AN ABSTRACT OF A THESIS

OPTIMAL CONTROL OF A MILD HYBRID ELECTRIC VEHICLE USING WEIGHT FUNCTIONS AND GENETIC ALGORITHMS

Jeffery L. McGehee Jr.

Master of Science in Mechanical Engineering

As a viable alternative to the conventional hybrid electric vehicles, so called "mild" hybrid drivetrains are currently being implemented in production vehicles. These mild hybrid electric vehicles use an Integrated Starter Generator (ISG) to simply assist the internal combustion (IC) engine rather than drive the vehicle independently of the IC engine. Some production mild hybrid vehicles have been shown to achieve over a10 % increase in fuel efficiency with minimal additional costs compared to a conventional vehicle. In this thesis, lookup table-based control schemes for the optimal control of the ISG and the IC engine on a mild hybrid vehicle are presented. The lookup table is optimized using two different methods: a traditional weighted cost function approach and a genetic algorithm optimization approach. The developed control logic is implemented in Matlab/Simulink along with a mild hybrid vehicle model, which is based on the EPA's light-duty vehicle model.

The simulation results show that the ISG control lookup table constructed by using a weighted cost function yields better performance than a standard rule-based control strategy. In addition, the lookup table is easily implementable for real-time control on the control hardware in current production vehicles. However, it was suspected that the optimization of the ISG control lookup table for an MHEV may require

a method designed to handle more complex nonlinear problems than can be optimized by a weighted cost function. Thus, a widely-used nonlinear global optimization technique called genetic algorithm was utilized for further optimization of the ISG control lookup table. The genetic algorithm forms an optimal table with better performance characteristics than the weighted cost function method, which suggests that genetic algorithm optimization is a good choice for the optimization of HEV lookup tables.

OPTIMAL CONTROL OF A MILD HYBRID ELECTRIC VEHICLE USING WEIGHT FUNCTIONS AND GENETIC ALGORITHMS

A Thesis

Presented to

The Faculty of the Graduate School

Tennessee Tech University

by

Jeffery L. McGehee Jr.

In Partial Fulfillment

Of the Requirements for the Degree

MASTER OF SCIENCE

Mechanical Engineering

May 2013

CERTIFICATE OF APPROVAL OF THESIS

OPTIMAL CONTROL OF A MILD HYBRID ELECTRIC VEHICLE USING WEIGHT FUNCTIONS AND GENETIC ALGORITHMS

by

Jeffery L. McGehee Jr.

Graduate Advisory Committee:

Sally Pardue, Chairperson	Date
Hwan-Sik Yoon	Date
Stephen Canfield	Date

Approved for the Faculty:

Francis Otuonye
Associate Vice President
for Research and Graduate Studies

Date

DEDICATION

This thesis is dedicated to my parents who raised

me in a way that made me get the job done.

ACKNOWLEDGEMENTS

I would like to thank Dr. Yoon who worked to make sure that my research and writing met the standards of the automotive industry. I would also like to thank Dr. Pardue for pushing me to complete this thesis on time and for encouraging me to push my limits academically. To Mike Renfro and Dr. Talbert, thank you for the guidance you gave me regarding genetic algorithms and for helping me reach my computational goals for this thesis.

My roommate Tim Nummy was a great help in the brainstorming processes throughout this project, and helped me especially in the formulation of the successful genetic algorithm in Chapter 5. Also, my fiancé Sarah Beth Hawkins has been extremely tolerant of my endless chatter regarding my research and has accepted the time burden associated with my work by supporting all that I have done towards the completion of this thesis.

Lastly, I would like to thank myself for the hard work and dedication that I have shown in the completion of this thesis. Nobody has come close to putting forth the amount of work towards this research as I have. I have denied myself many instantaneous rewards in pursuit of the delayed gratification that the completion of this work has given me, and I have put forth my best effort in the derivation of original and valid methods presented in this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Due to the growing concern about the environmental impact of automobiles as well as the rising cost of petroleum-based fuels, numerous new vehicle technologies are being actively researched. One of these technologies is the hybrid electric vehicle. By combining two different power sources with different strengths (IC engine and electric motor), it is possible to obtain a much more desirable overall vehicle efficiency [1]. This makes hybrid electric vehicles a very effective method for reducing the environmental impacts that arise with the combustion of fossil fuels.

Hybrid electric vehicles can be classified by their "degree" of hybridization, that is, their level of ability to use each power source independently [2]. This creates two basic classes: full hybrids and mild hybrids. A full Hybrid Electric Vehicle (HEV) is able to drive the wheels with the electric motor only, the IC engine only, or a combination of both. A Mild Hybrid Electric Vehicle (MHEV) is equipped with electric hardware similar to a full hybrid, but this hardware is not designed to drive the wheels on its own. Instead, the hardware is used to provide an assistive torque source that allows the engine to be turned off and restarted quickly while the vehicle is braking or stopped. It is also possible to use the electric drive components to assist the IC engine when it is not in its most efficient operating mode. This mild hybrid architecture is the focus of this thesis.

A conventional vehicle powertrain has two electric machines in the form of a starter and a generator/alternator. The starter converts electric energy into mechanical energy which is used to initiate the rotation of the ICE crankshaft. The generator takes mechanical energy from the crankshaft once the engine is running, and converts it to electrical energy to charge the vehicle's battery system. A mild hybrid powertrain is equipped with an electric machine known as an Integrated Starter Generator (ISG). The ISG is an electromechanical component that performs the duties of both the starter and the alternator, but has a larger power rating (for both mechanical and electrical power) than typical starters and alternators. This larger power output allows the ISG to use electrical power from an energy storage device to produce mechanical power that will not only start the ICE, but can also assist the ICE through a range of torques. The ISG can also draw mechanical power from the powertrain and convert it to electrical energy for the vehicle's energy storage device.

In MHEVs, there are three main locations in which the ISG can be placed. . The C-ISG is mounted to the crankshaft flywheel similar to a typical starter, the D-ISG is mounted to the drivetrain between the clutch and the transmission, and the B-ISG is mounted as belt driven engine accessory similar to a conventional alternator [2].

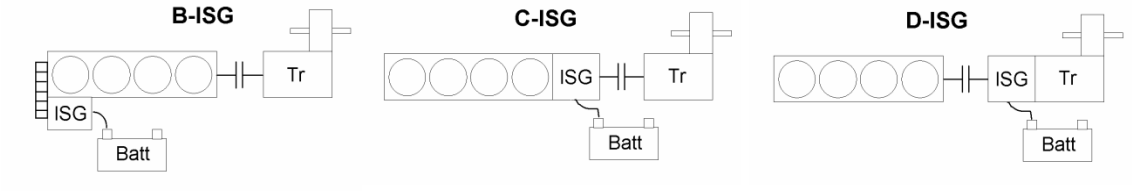


Figure 1.1 MHEV Architectures [2]

In the early days of the mild hybrid vehicle development, the electric components were not used to their full potential with respect to energy efficiency. In fact, they were only used to allow the IC engine to be turned off during braking and stop and go traffic situations. However, in more recent times researchers have been finding methods to utilize the electric hardware in ways that will allow for optimum energy efficiency. This involves the derivation of sophisticated control methods which requires an advanced understanding of how the electric components and IC engine of the vehicle interact. Research of these control methods is a relatively new field, but several control techniques for this problem have already been established.

The focus of this research is *supervisory control*. The goal of this control method is to provide lower level controllers with reference signals regarding the desired state of the ISG and ICE. Controlling and designing power electronics for the lower level systems is still a very viable research area, but it is outside the scope of this work [3].

Upon review of the methods being researched in this field, a new lookup table-based control method for the optimal control of the ISG and the IC engine on a mild hybrid vehicle is formulated. The lookup table is optimized using two different methods: a traditional weighted cost function approach and a genetic algorithm optimization approach. The developed control logic is implemented in Matlab/Simulink along with a mild hybrid vehicle model, which is based on the EPA's light-duty vehicle model.

CHAPTER 2

LITERATURE REVIEW

From a controls perspective, a hybrid electric vehicle is a very complicated engineering system. The multiple power sources, different efficiencies of various components, unknown drive cycle, and unpredictable driver behavior are among some of the characteristics that complicate the control of the system. As mentioned previously, the focus of this research (like that of many other researchers) is not only to control the MHEV, but also to control it in an optimal manner. This complicates the problem further, which has led to the creation of a wide variety of optimal control strategies. These strategies range from simple heuristic methods [4, 5] to mathematically complex optimization methods [2, 6, 7, 8, 9]. While the highly mathematical methods can achieve "near-perfect" optimality, they can be too computationally intensive to be implemented in a real-time controller. The simpler methods are more appropriate to be implemented in real time, but may fall short of the optimality achievements of their more complicated counterparts. The purpose of this literature review is to examine the various methods being researched for optimal HEV control and determine which will serve as an ideal approach for a new control scheme. Because the focus of this research is to develop an optimal control strategy that is implementable in real time, the existing HEV control methods will be classified into two main categories: non-real time and real time.

2.1 Non-Real Time Strategies

Non-real time strategies are those that cannot be implemented for real time vehicle control. This is because they have one or both of the following characteristics. First, the control/optimization methods require intense calculations that current automotive processing hardware is unable to perform in real time. Second, the strategies require knowledge of future driver actions to obtain an optimal solution. While these strategies lack the ability to be implemented in a vehicle, some of the methods used are able to obtain true mathematical optimality for the system [2, 6, 7, 8, 9]. Because of this advantage, these methods were studied in depth to examine if any of them could be adapted for real time use.

2.1.1 Dynamic Programming

The dynamic programming method is founded on a simple concept known as the "principle of optimality". The principle of optimality states that the optimal solution of a large problem is the sum of the optimal solutions of sub-problems within the original problem [10]. For example, if a person wished to travel from point A to points D through points B and C, the shortest distance one can travel is the sum of the shortest distances from A to B, B to C, and C to D. This method allows for true mathematical optimality to

be achieved (with respect to a given cost function), but can only be solved with knowledge of the entire path.

In the case of hybrid vehicle control, it is possible to determine the optimal path through the state-space of the vehicle for a certain drive cycle by finding the optimal path for discretized time steps within the drive cycle. This has been done by many researchers, and the results are generally accepted as "*the optimal solution*" for a given drive cycle [10]. However, a real-time implementation of dynamic programming over a general drive cycle is impossible due to its requirement on the knowledge of future drive cycle.

Rousseau et al. used dynamic programming to optimize various battery and electric motor sizes in a natural gas ICE mild hybrid vehicle [6]. This problem formed a good application for dynamic programming over a drive cycle, which ensures that each set of components is utilized in an optimal manner. The largest improvement in fuel economy achieved in this research was approximately 15%, but this was compared to the natural gas ICE, which is more efficient than typical gasoline powered ICE [6].

Lin et al. developed a dynamic programming method to optimize both the ICE/EM power split as well as the drivetrain gear shifting for fuel economy and emission reduction [8]. With this method, the authors were able to achieve an improvement of 21.54% over the rule-based control with the following performance measure equation:

$$Cost = fuel(\frac{g}{mi}) + 40 * NOx(\frac{g}{mi}) + 800 * PM\ (\frac{g}{mi}) \quad (2.1)$$

, where PM is exhaust particulate matter.

A rule based strategy was then devised in an attempt to reproduce the behavior of the dynamic programming algorithm. However, this rule-based method failed to perform well in all instances [8]. In [7], Lin et al. attempted to improve upon their work by using dynamic programming on six "representative driving patterns", and again creating a rule-based method which was made to follow the behavior of the dynamic programming solution [7]. Then, a "driving pattern recognition" algorithm was used to determine which set of rules should be implemented during real time control. Again, this method showed mixed results, and when compared for several drive cycles to the previous method, the greatest improvement gained on the performance criteria was approximately 5%, and in some cases the new method performed worse [7].

Lammers et al. performed a dynamic programming optimization on a vehicle model that consisted of a diesel engine with a 12v belt-driven ISG [2]. With the 12v system and dynamic programming, the authors found that the maximum fuel consumption benefit lies between 1.5% and 2.0% [2]. Lammers also implemented a heuristic strategy in an attempt to replicate the dynamic programming results. Unlike [7, 8], Lammers was able to achieve very similar fuel economy values over various drive cycles with the heuristic strategy. However, it is suspected that this could be due to the 12v system which does not allow as much "room" for optimization as the larger system used in [7, 8].

Arnolds et al. developed a dynamic programming strategy for a series hybrid vehicle in [9]. As mentioned, with a series hybrid vehicle, it is much easier to control the ICE operating point due to its decoupling from the drivetrain, which can result in a simpler control problem. With the method developed, the authors were able to show an increase of 2% in the vehicles fuel economy over that of the baseline heuristic controller [9].

In the above papers, dynamic programming over a known drive cycle has been shown to be a reliable method to achieve optimal power management through simulations of vehicle models. Although dynamic programming (over an entire known drive cycle) lacks real time implementability, it is a very powerful tool that should be considered in the development of a real time control strategy. Several authors attempted this by trying to mimic its behavior with heuristics strategies, but were only moderately successful [2, 7, 8] . Because of the powerful idea behind the principle of dynamic programming, a method inspired by dynamic programming is used in this research to create an implementable optimal control scheme.

2.1.2 Model Predictive Control

Model predictive control is a strategy that involves using a vehicle model coupled with a drive cycle prediction algorithm to predict future drive cycle characteristics and then optimize the controller based on this prediction. While this method shows promise for real-time implementation, the actual implementability depends greatly on the level of

detail present in the vehicle model, the optimization method, and the prediction algorithm. Although some researchers have developed model predictive control methods that are simple enough to implement in real time, this section of the thesis will focus on the methods that lack real time implementability.

Because it is generally accepted that dynamic programming provides a true optimal solution for supervisory control of the various power elements of hybrid vehicles, some researchers choose to implement dynamic programming over a predicted "horizon" [11]. However, any optimization method could be used for a predicted horizon. In fact, various vehicle models and prediction algorithms have been utilized for this purpose [11, 12, 13].

Arsie et al. created a drive cycle prediction algorithm for a parallel hybrid vehicle using a neural network with a prediction horizon of 20 seconds [11]. After the future driving condition was predicted by the neural network, dynamic programming was used in conjunction with the vehicle model to optimize the control variables based on the predicted drive path [11]. This method produced a 45% improvement in fuel economy over an equivalent non-hybrid vehicle [11]. However, due to the high computational demand of the dynamic programming optimization, this method cannot be implemented in real time [11].

Koot et al. used a more conventional prediction algorithm and a linear programming optimization strategy on a parallel hybrid vehicle with an ISG connected to the drivetrain after the clutch [12]. In order to improve the implementability of this strategy, the authors used a prediction horizon of 10 seconds and observed faster than

real-time simulation speeds on a 2.4 GHz Pentium IV computer [12]. With this method, the authors were able to achieve an improvement of 19.1% in fuel economy and approximately 24% improvement in overall emissions compared to a conventional non-hybrid vehicle. However, this method cannot be implemented in Electronic Control Units (ECUs) on current production vehicles due to the fact that most ECUs compute at speeds in the MHz range.

Stromberg et al. developed a model predictive strategy that uses the *fmincon* function from MATLAB's optimization toolbox to find the optimal control variable values over the predicted horizon [13]. However, this method relies on the assumption of a purely convex optimization function which ensures that a local minimum is the same as the global minimum of the function. Also, rather than devising a true prediction algorithm, the authors simply assumed that the actual drive cycle was known for a certain prediction horizon. Within the predictive strategy, a study was done to examine how the prediction horizon time influenced the overall fuel economy. Intuitively, as the prediction horizon increases (which for this case was assumed to be an accurate prediction), the vehicle fuel economy increases as well. However, it was noted that for a prediction horizon of 100s, the fuel economy increased by 20.6% compared to no prediction horizon, but when the horizon was increased to 1200s (the full drive cycle), fuel economy increased only by an extra 1% overall [13]. This means that the improvement almost saturated at a prediction horizon of 100s. This trend shows that if predictions are accurate, it may be possible to use model predictive control in real time controllers and achieve near optimal results. While all these methods are promising,

currently, there does not seem to be a model predictive control method which is capable of being used for real-time vehicle control.

2.2 Real-Time Strategies

Real-time strategies are control methods that can be implemented in current production vehicles. Thus, it requires that the strategies possess two main capabilities. First, the control hardware can calculate the optimal action at any operating point in real time. Second, the method can operate without "knowing" what the driver's next action may be.

2.2.1 Heuristic

Heuristic techniques provide an easily implementable control strategy since the methods do not require any real time optimization calculations. The heuristic control relies on the engineer's knowledge of the system to build rules that should govern the system at any given state. This can require extensive testing and evaluations that would need to be performed for each different vehicle. This type of approach has already been discussed in the "dynamic programming" section, where the authors of [2, 7, 8] developed complicated DP optimization processes in order to extract a set of control rules for a heuristic strategy.

Along with the authors mentioned in the dynamic programming section, Fulks et al. developed a heuristic control strategy for a hybrid vehicle control. The vehicle chosen for the strategy was a mild hybrid with a belt-operated ISG. The developed strategy focused on the specific challenges of a belt-driven ISG system, such as ensuring the ISG could indeed start the motor on demand [4]. However, the strategy could still show a 3.5% improvement in fuel economy over a vehicle without stop/start technology.

Nesbitt et al. used electronic throttle control on a parallel hybrid truck to create a heuristic control strategy with the purpose of improving fuel economy [5]. The strategy divides the HEV's operation into four specific modes: Launch, Powersplit, Parallel, and Regenerative Braking [5]. Each mode possesses a different set of rules regarding power distribution, and the vehicle enters each mode based on a set of rules as well. The strategy showed an improvement in fuel economy of 15% over the stock vehicle, with a 10% increase in vehicle weight. However, the strategy involved no true optimization technique, which means the authors could not prove if there was (or was not) any room for improvement in the strategy.

Heuristic control strategies, while simple to be implemented, still rely heavily on the designer's understanding of the system, and the operating conditions he/she is able to foresee. Because of the wide variety of ICE/ISG pairs, it is difficult to decide which heuristic rules should be used for each vehicle and whether or not the rules are truly optimal with regards to some desired metrics. This means that heuristic methods should be avoided for optimal supervisory control.

2.2.2 Low Pass Filter

This method is currently being utilized in series hybrid vehicles. In a series hybrid vehicle, it is possible to control the operating point of the ICE independently because it is completely decoupled from the drivetrain. This decoupling allows the ICE to operate in a smooth and efficient manner, even when the power demand from the driver can be very erratic [14]. Similar to all the other methods, the battery State of Charge (SOC) must be maintained along with the desired improvements in fuel efficiency. Because the ICEs in series hybrid vehicles are used only to charge the battery, it can be said that in order to maintain the SOC, the generator charging power should be nearly the same as the wheel output power over time [14]. The system can be modeled as a fluid flow system as shown in Figure 2.1

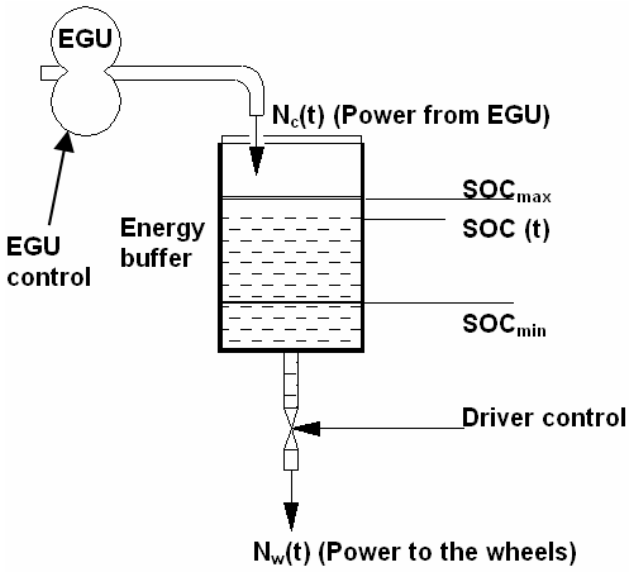


Figure 2.1 Series Hybrid Fluid Flow Model [14]

In the diagram, the “energy buffer” represents a battery, which can store or discharge extra power as needed. Because of the desired relationship between generator power and wheel power, the time signal of the wheel power can be sent through a low pass filter in order to obtain the general trend of the power sent from the batteries to the wheels [14]. This filtered signal then becomes the control signal for the ICE power, which should avoid transient operation in order to maintain high levels of efficiency [14]. The cut-off frequency for the low pass filter not only affects the operation of the engine, but also determines how far the battery SOC will deviate from its initial value [14]. Although it was not discussed by Konev, an optimization process could be performed to find an optimal cut-off frequency for the method.

The main strength of this method is its ability to allow the ICE to operate at a semi-steady state. Controlling the ICE operating points in this manner is simply impossible in a parallel mild hybrid vehicle. However, an in depth understanding of the optimal operation of the ICE can be of great use in the formulation of a control method for a MHEV.

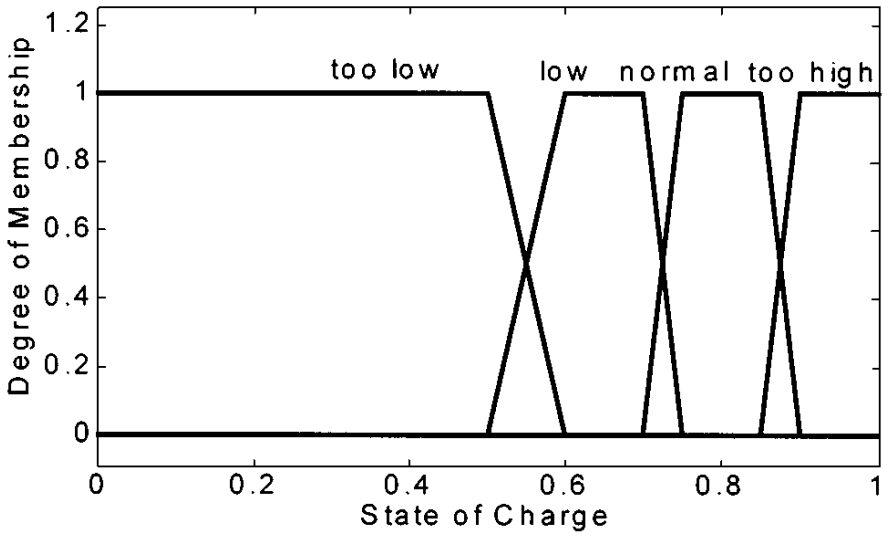


Figure 2.2 Fuzzy Logic Membership Functions [15]

2.2.3 Fuzzy Logic

Fuzzy logic is a control method that is based on degrees of truth rather than typical true or false type binary logic [15, 16]. Therefore instead of a control signal having a value of 0 or 1, it can have a value ranging from 0 to 1. The degrees of truth are determined by membership functions, which can be most easily explained by Figure 2.2 [15].

In the figure, there are four membership functions for the battery state of charge: too low, low, normal, and too high. Depending on the value of the SOC, it may have a certain degree of membership (or level of truth) in multiple functions. While this method is more conducive to optimality, it is not much different from the simpler heuristic methods.

Both Schouten et al. and Glenn et al. developed a fuzzy logic controller for a parallel hybrid vehicle which optimizes the fuel economy through understanding of the efficiencies of the ICE, EM, and battery [15, 16]. After analyzing these components, various membership functions were created based upon the operation efficiencies. The authors then derived a set of IF-THEN rules to control the vehicle based on the values of the membership functions at any point in time. Schouten et al. focused on decreasing energy losses in the system (through lack of component efficiency) and observed a decrease in energy losses ranging from 8%-10% [15]. Glenn et al. simply focused on increasing vehicle fuel economy and achieved an improvement of approximately 16% in fuel economy over a conventional non-hybrid vehicle.

As mentioned, the fuzzy logic control method does not enforce any true optimality metrics. This means that the control designer must still derive a set of rules regarding the processing of the membership function values. These rules are simply based on the designer's interpretation of the component efficiency maps and "optimality". Therefore, the most a designer can do is to try to discover better rule sets or better membership functions.

2.2.4 Lookup Table

The lookup table method involves creating a table of desired values for control variables in the state space of the system. In fact, this has been a common method for the control of fuel flow rates using ECUs for ICEs. For fuel flow, a table of desired flow

rates is constructed at each point in the state space made up of engine torque and engine speed. This yields a three-dimensional lookup table, as shown in Figure 2.3 that is referenced by the ECU to determine the amount of injected fuel at any engine operating point. For HEV supervisory control, a table is built to provide optimal operation of the EM and ICE. The methods used to form these tables, and the states utilized within the table can vary.

Wang et al. used the idea of "equivalent fuel consumption" of the EM and battery to build multiple lookup tables of optimal EM torque values [17]. Each of these tables shows the optimal EM torque values for a given voltage of the HEV's supercapacitor. The optimal torque values for each voltage are the ones that minimize the sum of the ICE fuel consumption and the EM equivalent fuel consumption at a specific ICE speed and torque demand. This method produces several lookup tables that are referenced for specific supercapacitor voltages like the one shown in Figure 2.3. With this strategy, the authors were able to achieve a 5.4% improvement in fuel economy compared to a heuristic strategy implemented on the same vehicle.

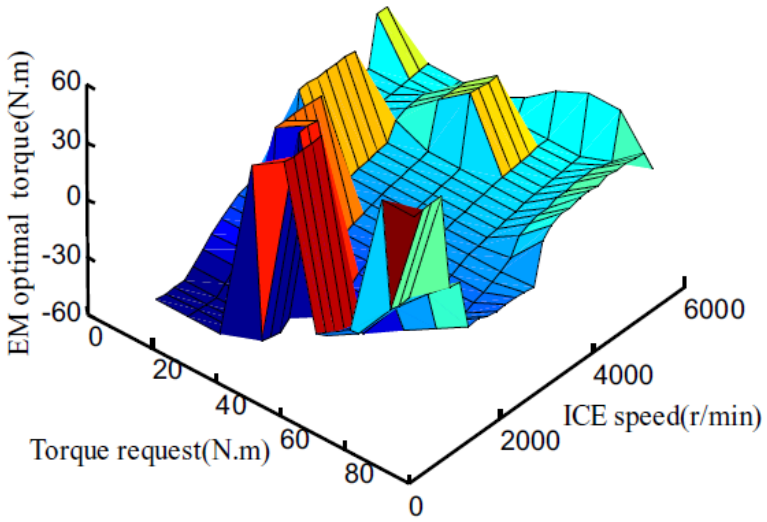


Figure 2.3 EM Torque Lookup Map for a Given Supercapacitor Voltage [17]

Ormerod and Fussey devised a lookup table to reduce fuel consumption and emissions using the following objective function [18].

$$Objective\ Function = \sqrt{\sum_i (w_i (Species)_i)^2} \tag{2.2}$$

Where

Species = normalized quantity of each species including fuel consumption, HC, CO, NOx, PM, NVH, etc.

w = weight for each species

i = index for each species

Rather than using the table as a pure lookup table for control of a component, it is used as a 3D objective function which is a function of engine speed and load.

The objective function represents a cost to generate the EM output power. If the cost is lower than a given threshold, the EM will generate power. If the cost is higher than a certain threshold, the EM will assist the ICE. With this method, the authors observed up to a 5% reduction in fuel consumption and 7% reduction in emissions.

Yoon and Lee from Hyundai Motor Company created a lookup table based on power demand rather than torque demand [19]. Also, rather than using the demand on the engine, the authors used the demand at the wheels for the control method. This demand is interpreted from the driver accelerator/brake petal input and is positive for acceleration and negative for braking. Similar to the previously mentioned lookup tables, a cost function was used to determine the optimum values for control variables (in this case EM power). This yielded the table shown in Figure 2.4.

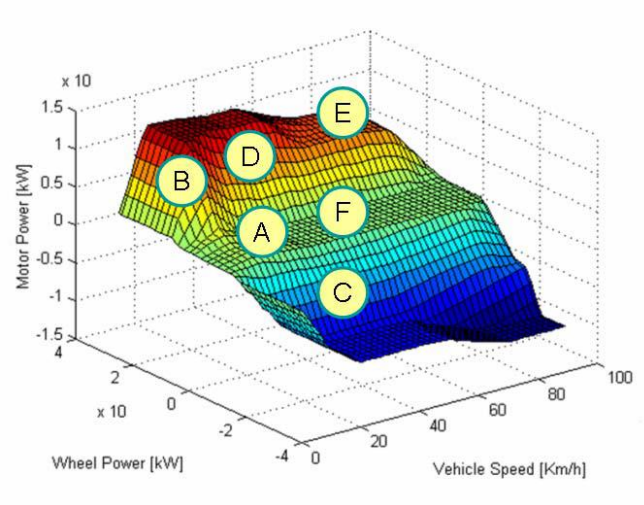


Figure 2.4 Power Demand Lookup Table [19]

Yoon and Lee also created a lookup table for control of the vehicle's CVT, but that is out of the scope of this thesis. This method seems to promise good results, but no specific numbers regarding its performance were presented in the paper.

These lookup table-based methods approach the HEV control problems from a practical perspective. Because lookup tables are a current standard for vehicle control, it is very simple to implement them for the purpose of supervisory power management. This means that a lookup table can be implemented just as easily as a heuristic control strategy. However, unlike the heuristic strategies, a lookup table can be shown to be optimal (for a certain cost function), and this optimality can extend throughout all possible operating points a vehicle may experience. The use of a cost function is very much similar to the dynamic programming methods, except that the vehicle operation is broken down into discrete states instead of a drive cycle being broken down into smaller sub-problems like in the dynamic programming method. The generation of the lookup table is all conducted offline, which means that very little calculation is required for real-time control by an onboard computer. Based on these observations, a lookup table strategy is used in this research.

CHAPTER 3

MILD HYBRID ELECTRIC VEHICLE MODEL

For the performance evaluation as well as the controller development for a mild hybrid vehicle, a conventional light-duty vehicle model developed by US Environmental Protection Agency (EPA) was modified to include the necessary components for the mild hybrid drivetrain.

3.1 EPA Light Duty Vehicle Model

The EPA model called ALPHA is a full vehicle model developed to simulate various types of light duty vehicles [20]. Among several options for vehicle configurations available in ALPHA, this research is focused on a small passenger car model. The vehicle specifications for the small car are shown in Table 3.1.

Table 3.1 Parameters of EPA ALPHA Model with Small Car Option

Vehicle Weight	1191 kg
Vehicle Frontal Area	2.291 m^2
Vehicle Drag	.29
Engine Peak Power	82.3 kW
Engine Peak Torque	145 Nm
Base Fuel	36.16 mpg

The ALPHA model employs various lookup tables for engine control algorithms, which is advantageous in the controller design for two reasons. First, models based on lookup tables can run a certain drive cycle simulation in a relatively short amount of time, which allows for rapid development of the control algorithm. Secondly, the values in the lookup tables can be used in the formulation of the control scheme. As an example, the engine fuel map of the ALPHA model, which is a necessary part of the current control design, is shown in Figure 3.1.

It should be noted that the ALPHA model is not a high-fidelity physics-based model. This means that the simulation results may not be as accurate as other high fidelity vehicle models. However, for the purposes of controller development and performance evaluation for a hybrid electric vehicle, which is the focus of this research, the accuracy of this model is deemed acceptable.

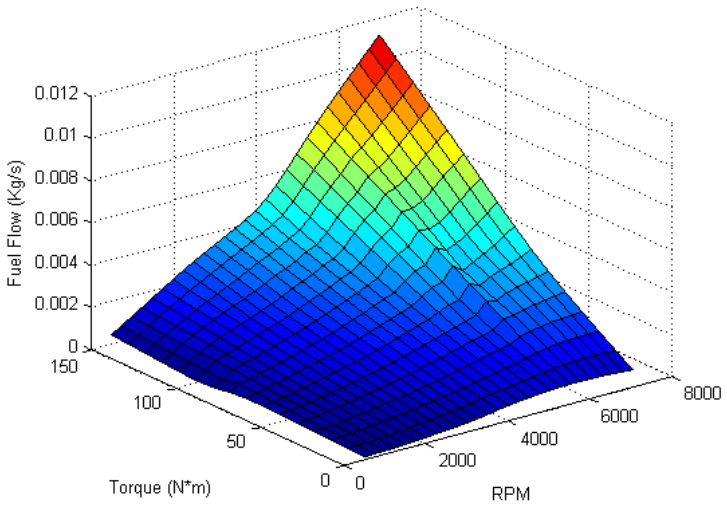


Figure 3.1 Engine Fuel Map of EPA ALPHA Model with Small Car Option

3.2. Mild Hybrid Electric Vehicle Model

For the purpose of this study, the baseline ALPHA model was modified to include the necessary components of a mild hybrid electric vehicle. These include an electric motor/starter and an enlarged battery pack. Models for those two components were created and integrated into the ALPHA model to produce a mild hybrid vehicle model.

3.2.1 Battery Model

The battery of a mild hybrid powertrain must have a larger capacity than that of a conventional IC engine powertrain. This is not only due to the fact that it must power the electric motor, but also it must run all necessary accessories when the IC engine is turned off. These accessories can include such systems as the air conditioning and the power steering. Although the IC engine may be turned off during breaking, it is desirable to provide power to these accessories without interruption during this event. Because of these reasons the battery model for the original ALPHA model had to be modified.

The battery model used in this research was first developed by Tremblay and Dessaint in 2009 for use in a hybrid vehicle simulation. Tremblay and Dessaint modeled four types of batteries: lead-acid, nickel metal hydride, nickel cadmium, and lithium ion [21]. Each model underwent experimental validation and could accurately predict the behavior of each battery type. In this research, a lithium ion battery model was adopted as this is the current trend in hybrid vehicles [22]. However, it should be noted that the

control strategy itself will be applicable to vehicles with any battery type. The charge and discharge equations for the lithium ion battery model are [21]:

Charge

$$V_{batt} = E_0 - iR - K\frac{C}{it-0.1C}i^* - K\frac{C}{C-it}it + Ae^{-Bit} \quad (3.1)$$

Discharge

$$V_{batt} = E_0 - iR - K\frac{C}{C-it}(it + i^*) + Ae^{-Bit} \quad (3.2)$$

where

V_{batt}=battery voltage (V)

E_0=battery constant voltage = 3.336 (V)

K=polarization constant =.0076 (V/Ah)

C=battery capacity = 2.3(Ah)

i=battery current (A)

$it = \int i\,dt$=actual battery charge (Ah)

i^*=filtered current (A)

R=internal resistance (Ω)

A=exponential zone amplitude = .26422 (V)

B=exponential zone time constant inverse = 26.5487 $(Ah)^{-1}$

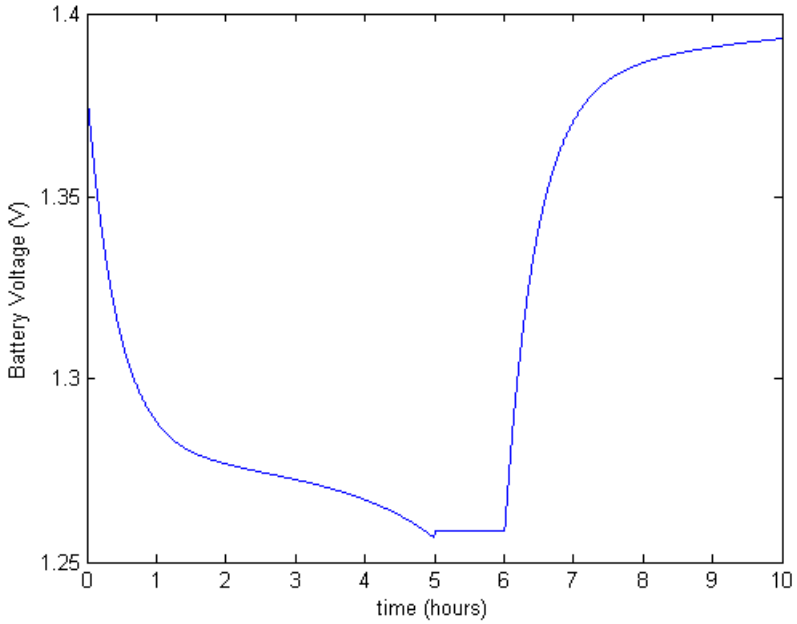


Figure 3.2 Discharge and Charge Characteristics of Lithium-Ion Battery

Using these equations by Tremblay and Dessaint, a model was created and its discharge and charge behaviors were plotted in Figure 3.2. The charge and discharge characteristics of the new battery model were found to be consistent with those in [21].

3.2.2 Electric Motor Model

Currently, there exist two major types of motors in hybrid drivetrains - induction and DC motors [23, 24]. Each type of motor has its own advantages, but modeling the DC motor is much simpler than modeling its AC counterpart [24]. Because the focus of this research is control system design rather than vehicle modeling, the electric motor, or integrated starter generator (ISG) was modeled as a Permanent Magnet DC motor.

However, it should be noted that the proposed control method is directly applicable to AC drive systems without any major modification to the control method itself.

A DC motor is governed by relatively simple well-known equations. The equations are as follows:

Torque

$$T = K_t i \tag{3.3}$$

Back EMF

$$V_b = K_b \omega \tag{3.4}$$

Voltage

$$V = iR + L\frac{di}{dt} + V_b \tag{3.5}$$

, where

T = motor torque

K_t= motor torque constant

i = motor current

V_b= back EMF voltage

K_b= motor back EMF constant

ω= motor angular velocity

V = motor voltage

R = motor resistance

L = motor inductance

For the mild hybrid model, the desired torque and the rotational speed of the motor will be the input to these equations, and a corresponding current and voltage will be determined and provided by the battery. For the control strategy implemented, it will be assumed that the motor can safely add/subtract a maximum of 50N·m of torque to/from the system.

3.2.3 Model Integration

The original ALPHA model includes an electric motor that adds a torque load to the crankshaft based on the state of the vehicle's electronic components. So, for the model integration, it was only required to add these new components to the electric system model in a manner that would allow for the torque of the ISG to be added to or subtracted from the crankshaft torque.

CHAPTER 4

LOOKUP TABLE-BASED CONTROL

4.1 Weighted Cost Function Control Method

When a vehicle is in operation, the powertrain's main function is to propel the vehicle at the velocity desired by the driver. The adherence of the control system to this constraint can be measured by the vehicle's ability to match the velocity profile of a certain drive cycle. Since the actual driving force to a vehicle system is a torque produced by the crankshaft, matching of the vehicle speed to a desired velocity profile is equivalent to matching the powertrain output torque to a desired torque profile.

In the mild hybrid architecture, the torque produced by the powertrain is the sum of the torque from the ISG and the torque from the ICE. In order to ensure that torque demand from the driver is met, the relationship in Equation (4.1) should be satisfied.

$$T_{demand} = T_{ISG} + T_{ICE} \tag{4.1}$$

Equation (4.1) simply shows that in order to achieve the desired vehicle velocity, the torque demanded by the driver must be met by the two power sources that produce torque. By applying the equality constraint shown in equation (4.1) to a given torque demand, one can find that there is only one independent variable, which is either T_{ISG} or T_{ICE}. In this study, we choose the ISG torque to be the design variable that needs to be optimized. This allows the ICE torque to be determined with equation (4.1).

4.2 Determining Optimal ISG Torque

The optimal value for the ISG torque with respect to fuel economy at any given time clearly depends on how much fuel is being consumed at that time. From equation (4.1), it can be understood that as the ISG torque increases, the ICE torque will decrease for any value of torque demand. Also, from the fuel map in

Figure **3.1**, it can be observed that for any engine RPM, the fuel consumption decreases as the torque demand on the engine decreases. Therefore, the vehicle will consume the least amount of fuel when the ISG is operating at maximum torque. However, if the ISG were to be operating at maximum torque for an extended amount of time, the battery charge would be depleted quickly. Therefore, the state of charge (SOC) of the battery must be taken into account in the optimization as well.

Now that it has been determined that fuel consumption and battery SOC are to be minimized, these values will become part of a cost function that will be used to determine the optimum ISG torque. A general form of the cost function can be defined as

$$cost\ function = f(\dot{m}_{fuel}, SOC) \tag{4.2}$$

where the mass flow rate of the fuel, $\dot{m}_{fuel}$ is determined by the ICE torque and RPM as

$$\dot{m}_{fuel} = fuel\_rate(T_{ICE}, RPM) = fuel\_rate'(T_{ISG}, RPM) \tag{4.3}$$

Note that in Equation (4.3), the relationship in Equation (4.1) is used to rewrite the fuel flow rate as a function of the ISG torque.

Meanwhile, the battery SOC is related to the initial SOC minus the time integral of the discharged current, which is a function of the ISG torque. More specifically,

$$SOC = \left(\frac{C \times SOC_{init} - \int_0^t i(T_{ISG})dt}{C}\right) \tag{4.4}$$

Therefore, by substituting Equations (4.3) and (4.4) into Equation (4.2), one can see that the cost function is eventually a function of only one design variable, T_{ISG}. All the other variables are either given or determined from the dynamics of the system.

4.2.1 Formation of Cost Function

The operating state of an IC engine can be described by two variables: output torque and RPM. In addition to the state of the IC engine, the battery SOC is also used to determine the operating condition of the mild hybrid electric vehicle.

Therefore, at any time instance, the powertrain system is in a specific state defined by the following three variables.

1. Torque Demand, T_{demand}
2. Engine Speed, RPM
3. Battery State of Charge, SOC

Although the battery SOC involves the time history of the ISG torque that is dependent on a specific drive cycle, it is possible to develop a drive cycle-neutral control algorithm if the time integral term is substituted by an instantaneous value of SOC with an appropriate modification. Because the control effort should be made such that the fuel consumption is minimized while the battery SOC is maintained at a certain level, the cost function can be defined as

$$cost\ function\ =\ w_1\dot{m}_{fuel} + w_2 T_{ISG}/SOC\ . \tag{4.5}$$

The cost function in equation (4.5) is a linear combination of the two optimization criteria, $\dot{m}_{fuel}$ and SOC, with appropriate weighting factors w_1 and w_2. The mass fuel rate is included in the first term to reduce the fuel consumption during the vehicle operation. In the second term, the ISG torque is included to oppose the tendency for the controller to run the ISG longer than necessary to reduce the fuel consumption, which will continuously decrease the SOC. In addition, the battery SOC is inversely multiplied to the second term to directly apply penalty to low SOC. Note that the both terms in the cost function are dependent only on a single design variable, T_{ISG}.

Now that the cost function is defined, the optimal ISG torque value can be obtained for each operating point in the state space defined by the three parameters: T_{demand} , RPM, and SOC.

4.2.2 Optimal Balancing of Cost Function Weights

It can be seen that the optimum value for the ISG torque will depend on the values for the weights w_1 and w_2 in the cost function. Therefore, it needs to be studied how different combinations of weights affects the overall performance of the system.

As already discussed, the minimum $\dot{m}_{fuel}$ will be reached when the ISG operates in its maximum capacity, which can be achieved through optimization with $w_1 >> w_2$. Conversely, $w_2 >> w_1$ will ensure that the battery charge stays high enough regardless of its impact on $\dot{m}_{fuel}$. Therefore, for the ratio of the two weights

$$\bar{W} = w_1/w_2, \tag{4.6}$$

there must exists an optimal value, $\bar{W}_{opt}$ that will allow for the best fuel economy possible while ensuring the SOC maintains an appropriate high value. In this research, the optimality condition is set such that the change in SOC over the entire drive cycle must be zero. For the FTP drive cycle, it can be written as

$$\Delta SOC_{FTP} = 0. \tag{4.7}$$

After optimal values for the ISG torque were obtained at each operating point for the cost function with different weight ratios, simulation was conducted using the optimal ISG torque values for the FTP drive cycle and the resulting ΔSOC_{FTP} versus $\bar{W}$ is plotted in Figure 4.1.

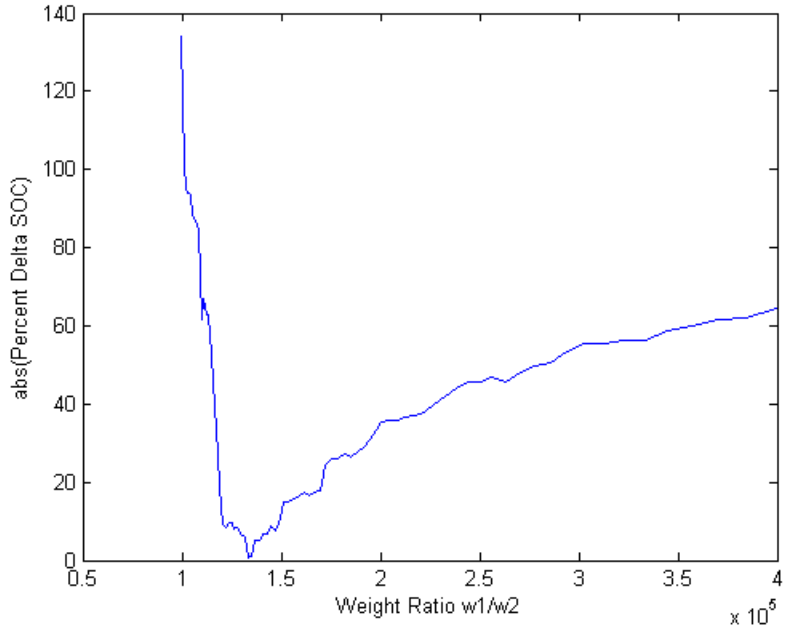


Figure 4.1 Absolute Value of Percent ΔSOC_{FTP} versus ΔSOC_{FTP}

In Figure 4.1, it can be seen that the ΔSOC_{FTP} approaches zero at

$$\bar{W}_{opt} = 1.330 \times 10^5 \tag{4.8}$$

This optimal value is relatively large due to the large discrepancy between the numerical values of the first and the second terms in the cost function. Because $\bar{W}$ is a ratio, w_2 can be set to unity, which will render

$$w_1 = 1.330 \times 10^5. \tag{4.9}$$

4.2.3 ISG Map with Optimum Cost Function

With the optimal weight values, the cost function can be written as

$$cost\ function_{opt} = (1.330 \times 10^5)\, \dot{m}_{fuel} + T_{ISG}/SOC \tag{4.10}$$

By minimizing the cost function with respect to T_{ISG} for each engine state, a 3D map or lookup table for the optimal values of T_{ISG} can be obtained as shown in Figure 4.2. This map represents a 4D array with the three operating variables defining a location in 3D space and the color of that location representing the optimum value for T_{ISG} at that state. This map can be used to control the powertrain of the mild hybrid model.

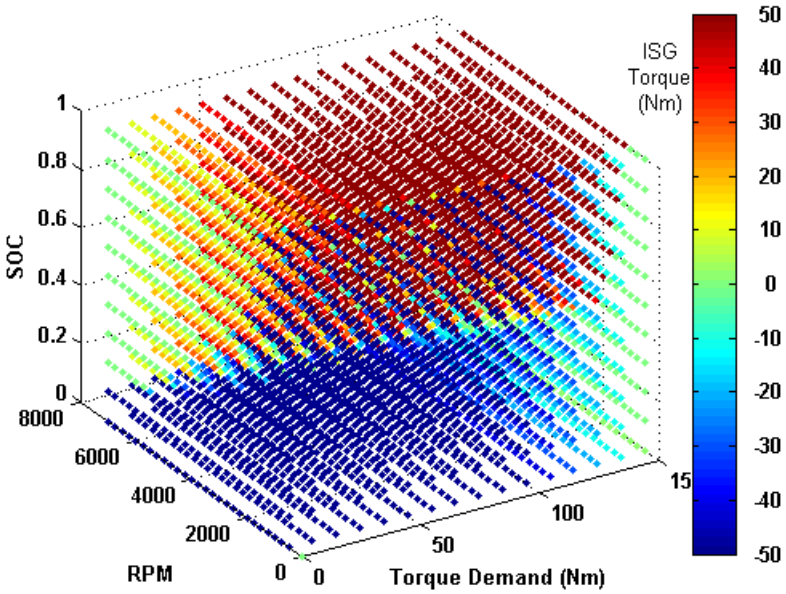


Figure 4.2 Optimal ISG Torque Map

4.3 Results

The control method based on the 3D lookup table is tested on the mild hybrid vehicle model. To compare the performance of this control method to those of other methods, two reference cases are simulated. The first reference is the performance of the original ALPHA model, which is a conventional vehicle model. The second reference is the mild hybrid model with a simple rule-based control strategy.

4.3.1 Reference Simulations

Simulations of the reference cases were conducted for the FTP drive cycle as mentioned previously. While the simulations allow many comparisons to be made, there are three main metrics of interest in the current study.

1. Fuel Economy (mpg)
2. ΔSOC_{FTP}
3. Drivability

Although the first and second metrics can be easily quantified using the simulation result, the third metric, drivability, must be defined. This can be done by comparing the actual vehicle speed for the FTP simulation to the desired speed of the FTP cycle. Then, it is assumed that any deviation of the actual vehicle speed greater than two percent from the desired speed is considered undesirable. Finally the percentage of the total time that

the vehicle spends outside of this two percent error range will be used as the drivability metric. This can be seen in equation (16).

$$Error\ Time\% = \frac{\sum time\ |_{error \geq 2\%}}{total\ time} \times 100 \tag{4.11}$$

4.3.1.1 Original ALPHA Model

The simulation results of the original ALPHA model for the FTP cycle are summarized in Table 4.1.

Table 4.1 Simulation Results of EPA ALPHA Model

Fuel Economy	36.16 mpg
$\Delta\ SOC_{FTP}$	NA
$\boldsymbol{Error\ time}$ %	1.04%

In Figure 4.3, the vehicle speed versus time plot from the simulation result is shown together with the desired vehicle speed for the FTP cycle. It can be seen that the two plots are in good agreement.

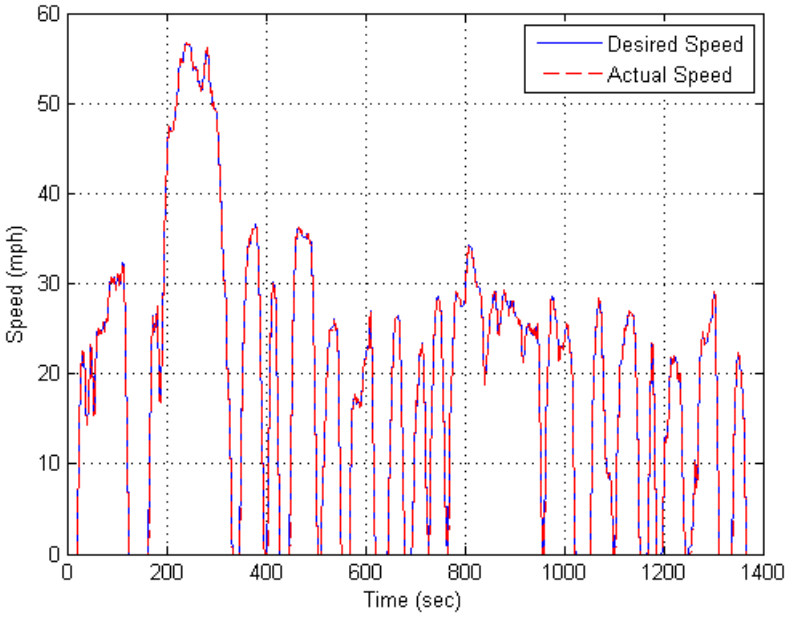


Figure 4.3 Speed Variation of EPA ALPHA Model

Figure 4.4 and Figure 4.5 show several variables of interest from the electrical system in the simulation. The first plot is the torque required by the electrical system (in this case only the alternator). The remaining plots show the battery SOC, the battery voltage and the battery current, respectively.

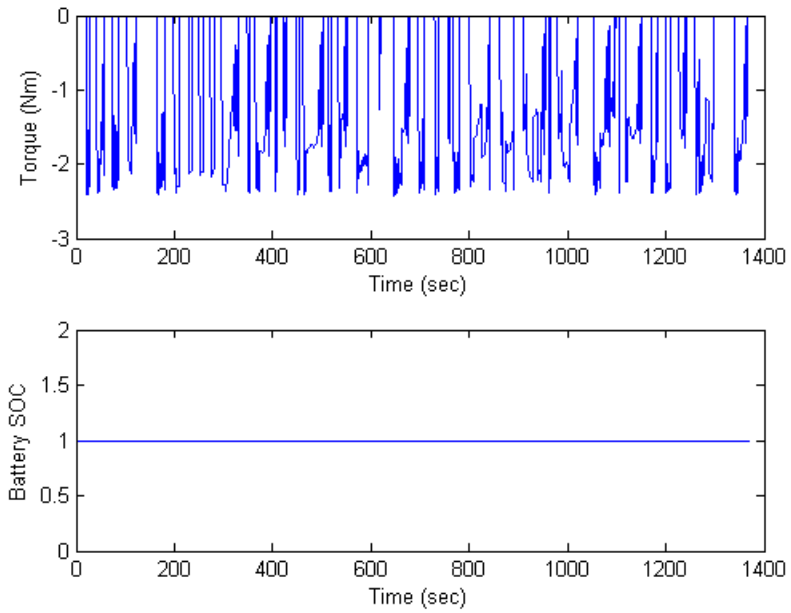


Figure 4.4 ISG Torque and Battery SOC of EPA ALPHA Model

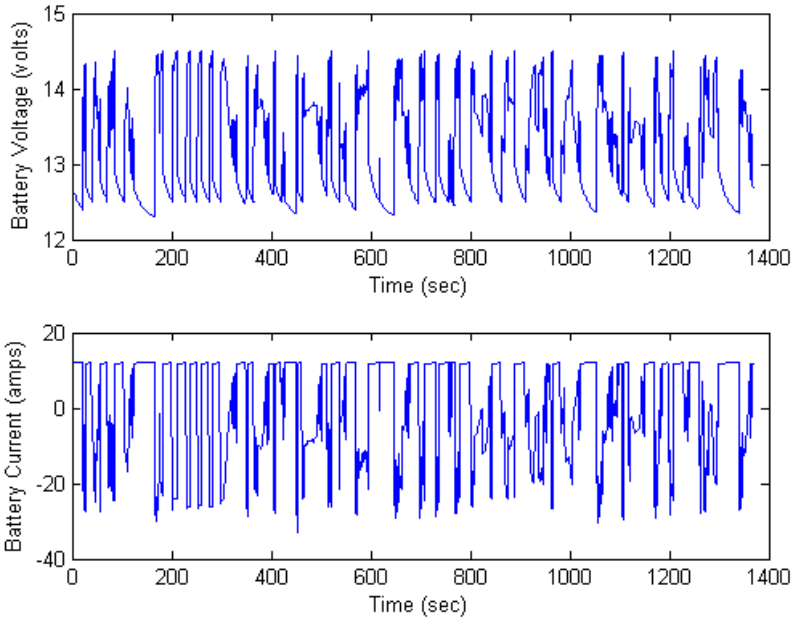


Figure 4.5 Battery Voltage and Battery Current Draw of EPA ALPHA Model

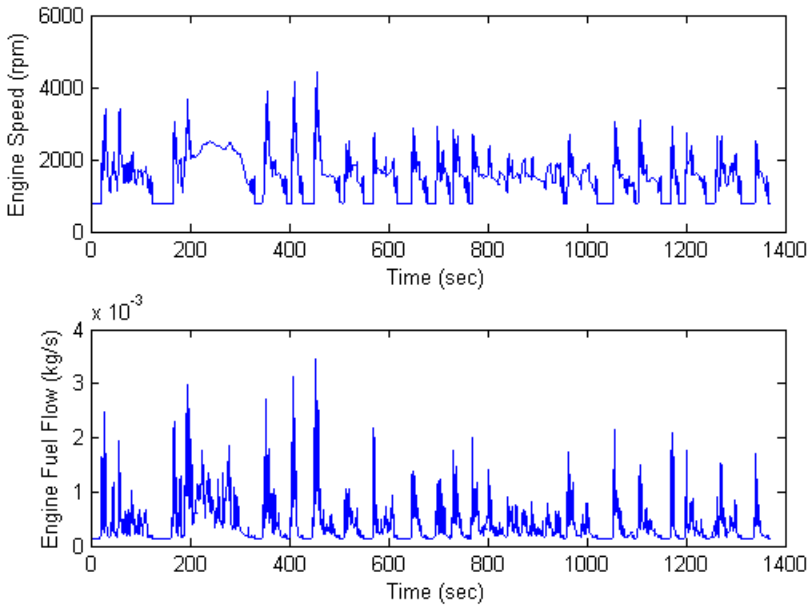


Figure 4.6 Engine Performance of EPA ALPHA Model

Since the behavior of the IC engine is also of interest in this study, the time variation of the engine RPM and the fuel flow rate are plotted in Figure 4.6.

4.3.1.2 Mild Hybrid Model with Rule-Based Control

The rule-based control algorithm employs the following three rules.

1. If the driver is braking, charge the battery
2. If the accelerator is greater than half throttle, assist the ICE.
3. If the vehicle is stopped, turn off the ICE.

The simulation results of the mild hybrid vehicle model using the rule-based control logic are summarized in Table 4.2.

Table 4.2 Simulation Results of Mild Hybrid Vehicle with Rule-Based Control

Fuel Economy	40.75mpg
$\Delta\, \boldsymbol{SOC_{FTP}}$	0.0%
Error time %	0.39%

Notice that the fuel economy of this system has been increased approximately by 12.5% over that of the original ALPHA model. In addition, the speed variation matches well with the desired speed profile as shown in Figure 4.7.

In Figure 4.8, the first plot shows the torque produced by the ISG in a manner similar to Figure 4.4. However, it should be noted that this now includes the torque of the ISG, so it varies over a much wider range. In the plot, the positive torque values mean that the ISG produces output torque and the negative values mean that the ISG applies a load torque to charge the battery.

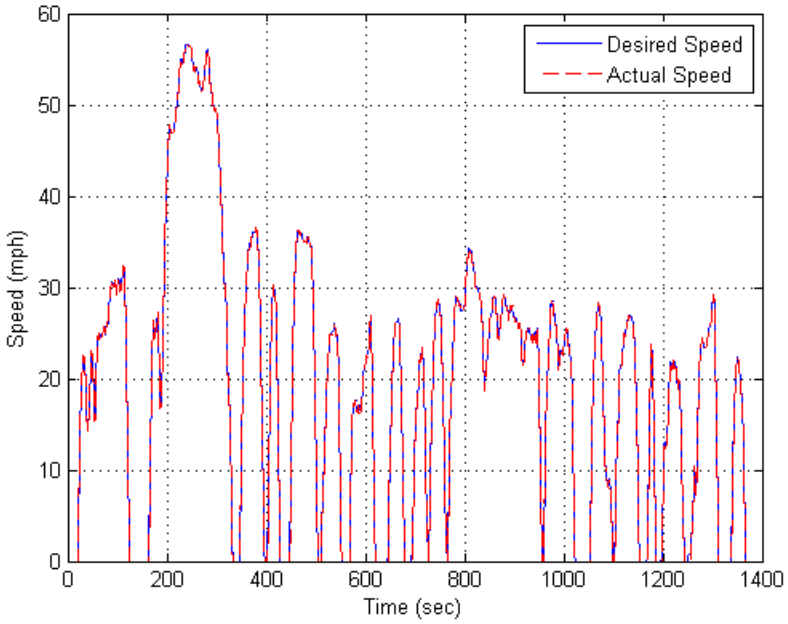


Figure 4.7 Speed Variation of Mild Hybrid Vehicle with Rule-Based Control

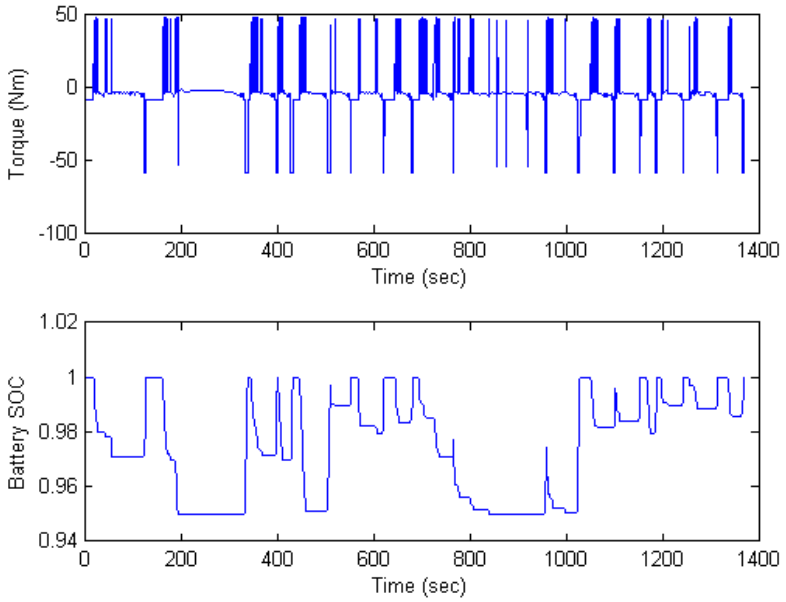


Figure 4.8 ISG Torque and Battery SOC of Mild Hybrid Vehicle with Rule-Based Control

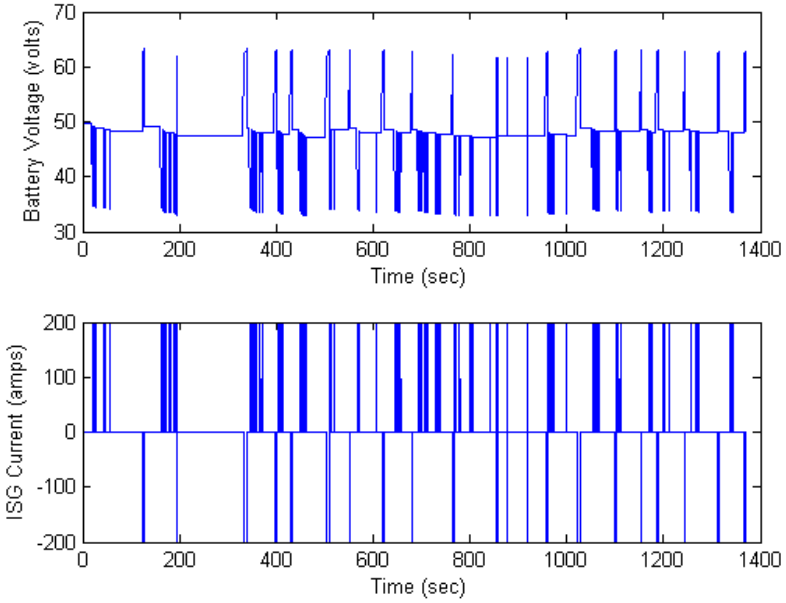


Figure 4.9 Battery Voltage and ISG Current Draw of Mild Hybrid Vehicle with Rule-Based Control

Some of the performance parameters of interest in the IC engine are shown in Figure 4.10. Notice how the fuel flow rate becomes zero during the stopped portion of the FTP cycle, which indicates that the IC engine is turned off as the control logic commands.

It can be seen from the simulation results of the rule-based control logic that a mild hybrid system does indeed have advantages in fuel economy over the conventional ICE vehicle. These results will be compared with the performances of the optimally controlled mild hybrid vehicle system in the next section.

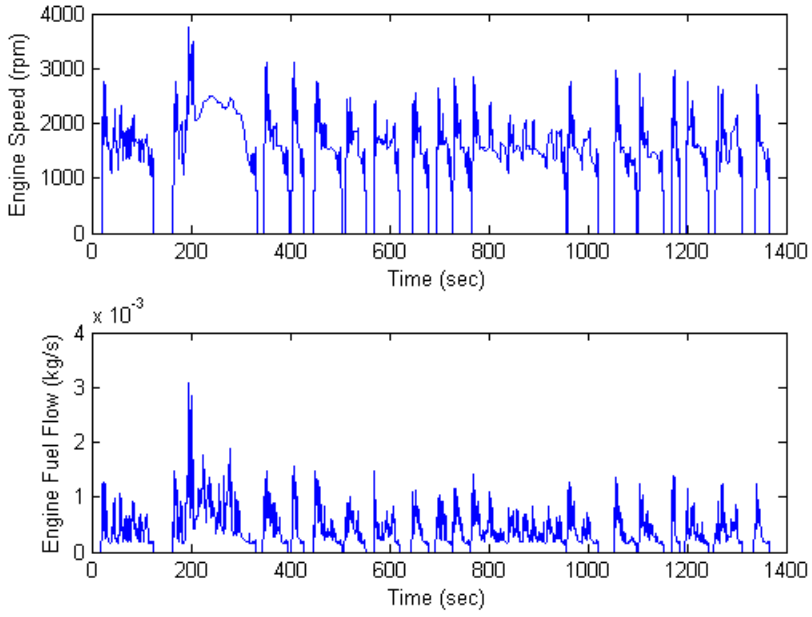


Figure 4.10 Engine Performance of Mild Hybrid Vehicle with Rule-Based Control

4.3.2 Optimized Lookup Table-Based Control

An optimized 3D control map or lookup table was produced using the cost function shown in equation (15). When the control law based on the lookup table was applied to the mild hybrid vehicle model and simulated for the FTP cycle, significant improvements were observed as summarized in Table 4.3.

Table 4.3 Simulation Results of Mild Hybrid Vehicle with Optimized Lookup Table-Based Control

Fuel Economy	47.40mpg
$\Delta\, SOC_{FTP}$	+0.2%
$Error\ time$ %	6.54%

Note that the fuel economy using the optimized lookup table has improved by 16% over that of the basic rule-based control and by 31% over the original ALPHA model. However, the time spent at greater than 2% error in speed is much larger at 6.54% as can be seen in Figure 4.11.

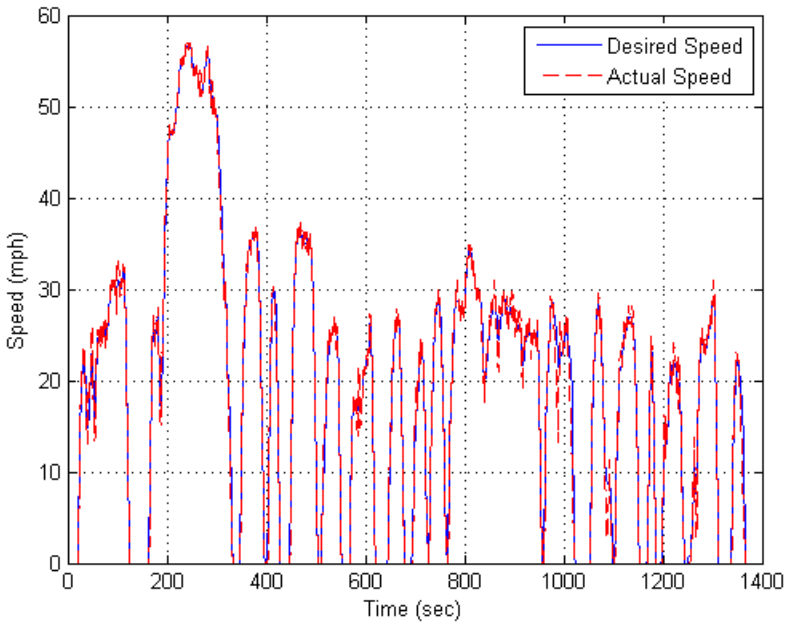


Figure 4.11 Speed Variation of Mild Hybrid Vehicle with Optimized Lookup Table-Based Control

The reason for the relatively large error between the desired speed and actual speed can be explained by Figure 4.12 and Figure 4.13. Due to the nature of the optimized ISG torque map, the control signal sent to the ISG changes very rapidly, which makes it difficult for the ICE to compensate for the changes in its required torque production.

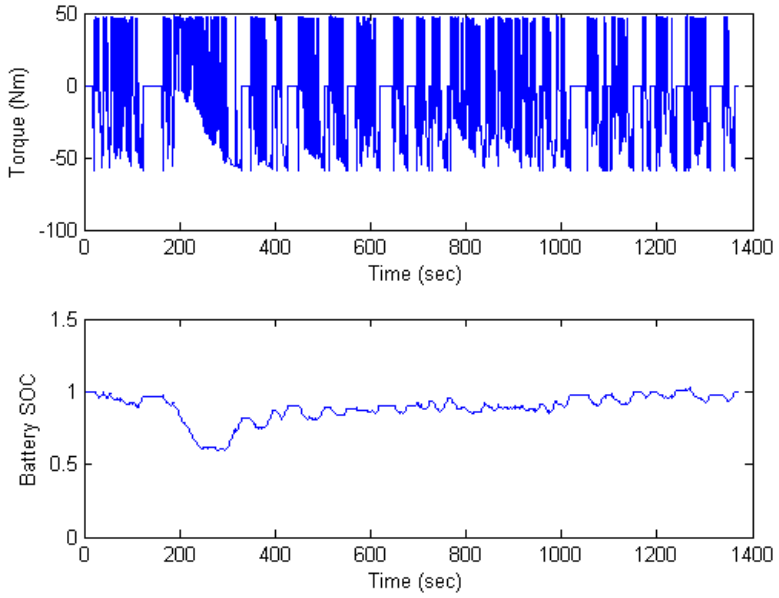


Figure 4.12 ISG Torque and Battery SOC of Mild Hybrid Vehicle with Optimized Lookup Table-Based Control

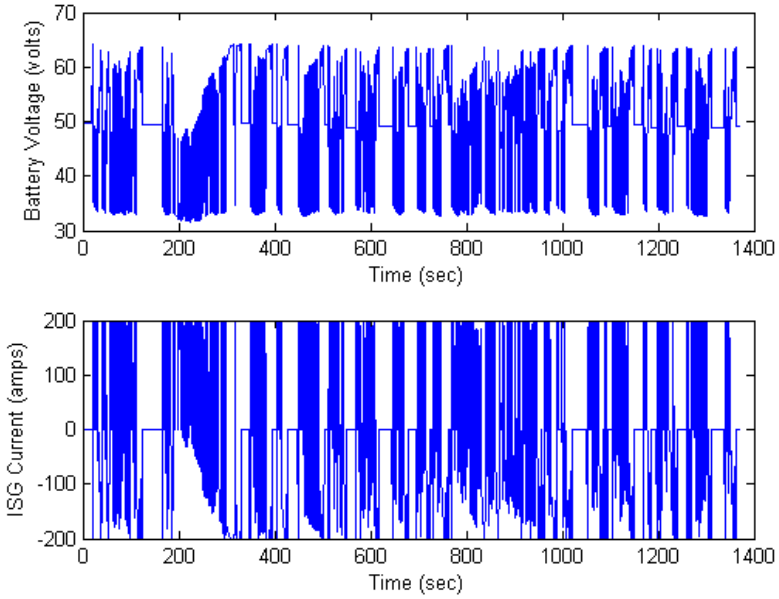


Figure 4.13 Battery Voltage and ISG Current Draw of Mild Hybrid Vehicle with Optimized Lookup Table-Based Control

Figure **4.14** shows the ICE parameters of interest. Similarly to the mild hybrid vehicle model controlled by the basic rule-based logic, the fuel flow rate becomes zero during the stopped portion of the FTP cycle. However, because the torque load experienced by the ICE is now much different, it can be seen that the fuel flow rate is different as well.

It can be seen from these results that the optimized lookup table-based control strategy produces a significant improvement in fuel economy over the basic rule-based control strategy. However, while a fuel economy has shown improvement, the drivability has not. Therefore, a modified method to alleviate the negative impact on the drivability needs to be developed.

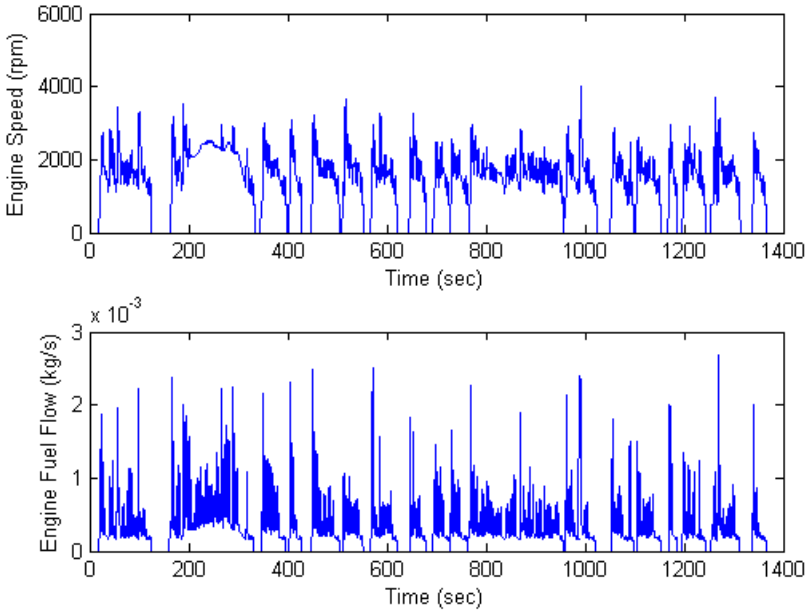


Figure 4.14 Engine Performance of Mild Hybrid Vehicle with Optimized Lookup Table-Based Control.

4.3.3 Rate-Limited Lookup Table-Based Control

As explained in the previous section, the optimized lookup table-based control algorithm showed an improvement in fuel economy with a sacrifice in drivability. This decrease in drivability is due to the rapid changing of the torque control signal at each sampling instance. In other words, within a short amount of time the state of the drivetrain is moving through a large ISG torque range in the ISG torque map. While each value in the ISG torque map is considered to be optimal with respect to the fuel consumption and the battery SOC at each time instance, the cost function does not take into account the variation rate of the torque values.

In order to alleviate this problem, a rate limiting algorithm was added to the controller to limit the rate at which the ISG torque signal could change. This allows for the output of the rate limiter to follow the optimal "trend" of the ISG torque value, but prevents the output signal from changing as rapidly as the input ISG torque signal. The result of this change is an ISG torque control signal that follows the optimal ISG torque as previously, but does not show as many rapid changes as with the previous control method.

Since the control scheme has been modified, it is necessary to find a new optimum weight ratio $\bar{W}_{opt}$ by the same method described previously. This yielded

$$\bar{W}_{opt} = 1.333 \times 10^5. \tag{17}$$

By setting w_2 to unity, the following cost function for the ISG torque map can be obtained and the corresponding 3D lookup table can be produced.

$$cost\ function_{opt} = (1.333 \times 10^5)\dot{m}_{fuel} + T_{ISG}/SOC \quad (18)$$

4.3.3.1 Simulation Results

The mild hybrid model was simulated with the new rate limiting algorithm and the new optimized cost function. The results are summarized in Table 4.4.

Table 4.4 Simulation Results of Mild Hybrid Vehicle with Rate-Limited Lookup Table-Based Control

Fuel Economy	45.84mpg
$\Delta\ \boldsymbol{SOC_{FTP}}$	+3.0%
$\boldsymbol{Error\ time}$ %	3.18%

It can be seen that with the rate limiting algorithm, the Error Time% is decreased by over 50% compared to the control algorithm without the rate limiter. However, there is a slight decrease in fuel economy although it is still over a 12% increase over the rule-based model. The simulated vehicle speed with this modified control algorithm is shown in Figure 4.15.

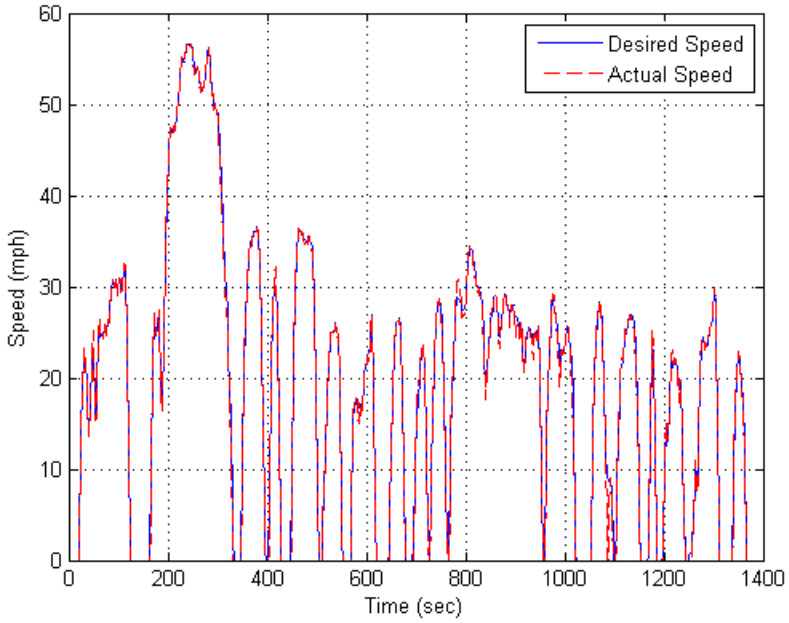


Figure 4.15 Speed Variation of Mild Hybrid Vehicle with Rate-Limited Lookup Table-Based Control

As predicted, the ISG torque follows a similar trend to that of the optimal controller without the rate limiting algorithm, but with reduce occurrence rate of abrupt switching. These characteristics can be observed in

Figure **4.16** and

Figure **4.17**.

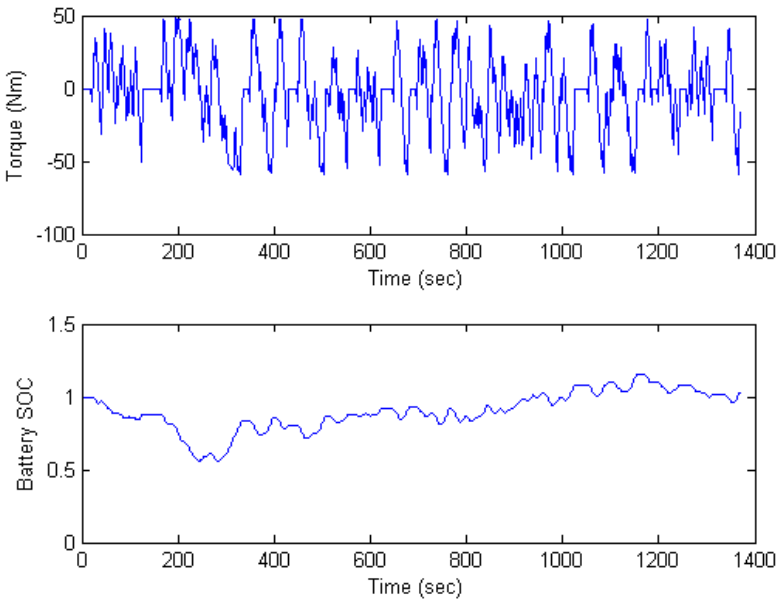


Figure 4.16 ISG Torque and Battery SOC of Mild Hybrid Vehicle with Rate-Limited Lookup Table-Based Control

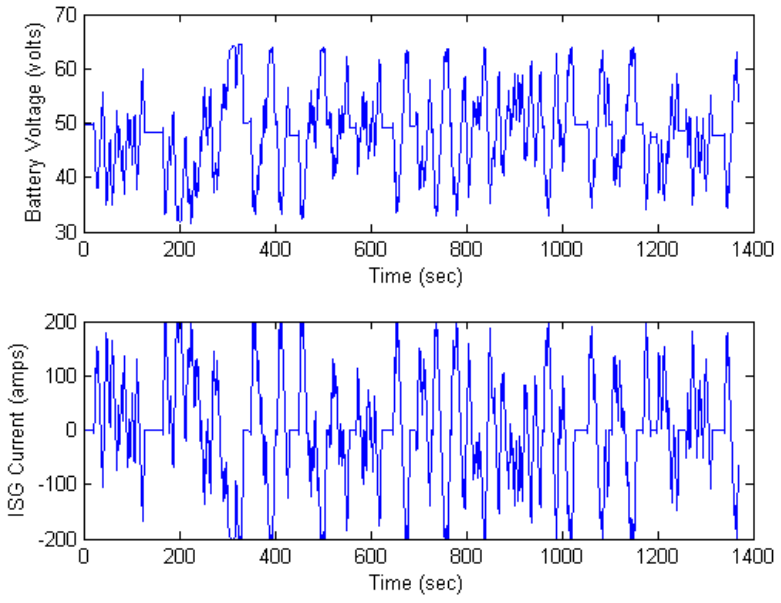


Figure 4.17 Battery Voltage and ISG Current Draw of Mild Hybrid Vehicle with Rate-Limited Lookup Table-Based Control

Figure 4.18 shows the more improved engine speed and fuel flow rate as well. These figures show that the rate limiter algorithm does improve the drivability of the vehicle while maintaining an improvement in fuel economy over the rule-based control model.

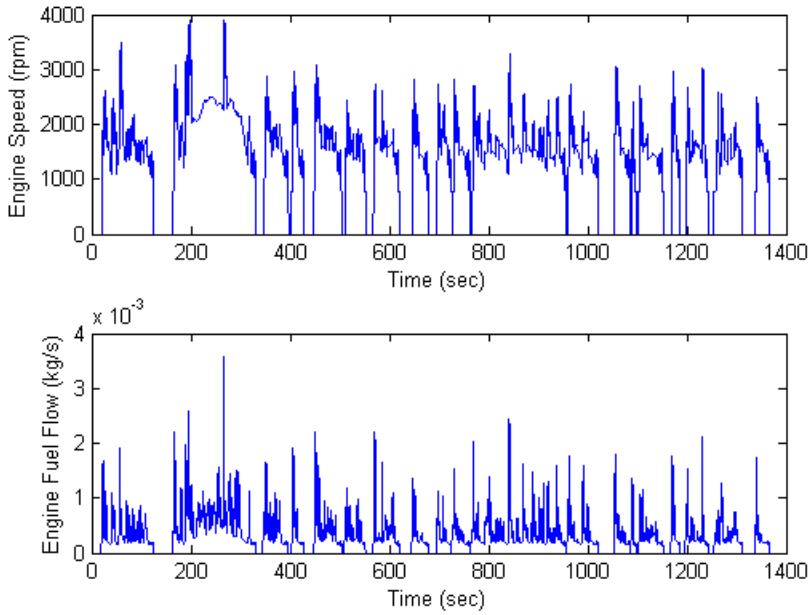


Figure 4.18 Engine Performance of Mild Hybrid Vehicle with Rate-Limited Lookup Table-Based Control

CHAPTER 5

GENETIC ALGORITHM FOR LOOKUP TABLE OPTIMIZATION

The method used for choosing the optimal values for ISG torque in Chapter 4 involved the minimization of a weighted cost function. Therefore, the choice of the cost function greatly influenced the optimization results. This means that there exist an arbitrarily large number of cost functions that can yield an infinite number of optimization results. It may be possible to use some educated "guess" to derive a meaningful cost function, but the question still remains—is the cost function optimal? Therefore, it is desirable to approach the optimization problem using a more comprehensive method.

5.1 Introduction to Genetic Algorithms

Genetic algorithms (GA) were developed in the 1960s by John Holland at the University of Michigan. Contradictory to the current use of GAs, Holland's original goal was to formally study the phenomenon of adaptation as it occurs in nature and to develop ways in which the mechanisms of natural adaptation might be imported into computer systems [25]. Holland's algorithm was a major innovation at the time of its conception. It was the first population-based algorithm with crossover, inversion, and mutation [25]. Over time, researchers realized the potential of the GA for use in solving complex computational problems like optimization.

There is no rigorous definition for "genetic algorithm" generally accepted by the evolutionary computation community to distinguish GAs from other evolutionary computation methods. However, most GAs have at least the following elements in common [25].

1. A population of "Parents"
2. Selection of "Parents" according to fitness
3. Crossover to produce "Children"
4. Random mutation of new "Children"

These steps are executed iteratively with the children formed in step 4 being used as the parents in the next iteration. Because the parents of each child are selected according to their fitness, the population will evolve to become more "fit" over time.

5.2 GAs for Optimal MHEV Lookup Table

In a similar fashion to the weight function optimization in Chapter 4, the goal of GA optimization for MHEV control is to create an optimal lookup table for the control of the ISG torque. Based on this, the population of parents is chosen to be an array of 3D lookup tables similar to the one produced by the weight function optimization. Although genetic algorithms are simple to describe and program, their behavior can be complicated, and many open questions exist about how they work and for what types of problems they are best suited [25].

No previous research is available in literature regarding the use of GAs for HEV control. So, this research intends to lay a foundation for further exploration in this area. As such, the GAs proposed in this research are not claimed to be the "best" for HEV control, but rather a good starting point for further development of a framework. Two different genetic algorithms, and explanations and analyses of their results are presented in this section. The aim of the results analyses is not only to understand how these results compare to the previous optimization methods, but also to understand why the GAs produce these results.

5.2.1 First Genetic Algorithm

The algorithm presented in this section is the first attempt at GA optimization for HEV lookup tables. As mentioned previously, while the description and implementation of GAs are fairly simple, the behavior of them could be complex and oftentimes not fully understood. The results of this algorithm are shown to be rather poor, but the algorithm and the results will be analyzed to gain perspective on why this GA does not perform as desired. The lesson learned during the formulation of this algorithm, and the analysis of provides insight for the design of the second GA.

5.2.1.1 Form initial Population

The first part of any GA is to form the initial population. Due to computation constraints, a population size of 20 members was chosen. Although it is small, it was

decided that this population size would be large enough to test the validity of GA-based optimization for HEV control.

Based on the previous work, it was believed that an initial population based on a lookup table resulted from the weighted cost function method would provide a good starting point for the algorithm. Therefore, the lookup table produced by the weighted cost function was used as a "seed" table. However, the seed table was not produced using weights that would yield a ΔSOC of zero. This provides an opportunity to analyze the GA's ability to reduce ΔSOC. Each consecutive member of the population, $\boldsymbol{M}_i$, would be formed by multiplying the original lookup table, $\boldsymbol{T}_{seed}$, by $\boldsymbol{T}_{rand}$, a 3D array of equal dimension with random values between -1 and 1.

$$\boldsymbol{M}_{(m\times n\times p),i} = \boldsymbol{T}_{seed,(m\times n\times p)} \text{ o } \boldsymbol{T}_{rand}|^{1}_{-1(m\times n\times p)};\ i = 2 \dots 20, \tag{5.1}$$

where o represents the entrywise matrix product called the Hadamard product. Equation (5.1) shows how the initial population will be formed for the first iteration of the GA. Each consequent population will be formed from steps 3 and 4 of the previous iteration.

5.2.1.2 Perform Fitness Evaluation

After forming the population, it is necessary that each member undergoes a fitness evaluation. For the current application, the evaluation is the vehicle simulation performance over the FTP drive cycle. Like before, the performance metrics are chosen to be fuel economy, ΔSOC, and Error Time%. Because there are three performance

metrics, a weighted linear combination of the three metrics is calculated and assigned to each member as a single score. However, it was determined that a ΔSOC greater than 20%, or Error Time% greater than 8% would receive an automatic score of 0. Therefore, the score function for this algorithm was chosen as:

$$score_i = \begin{cases} MPG - 10|\Delta SOC| - \%Error\ time \\ 0\ if\ \ |\Delta SOC| > 0.2\ \ or\ \%Error\ time > 0.08 \end{cases};\ \ i = 1 \dots 20 \qquad (5.2)$$

Note that GAs are quite flexible in nature, and that many other methods for scoring could be explored.

5.2.1.3 Create new population with Elite Passthrough, Reproduction, and Mutation

The next step in the algorithm is to sort the members based on the score obtained from the scoring function. Using a technique called Elite Passthrough, the two highest scoring tables pass through to the next generation without any modification. Next, a weighting factor is assigned to each member based on its score. This weighting factor determines the statistical likelihood that the member will reproduce. For this algorithm, the weights were assigned as follows:

Table 5.1 Reproduction Weighting Based on Fitness Percentile

Percentile	Weight
80-100	0.7
60-80	0.6
40-60	0.5
20-40	0.4
0-20	0.3

To form the remainder of the next generation, a "father" and "mother" are chosen from the current population. The choice is a pseudorandom function (using MATLAB function randsample) based on the weight factor assigned to the members. This means that every member has the chance to reproduce, but the higher-scored members are more likely to be selected. When two parents are chosen, there is a 0.1% chance the child will be a mutation. This produces a new table with completely random values between -50N·m and 50N·m. If the mutation does not occur, a child is produced by averaging the values from the mother and father table. This is repeated until a new population of 20 members is formed.

5.2.1.4 Results of First Genetic Algorithm

The results of this first GA were poor. Upon the start of the optimization process, the GA converged quickly to the original lookup table used as the seed. While this result was undesired, two things can be easily seen from it.

1. The lookup table based on the weighted cost function is likely to be a local maximum in the optimization space.
2. The search space created by the GA was not large enough to explore beyond the local maximum, causing very quick convergence.

It was possible to extract some information on the evolution of the population from the data that was collected from the GA-based optimization. Figure 5.1 shows the evolution of the population's performance in the three selected metrics and the overall

score. It can be seen that, though the population has a fairly wide variance initially, it quickly converges to the values that were produced by the seed lookup table. Also, Figure 5.2 shows that the top performer does not change in each generation, and is the original seed lookup table. This means that the initial population and their children were not capable of moving further out of the local maximum and failed in reaching another maximum in the optimization space.

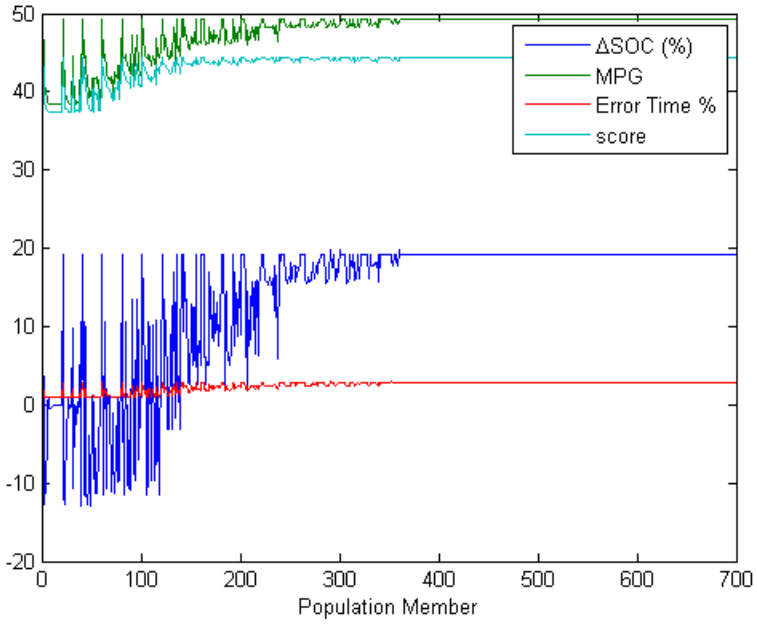


Figure 5.1 Evolution of Population Performance for First GA

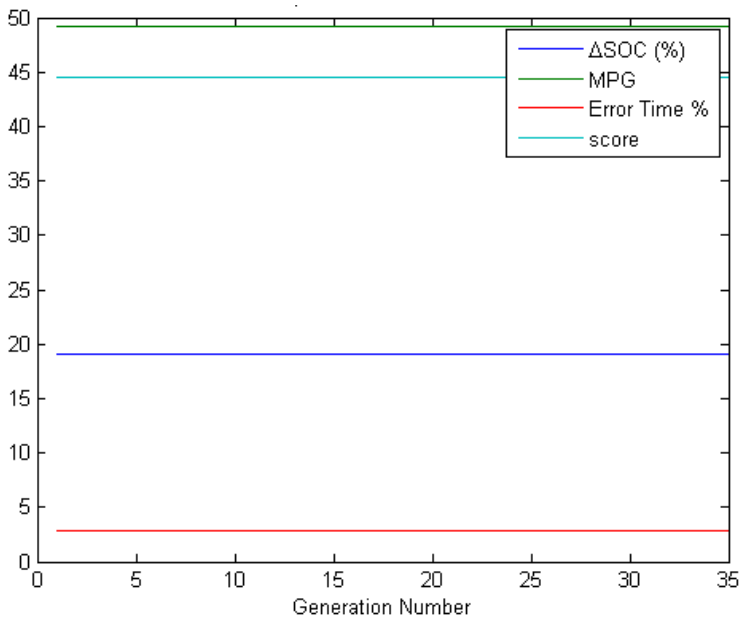


Figure 5.2 Evolution of Top Performers in First GA

5.2.1.5 Discussion of First Genetic Algorithm

The idea of having a "dominant" member in the initial population can be good because it can provide the GA with a good starting point in the optimization procedure. However, when every member of the population is derived in a similar way or all members are derived from the single dominant member, the population lacks diversity. This lack of diversity greatly reduces the potential search area of the GA within the optimization space.

Creating a fitness function that is a linear combination of various performance metrics is a common practice in GAs [25]. However, assigning a score of zero to any member outside a specific range on a given metric will cause the weighting method used in parent selection to be ineffective. In other words, if a member is only classified as

"bad", it is impossible to know how "bad" this member really is. If multiple members are classified as "bad", they are viewed the same by the weighted parent selection algorithm. This can cause the diversity of the population to be eliminated rapidly.

Averaging parents is not a proper technique in GAs. Children should be produced using a crossover method in which "cells" from the mother and father are assembled to form a new population member [26]. When averaging of parents is coupled with the aforementioned initial population and fitness function, it can be seen that convergence will occur quickly, and the search area will be very localized. Another weakness of the mating algorithm was the implementation of the mutation phenomenon. Mutation should occur within certain "cells" of a child rather than causing the creation of an entire mutant child [26].

Based on the information gained from this genetic algorithm, substantial changes were implemented in the next attempt. The new algorithm is described next.

5.2.2 Second Genetic Algorithm

The performance of the previous genetic algorithm called for the creation of a new algorithm. Several changes were made based on a literature review and observations of the results from the previous algorithm. Changes were made to the initial population, the fitness function, and the reproduction and mutation strategies. As a result, the outcome of the new algorithm became much better than that of the previous algorithm.

5.2.2.1 Form initial Population

It was concluded that the initial population from the first algorithm did not expand the search space enough to explore other potential local maxima in the optimization space. However, it was still desired to allow the characteristics of the weight function-based lookup table to be a part of the initial population pool. Therefore the first member of the population would remain the same (lookup table based on the weighted cost function), but the remainder of the population would be tables filled with completely random values within the range of -50N·m to 50N·m. Each remaining member of the population, $\boldsymbol{M}_i$, is formed as:

$$\boldsymbol{M}_{(m*n*p),i} = \boldsymbol{T}_{rand}|_{-50_{(m*n*p)}}^{50};\ i = 2 \dots 20 \tag{5.3}$$

Equation (5.3) shows how the initial population is formed for iteration 1 of the new genetic algorithm. Each consequent population is formed from step 3 of the previous iteration.

5.2.2.2 Perform Fitness Evaluation

It was considered that the method of using a weighted linear combination of the performance metrics as done previously would still be acceptable. However, a modification to the specific score function used previously could yield better results. Therefore, the scoring function is chosen as:

$$score_i = \begin{cases} If\ \Delta SOC > 0\ ,\ MPG - 5|\Delta SOC| - \%Error\ time \\ If\ \Delta SOC < 0,\ MPG - 50|\Delta SOC| - \%Error\ time \end{cases};\ i = 1 \dots 20 \tag{5.4}$$

This scoring function allows each member to receive a score relative to its performance rather than assigning a score of zero to any member that performs "poorly".

5.2.2.3 Create new population with Elite Passthrough, Reproduction, and Mutation

Again, the first step is to sort the members based on fitness score. Two members are passed through to the next generation, just as in the first GA. The weighting factors also remain the same.

Table 5.2 Reproduction Weighting Based on Fitness Percentile for Second GA

Percentile	Weight
80-100	0.7
60-80	0.6
40-60	0.5
20-40	0.4
0-20	0.3

To form the remainder of the next generation, a "father" and "mother" are chosen from the current population. The choice is a pseudorandom choice (using MATLAB function *randsample*) based on the weighting factor assigned to the members. This is where the similarities between the first and second algorithms end. As mentioned, the mating procedure of simply averaging the mother and father is not a proper technique. In GAs, it is desirable to have an organism made of "cells", which are then used to produce children [26]. Therefore, once the parents are selected, each one is divided into eight equal sized "cells", and a child is formed by combining the cells of each parent stochastically. This procedure is described by Equation (5.5).

$$Cell_{i,child} = \begin{cases} Cell_{i,mother} \\ Cell_{i,father} \\ avg(Cell_{i,mother}, Cell_{i,father}) \end{cases} \quad equal\ probability \qquad (5.5)$$

This means that, with equal probability, each cell of the child will be the same respective cell from the father or mother, or it will be an average of the mother's and father's corresponding cells. This is a slightly modified form of the uniform crossover method [26].

The method for introducing mutation into the population was also changed for the second GA. After further study, it was determined that mutation should occur at parts *within* the child instead of creating an entirely new child. Therefore, for each child, a probability of 1% was implemented for each value from the table to be swapped for a random value between -50N·m and 50N·m. This essentially guarantees that each child is subject to some degree of mutation since the size of the table is 27×16×11.

5.2.2.4 Results of Second Genetic Algorithm

Unlike the first GA, the second GA was able to produce a lookup table, which is very different from that created by the weighted cost function. These results can be found in Table 5.3.

Table 5.3 Optimization Results of Second GA

Performance Metric	Second GA	Seed Table	Percent Diff
MPG	48.28	49.21	-1.89%
ΔSOC	.09%	19.14%	-99.53%
Error Time%	1.02%	2.84%	-64.08%

The evolution of the population for the second GA is shown in Figure 5.3. When compared to Figure 5.1 of the first algorithm, it can be seen that even after 200 generations, the second GA is still exploring the optimization space while the first GA stopped searching after approximately 20 generations. This is due to the change in the mating and mutation algorithms, which nearly guarantee that a child will never be the same as either of its parents. Also, it can be noted that the initial population of this algorithm is much more diverse than that of the first algorithm, which allows for a much larger initial search area within the optimization space.

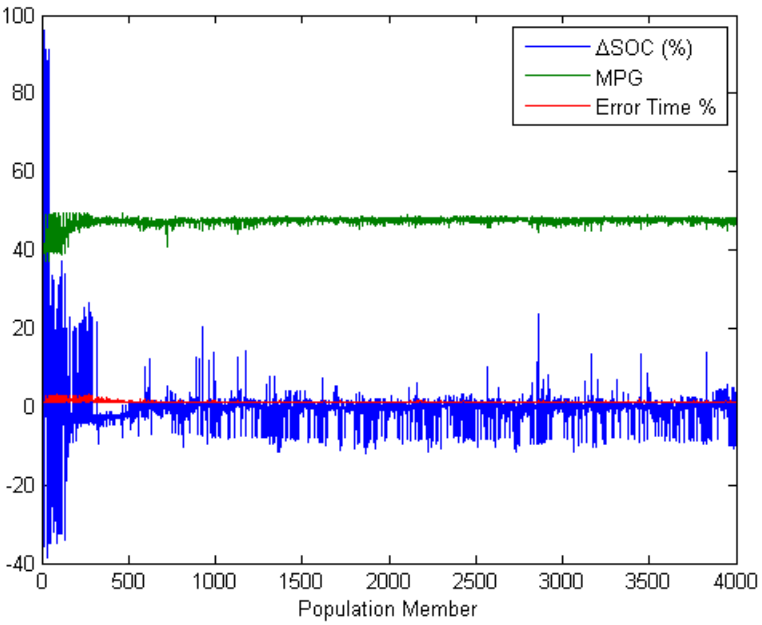


Figure 5.3 Evolution of Population of Second GA

Figure 5.4 shows a plot of the evolution of the top performer for the second GA. Unlike the case of the first GA, the performance does change from generation to generation, implying that the search space has expanded past the local maximum represented by the seed table. It can be seen as well that, though the increase in performance does not completely diminish until near generation 200, most of the performance gains were made in the first 40 generations. This may seem ideal, but as seen with the first GA, fast convergence in a complex optimization space could mean the algorithm is trapped in a local maximum.

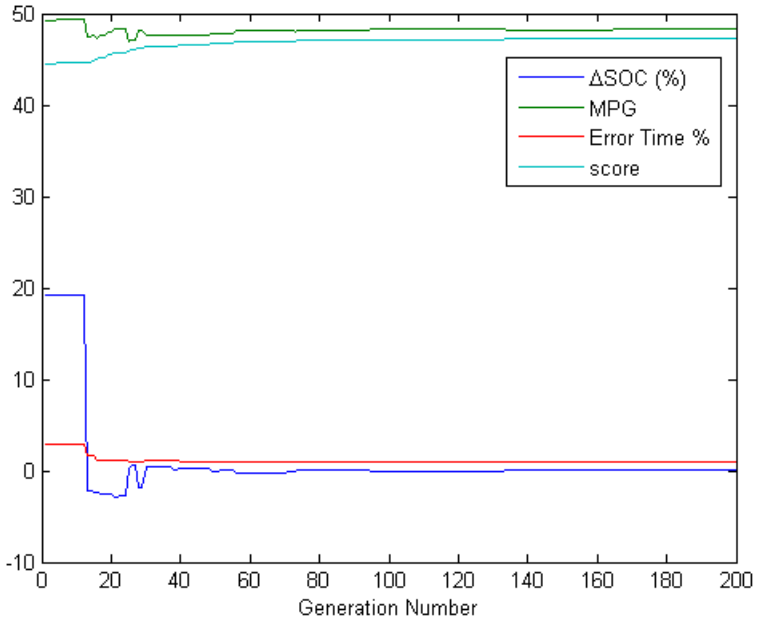


Figure 5.4 Evolution of Top Performers in Second GA

According to the fitness function scoring method, the top performer in generation 200 of the second GA receives a score of 47.5. In order to gain an understanding of why

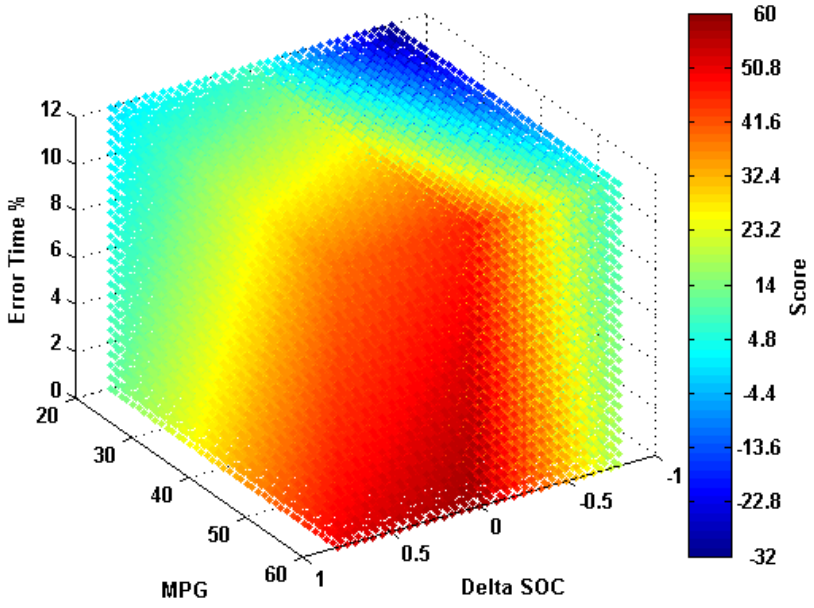


Figure 5.5 Fitness Score Function for Second GA

this score is reached, the scoring function in Equation (5.4) is plotted for ranges of ΔSOC, MPG, and Error Time%. The plot can be seen in Figure 5.5. In the plot, colored special points represent scores as shown in the color bar. Due to the conditionality of equation (5.4) being based on the sign of ΔSOC, the function is not symmetric across the $\Delta SOC = 0$ plane. The score of 47.5 achieved by the second GA is in the high range for this graph. However, it is shown that the score gradient in the MPG-ΔSOC plane is quite steep as $|\Delta SOC|$ increases (especially for negative values of ΔSOC). This means that the algorithm is relatively unwilling to trade an increase in ΔSOC for an increase in MPG. However, the score gradient in the MPG-Error Time% plane is less steep, meaning the algorithm is more willing to trade an increase in error for an increase in MPG. Both of these characteristics are shown in Table 5.3.

Another comparison was made between the seed lookup table and the algorithm's top performing table as shown in Figure 5.6. While the seed map is fairly "predictable" in its distribution of ISG torque values, the algorithm's top performing table has many sections that appear to be random. This is explained in Figure 5.8. For the FTP cycle,

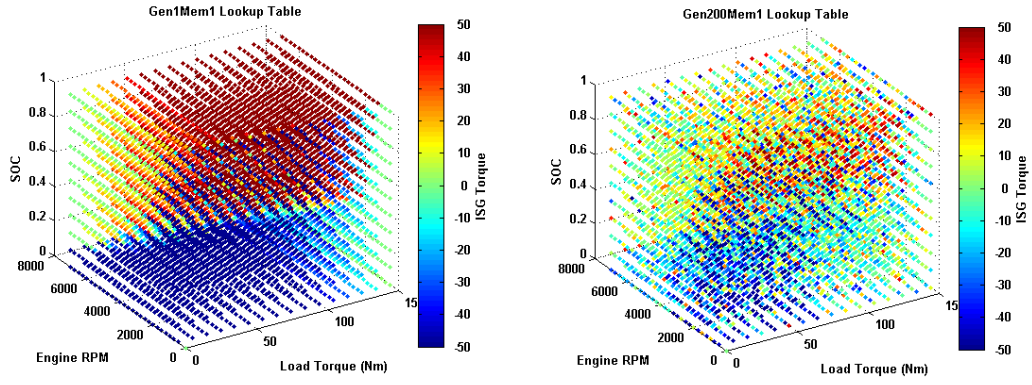


Figure 5.6 Seed Map and Top Performer Map for Second GA

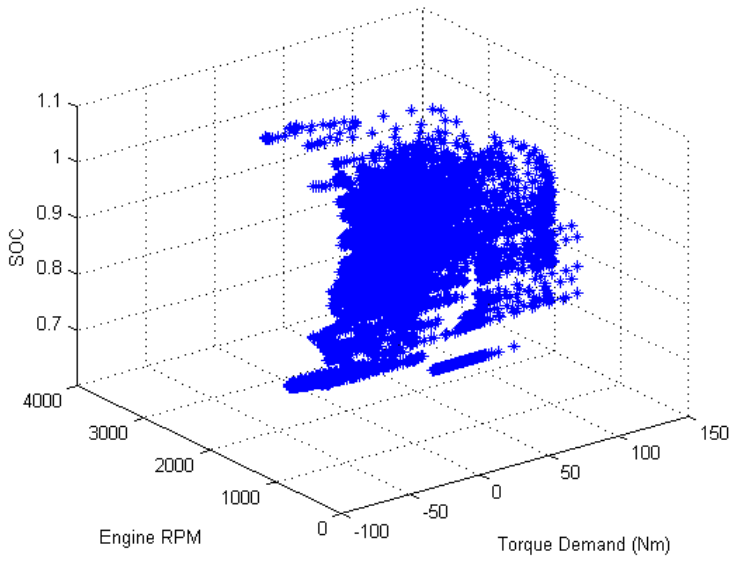


Figure 5.8 Vehicle Operating Points for Top Performer FTP Cycle Simulation

much of the regions in the lookup table are not entered by the vehicle. Therefore, the GA has no "reason" to change any of these values that are not utilized by the vehicle within the FTP drive cycle. So, it can be said that the GA "trains" an optimal lookup table for a particular drive cycle, which was the FTP drive cycle in this case.

The battery SOC versus time plot is very insightful in understanding how the lookup table changes from generation to generation. Figure 5.7 shows time traces of

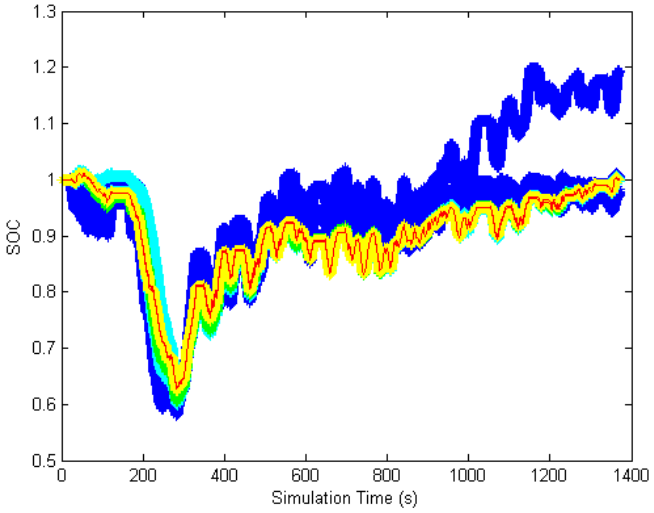


Figure 5.7 SOC vs Time for Second GA Top Performers

SOC for the top performers of generations from 1 to 200. The "cool" colors represent early generations, and the "warm" colors represent later generations. It can be seen that the final SOC path for the later generations remains very similar to that of the seed table from t = 0s to around t=300s. This implies that the weighted cost function method performs well in the operating points associated with this portion of the drive cycle.

5.2.2.5 Simulink Simulation Results of Top Performer for Second GA

For a detailed comparison between the GA-optimized table and the weighted cost function optimized table, the simulation results for the former are presented similarly to those in Chapter 4. The vehicle velocity profile for the FTP cycle can be seen in Figure 5.9 and the electric motor torque and SOC profiles can be seen in Figure 5.10.

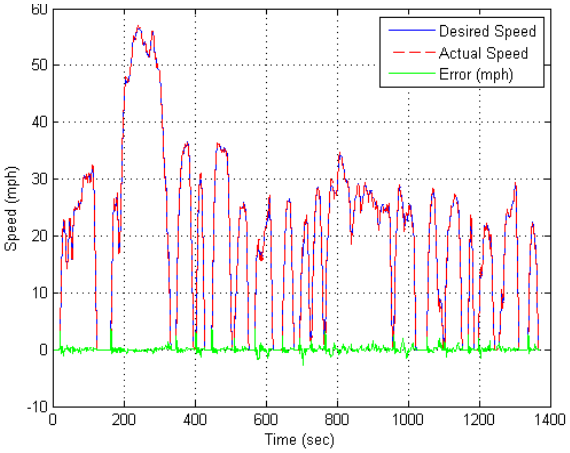


Figure 5.9 FTP Velocity Profile of Top Performer for Second GA

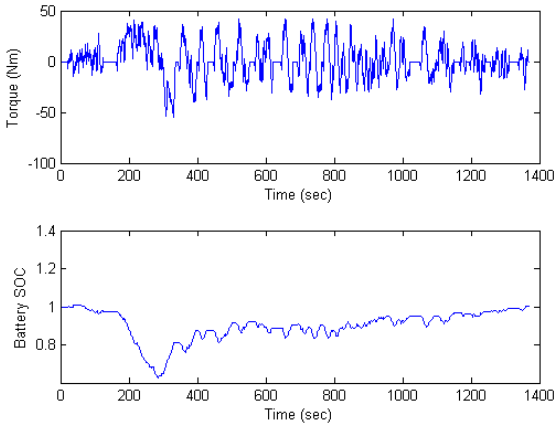


Figure 5.10 ISG Torque and SOC Profiles of Top Performer of Second GA

It can be seen that the lookup table created by the GA allows the SOC to drop to its lowest point during the longest portion of acceleration in the drive cycle (around at 200s). This is an effect of the GA being trained on the FTP cycle alone. The battery usage seems to be quite optimal for the FTP cycle, but if the battery SOC was diminished like this in an unknown drive cycle, it could result in complete battery depletion.

5.2.2.6 Performance Summary of Second GA

- The second GA was able to show a significant capability to optimize a lookup table for MHEV control.

- The second GA showed fast convergence rate, which could mean the algorithm narrows the initially wide search space too quickly.

- The fitness evaluation scoring function is relatively unwilling to trade an increase in ΔSOC for an increase in MPG, but is more willing to trade an increase in Error Time% for an increase in MPG.

- The lookup table generated by the genetic algorithms has been "trained" for the FTP drive cycle, which means it could have poor performance for a general unknown drive cycle.

CHAPTER 6

CONCLUSIONS AND FUTURE RESEARCH

This thesis presents two methods for the creation of an optimal lookup table for MHEV control. It should be noted that although this research is focused on MHEV technology, the same methods can be applied to traditional HEV architectures as well. In both methods, the lookup table is a 3D array that specifies an ISG torque value for a given ICE speed, drivetrain load torque, and battery SOC. The first method formulates a lookup table based on a weighted cost function. This method shows significant improvement over a simple rule-based control strategy. When considering the optimization space for an MHEV, a weighted cost function creates a simplified representation of a complex problem. Therefore, optimization techniques based on genetic algorithm were applied to explore the possibilities of even further improvement. After some tuning of the GA, optimization results show an even greater improvement in performance than that of the weighted cost function method. These results are summarized in Table 6.1.

Table 6.1 Final Results Summary

Simulated Model	**Fuel Economy(mpg)**	ΔSOC_{FTP}	**Error Time %**
Original ALPHA Model	36.16	NA	1.04%
Rule-Based Control	40.75	0.0%	.39%
Lookup Table-Based Control	47.40	+0.2%	6.54%
Lookup Table-Based Control with Rate Limiter	45.84	+3.0%	3.18%
Lookup Table Optimized with Genetic Algorithm	48.28	+.09%	1.02%

It is shown that the lookup table, optimized with genetic algorithm, results in greater fuel economy than the lookup table created by the weighted cost function without the rate limiting algorithm, and better drivability than the method with the rate limiter. This was also achieved while maintaining ΔSOC of essentially zero for the FTP cycle.

The genetic algorithm shows great promise as an optimization method for the creation of lookup tables for HEV supervisory control. However, further research should be conducted towards enhancing the GA used in this thesis. Increasing the number of initial population members would be a good start to test the capabilities of the current algorithm. There are also several parts of the current GA that could be altered in an attempt to achieve improved results.

Just like the weighted cost function in the first control method presented in Chapter 4, the scoring method used in the GA has a great influence over the results. It would be ideal to explore other scoring methods than those employed in this research. For instance, instead of scoring based on a linear combination of the performance

metrics, it may be possible to assign a score to the members for each individual metric, and use the three individual scores in the selection of mating pairs.

A technique common to GAs that was not explored in this thesis is dynamic mutation. This involves tracking the movement of "family trees" through the optimization space. When a family is far away from a maximum, that family is likely to produce large numbers of mutations in its children in order to widen the search space. On the contrary, when the family is approaching a maximum, the likelihood of mutation is decreased in order to ensure the maximum is not passed over [27].

In conclusion, genetic algorithms have shown to be a good choice for the optimization of control lookup tables for an MHEV. Their ability to search a large, complex solution environment makes them ideal for this type of problem [25]. However, there is much more work that could be done in searching for an ideal GA for HEV control.

REFERENCES

[1] M. Ehsani and Y. Gao, "Hybrid Drivetrains," in *Handbook of Automotive Power Electronics and Motor Drives*, Taylor & Francis Group, LLC, 2005.

[2] A. Lammers and M. Steinbuch, "Impact of Mild-hybrid Functionality on Fuel Economy and Battery Lifetime," Eindhoven University of Technology, Eindhoven, 2006.

[3] H. Kusumi, K. Yagi, Y. Ny, S. Abo, H. Sato, S. Furuta and M. Morikawa, "42V Power Control System for Mild Hybrid Vehicle," in *SAE World Congress*, Detroit, 2002.

[4] G. Fulks, G. Roth and A. Fedewa, "High Preformance Stop-Start System with Fourteen Volt Belt Alternator Starter," *SAE International Journal of Engines,* vol. 5, no. 3, 2012.

[5] R. Nesbitt, C. Muehlfeld and S. Pandit, "Powersplit Hybrid Electric Vehicle Control with Electronic Throttle Control," in *Powertrain and Fluid Systems Conference and Exhibition*, Pittsburg, 2003.

[6] G. Rousseau, D. Sinoquet and P. Rouchon, "Constrained Optimization of Energy Management for a Mild Hybrid Vehicle," *Oil and Gas Science and Technology,* vol. 62, no. 4, pp. 623-634, 2007.

[7] C.-C. Lin, S. Jeon, H. Peng and J. M. Lee, "Driving Pattern Recognition for Control of Hybrid Electric Trucks," University of Michigan, Seoul National University.

[8] C.-C. Lin, H. Peng, J. Grizzle and J.-M. Kang, "Power Management Strategy for a Parallel Hybrid Electric Truck," 2002.

[9] M. B. Arnolds and M. Steinbuch, "Series Hybrid Powertrain Optimization and Control," Eindhoven Univeristy of Techology, Eindhoven, 2005.

[10] D. Bertsekas, Dynamic Programming and Optimal Control, Belmont: Athena Scientific, 2005.

[11] I. Arsie, M. Graziosi, C. Pianese, G. Rizzo and M. Sorrentino, "Optimization of Supervisory Control Strategy for Parallel Hybrid Vehicle with Provisional Load Estimate," in *Advanced Vehicle Control (AVEC) Symposium*, 2004.

[12] M. Koot, J. Kessels and B. de Jager, "Fuel Reduction of Parallel Hybrid Electric Vehicles," *IEEE*, 2005.

[13] E. Stromberg, A. Froberg and L. Nielsen, "Optimal Control of Hybrid Electric Vehicles," Linkopings Universitet, 2003.

[14] A. Konev, L. Lezhnev and I. Kolmanovsky, "Control Strategy Optimization for a Series Hybrid Vehicle," in *SAE World Congress*, Detroit, 2006.

[15] N. J. Schouten, M. A. Salman and N. A. Kheir, "Fuzzy Logic Control for Parallel Hybrid Vehicles," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 3, pp. 460-468, 2002.

[16] B. Glenn, G. Washington and G. Rizzoni, "Operation and Control Strategies for Hybrid Electric Automobiles," *SAE Technical Paper Series*, 2000.

[17] F. Wang, X. Mao and B. Zhuo, "Integrated Starter Generater Hybrid Electric Car Torque Distribution Control," in *SAE International Powertrains, Fuels, and Lubricants Congress*, Detroit, 2008.

[18] J. Ormerod and P. Fussey, "Development of a Control System for a Mild Hybrid Vehicle," in *International Workshop on Modeling, Emissions, and Control in Automotive Engines* , Salemo, 2001.

[19] H.-J. Yoon and S.-J. Lee, "An Optimized Control Strategy for Parallel Hybrid Electric Vehicle," in *SAE World Congress*, Detroit, 2003.

[20] E. P. Agency. [Online]. Available: http://www.epa.gov/oms/climate/documents/420b12051.pdf.

[21] O. Tremblay and L. Dessaint, "Experimental Validation of a Battery Dynamic Model for EV Applications," *World Electric Vehicle Journal*, vol. 3, 2009.

[22] G. Marangoni and A. Beghi, "Battery Management System for Li-ion Batteries in Hybrid Electric Vehicles," University of Padova, 2010.

[23] J. E. Walters, R. J. Krefta, G. Gallegos-Lopez and G. T. Fattic, "Technology Considerations for Belt Alternator Starter Systems," in *SAE World Congress*, Detroit, 2004.

[24] A. Emadi and e. al., Handbook of Automotive Power Electronics and Motordrives, Taylor & Francis Group, LLC, 2005.

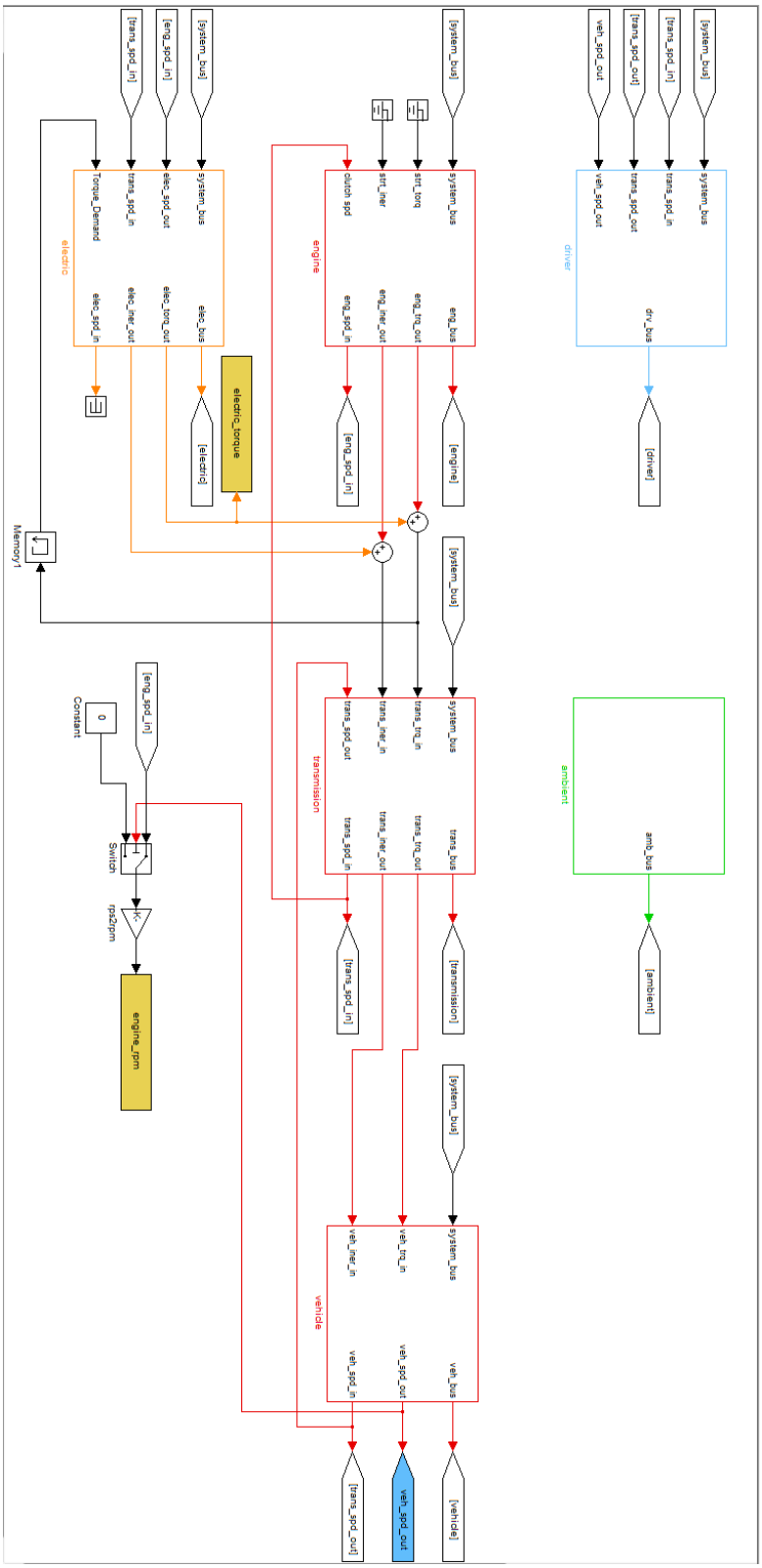
[25] M. Mitchell, An Introduction to Genetic Algorithms, Cambridge, MA: MIT Press, 1999.

[26] K. Sastry, D. Goldberg and G. Kendall, "Genetic Algorithms," pp. 97-125.

[27] D. Thierens, "Adaptive Mutation Rate Control Schemes in Genetic Algorithms," *IEEE*, no. 4, 2002.

APPENDIX

MHEV Simulink Model



Weighted Cost Function Simulation MATLAB Script

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
% Jeff McGehee -- Thesis Code Weight Function Optimization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
warning off
clc
clear all



% Safety Feature =======================
kph2mph = 0.621371192;
% ======================================


%-----------------------------------------------------------------------
---------
% UNIT CONVERSION FACTORS & SIMULATION PARAMETERS
%-----------------------------------------------------------------------
---------

% Determine Unit Conversion Factors
% ... mile to meter
mil2mtr = 1609.344;
% ... m/s to mph
mps2mph = 3600/mil2mtr;
% ... rad/s to RPM
rps2rpm = 30/pi;
% ... RPM to rad/s
rpm2rps = pi/30;
% ... kg to lb
kg2lb   = 2.20462262;
% ... lb to ton
lb2ton  = 0.0005;
% ... lbf to N
lbf2N   = 4.4482216283;
% ... kg to ton
kg2ton  = kg2lb*lb2ton;
% ... gallon to liter
gal2lit = 3.78541178;
% ... gallon/mile to CO2/mile (for gasoline)
gpm2CO2 = 8887;

% Determine Simulation Parameters
% ... simulation step time (sec)
sim_step_time = 0.01;
% ... simulation stop time (sec)
sim_stop_time = inf;


%-----------------------------------------------------------------------
---------
```

```
% VEHICLE MODEL PARAMETERS (Common Parameters Only)
%-------------------------------------------------------------------------
---------



% Download Vehicle Driving Cycle
% ... vehicle speed in m/sec
load('drive_cycles\FUDS')

% Define Vehicle Simulation Parameters
% ... simulation step time
sim_step_time = 0.01;
% ... simulation stop time
sim_stop_time = sch_drive(end,1);

% Download Ambient System Parameters
% ... call subroutine 'ambient_param'
run('param_files\ambient_param')

% Download Driver System Parameters
% ... call subroutine 'driver_param'
run('param_files\driver_param')

% Download Electrical System Parameters
% ... call subroutine 'electric_param'
run('param_files\electric_param')

% Download Engine Parameters
% ... call subroutine 'engine_base_smcar_param'
run('param_files\engine_base_smcar_param')

% define transmission parameters
run('param_files\transmission_dct_smcar_param')

% ... vehicle parameters
run('param_files\vehicle_smcar_param')

% Define Other Vehicle Simulation Parameters
% ... default A/C flag (off)
ac_load     = 0;
% ... default road-load flag (based on Cd/Crr)
veh_cstdwn  = 0;
% ... default start-stop flag (off)
%veh_spd_ss  = -100;     % vehicle speed when engages (m/s)
veh_spd_ss = 1/mps2mph;  % off_cycle_technology
sim_time_ss = 0;        % simulation time when engages in FTP cycle
ftp = 1;
veh_type = 0;

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

engine.cyl.fuel_torq     = [0:10:150]; %"rows" of fuel map
```

```
engine.cyl.fuel_spd      = [500:250:7000]; %"columns" of fuel map
engine.cyl.fuel_map      = (1e-3)*[...
    0.137784200794720   0.145429451947655   0.156382680284332
0.173195524080152   0.197393944164130   0.229105801613709
0.266408987860409   0.305130814888616   0.343081937033314
0.382667585950033   0.428626905825282   0.487485348742466
0.562730476619842   0.654276353564051   0.723948053534848
0.777367045220901   0.825399680423211   0.871535295899542
0.911467109480377   0.936528518075875   0.943348062327588
0.939923034520763   0.930979400432104   0.914596010154164
0.889888507065861   0.859158115297706   0.825884661086600
    0.178447245473081   0.186174126898410   0.193821693549619
0.229281947250495   0.268944346499218   0.313221147347984
0.367200490699964   0.421324166935333   0.464963942931740
0.518140982067324   0.573337548121601   0.641237195689641
0.709889742822316   0.832743172265829   0.900896106352274
0.968849257150545   1.041808370322380   1.115826093728330
1.189797263528230   1.257572907557640   1.309593850001830
1.357208749890870   1.387801874156830   1.398651682913730
1.395275776343720   1.383383357349100   1.367833022295900
    0.215802312967702   0.227675140757502   0.232286384261295
0.286766470172409   0.342544718480230   0.399587072529394
0.466008651188919   0.532922941661500   0.584512036299191
0.651828070287716   0.721086080583067   0.798562540448432
0.895137999171677   0.967413596238694   1.072599276057370
1.167708484123430   1.266013179148820   1.364476712477430
1.462832273844690   1.567248545395710   1.679110964184610
1.781141580945120   1.843587554595180   1.879521774921710
1.899484443515190   1.908545363194440   1.912324445948160
    0.246620740868960   0.268858136962213   0.291117601139029
0.355791144234713   0.422327372456642   0.490861852133899
0.579289762332647   0.652494711172791   0.713269471414382
0.788619169818959   0.874527372443634   0.965943278532812
1.063031757575180   1.174347325903810   1.283107645944950
1.390474838983970   1.502968037530540   1.626890781391480
1.728687200348660   1.833278845101550   2.079135123482510
2.208474450185690   2.289060508392720   2.353829102706320
2.402186513010150   2.436781576935140   2.465559910544890
    0.274935451053239   0.309289971000674   0.353352638875700
0.429296322037564   0.506099304185812   0.586009212724862
0.674286140957250   0.763885607584720   0.845872137182697
0.948133556209927   1.033949122827310   1.148932016719510
1.268578859931220   1.380420597701390   1.512599998106280
1.643227591125230   1.770103872234600   1.911689231950220
2.013959924917270   2.157710506594280   2.549376938317960
2.593134549245770   2.717477490391370   2.823666881567500
2.902503880088570   2.970567517735820   3.039022231513130
    0.304757297313362   0.352879740632899   0.422461321225047
0.504239569899791   0.593142849311854   0.686766441223438
0.786578210447621   0.885302157554547   0.997599700308058
1.093522527456920   1.207044244904640   1.339597480142180
1.475226927668060   1.607205292426290   1.744243249961790
1.899194123966910   2.039364976278560   2.192998339790720
2.332544283963070   2.500968277496060   3.020629848357870
2.923876948337440   3.155624780152180   3.287885769057890
3.394261661577790   3.514507590509400   3.650010954595190
```

0.330388985514560 0.407733087924073 0.491827880073060
0.579254351939491 0.683298097608957 0.802209736046632
0.905827189839588 1.032749466446620 1.148711689035400
1.257506974257530 1.381859775595540 1.530231956614420
1.681660035126600 1.832789569508600 1.986580439582240
2.155074922058620 2.309881595705230 2.520783784504530
2.738797312858450 2.853633432823880 3.492641409037500
3.267192376755820 3.540748619734460 3.719558548263370
3.876699569082420 4.084761713302820 4.315434549403270
0.324674084249453 0.463278348107089 0.560757022805280
0.662916783769039 0.782083710191668 0.916731801187185
1.040099403142220 1.168994199887980 1.290051101960990
1.426445417299760 1.566365718576840 1.721165535973930
1.888065847833100 2.058332612758290 2.234181838745480
2.410997202049000 2.584932222478870 2.900842491735900
3.191707973961970 3.309509246034520 3.966336715260320
3.613435646151410 3.868940017780900 4.125604615995560
4.387425048262150 4.714134638059620 5.036944050395440
0.253705472371187 0.513749904635680 0.629335259257198
0.750799106387447 0.881562721845528 1.020904167467550
1.156110910767710 1.305822836440080 1.464803965447070
1.597709058815800 1.753485905666100 1.925325665865860
2.101382388934410 2.286058014393740 2.481890629435960
2.675452941505540 2.861014968159040 3.280760543879490
3.611815552628840 3.777322272990910 4.443392206050360
4.208780558379180 4.376304333624880 4.691237401652390
5.064946007409210 5.429731377082080 5.793579800149720
0.124701617892350 0.327991786280844 0.696165142484334
0.840349653736285 0.986477541796325 1.130945665849060
1.285154225678900 1.441279444274440 1.607940231103820
1.769845055500050 1.943992184839290 2.134216328975160
2.333772885054880 2.528657497595260 2.732841554318950
2.946329489158050 3.163627712192770 3.661727664839750
4.044542892172460 4.274756997605060 4.919496941656410
4.821501746937240 5.025242320557140 5.375191677086350
5.793978274457150 6.185967430218310 6.568282318705810
0.089340066468506 0.407366980896065 0.766815105317886
0.929174852469753 1.104153709982370 1.269156919947450
1.424769434842180 1.584038805150390 1.761163570903930
1.946944766916970 2.138645657582700 2.370520222442560
2.568320568670940 2.775161763116810 3.013263411184590
3.219514430400780 3.512632414866040 4.043543594670780
4.478808101621810 4.772399313796620 5.395031114996550
5.433593305980850 5.677474224341020 6.077794152446380
6.542134859828310 6.944307640189880 7.344929072708620
0.134415175940221 0.472429569933182 0.782578906139948
1.020111989448310 1.222868549378480 1.407634158781170
1.587483873439580 1.752594973360310 1.949533153564860
2.145956143911230 2.354252601518090 2.614380514828010
2.803226245267630 3.021500004130710 3.294179213223660
3.495588924471920 3.862263271215460 4.424993801523920
4.905620443701080 5.271975828740250 5.870606153252860
6.045386096996220 6.329329725634240 6.769688694471050
7.220427100598010 7.673665756780660 8.117071796076750
0.205300094153800 0.523664432102514 0.820235085333885
1.089365959139160 1.327151516354810 1.545902874473060
1.763231542320510 1.948546013802310 2.154494401514830

```
2.374110650970730   2.600507372574610   2.861880121727730
3.045443147135970   3.268615576228110   3.574818496704490
3.772688828366140   4.214560745221630   4.806538330818020
5.331471317798090   5.753322539134660   6.345574312588370
6.637165615099130   6.985500772304380   7.458645337463510
7.925540836052870   8.402020761109140   8.879813570571630
    0.275233026741741   0.578442583440395   0.873971258650645
1.158353604767640   1.425826001454500   1.678082604389820
1.918483008752390   2.144418941539820   2.372757366259830
2.611308355320650   2.855165174673810   3.115667804819580
3.325158078917280   3.557041889293750   3.846525959834990
4.112967455649060   4.585807582447910   5.187970167323680
5.757927438837380   6.231462028264860   6.820202657586520
7.216498663015810   7.657208538529570   8.140046208650470
8.634840564862280   9.135186150465050   9.638090094841280
    0.339245864053882   0.637987449974570   0.936234484295948
1.232275114903380   1.522415003886300   1.803378502949640
2.073519600547620   2.334654498976310   2.589996708499750
2.844117636086850   3.097348000514730   3.347911998351400
3.590608006267930   3.847781075166680   4.126351084584140
4.449089704491610   4.957539091657710   5.569114688987870
6.184373683863820   6.750479224839140   7.300522930698100
7.811053203608610   8.318152201553810   8.830653707253650
9.349795820792950   9.872016764483650   10.39552418312740
    0.400051377928469   0.699307023849234   1.001769993207550
1.308872902100500   1.618776083358820   1.926963592751930
2.229147409008490   2.522750319614730   2.804603895391230
3.073679443956640   3.331580428278010   3.583157784218860
3.841212845720280   4.115545354387770   4.430800139725080
4.834596937887820   5.349931554639790   5.950414811191120
6.588996574979650   7.218471112904150   7.813456045773990
8.399594302907840   8.969915971660040   9.523464580678500
10.06846355152910   10.61093293095620   11.15318029089840];
%fuel flow in kg/s



ISG_map = zeros(5797,4);
ISG_map1 = zeros(27,16,11);

W1 = 1;



%%%%%%%%%%%%%%%For Loop Arrays%%%%%%%%%%%%%%%
W2_a = 7.50e-6:-.01e-6:7.4e-6;
[r_W2,c_W2] = size(W2_a);

rpm_a = 500:250:7000;
[r_rpm,c_rpm] = size(rpm_a);

load_torq_a = 0:10:150;
[r_load_torq,c_load_torq] = size(load_torq_a);

SOC_a = 0:.1:1;
```

```
[r_SOC,c_SOC] = size(SOC_a);

isg_torq_a = -50:2.5:50;
[r_torq,c_torq] = size(isg_torq_a);


for run_idx = 1:c_W2;
    W2 = W2_a(run_idx);
    tic
    for a = 1:c_rpm;%600:50:2200;
        rpm = rpm_a(a);
        for b = 1:c_load_torq;% = 0:10:150;%0:20:3000;
            load_torq = load_torq_a(b);
            for d = 1:c_SOC% = 0:.1:1
                SOC = SOC_a(d);

                for k = 1:c_torq
                    isg_torq = isg_torq_a(k);
                    fuel_flow = interp2(engine.cyl.fuel_spd,
engine.cyl.fuel_torq,engine.cyl.fuel_map,rpm,load_torq-isg_torq);
                    costfun(k) = W1*fuel_flow + W2*isg_torq/(SOC);

                end

                [c,i] = min(costfun);
                etorq = isg_torq_a(i);



                ISG_map(d,1) = load_torq;
                ISG_map(d,2) = rpm;
                ISG_map(d,3) = SOC;
                ISG_map(d,4) = etorq;

                ISG_map1(a,b,d) = etorq;




            end
        end
    end


    sim('ALPHA_v1p0_mod')

    toc
    %-------------------------------------------------------------------
-------------
    % POST-PROCESSING
    %-------------------------------------------------------------------
-------------
```

```
    % Perform Post-Processing of Simulation Results
    % ... calculate percent time trace missed by 2 mph [%]
    veh_spd_err      = (veh_spd_des-veh_spd_out)*mps2mph;
    num_err_plus2mph = find(veh_spd_err >= 2);
    per_err          =
length(num_err_plus2mph)/length(veh_spd_err)*100;
    % ... calculate total fuel consumption [mpg]
    fuel_mile_total  =
sum(veh_spd_out)/(sum(engine_fuel_flowrate)*engine.cyl.fuel_rate_cal);
    mpg_total        =
fuel_mile_total*(engine.cyl.fuel_desity*gal2lit/mil2mtr);
    % ... calculate CO2 [g/mile]
    CO2_total        = gpm2CO2/mpg_total;


    % figure(3)
    % plot3k(ISG_map(:,1:3),'ColorData',ISG_map(:,4))
    % xlabel('Torque')
    % ylabel('RPM')
    % zlabel('SOC')


    map_categories = char('rpm range for ISG Map',...
        'torque demand range for ISG Map',...
        'SOC range for ISG Map',...
        'ISG Torque range for ISG Map',...
        'costfun(k) = W1*fuel_flow + W2*isg_torq(k)/(SOC);');

    cats = cellstr(map_categories);

    map_stats  = cell(5);

    for i = 1:size(cats)
        map_stats(i,1) = cats(i);
    end
    map_stats(1,2) = num2cell(rpm_a(1));
    [r,c] = size(rpm_a);
    map_stats(1,3) = num2cell(rpm_a(c));

    % map_stats(2,1) = 'torque demand range for ISG Map';
    map_stats(2,2) = num2cell(load_torq_a(1));
    [r,c] = size(load_torq_a);
    map_stats(2,3) = num2cell(load_torq_a(c));

    % map_stats(3,1) = 'SOC range for ISG Map';
    map_stats(3,2) = num2cell(SOC_a(1));
    [r,c] = size(SOC_a);
    map_stats(3,3) = num2cell(SOC_a(c));

    % map_stats(4,1) = 'ISG Torque range for ISG Map';
    map_stats(4,2) = num2cell(isg_torq_a(1));
    [r,c] = size(isg_torq_a);
    map_stats(4,3) = num2cell(isg_torq_a(c));
```

```
    % map_stats(5,1) = 'costfun(k) = fuel_flow +
isg_torq(k)/(200000*SOC);';
    map_stats(5,2) = num2cell(W1);
    map_stats(5,3) = num2cell(W2);

    res_title = sprintf('Conv Results %d',run_idx);

save(res_title,'ISG_map','ISG_map1','map_stats','veh_spd_err','per_err'
,'mpg_total','CO2_total','total_sim_time','SOC')

end
```

Genetic Algorithm Simulation MATLAB Script

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
%Jeff McGehee 3/6/2013
%This script file manages the entire simulation operation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
clc
clear all
close all
load_system('ALPHA_v1p0_par')
numgen = 1000; %Number of Generations to simulate
nummem = 20; %Number of Members per generation
numkeep = 2; %Number of top scores to keep each generation

%         weights(1:round(nummem/3))=.3;
%         weights(round(nummem/3)+1:round(2*nummem/3)) = .3;
%         weights(round(2*nummem/3)+1:nummem) = .3;

for i = 0:round(nummem/2)-1
    if i==0 || i==1
        weights(round(i*nummem/10)+1:round((i+1)*nummem/10))=.7;
    elseif i==2 || i==3
        weights(round(i*nummem/10)+1:round((i+1)*nummem/10))=.6;
    elseif i>3&&i<6
        weights(round(i*nummem/10)+1:round((i+1)*nummem/10))=.5;
    elseif i==6||i==7
        weights(round(i*nummem/10)+1:round((i+1)*nummem/10))=.4;
    elseif i==8||i==9
        weights(round(i*nummem/10)+1:round((i+1)*nummem/10))=.3;
    end
end


if length(weights)~= nummem;
    disp('Weight array not same size as # of gen members')
    return
end

%% Prepare 1st Generation %%
%Lines below commented in order to restart code at gen 46
%load('Base_ISG');%ISGgen1mem1_3_6_2')
%save('ISGgen1mem1','ISG_map1')

% for L=2:nummem
%     %ISG_map1=rand(27,16,11).*ISG_map1; %Random multiple of original
table
%     ISG_map1 = 50.*(rand(27,16,11)-rand(27,16,11)); %Completely
Random Table
%     savestring = sprintf('ISGgen1mem%d',L);
%     save(savestring,'ISG_map1')
% end

%% Begin GA %%
for K = 46:numgen
```

```
    K
    messagestring = sprintf('Your genetic algorithm has reached
generation %d',K);
    %sendmail('jlmcgehee21@gmail.com',messagestring,'I think I can...')
    if rem(K,10) == 0
        flag = isnet();
        if flag == 1
            send_text_message('931-628-2298','AT&T',messagestring);
        end
    end
    for L = 1:nummem
        L
        loadstring = sprintf('ISGgen%dmem%d',K,L);
        load(loadstring)
        genKmemL(K,L,ISG_map1)
    end
    %% Reproduction %%
    for L=1:nummem
        loadstring=sprintf('score_gen%dmem%d',K,L);
        load(loadstring)
        scores(L) = score;
    end

    %% Sort Tables by Score %%
    [Y,I] = sort(scores,'descend');

    scoremat(:,1) = I;
    scoremat(:,2) = Y;

    %% Keep highest Scoring Tables %%
    for i=1:numkeep
        loadstring = sprintf('ISGgen%dmem%d',K,scoremat(i,1));
        load(loadstring)
        savestring = sprintf('ISGgen%dmem%d',K+1,i);
        save(savestring,'ISG_map1')
    end

    %% Mate Tables %%
    for i=numkeep+1:nummem



        male = randsample(scoremat(:,1),1,true,weights);
        female = randsample(scoremat(:,1),1,true,weights);

        loadstring = sprintf('ISGgen%dmem%d',K,male);
        load(loadstring)
        male_map = ISG_map1;
        loadstring = sprintf('ISGgen%dmem%d',K,female);
        load(loadstring)
        female_map = ISG_map1;
```

```
        dom = randsample([1 0], 1, true,[.5 .5]); %Ensure child is same
size as parents
        if dom==1
            child= male_map;
        else
            child= female_map;
        end

        dim1 = [(1:13);(14:26)];
        dim2 = [(1:8);(9:16)];
        dim3 = [(1:5);(6:10)];

        qq=0;
        for ii = 1:2
            for jj = 1:2
                for kk = 1:2
                    qq = qq+1;

                    male_block =
male_map(dim1(ii,:),dim2(jj,:),dim3(kk,:));

                    female_block =
female_map(dim1(ii,:),dim2(jj,:),dim3(kk,:));

                    if qq==1
                        male_block1 = male_block;
                        female_block1 = female_block;
                    elseif qq==2
                        male_block2 = male_block;
                        female_block2 = female_block;
                    elseif qq==3
                        male_block3 = male_block;
                        female_block3 = female_block;
                    elseif qq==4
                        male_block4 = male_block;
                        female_block4 = female_block;
                    elseif qq==5
                        male_block5 = male_block;
                        female_block5 = female_block;
                    elseif qq==6
                        male_block6 = male_block;
                        female_block6 = female_block;
                    elseif qq==7
                        male_block7 = male_block;
                        female_block7 = female_block;
                    else
                        male_block8 = male_block;
                        female_block8 = female_block;
                    end
                end
            end
        end
        qq = 0;
        for ii = 1:2
            for jj = 1:2
```

```
                for kk = 1:2
                    qq = qq+1;

                    if qq==1

                        dom = randsample([1 0 -1], 1, true,[.5 .5 .4]);
                        if dom==1
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
male_block1;
                        elseif dom==0
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
female_block1;
                        else
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
(male_block1+female_block1)./2;
                        end
                    elseif qq==2
                        dom = randsample([1 0 -1], 1, true,[.5 .5 .4]);
                        if dom==1
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
male_block2;
                        elseif dom==0
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
female_block2;
                        else
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
(male_block2+female_block2)./2;
                        end
                    elseif qq==3
                        dom = randsample([1 0 -1], 1, true,[.5 .5 .4]);
                        if dom==1
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
male_block3;
                        elseif dom==0
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
female_block3;
                        else
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
(male_block3+female_block3)./2;
                        end
                    elseif qq==4
                        dom = randsample([1 0 -1], 1, true,[.5 .5 .4]);
                        if dom==1
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
male_block4;
                        elseif dom==0
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
female_block4;
                        else
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
(male_block4+female_block4)./2;
                        end
                    elseif qq==5
                        dom = randsample([1 0 -1], 1, true,[.5 .5 .4]);
                        if dom==1
```

```
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
male_block5;
                        elseif dom==0
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
female_block5;
                        else
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
(male_block5+female_block5)./2;
                        end
                    elseif qq==6
                        dom = randsample([1 0 -1], 1, true,[.5 .5 .4]);
                        if dom==1
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
male_block6;
                        elseif dom==0
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
female_block6;
                        else
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
(male_block6+female_block6)./2;
                        end
                    elseif qq==7
                        dom = randsample([1 0 -1], 1, true,[.5 .5 .4]);
                        if dom==1
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
male_block7;
                        elseif dom==0
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
female_block7;
                        else
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
(male_block7+female_block7)./2;
                        end
                    else
                        dom = randsample([1 0 -1], 1, true,[.5 .5 .4]);
                        if dom==1
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
male_block8;
                        elseif dom==0
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
female_block8;
                        else
                            child(dim1(ii,:),dim2(jj,:),dim3(kk,:))=
(male_block8+female_block8)./2;
                        end
                    end
                end
            end
        end

        for ii = 1:27
            for jj = 1:16
                for kk = 1:11
                    mutation_prob = 1e-2;
                    mutation = randsample([1 0],1,true,[mutation_prob
1-mutation_prob]);
```

```
                    if mutation == 1
                        child(ii,jj,kk) = 50.*(rand-rand);
                    end
                end
            end
        end


        ISG_map1 = child;

        savestring = sprintf('ISGgen%dmem%d',K+1,i);
        save(savestring,'ISG_map1')
    end

end
```