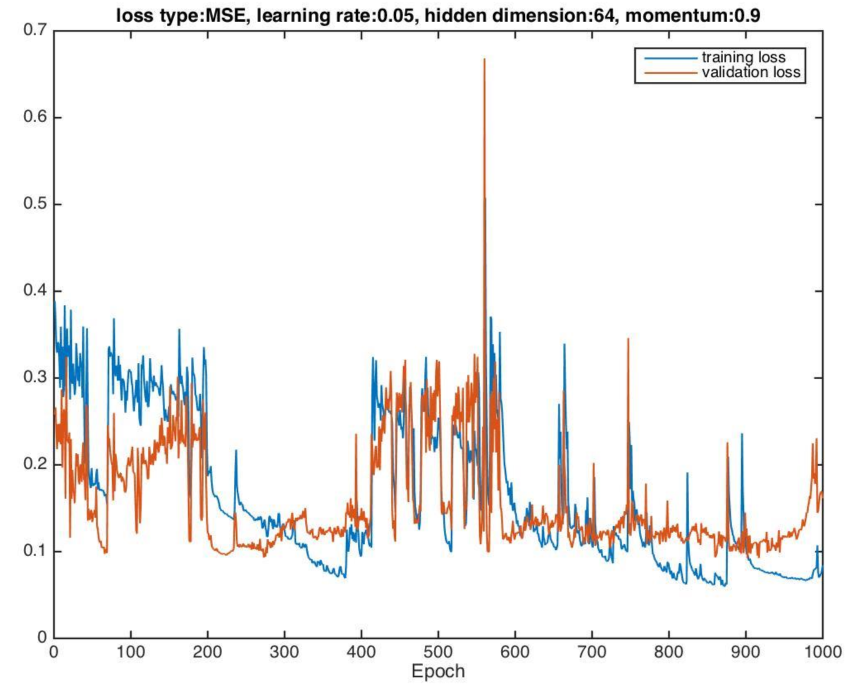


Figure 4.44. Training loss and validation loss for 0.05 learning rate

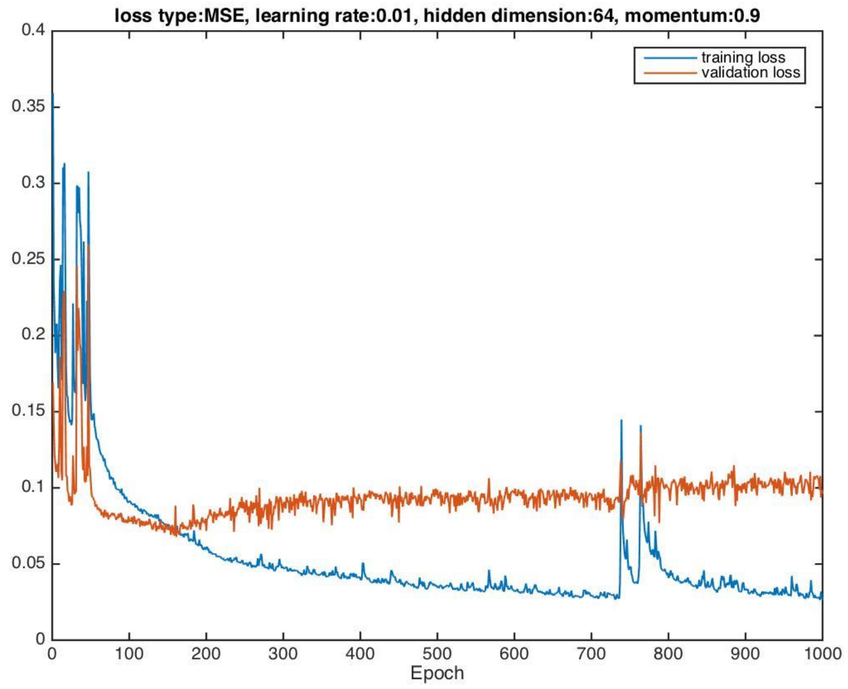


Figure 4.45. Training loss and validation loss for 0.01 learning rate

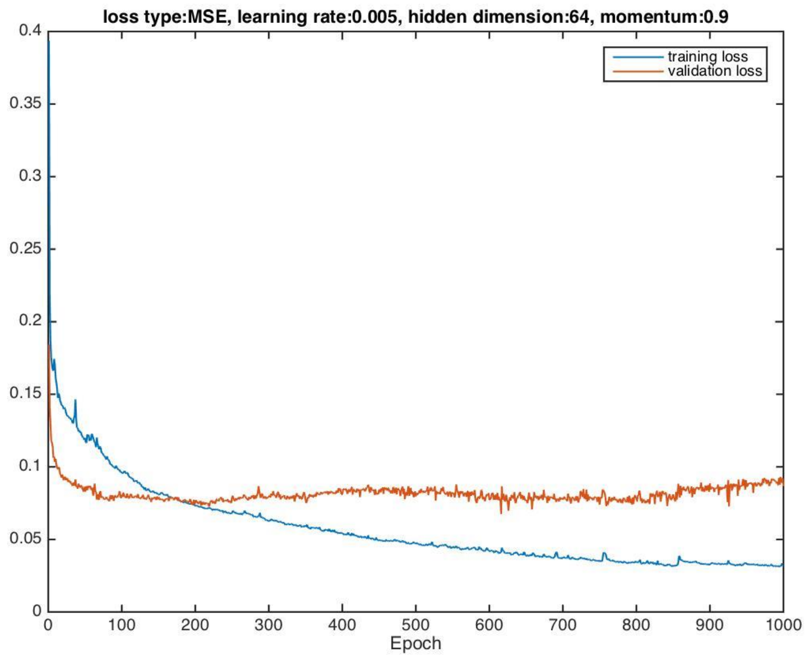


Figure 4.46. Training loss and validation loss for 0.005 learning rate

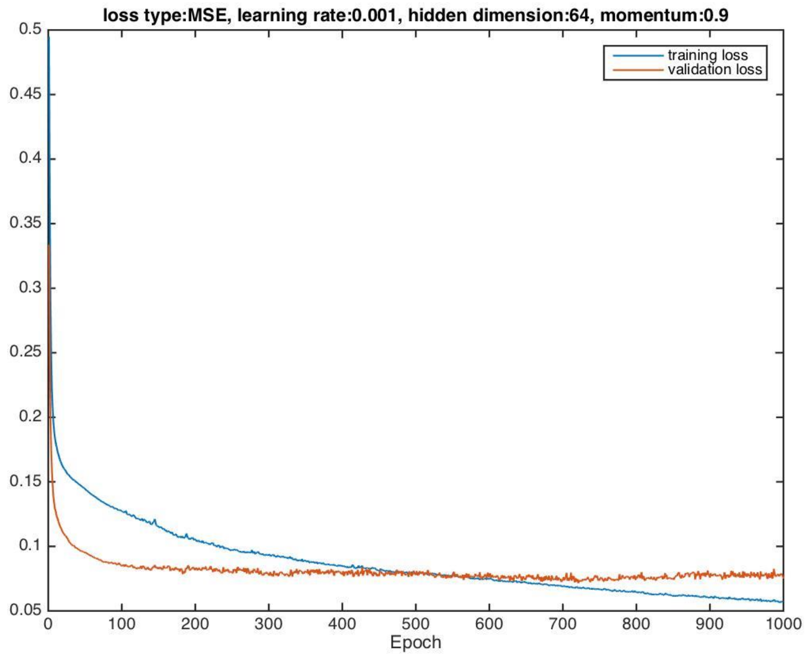


Figure 4.47. Training loss and validation loss for 0.001 learning rate

In Figure 4.43 and Figure 4.44, when learning rate equals to 0.1 or 0.05, we can see the training loss oscillate dramatically, indicating the occurrence of overshooting, which means the learning rate is too large. However, although Figures 4.46 and 4.47 with 0.005 and 0.001 learning rate show smoother loss curves, it is very time consuming to reach the 0.05 training loss. Learning rate 0.005 loops approximately 400 epochs to achieve 0.05 training loss and setting learning rate to 0.001, the training loss is always beyond 0.05 until the end of 1,000 epoch training. Taking both speed and convergence into consideration, we finalize learning rate to be 0.01.

- Choice of momentum:

In a neural network, a gradient descent optimization algorithm is used to minimize the loss function to reach a global minimum. As mentioned previously, it is not guaranteed to find the global minimum because the real loss surface is more complex and may contain many local minima. Since the local minimum may cause the optimization to get stuck, the algorithm assumes it has already reached the global minimum, leading to sub-optimal results. In order to avoid this case, we introduce a momentum term here added to the objective function. Momentum is a scalar ranging from 0 to 1, which increases the size of the leap taken towards the minimum by trying to jump out of a local minimum. Normally, a large value of momentum also leads to faster convergence. To see the effects of momentum, the training loss and validation loss results without momentum and with 0.9 momentum are shown in Figure 4.48 and Figure 4.49, respectively. To control other factors, the hidden dimension is set to 64, the MSE loss type is chosen, and the Learning rate is set to 0.01.

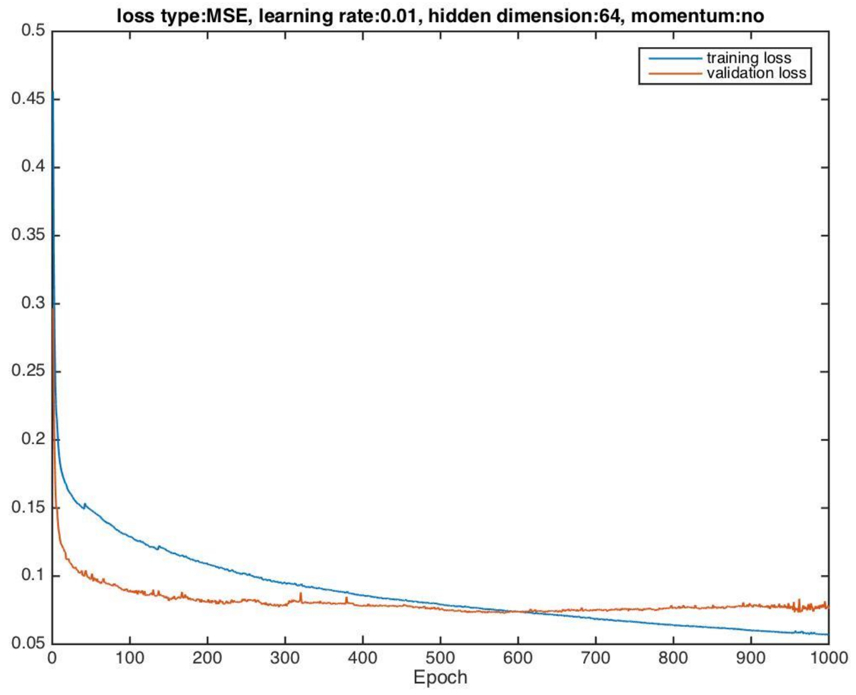


Figure 4.48. Training loss and validation loss without using momentum

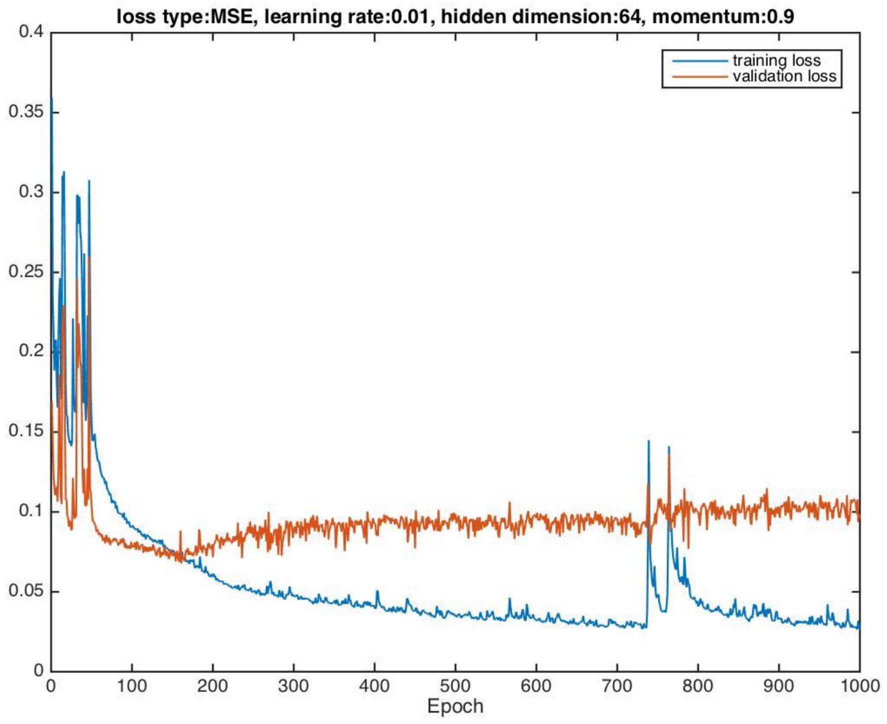


Figure 4.49. Training loss and validation loss with 0.9 momentum

Figure 4.48 shows the changing loss tendency with no momentum added. Although both curves present smoothness, the training loss goes down very slowly and never decreases below 0.05 before the end of 1,000 epoch training. It is obvious that, without momentum, the algorithm has difficulty to avoid the local minimum trap, thus has no chance to reach the global minimizer. However, in Figure 4.49 with only a little oscillation, momentum assists to avoid the local minima by reaching the 0.05 training loss within 200 training epochs. According to other researchers' experience of tweaking neural network hyper-parameters, 0.9 is a magic value for momentum which usually has good performance. Thus, we decide momentum to be 0.9 in our optimal model.

- Choice of epoch number:

To better understand neural network training, we have to know 'over-fitting' and 'under-fitting'. With the knowledge of these two phenomena, we are able to choose a suitable iteration number and get the optimal model.

Neural network is a data-based algorithm that relies on the training dataset to learn the features. On the one hand, if we over-train the model (e.g., train too many epochs with suitable hyper-parameters), it may happen that the model learns too many detailed features and even noise, thereby memorizing the training data. In this case, the model is no more applicable to the test data and is negatively impacted for generalization. On the other hand, if the model is not trained enough (e.g., train too few epochs even with suitable hyper-parameters), the model only learns some basic features of the task and is unable to give correct prediction on the training data. In such case, we need to train the model with more epochs. It is tricky to choose the right epoch number. However, a validation dataset is introduced to solve this problem. During training, we normally test the current model on the validation dataset after several epochs of training. With suitable hyper-parameters, it is always the case that training loss always goes down and validation loss first decreases and then increases. The turning point where the validation loss starts to increase is the epoch number that will be chosen. Two examples of under-fitting and over-fitting are presented in Figures 4.50 and 4.51.

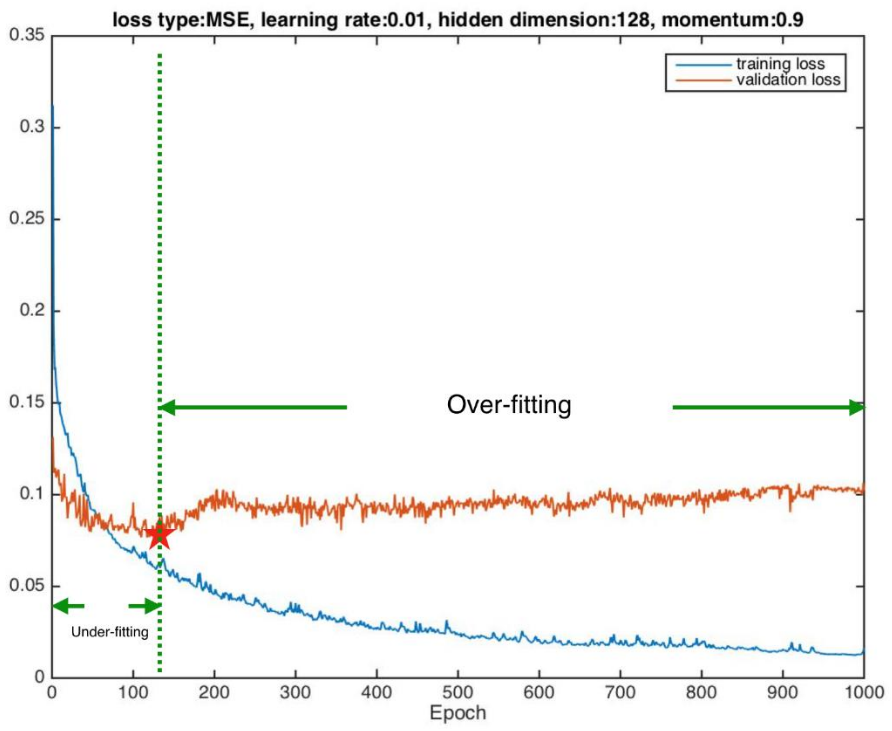


Figure 4.50. Example of under-fitting and over-fitting around epoch 130

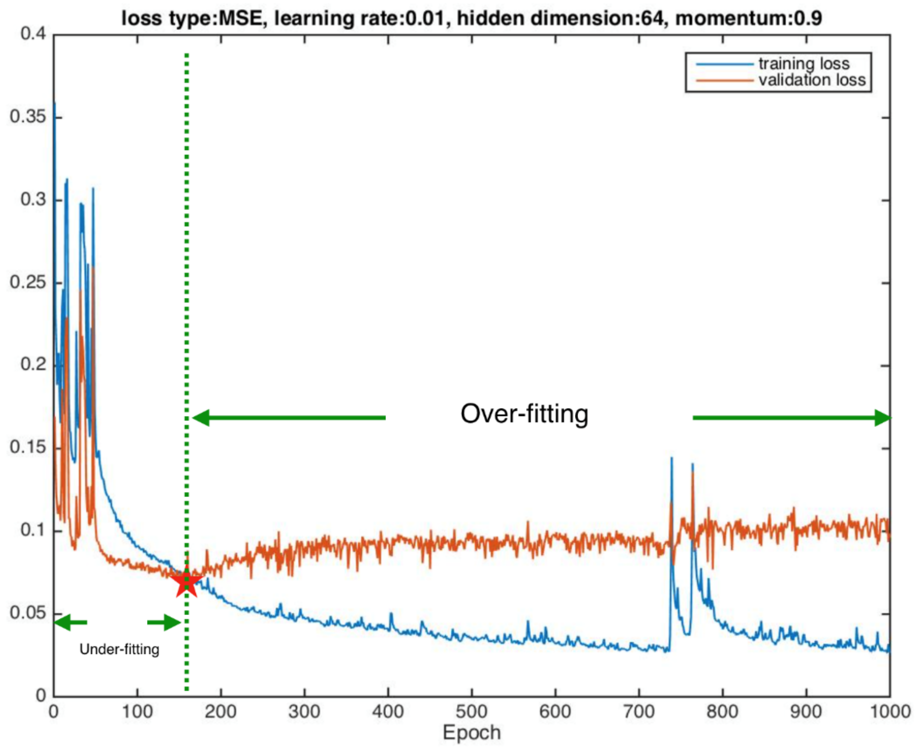


Figure 4.51. Example of under-fitting and over-fitting around epoch 160

4.6.5 Optimal Hyper-parameter Settings

The LSTM Torque network is implemented in Pytorch. Pytorch is an open-source machine learning library for Python, based on Torch, used for applications such as deep learning. According to our 10 experiments in 4.6.4, we finalize our hyper-parameter settings as follows:

1. Hidden Dimension = Cell State Dimension : 64
2. Loss Criterion: Mean Square Error Loss
3. Learning Strategy: Fixed
4. Learning Rate: 0.01
5. Momentum: 0.9

6. Epoch Number: 160

4.6.6 Neural Network Model Results in UDDS Drive Cycle

In the city UDDS drive cycle, the torque split result is shown in Figure 4.52.

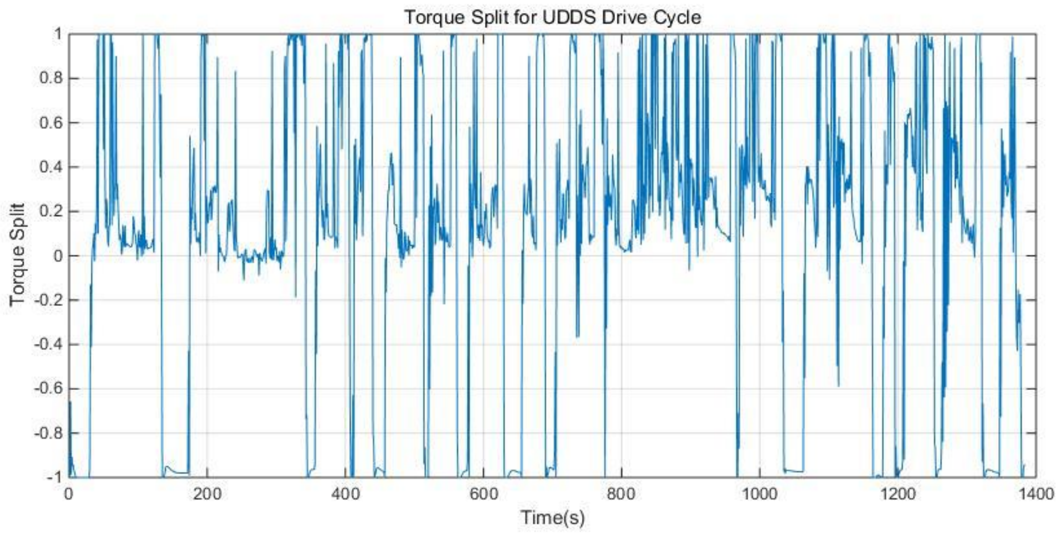


Figure 4.52. Neural networks model torque split in UDDS drive cycle

The battery state of charge (SOC) is maintained at 55 percent, as Figure 4.53 shows.

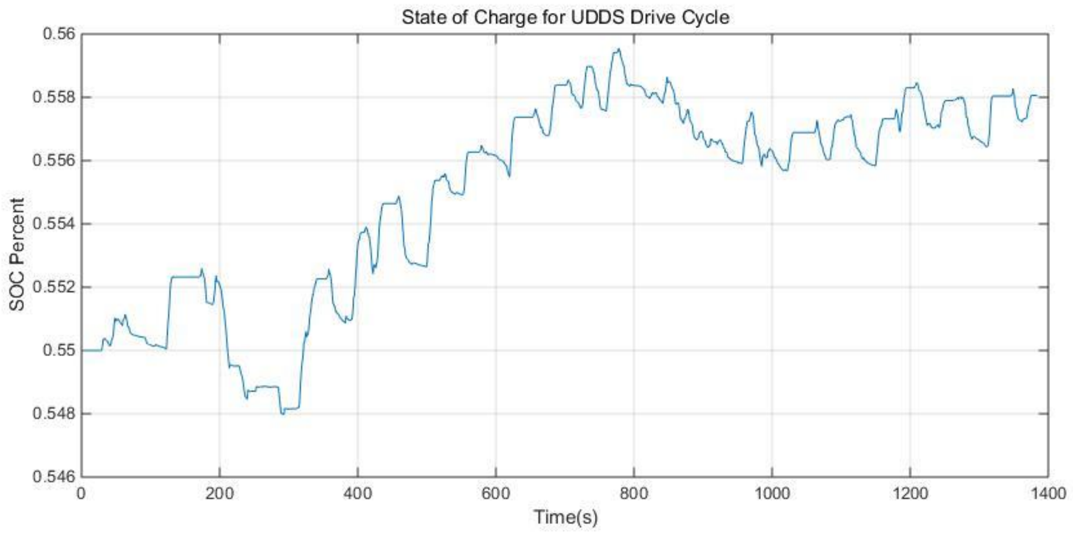


Figure 4.53. Neural networks model battery SOC for UDDS drive cycle

During the whole drive cycle, the total energy consumption (E_{total}) can be calculated from formula (4.22) , as 7.6167×10^6 (J).

The required power at each time step is shown in Figure 4.54.

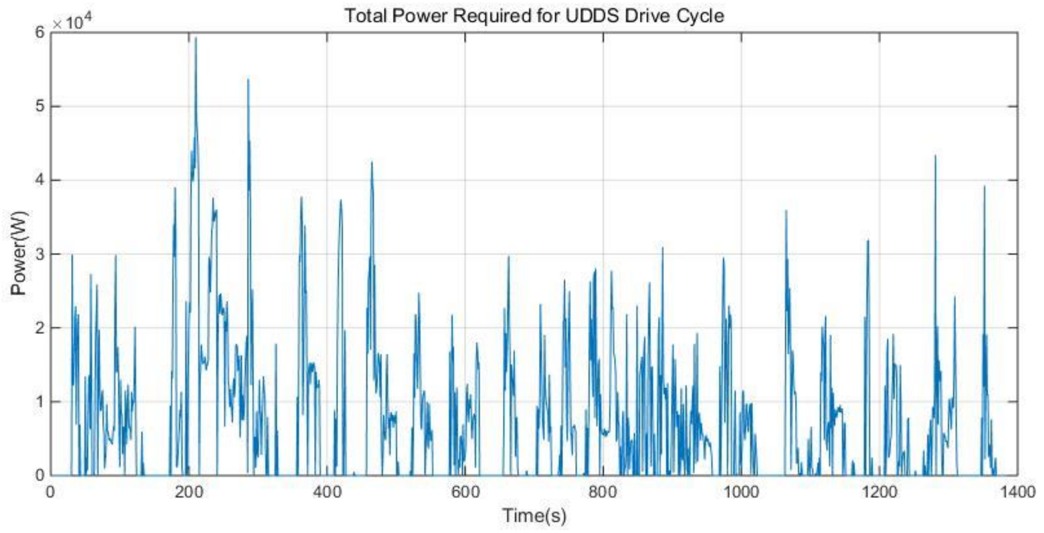
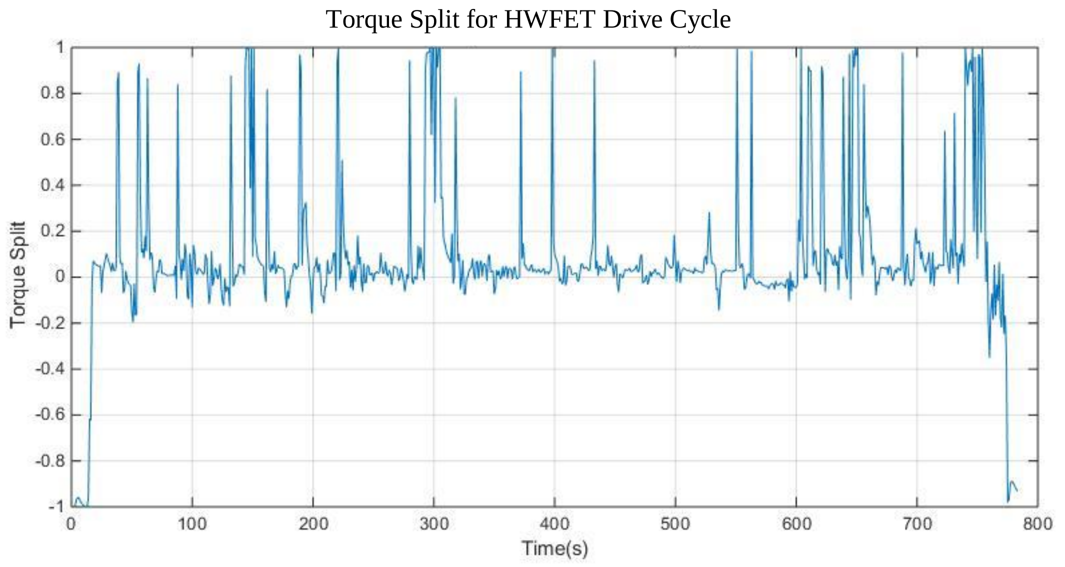


Figure 4.54. Neural networks model required power for UDDS drive cycle

4.6.7 Neural Network Modeling Results in Highway FET Drive Cycle

In the city HWFET drive cycle, the torque split result is shown in Figure 4.55.



The battery state of charge (SOC) is maintained at 55 percent, as Figure 4.56 shows.

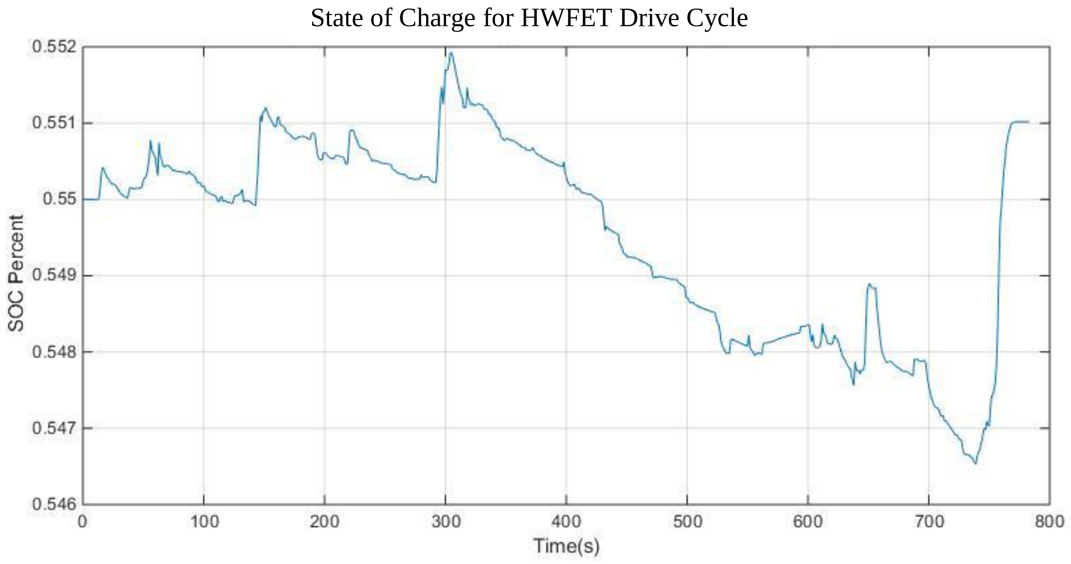


Figure 4.56. Neural networks model battery SOC for HWFET drive cycle

During the whole drive cycle, the total energy consumption (E_{total}) can be calculated from formula (4.22) , as 1.2192×10^7 (J).

The required power at each time step is shown in Figure 4.57.

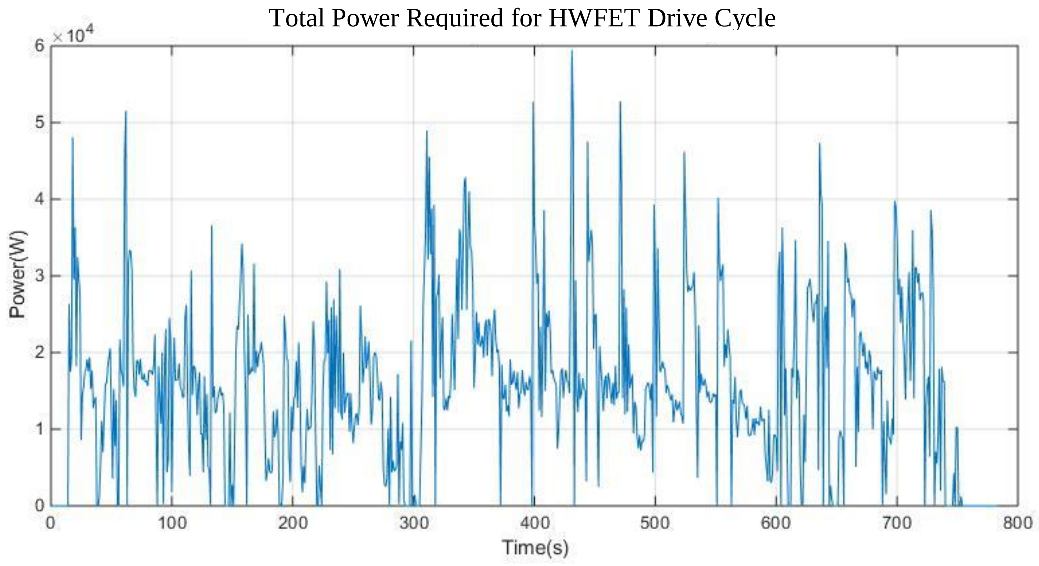


Figure 4.57. Neural networks model required power for HWFET drive cycle

4.7 Comparison and Summary

4.7.1 Comparison between Regression Model and Neural Network Model

In Rohinish Gupta's work [7], the Regression Model is proven as the best real-time implementable strategy for power management. Thus, the Neural Network Model is compared with the Regression Model.

Since both the Regression Model and the Neural Network Model are developed based on DP, which also acts as benchmark, the results of these two models need to be compared with the results from DP. If the results of the Neural Network Model are more similar to the DPM than the Regression Model, the Neural Network Model is the better solution. The UDDS drive cycle and Highway FET drive cycle are used as the test samples.

4.7.2 UDDS Test sample Comparing Regression Model and Neural Network Model

Figure 4.58 shows the Regression Model's prediction result of torque split compared with DPM's prediction. The absolute difference（$E_{regression}$）shows the similarity between them, calculated as：

$$E_{regression} = \frac{1}{n}\sum_{t}\left(\frac{\left|\tau_{dpm} - \tau_{regrssion}\right|}{\tau_{dpm}}\right) = 15.71\% \tag{4.32}$$

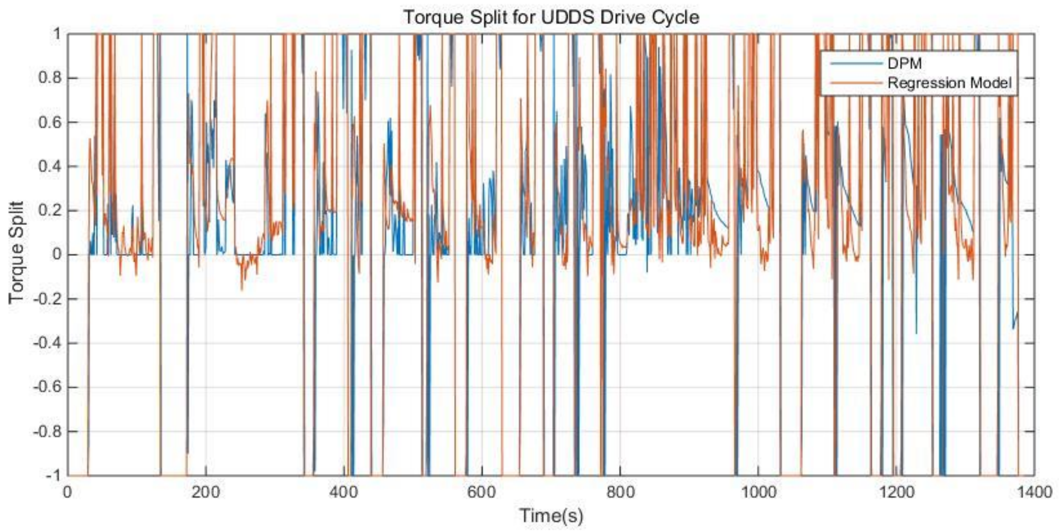


Figure 4.58. Torque split comparison for DP and regression model in UDDS drive cycle

Figure 4.59 shows the Neural network model's prediction result of torque split compared with DP Model prediction. The absolute difference (E_{nn}) is calculated as :

$$E_{nn} = \frac{1}{n}\sum_{t}\left(\frac{\left|\tau_{dpm} - \tau_{nn}\right|}{\tau_{dpm}}\right) = 11.84\% \tag{4.33}$$

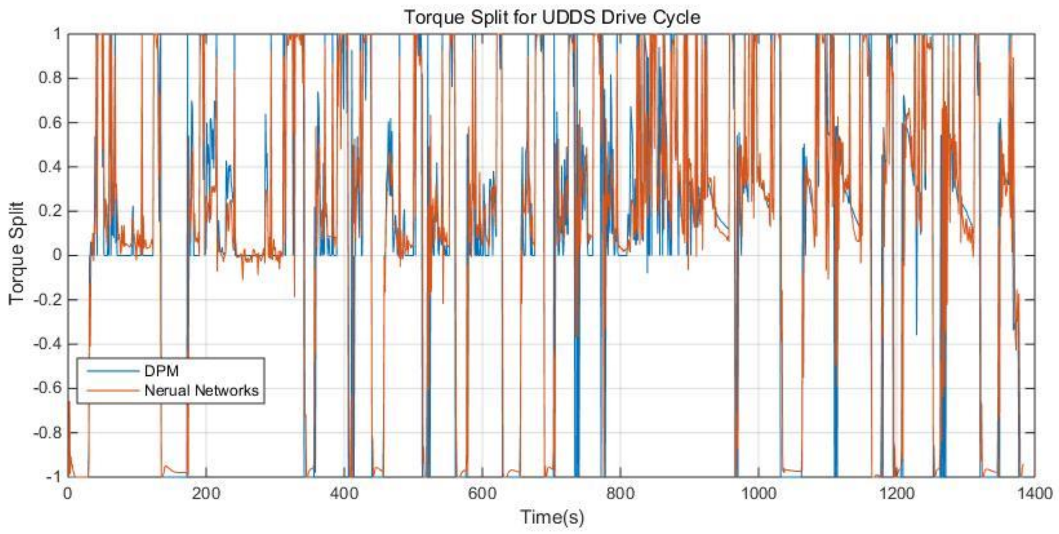


Figure 4.59. Torque split comparison for DPM and NNM in UDDS drive cycle

Thus, NNM has less difference compared with the DP result than the Regression Model.

For further comparison, the difference value of torque split between Regression Model and DPM as well as the difference value of torque split between NNM and DPM are shown in Figure 4.60.

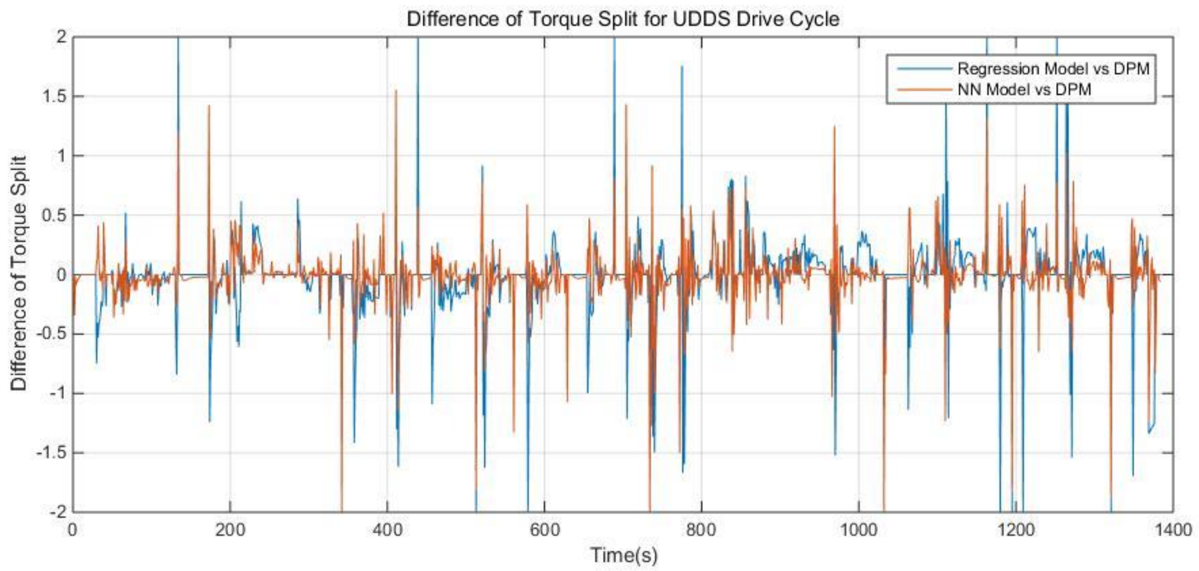


Figure 4.60. The difference value of torque split for Regression Model with DPM and NNM with DPM in UDDS drive cycle

The state of charge and total power required for these two models are also compared with DPM. Figures 4.61 and 4.62 show the total power required comparison between DPM and Regression Model, as well as DPM and NNM.

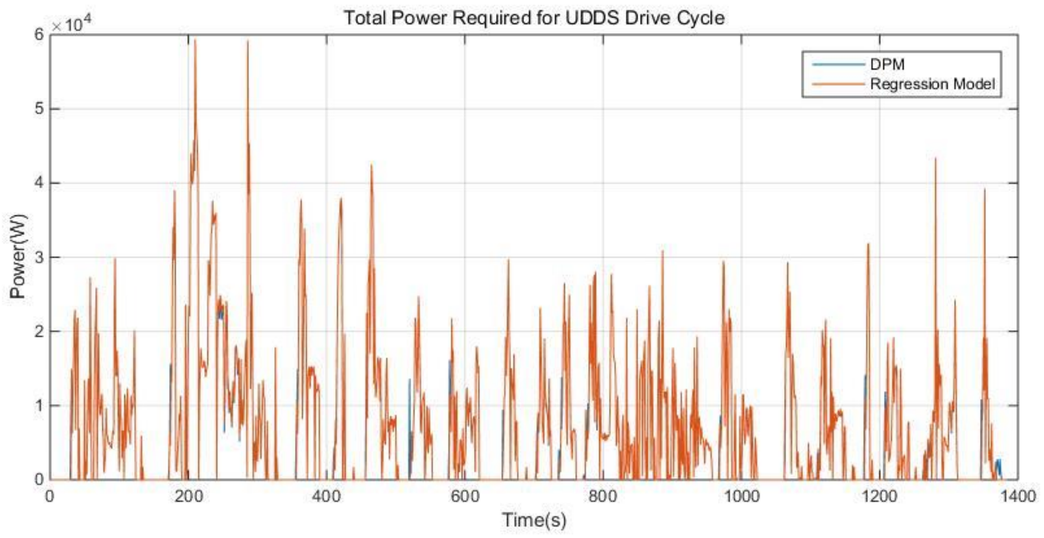


Figure 4.61. Total power required comparison for DPM and regression model in UDDS

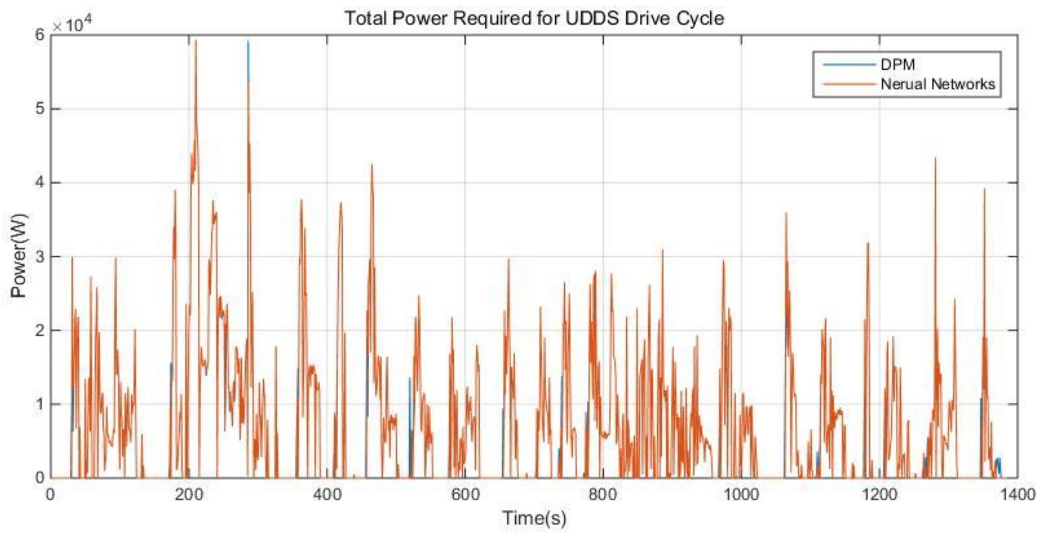


Figure 4.62. Total power required comparison for DPM and NNM in UDDS

For further comparison, we look at the total power required, which is the sum of the motor power required and the engine power required. Figure 4.63 shows the motor power required comparison between DPM, Regression Model and NNM. And Figure 4.64 shows the engine power required comparison between DPM, Regression Model and NNM.

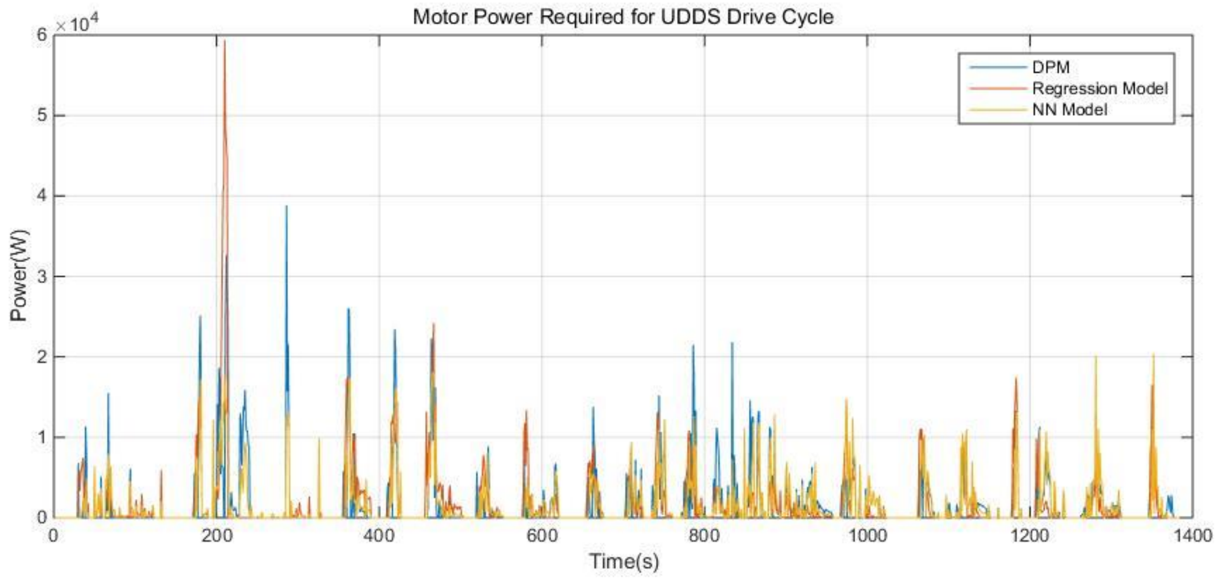


Figure 4.63. Motor power required comparison between DPM, Regression Model and NNM for UDDS Drive Cycle

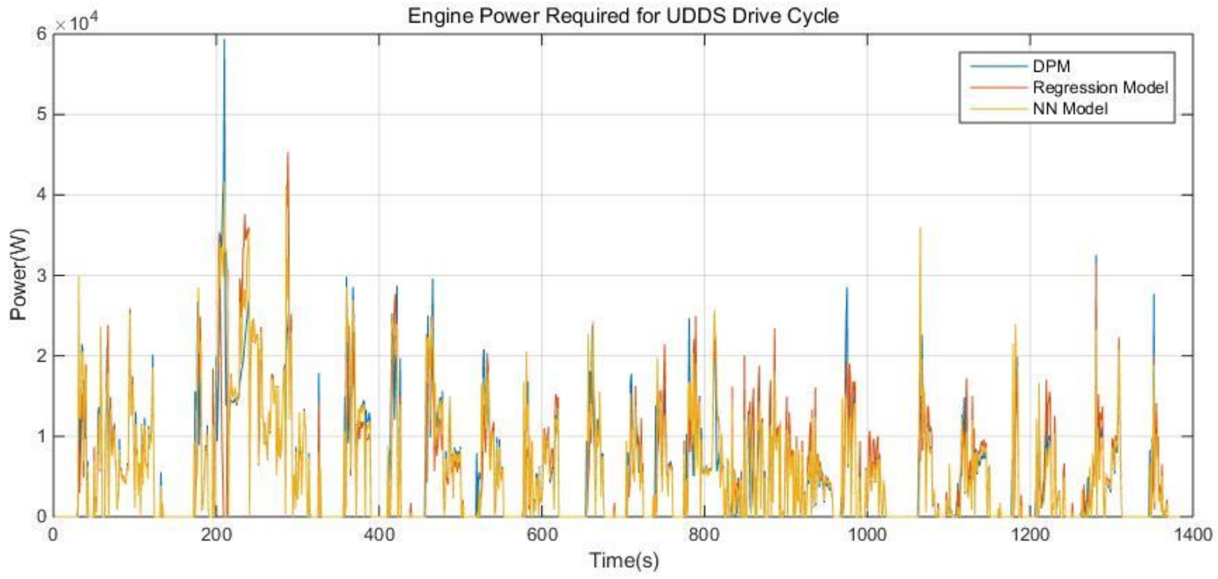


Figure 4.64. Engine power required comparison between DPM, Regression Model and NNM for UDDS Drive Cycle

Table 4.21. shows the engine energy consumption (E_{engine}), motor energy consumption (E_{motor}) and the total energy consumption (E_{total}). The total energy consumption of the Regression Model is 0.7% higher than the DPM, and the total energy consumption of the NNM is 1.25% higher than the DPM, which is not a significant difference.

Table 4.21 Energy consumption comparison for UDDS drive cycle

	Engine Energy Consumption (E_{engine})	Motor Energy Consumption (E_{motor})	Total Energy Consumption (E_{total})
DPM	7.5592×10^6 (J)	2.0490×10^6 (J)	9.6082×10^6 (J)
Regression	7.4058×10^6 (J)	2.2770×10^6 (J)	9.6828×10^6 (J)
Neural Networks Model	7.6167×10^6 (J)	2.1115×10^6 (J)	9.7282×10^6 (J)

Figure 4.65 and Figure 4.66 show the state of charge comparison between DPM and Regression Model, as well as DPM and NNM. The SOC remains at 0.55. The final SOC for Regression Model is close to DPM, and for NNM is around 5% higher than DPM. However, NNM keeps more features from DPM and higher final SOC. Therefore, for NNM, more energy is stored in the battery, which explains why NNM has more engine energy consumption and total energy consumption.

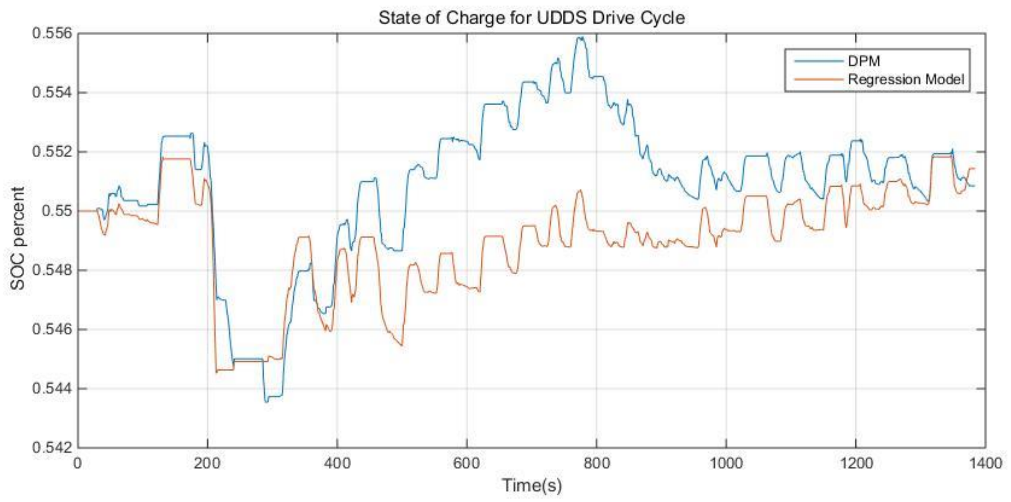


Figure 4.65 State of charge comparison for DPM and regression model in UDDS

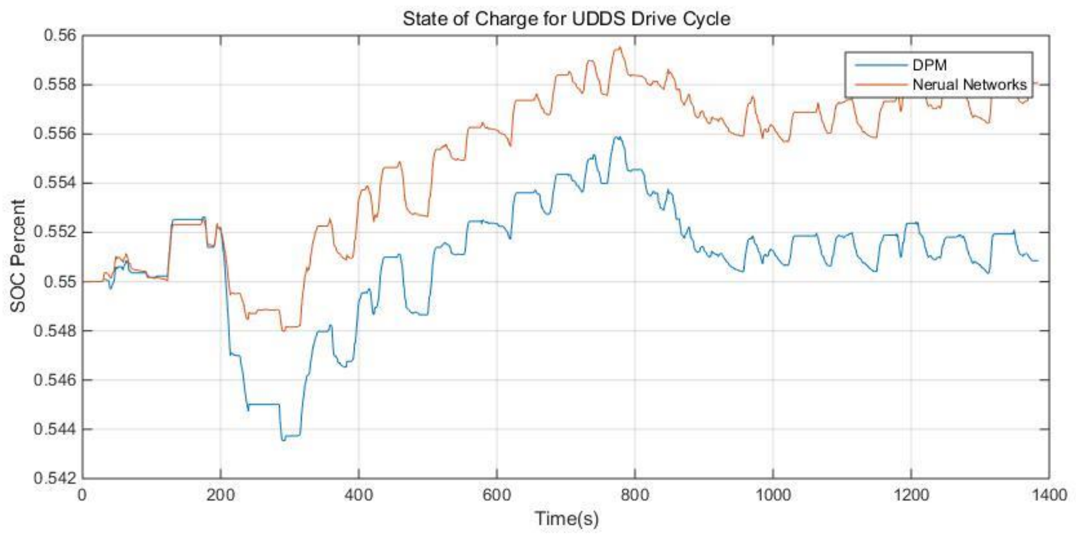
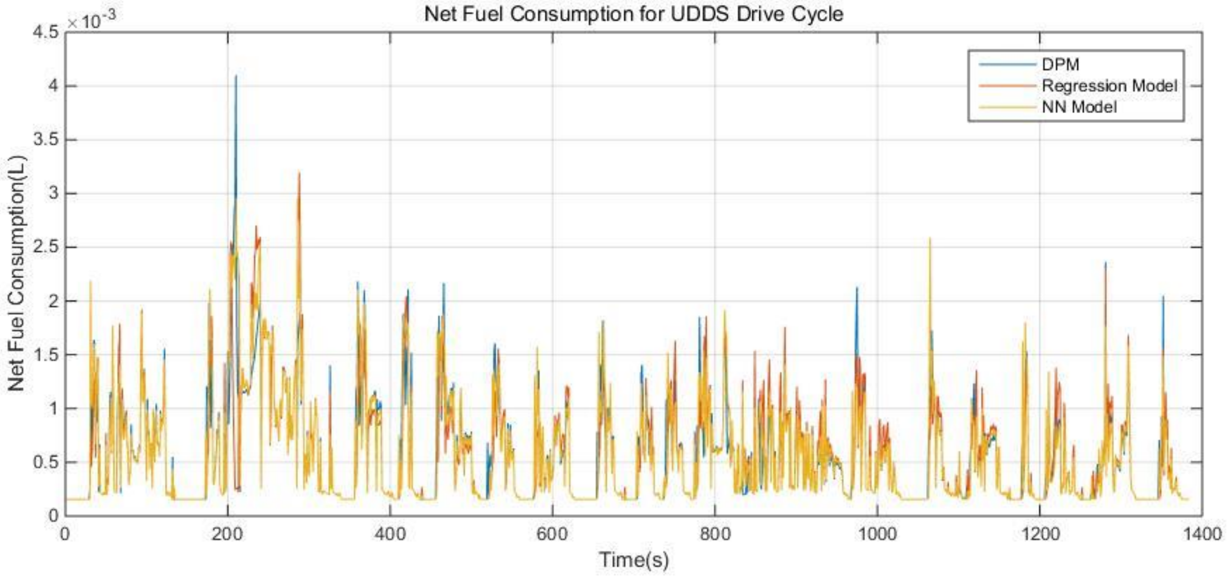


Figure 4.66. State of charge comparison for DPM and NNM in UDDS

The fuel consumption comparison for DPM, Regression Model and NNM is shown in Figure 4.67.

Figure 4.67. Fuel consumption comparison for DPM, Regression Model and NNM for



UDDS Drive Cycle.

The GHG_{WTW} emissions for DPM, Regression Model and NNM are shown in Figure 4.68.

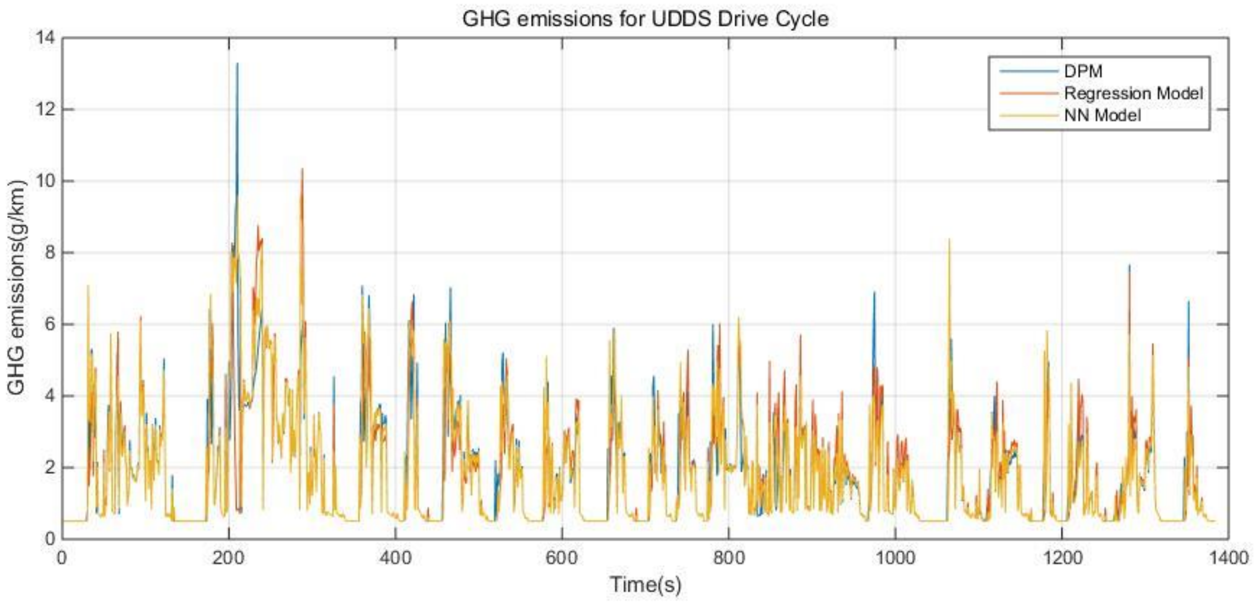


Figure 4.68. GHG_{WTW} emissions comparison for DPM, Regression Model and NNM for UDDS Drive Cycle

The total fuel consumption and GHG_{WTW} emissions during the UDDS drive cycle are shown in Table 4.22. For total fuel consumption, the result for Regression Model is 0.87% lower than for DPM, and the result for NNM is 0.89% higher than for DPM. For the GHG_{WTW} emissions, the result for Regression Model is 0.86% lower than for DPM, and the result for NNM is 0.89% higher than for DPM.

Table 4.22. Total fuel consumption and GHG_{WTW} emissions during the UDDS drive cycle

	Total Fuel Consumption	Total GHG_{WTW} emissions
DPM	0.7789(L)	2.5260×10^3 (g/km)
Regression	0.7721(L)	2.5042×10^3 (g/km)
Neural Networks Model	0.7858 (L)	2.5484×10^3 (g/km)

4.7.3 Highway FET Test sample Comparing Regression Model and Neural Network Model.

Figure 4.69 shows the Regression Model's prediction result of torque split compared with DPM's prediction.

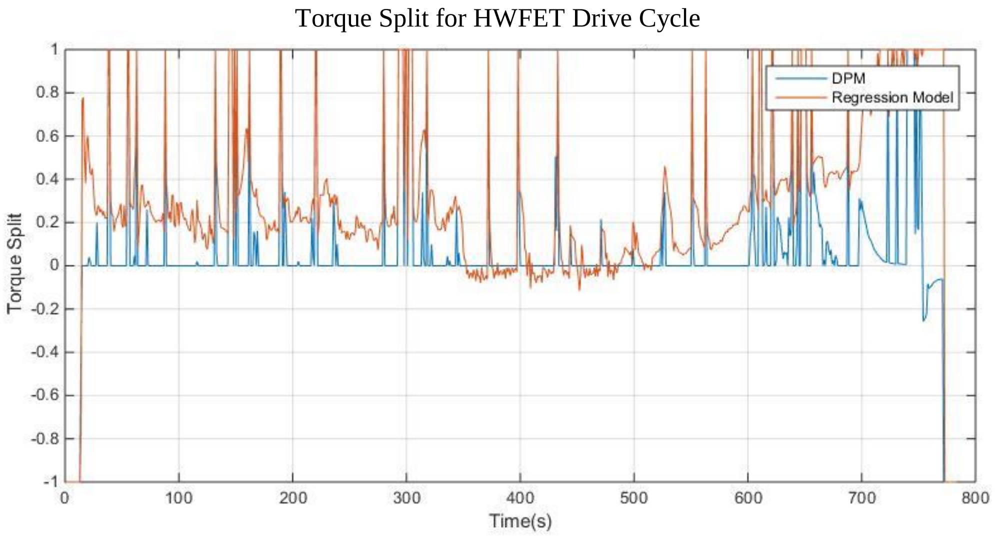


Figure 4.69. Torque Split Comparison for DP and Regression Model in HWFET Drive Cycle

The absolute difference（ $E_{regression}$ ）shows the similarity between them, calculated as：

$$E_{regression} = \frac{1}{n}\sum_{t}\left(\frac{|\tau_{dpm} - \tau_{regrssion}|}{\tau_{dpm}}\right) = 22.82\% \tag{4.34}$$

Figure 4.70 shows the neural networks model’s prediction result of torque split compared with DP Model prediction.

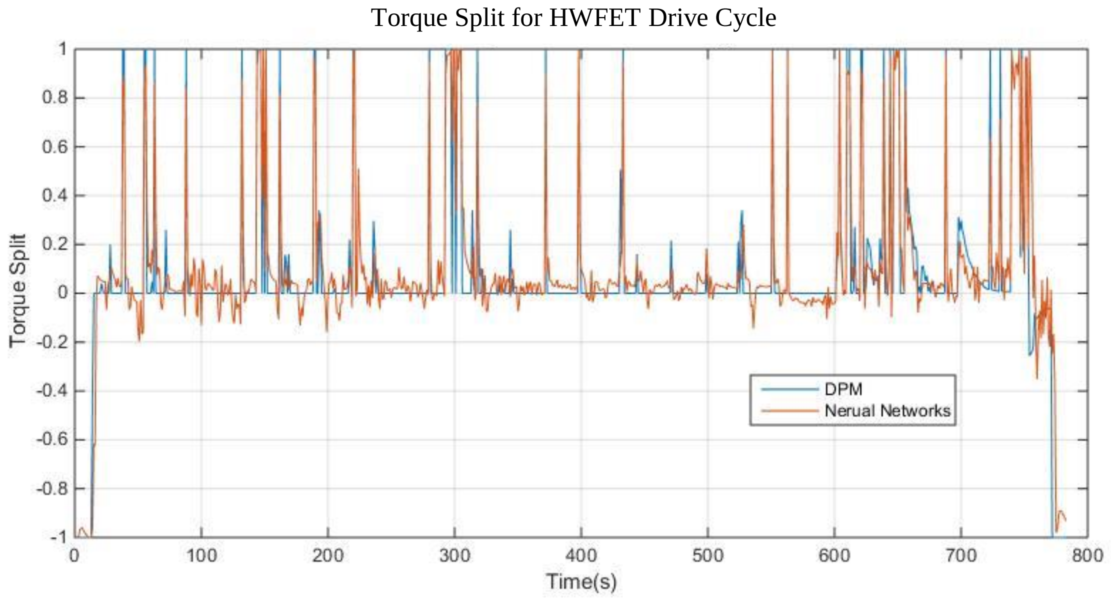


Figure 4.70. torque split comparison for DPM and NNM in HWFET Drive Cycle

The absolute difference（ E_{nn} ）is calculated as：

$$E_{nn} = \frac{1}{n}\sum_{t}\left(\frac{|\tau_{dpm} - \tau_{nn}|}{\tau_{dpm}}\right) = 6.78\% \qquad (4.35)$$

Thus, NNM has less difference compared with the DP result compared with Regression Model.

For further comparison, the difference value of torque split between the Regression Model and DPM as well as the difference value of torque split between NNM and DPM are shown in Figure 4.71.

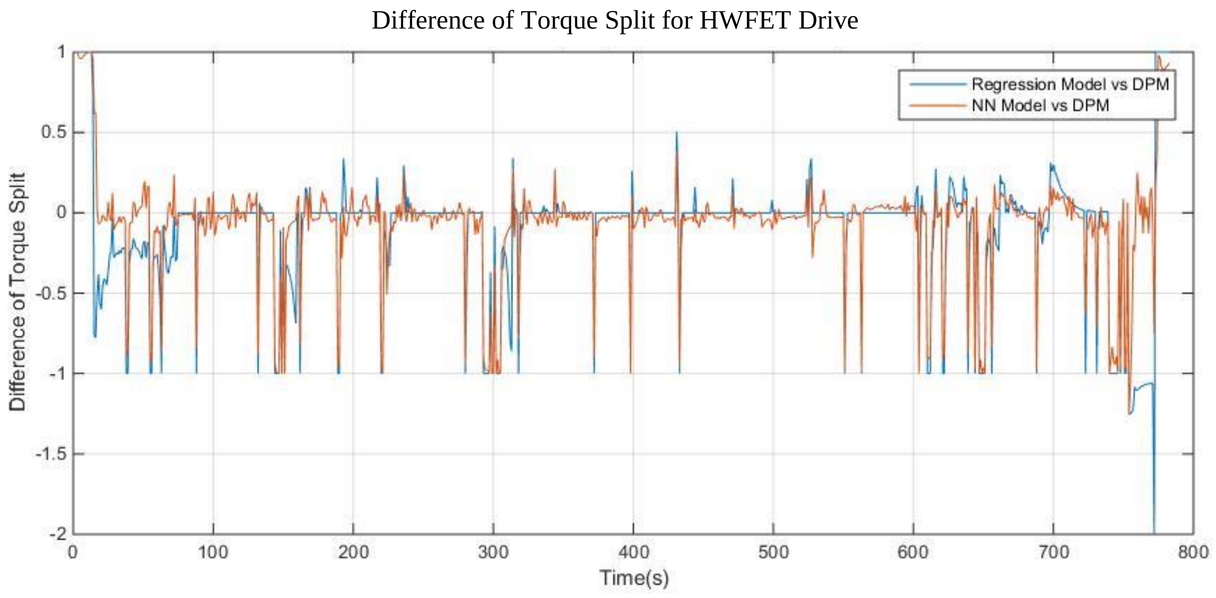


Figure 4.71. The difference value of torque split for Regression Model with DPM and NNM with DPM in HWFET drive cycle

The state of charge and total power required for these two models are also compared with DPM. Figure 4.72 and Figure 4.73 show the total power required comparison between DPM and Regression Model, as well as DPM and NNM.

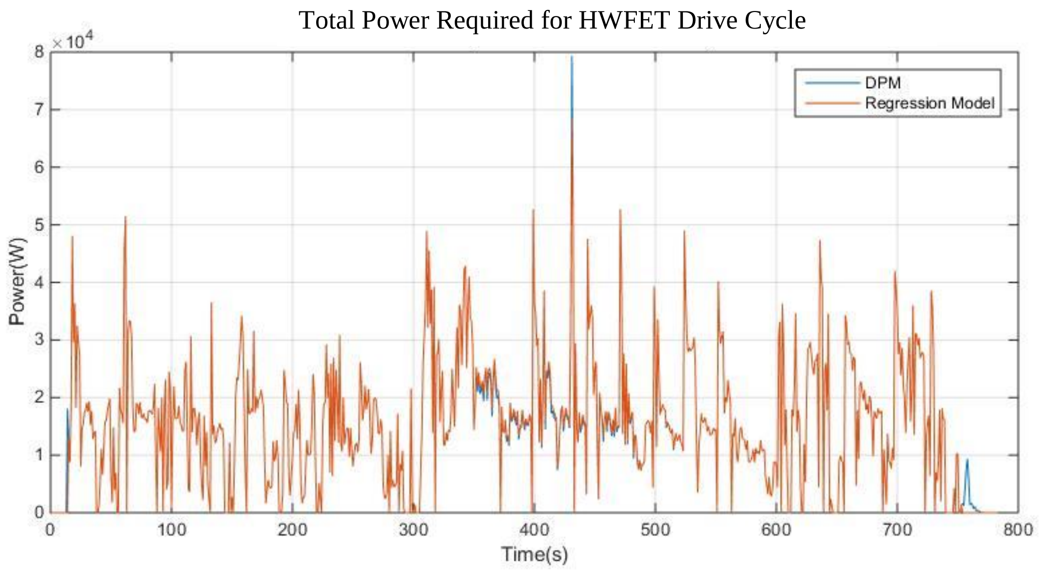


Figure 4.72 Total power required comparison for DPM and regression model in HWFET

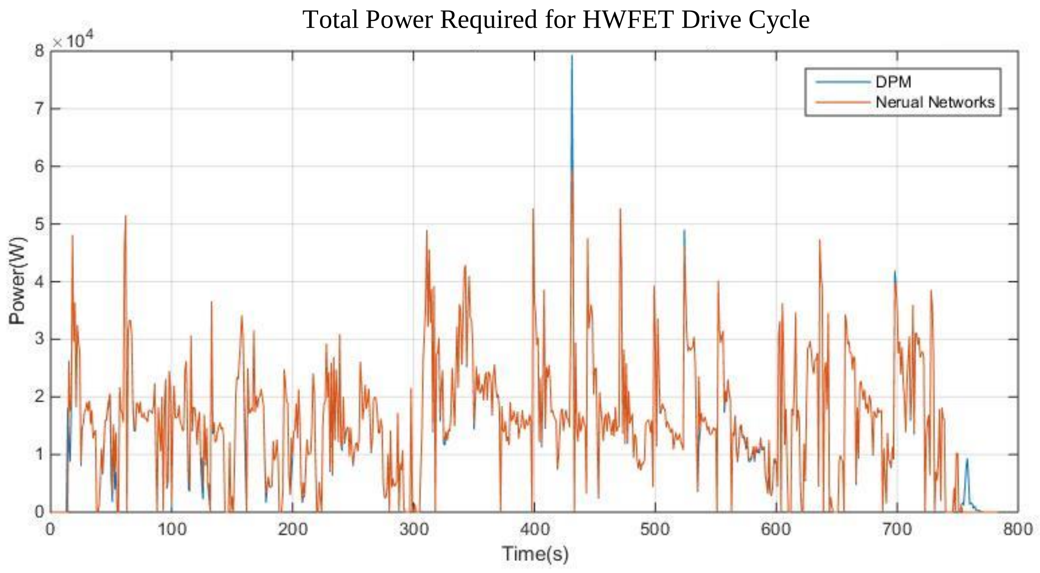


Figure 4.73. Total power required comparison for DPM and NNM in HWFET

For further comparison, since the total power required is the sum of the motor power required and the engine power required, Figure 4.74 shows the motor power required comparison between DPM, Regression Model and NNM. And Figure 4.75 shows the engine power required comparison between DPM, Regression Model and NNM.

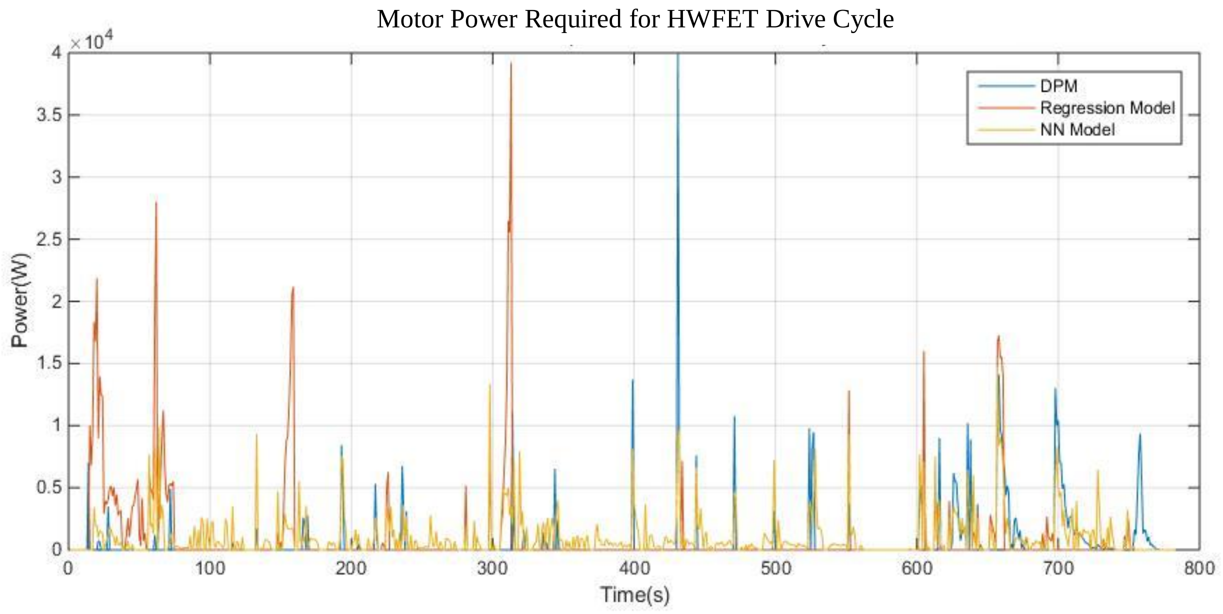


Figure 4.74. Motor power required comparison between DPM, Regression Model and NNM for HWFET Drive Cycle

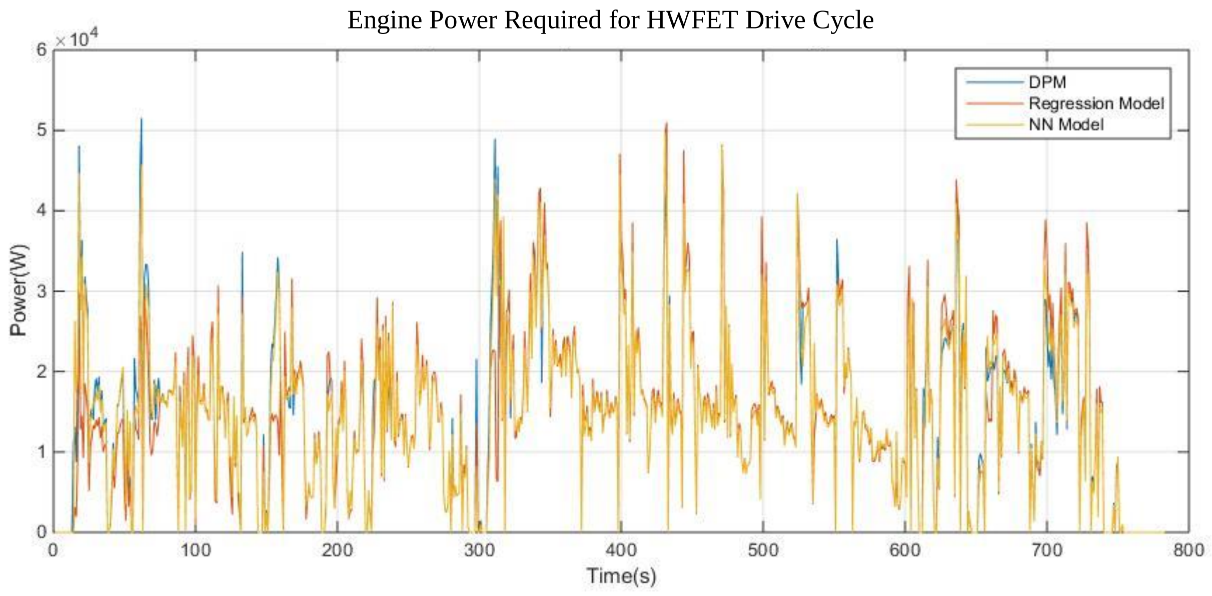


Figure 4.75. Engine power required comparison between DPM, Regression Model and NNM for HWFET Drive Cycle

Table 4.23 shows the engine energy consumption (E_{engine}), motor energy consumption (E_{motor}) and the total energy consumption (E_{total}). The total energy consumption of Regression Model is 1.05% lower than the DPM, and NNM is the 0.51% higher than the DPM. However, NNM keeps more features from DPM and higher final SOC. Therefore, for NNM, more energy is stored in the battery, which explains why NNM has more engine energy consumption and total energy consumption.

Table 4.23. Energy consumption comparison for HWFET drive cycle

	Engine Energy Consumption (E_{engine})	Motor Energy Consumption (E_{motor})	Total Energy Consumption (E_{total})
DPM	1.1636×10^7 (J)	4.9428e×10^5(J)	1.2130×10^7 (J)
Regression	1.1247×10^7 (J)	7.5568×10^5(J)	1.2003×10^7 (J)
Neural Networks Model	1.1411×10^7 (J)	7.8134×10^5(J)	1.2192×10^7(J)

Figure 4.76 and Figure 4.77 show the state of charge comparison between DPM and Regression Model, as well as DPM and NNM. The SOC remains at 0.55. The final SOC of Regression model is slightly lower than DPM, and NNM is slightly higher than DPM. However, NNM keeps more features from DPM.

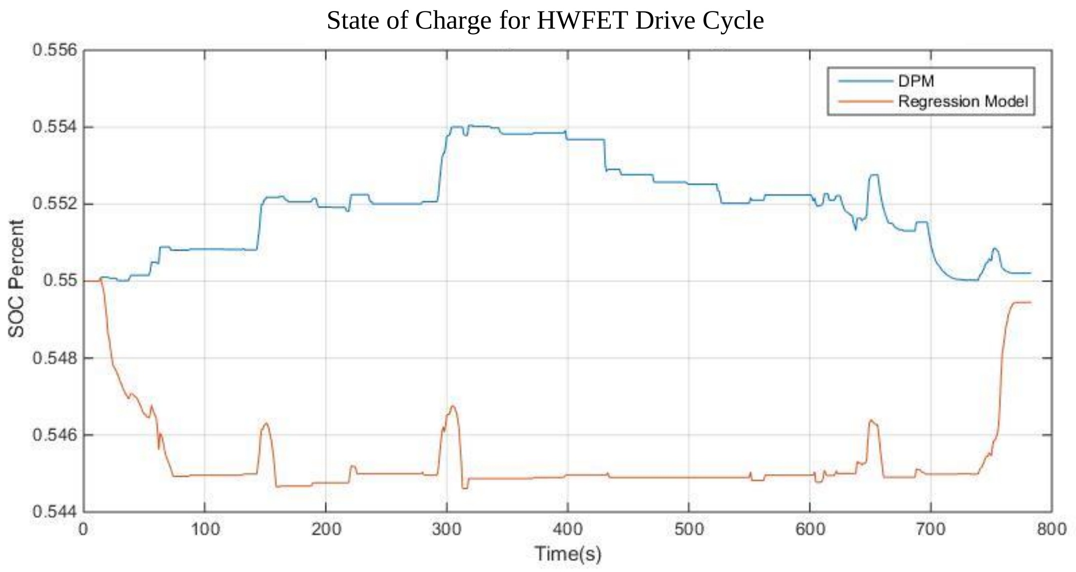


Figure 4.76. State of charge comparison for DPM and Regression model in HWFET

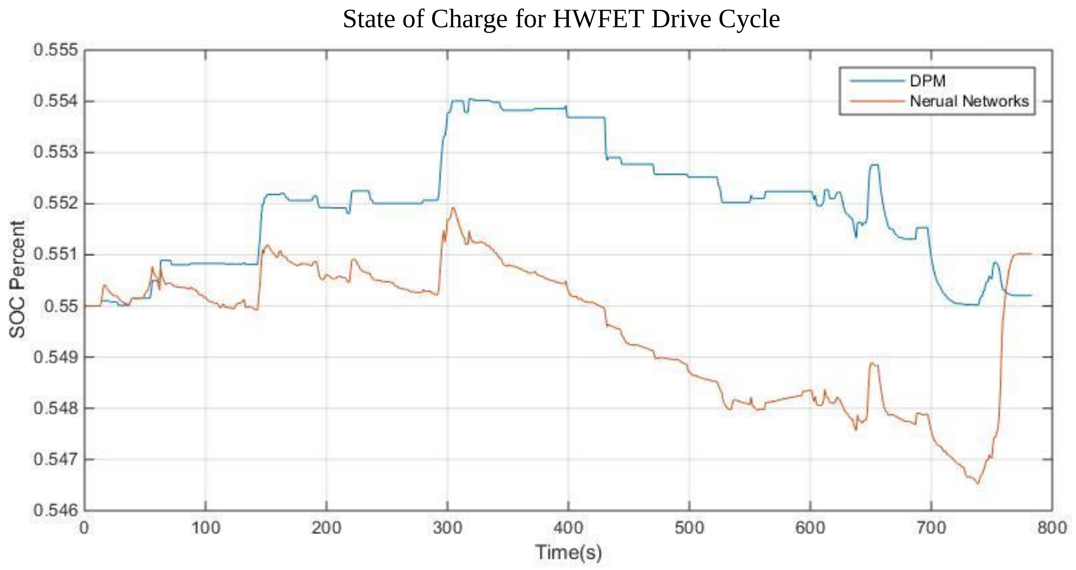


Figure 4.77. State of charge comparison for DPM and NNM in HWFET

In summary, in UDDS drive Cycle, NNM has a significant advantage over the Regression Model (11.84% error compared to 15.71% error). And this advantage is amplified in the HWFET drive cycle (6.78% error compared to 22.82% error). In addition, NNM keeps more features for change of SOC.

The Net fuel consumption comparison for DPM, Regression Model and NNM are shown in Figure 4.78.

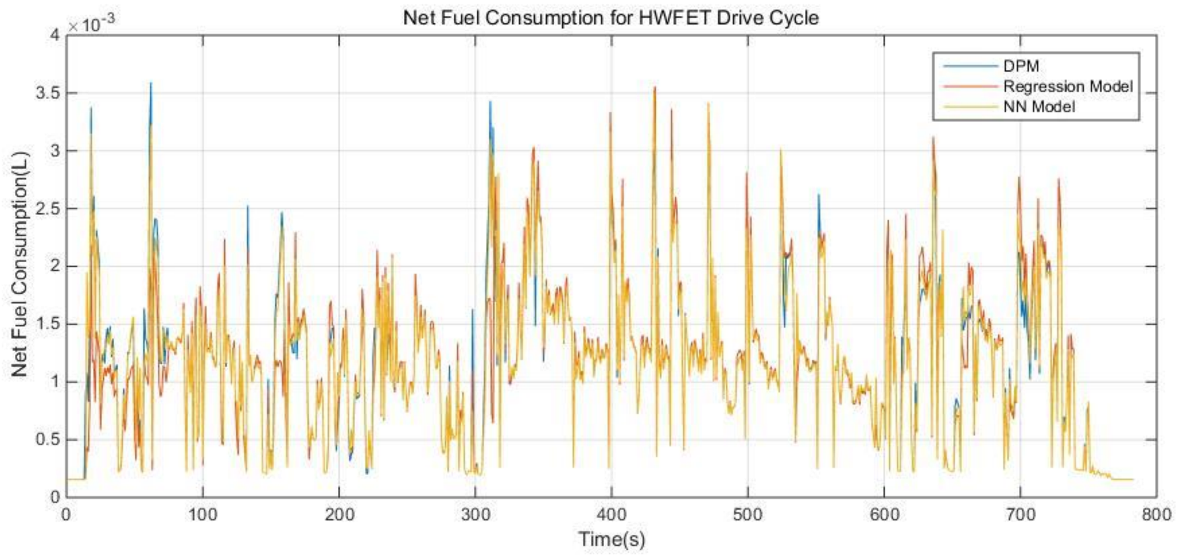


Figure 4.78. Fuel consumption comparison for DPM, Regression Model and NNM for HWFET Drive Cycle.

The GHG_{WTW} emissions for DPM, Regression Model and NNM are shown in Figure 4.79.

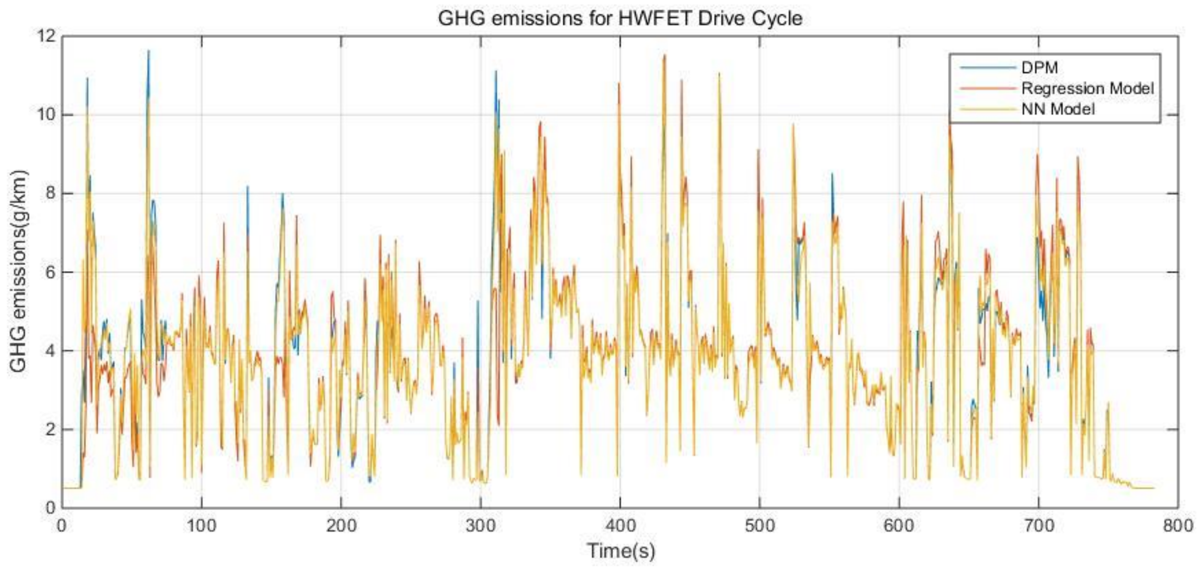


Figure 4.79. GHG_{WTW} emissions comparison for DPM, Regression Model and NNM for HWFET Drive Cycle

The total fuel consumption and GHG_{WTW} emissions during the HWFET drive cycle are shown in Table 4.24. For total fuel consumption, the result for the Regression Model is 1.66% lower than for DPM, and the result for NNM is 1.45% lower than for DPM. For the GHG_{WTW} emissions, the result for Regression Model is 2.47% lower than for DPM, and the result for NNM is 1.23% lower than for DPM.

Table 4.24. Total fuel consumption and GHG_{WTW} emissions during the HWFET drive cycle

	Total Fuel Consumption	Total GHG_{WTW} emissions
DPM	0.9363 (L)	3.0366×10^3 (g/km)
Regression	0.9132 (L)	2.9615×10^3 (g/km)
Neural Networks Model	0.9227 (L)	2.9991×10^3 (g/km)

The total fuel consumption as well as the total energy consumption for the Regression and Neural Network Models are lower than for DPM, which is inconsistent with the expectation that the DP solution is the most energy-efficient. One potential reason for this is the degree to which the output speed matches that of the drive cycle. The output speed profile for DPM is exactly the same as the speed profile in the drive cycle (hence has zero error), however, the output speed profiles for Regression and Neural Network Models have errors compared to the drive cycle.

4.7.4 Comparing Among all five Models

From Chapter 4.2 to Chapter 4.6, five Models and their results are introduced, including the Dynamic Programming Model (DPM), Equivalent Consumption Minimization Strategy (ECMS), Proportional State-of-Charge Algorithm (pSOC), Regression Model, and Neural Network Model (NNM). All their results are compared as follows.

Figure 4.80 shows the total power required comparison among all five models in UDDS drive cycle.

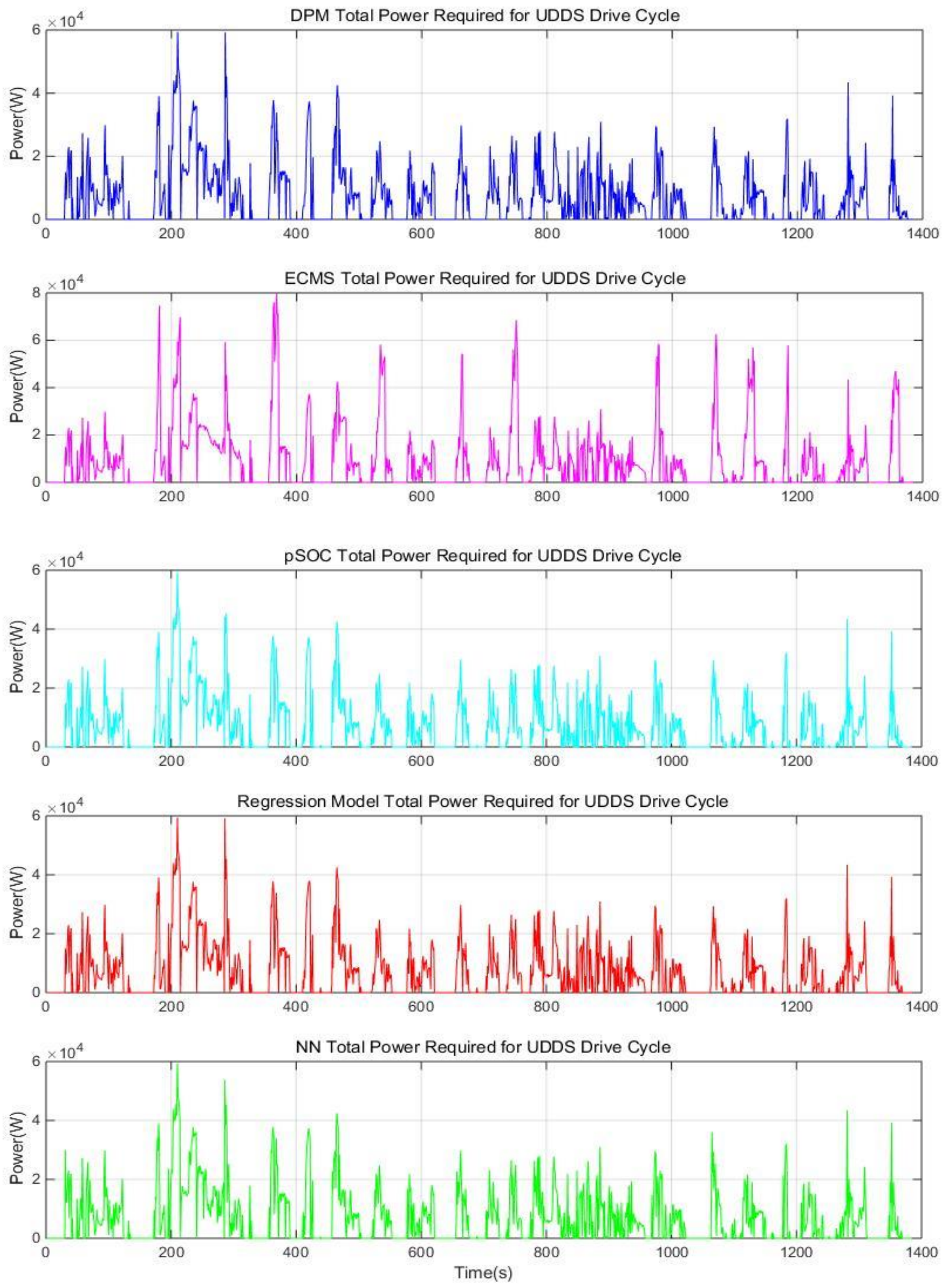


Figure 4.80. Total power required comparison for all five models in UDDS

Figure 4.80. continued

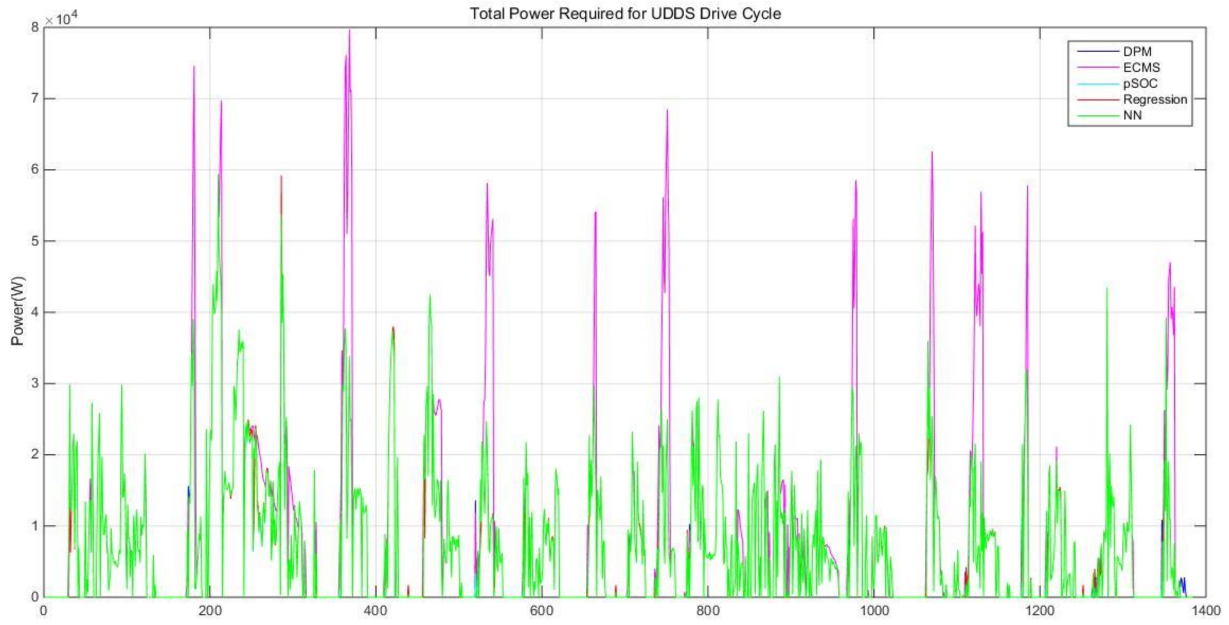


Figure 4.81 shows the total power required comparison among all five models in HWFET drive cycle.

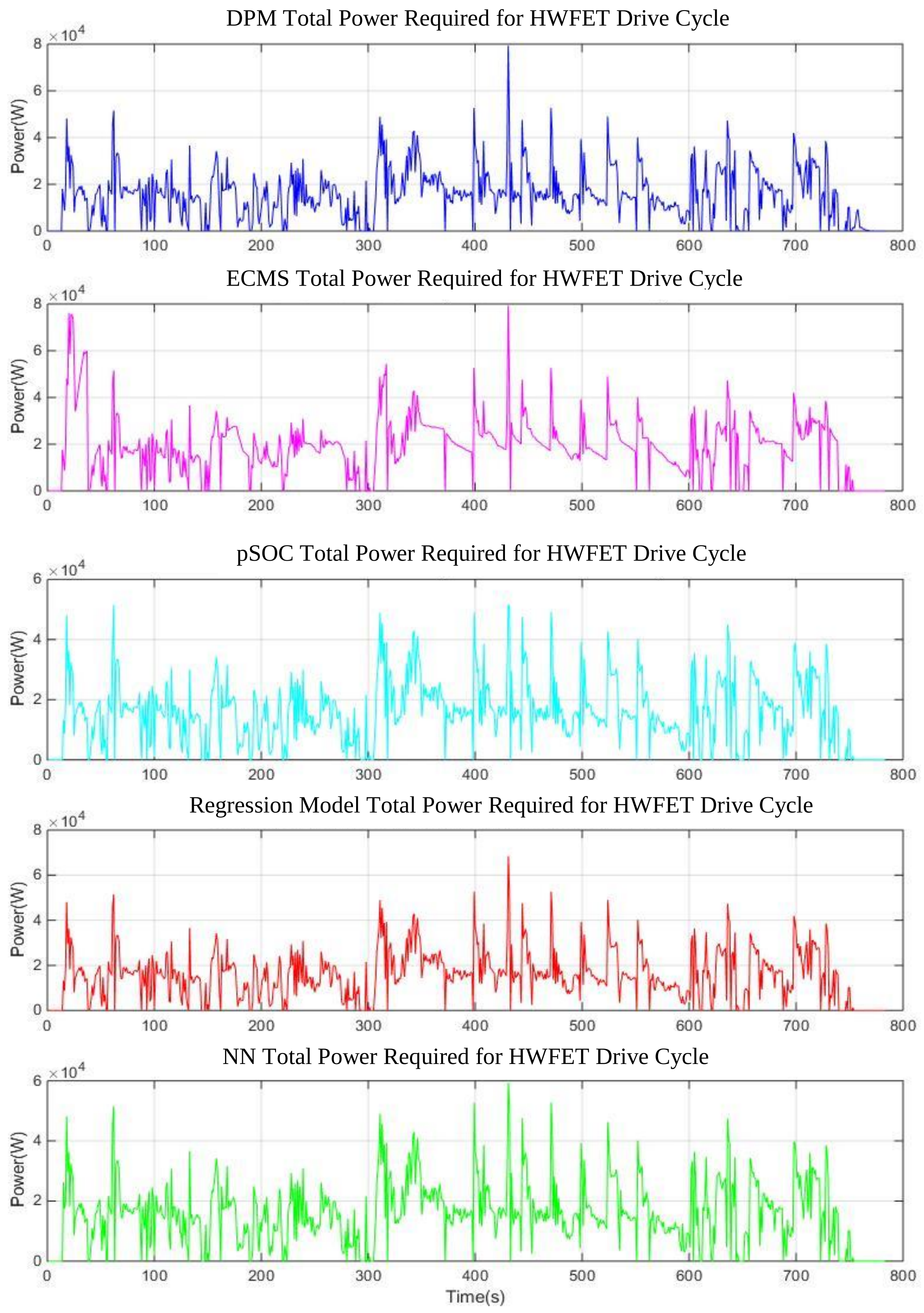


Figure 4.81. Total power required comparison for all five models in HWFET

Figure 4.81. Continued

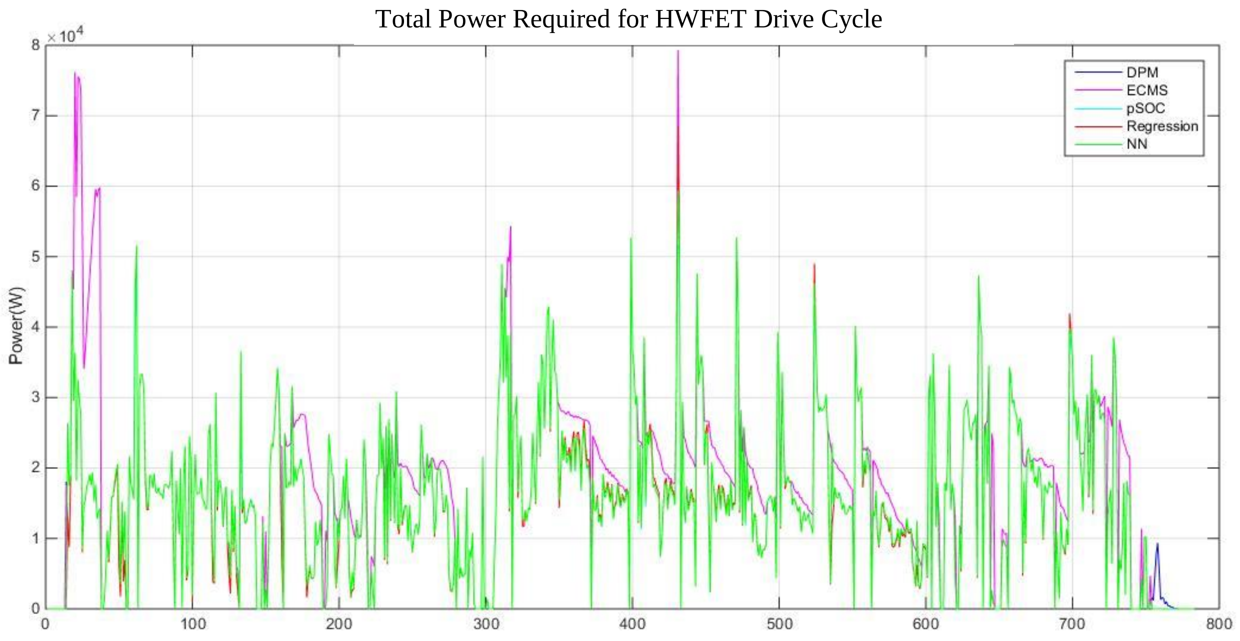


Table 4.25 shows the total energy consumption (E_{total}) among all five models in both UDDS and HWFET drive cycles.

Table 4.25. Total energy consumption comparison

	Total Energy Consumption (E_{total}) UDDS	Total Energy Consumption (E_{total}) HWFET
DPM	9.6082×10^6 (J)	1.2130×10^7 (J)
ECMS	1.3238×10^7 (J)	1.4852×10^7 (J)
pSOC	9.7474×10^6 (J)	1.2204×10^7 (J)
Regression	9.6828×10^6 (J)	1.2003×10^7 (J)
NNM	9.7282×10^6 (J)	1.2192×10^7(J)

DPM has the lowest Energy Consumption, Regression and NNM are both very similar to DPM and better than others. ECMS has significantly higher energy consumption than others.

Figures 4.82 and 4.83 show the state of charge comparison among all five models in UDDS drive cycle and HWFET Drive Cycle.

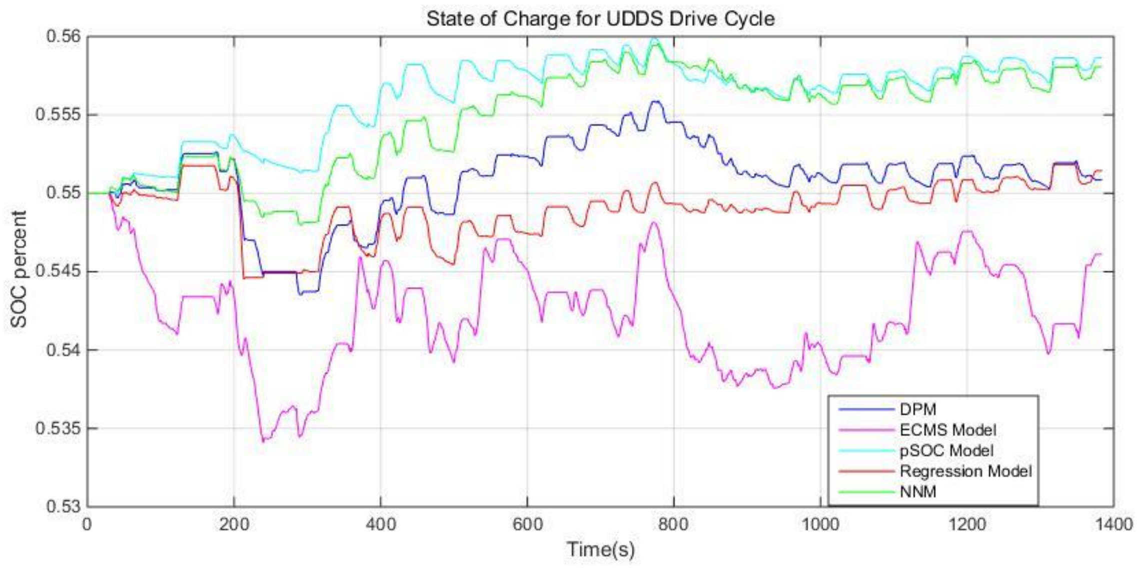


Figure 4.82. State of charge comparison among all five models in UDDS drive cycle

State of Charge for HWFET Drive

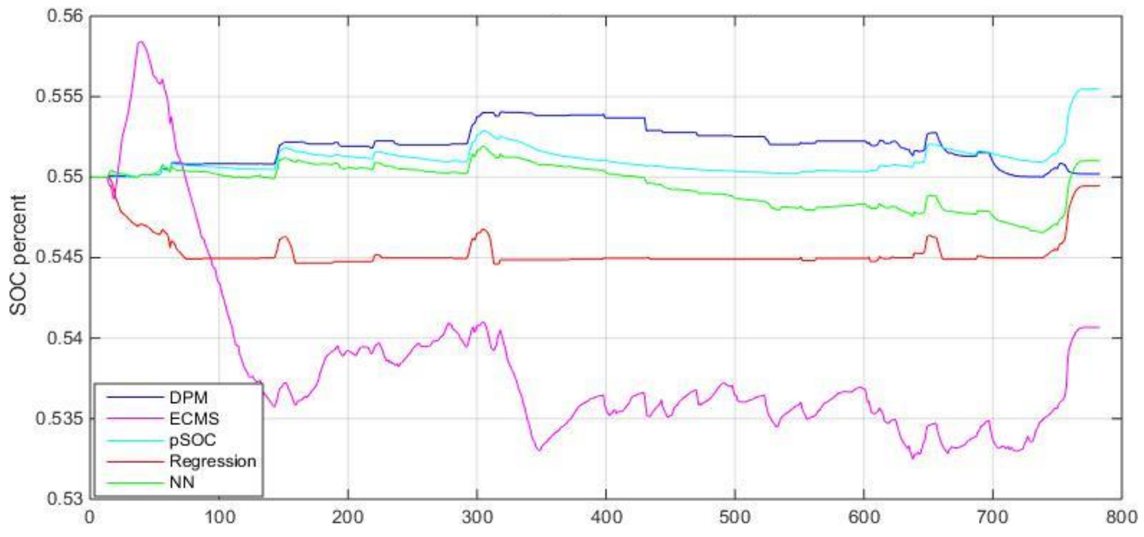


Figure 4.83. State of charge comparison among all five models in HWFET drive cycle

In both drive cycles, pSOC has the highest final SOC, NN is higher than DPM and Regression is lower than DPM. PSOC and NN keep most features of DPM. ECMS has the lowest final SOC.

5. CONCLUSIONS AND FUTURE WORK

5.1 Key contributions

Comparing to the standard driving cycle tests, such as Urban Dynamometer Driving Schedule and Highway Fuel Economy Test, the overall vehicle model can fairly accurately predict the real-world driving range. In consequence, it demonstrates that there is a way to study the behavior of a vehicle without any physical testing. Within these error bounds, the vehicle model can be utilized to work on the supervisory control of the vehicle and is expected to predict the characteristics of the vehicle in the real world.

All the power management strategies are developed based on simplified models. Even though each strategy has its own constraints, there are many advantages, which will be described in the following.

5.1.1 Dynamic Programming Strategy (DPM)

For a given drive cycle, the optimal results are able to be generated by implementing a dynamic programming strategy. However, this strategy requires a backward-facing mode. Thus, it cannot be implemented for real-time calculation. Therefore it is utilized as a benchmark for other control strategies.

5.1.2 Equivalent Consumption Minimization Strategy (ECMS)

The result obtained through equivalent consumption minimization strategy shows that the energy consumption is suboptimal. Since this strategy is a real-time implementable power management strategy, if this method is carefully tuned for a particular drive cycle, it can be used as an alternate strategy.

5.1.3 Proportional State-of-Charge Algorithm (pSOC)

The proportional state-of-charge algorithm power management strategy is a sub-optimal power management strategy, but it still can provide benefits in terms of energy consumption compared to ECMS and ease of implementation.

5.1.4 Regression Model

Regression is also a sub-optimal power management strategy. It can provide an overall good result which is better than ECMS and pSOC. The regression model can be used to predict the dynamic programming trends, and the outputs are similar to the results from DPM.

5.1.5 Artificial Neural Networks Modeling (NNM)

Artificial neural networks algorithm has the best results compared to ECMS, pSOC and Regression. DPM is used as the benchmark for both NNM and regression model. The absolute difference for UDDS drive cycle is 11.84% between DPM and NNM compared to 15.71% between DPM and regression model, and for HWFET drive cycle is 6.78% between DPM and NNM compared to 22.82% between DPM and regression model. Thus, NNM has significant advantages compared to the regression model. The disadvantage of NNM is the complexity of calculations.

NNM is actually a general implementable substitution of DPM or any other algorithms that cannot be implemented in real life. For example, if there is an algorithm even better than DPM, but it needs to be tuned manually for different drive cycle, NNM can be used to imitate this algorithm and get a similar result. Furthermore, with few parameters changed, NNM can also be used for other vehicle platforms including both hybrid vehicle or electric vehicle, especially electric vehicle with multiple motors.

NNM can also imitate the performance and experience of real drivers. If a vehicle can be driven by an experienced driver who is able to perform the best combination of energy consumption and comfort of passengers for this vehicle over several days or dozens of drive cycles, NNM can be used to achieve a solution with similar performance to that of the driver for real-world driving.

5.2 Future work

- The current NNM is developed in PyTorch platform. It cannot be interfaced with the current hardware-in-loop simulation setup which is dSPACE MicroAutoBox. Also, the dSPACE MicroAutoBox does not have enough computing power and memory space for NNM. However, a vehicle which is equipped with internet access, which is also called connected vehicle, is very common in the market. Therefore, we can either improve the local hardware controller or connect it with internet and using cloud computing to solve this problem. And the best solution is to combine both of them.

- The simplified model needs to be improved by including an automatic transmission model. And more details need to be included such as a motor efficiency map and friction brake model.

- Only thirty-six drive cycles are used. The accuracy of NNM highly depends on the quantities of training set. To improve the accuracy, more drive cycles should be collected.

- Autonomous hybrid or electric vehicles are going to replace the traditional vehicles in several decades. NNM is more valuable and suitable to implement on the autonomous driving control strategy. It is not difficult to achieve a good energy consumption solution by using rule-based computational algorithm. However, it is not easy to compute when other factors besides energy consumption are important as well, for example, when the comfort of the passengers also needs to be considered. Since NNM can imitate the performance and experience of the real drivers, it is a general solution to achieve self-driving vehicle control strategy if there are some experienced drivers that can test the vehicle on real roads.

REFERENCES

[1] U.S. Energy Information Administration, *Monthly Energy Review*, Table 2.1, April 2018

[2] United States Environmental Protection Agency, Inventory of U.S. Greenhouse Gas Emissions and Sinks: 1990-2016, 2016

[3] The Statistics Portal, https://www.statista.com/statistics/183505/number-of-vehicles-in-the-united-states-since-1990/, 2018

[4] U.S. Energy Information Administration, Greenhouse gas emissions, 2017

[5] National Oceanic and Atmospheric Administration's National Climatic Data Center, https://urbanmilwaukee.com/, 2015.

[6] Hannah Ritchie and Max Roser. "CO_2 and other Greenhouse Gas Emissions". https://ourworldindata.org/co2-and-other-greenhouse-gas-emissions, 2018

[7] Rohinish Gupta, "Modeling and control of a parallel through-the-road plug-in hybrid vehicle", Unpublished master's thesis, Purdue University, West Lafayette, Indiana, 2015.

[8] K.T. Chau, Y.S. Wong, *Energy Conversion and Management* 43 (2002) 1953–1968

[9] C. Musardo, G. Rizzoni, and B. Staccia, "A-ECMS: an adaptive algorithm for hybrid electric vehicle energy management" in Proc. 44th *IEEE Conf. Decision Control*, 2005 European Control Conf., Seville, Spain, 2005, pp. 1816–1823

[10] Y. Zhu, Y. Chen, G. Tian, H. Wu, and Q. Chen, "A four-step method to design an energy management strategy for hybrid vehicles," in *American Control Conf.*, Boston, MA, pp. 156–161, 2004.

[11] Peng H. Grizzle J. W. Lin, C.-C. and J.-M. Kang. "Power management strategy for a parallel hybrid electric truck." In *IEEE Transactions on Vehicle Technologies*, 2003.

[12] Ambhl D. Sundstrm, O. and L. Guzzella. “On implementation of dynamic programming for optimal control problems with state constraints.” In *Oil Gas Science Technology*, 2009.

[13] L. Pontryagin, V. Boltyanskii, R. Gamkrelidze, and E. Mishchenko, “The mathematical theory of optimal processes.” New York: *Interscience Publishers*, 1962.

[14] D. Bertsekas, “Dynamic Programming and Optimal Control.” Belmont, MA: *Athena Scientific*, 1995.

[15] H. P. Geering, “Optimal Control with Engineering Applications.” Berlin Heidelberg: *Springer*, 2007.

[16] G. Paganelli, T. Guerra, S. Delprat, J. Santin, M. Delhom, and E. Combes, “Simulation and assessment of power control strategies for a parallel hybrid car,” *Journal of Automobile Engineering*, vol. 214, no. 7, pp. 705–717, 2000.

[17] G. Paganelli, G. Ercole, A. Brahma, Y. Guezennec, and G. Rizzoni, “General supervisory control policy for the energy optimization of charge-sustaining hybrid electric vehicles,” *JSAE Review*, vol. 22, no. 4, pp. 511–518, 2001.

[18] P. Pisu and G. Rizzoni, “A comparative study of supervisory control strategies for hybrid electric vehicles,” Control Systems Technology, *IEEE Transactions* on, vol. 15, no. 3, pp. 506–518, 2007.

[19] B. Gu and G. Rizzoni, “An adaptive algorithm for hybrid electric vehicle energy management based on driving pattern recognition,” *ASME International Mechanical Engineering Congress and Exposition*, 2006.

[20] C. Musardo, G. Rizzoni, Y. Guezennec, and B. Staccia, “A-ECMS: An adaptive algorithm for hybrid electric vehicle energy management,” *European Journal of Control*, vol. 11, no. 4-5, pp. 509–524, 2005.

[21] Schouten N. Salman, M. and N. Kheir. “Control strategies for parallel hybrid vehicles.” In *Proceedings of the 2000 American Control Conference*, 2000.

[22] Parten M. He, X. and T. Maxwell. “Energy management strategies for a hybrid electric vehicle.” In *Proceedings of the 2005 IEEE Vehicle Power and Propulsion Conference*, 2005.

[23] J. Xu Z. Wei and D. Halim. “Hev energy management fuzzy logic control based on dynamic programming.” In *IEEE, editor, Vehicle Power and Propulsion Conference*, 2015.

[24] Stuart Dreyfus. "Richard Bellman on the birth of Dynamical Programming", *Operations Research*, Vol. 50, No. 1, January–February 2002, pp. 48–51

[25] Bellman, R. Dynamic Programming. Princeton, NJ: Princeton University Press. 1957.

[26] Sepp Hochreiter, “Recurrent Neural Net Learning and Vanishing Gradient”, *International Journal of Uncertainty*, Fuzziness and Knowledge-Based Systems 6(2):107-116, 1998.

[27] Yoshua Bengio, Patrice Simard, and Paolo Frasconi, “Learning Long-Term Dependencies with Gradient Descent is Dificult”, *IEEE Transactions on Neural Networks,* Vol. 5, No.2 2, March 1994.

[28] Sepp Hochreiter and Jürgen Schmidhuber, “Long Short-Term Memory”, *Neural Computation*, Volume 9, Issue 8, November 15, 1997 p.1735-1780.

[29] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, “Speech Recognition with Deep Recurrent Neural Networks”. *2013 IEEE International Conference on, pages 6645–6649. IEEE, 2013.*

[30] Dspace MicroAutoBox II https://www.dspace.com/en/inc/home/products/hw/micautob/ microautobox2.cfm, 2018

[31] “CAN Bus Explained - A Simple Intro” https://www.csselectronics.com/screen/page/simple-intro-to-can-bus/language/en, 2018

[32] “Threat Modelling for Automotive” https://radiojitter.wordpress.com/2018/04/18/threat-modelling-for-automotive-part-2, 2018

[33] O. Sundstrom and L. Guzzella. A generic dynamic programming matlab function. In *2009 IEEE Control Applications*, (CCA), St.Petersburg, Russia, pages 1625-1630, 2009.

[34] “3 Types of Gradient Descent Algorithms for Small & Large Data Sets” https://www.hackerearth.com/blog/machine-learning/3-types-gradient-descent-algorithms-small-large-data-sets, 2017

APPENDIX A. DETAILS OF DRIVE CYCLES USED FOR NEURAL NETWORK MODEL

List of drive cycles used to the train and validate dataset for neural network model.

Data File	Drive Cycle
City 1	Air Resource Board Drive Cycle No. 2
City 2	Assessment and Reliability of Transport Emission Models and Inventory Systems (ARTEMIS) Drive Cycle
City 3	ARTEMIS Extra Urban
City 4	ARTEMIS Urban
City 5	The Central Business District Cycle (included 14 Repetitions)
City 6	Combined International Local and Commuter Cycle
City 7	Extra Urban Drive Cycle HYRROUT
City 8	Urban Drive Cycle HYZROUT
City 9	City Suburban Heavy Vehicle Route
City 10	Composite Urban Emissions Drive Cycle
City 11	Composite Urban Emissions Drive Cycle-Arterial
City 12	Composite Urban Emissions Drive Cycle-Congested
City 13	Composite Urban Emissions Drive Cycle-Residential
City 14	Economic Commission of Europe Drive Cycle
City 15	EPA LA92
City 16	Urban Dynamometer Driving Schedule (Cold-Start, 505secs)
City 17	Heavy-Heavy-Duty Diesel Truck Transient
City 18	Hybrid Truck Users Forum Class 4Parcel Delivery Cycle
City 19	Hybrid Truck Users Forum Refuse Truck cycle
City 20	India Urban Drive Sample
City 21	INRETS Urban
City 22	INRETS Urban1
City 23	INRETS Urban3
City 24	INRETS Road
City 25	INRETS Road1
City 26	INRETS Road2
City 27	Japan 10 Mode Drive Cycle
City 28	Japan 15 Mode Drive Cycle

City 29	Japan 1015 Mode Drive Cycle
City 30	Japanese JC08 Cycle
City 31	Nuremberg R36 City Bus Drive Cycle
City 32	New York City Cycle
City 33	New York Garbage Truck Cycle
City 34	US EPA Air Conditioning Drive Cycle (SC03)
City 35	West Virginia University City Drive Cycle
City 36	West Virginia University Suburban Driving Cycle