Supervisory Control Validation of a Fuel Cell Hybrid Bus Using Software-in-the-Loop and Hardware-in-the-Loop Techniques

THESIS

Presented in Partial Fulfillment of the Requirements for the Degree Master of Science in the Graduate School of The Ohio State University

By

Steven Abraham Ramirez

Graduate Program in Mechanical Engineering

The Ohio State University

2013

Master's Examination Committee:

Dr. Shawn Midlam-Mohler, Advisor

Dr. Yann Guezennec

Abstract

The work presented within this thesis consists of the validation of a supervisory controller and vehicle simulator for the ECO Saver IV demonstration bus being developed as part of the National Fuel Cell Bus Program (NFCBP). The goal of the NFCBP is to develop fuel cell transit buses such that a U.S. industry for fuel cell bus technology can be established through both technology innovation and increased public awareness of fuel cell vehicles. The use of fuel cells in vehicles is desirable due to their high efficiencies and zero emissions, allowing the transportation sector to rely less heavily on petroleum and carbon based fuels that emit hazardous greenhouse gases. The ECO Saver IV, as designed by the DesignLine Corporation through a contract with the Center for Transportation and the Environment, is a battery dominant fuel cell hybrid bus that takes advantage of the benefits of hybridization in conjunction with the benefits of the fuel cell. The team of researchers at The Ohio State University (OSU) Center for Automotive Research (CAR) served as a subcontractor to develop a supervisory controller and fuel cell hybrid bus simulator, modeled after the chosen powertrain architecture.

The validation performed involved the use of software-in-the-loop and hardware-in-the-loop simulations, where the results were compared to baseline model-in-the-loop simulations. The driving conditions of the intended application of the demonstration bus,

i.e., integration into the OSU Campus Area Bus Services (CABS) fleet, were taken into consideration through the development of real-world drive cycles that were representative of actual CABS bus routes. A new driver model was developed that solved issues related to tracking distance, velocity and road grade to enable the use of real-world drive cycles. The results of the validation are to be used in the final phases of development and construction of the ECO Saver IV fuel cell hybrid transit bus to prove the effectiveness of using the developed control algorithm within the bus' control hardware. To aid in the evaluation phase of the demonstration bus project, a CAN based data acquisition system was developed and tested on the HIL test bench. The logged data will be used to evaluate the successfulness of the fuel cell hybrid transit bus while providing evidence of the viability of such a vehicle.

Dedication

This document is dedicated to God, my family, and friends who have fueled my passion and taught me to strive to be the best I can be.

Acknowledgments

I would like to acknowledge those directly involved with the OSU project team, namely, my advisor Dr. Shawn Midlam-Mohler for his guidance, support, and trust through all the bumps along the way, Dr. Yann Guezennec for taking the time to ensure I was on the right track, Dr. Simona Onori for her vast knowledge concerning controls and optimization, Frank Ohlemacher for allowing me to help in the planning of the hydrogen fueling station, and my colleagues and friends Bharatkumar Hegde, Kyle Simmons, and Siddharth Mulay for the countless help they contributed during the completion of my research. A special thanks goes out to Dr. Giorgio Rizzoni for being the first Professor at The Ohio State University to believe in me and for recommending me for the project. I'd like to also acknowledge the contributions of the many members of NI and the NI forums for all the knowledge they shared. Many thanks to the OSU EcoCAR Team for letting me use their equipment, and to Tamal Mukherjee for allowing me to use his laptop so I could complete my work and for providing advice on HIL simulations. My gratitude also goes out to Andrew Spiegel and David Hillstrom for proofreading this thesis, and to the friends I've made at the Center for Automotive Research that made working here so much fun. Finally, I would like to thank all my friends and brothers in Christ for their support and prayers through my Graduate Studies without which I might not have made it.

Vita

2005 .....Elsik High

2010 .....B.S. Mechanical Engineering, University of Houston

2012 to present .....Graduate Research Associate, Center for Automotive Research, The Ohio State University

Fields of Study

Major Field: Mechanical Engineering

Table of Contents

List of Tables

List of Figures

CHAPTER 1: Introduction

1.1 Motivation

For years, the transportation industry has depended on carbon based fuels ranging from coal, gasoline, and diesel to power a wide array of vehicles, from passenger cars to transit buses. However, this dependency has led to many problems that affect the environment. These problems are usually caused by the toxic exhaust of conventionally powered vehicles. When the constituents of a vehicles exhaust, *i.e.* carbon dioxide, carbon monoxide, and nitrous oxide, among others, is allowed to interact with the environment, the result is air pollution, acid rain, soot, and often climate change due to the trapping of heat in Earth's atmosphere. In addition to these harmful effects on the environment, these toxins are a hazard to the health of humans and animals alike.

The destructive properties of carbon based fuel is not only limited to its use in vehicles but also from the process employed in obtaining these fuels. Petroleum drilling

has been known to have adverse effects on the immediate and surrounding drilling sites, especially when incidents like that of the 2010 BP oil spill occur. Unfortunately, the 2010 oil spill off of the Gulf Coast has not been the only time a drilling site has had a spill that endangered the wildlife and environment around it. Even without the consideration of the damage done by oil spills, the process of drilling often releases chemicals like methane directly into the atmosphere where it serves as a potent greenhouse gas.

Yet another problem with carbon based fuels is the dependence of foreign supplies of oil and gas. Supporters of a process called hydraulic fracturing or "fracking", in which natural gas is extracted from deep underground pockets of shale gas, claim that it will allow the US to decrease its dependence on foreign oil and gas while creating jobs that will boost the economy. Though the economic benefits claimed by supporters of this process may be true, they neglect the environmental and health problems prevalent in and around fracking sites as well as the eventual depletion of oil and gas, a problem that has come to be known as the energy crisis. While some may argue that petroleum is a long way from running out, the consequences of its continual use cannot be denied nor ignored any longer. [1], [2]

Research has been performed during recent years into several possible clean energy solutions to a substitute for gasoline and diesel powered vehicles. Most of these solutions are centered on the adoption of electric drive systems in vehicles, whether they be powered by batteries, super and ultra-capacitors, solar cells, or fuel cells. Much

of the focus has been on the development of vehicles that use energy storage devices such as battery electric vehicles (BEV). However, BEVs have been unable to overcome the limited range of operation before the need of recharging arises, as well as the long charging times needed to get BEVs back on the road. While some improvements have been made to the charging systems, there still exists an inherent use of the electric grid, which tends to be powered through the burning of coal.

To solve these issues, hybrid electric vehicles (HEV) were developed. HEV's typically use both a gasoline powered engine and batteries to power a vehicle. This overcomes several problems with battery powered vehicles and conventional vehicles, such as efficiency, mileage, range, and emissions. Nevertheless, HEVs still use petroleum and, if the vehicle is a plug-in hybrid electric vehicle (PHEV), possibly coal burned at electric power plants.

Out of the list of alternative propulsion devices, none has shown more promise than that of a fuel cell powered vehicle (FCV) for the development of a clean energy vehicle. [3] Fuel cells are electrochemical devices like batteries with the difference that they do not require charging but rather a fuel source. While some vehicles use hydrogen derived from hydrocarbons as fuel, others use hydrogen obtained through the electrolysis of water. The former essentially makes the fuel cell an energy production device whereas the latter is seen as another form of an energy storage device. As seen from Figure 1, the well-to-wheel analysis of various fuels for fuel cell vehicles, each has its advantages and disadvantages.

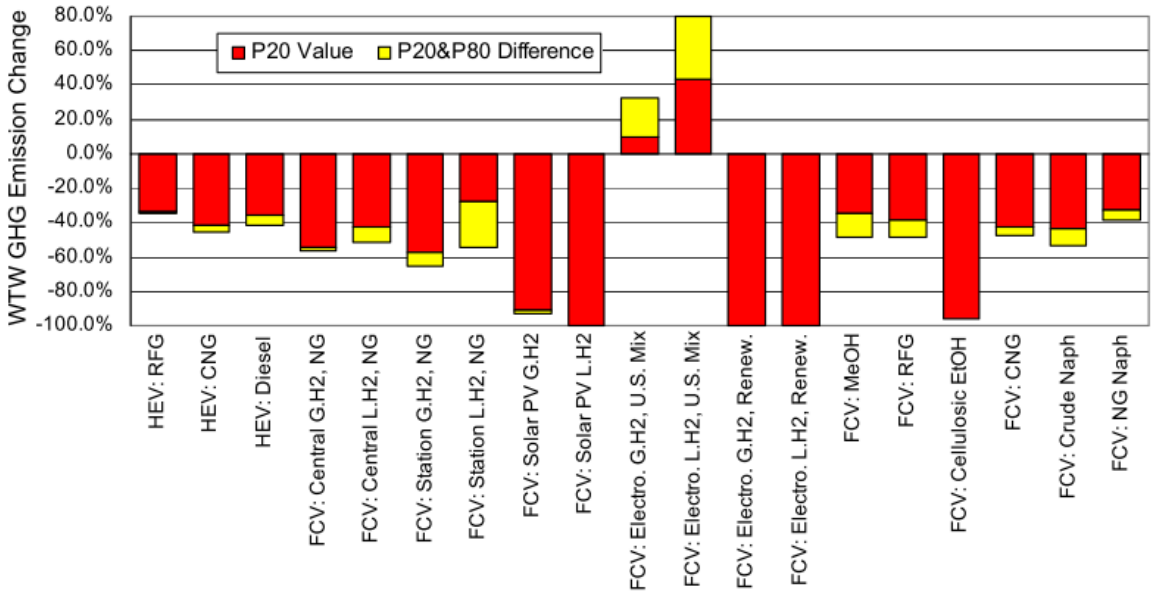

Figure 1: Well-to-Wheel Emissions Analysis of FCV Fuel Sources

Nevertheless, when hydrogen and oxygen, or air, is used as the fuel source, the only tailpipe products are electricity, water, and heat. However, as with the other alternative power sources, fuel cells have their disadvantages when used as the sole power source for vehicles. These disadvantages are mostly linked to the fuel cells sensitivity and response to dynamics in the fuel system, air delivery system, and load, to name a few.

Thus, work on the development of fuel cell hybrid electric vehicles (FCHEV) has been done. As with HEVs, FCHEVs take the advantages of both fuel cells and batteries to overcome the disadvantages of both. The result is a clean vehicle with improved efficiency, better dynamic system response, and true zero emissions. Much work and research is still being done to bring these vehicles into production, including

technological demonstration projects supported by the government, industry, and academic research institutes.

1.2 Technology Demonstration

Technology demonstration projects involve the development of a product focused on bringing an emerging technology towards the public eye with hopes of garnering support and excitement about the particular technology. Several of these projects have been completed through the years that demonstrate the viability of electric and hybrid electric vehicles. These projects tend to be initiated by government funded programs such as the National Fuel Cell Bus Program (NFCBP), which focuses on the development of fuel cell transit buses that will be added to bus fleets of cities or institutions and road tested over the course of several years. As part of the NFCBP, the Center for Transportation and the Environment (CTE), in conjunction with the Federal Transit Administration (FTA), has been appointed to overlook the building and progress of these fuel cell hybrid electric bus demonstration/research projects.

One such project, the ECO Saver IV Hybrid Electric Fuel Cell Bus Demonstration, has been undertaken by DesignLine and the Center for Automotive Research (CAR) at the Ohio State University (OSU) through a contract with CTE. DesignLine was tasked with the integration of a Ballard fuel cell system with the building of a commercially viable heavy duty transit bus. The fuel cell hybrid bus architecture includes a 75kW Ballard fuel cell system and a 600 kW Lithium Iron Phosphate (LFP) battery pack. More

details concerning the vehicle powertrain architecture will be provided in the following chapters.

Vehicle modeling and simulation was performed at CAR with the intent to validate the performance of the DesignLine bus architecture. In addition, an improved control algorithm for use with the bus' electronic control unit (ECU) was to be developed and proposed. The technology demonstration is to be completed at OSU through integration with the Campus Area Bus Services (CABS) fleet and will last a total of two years in operation.

1.3 Thesis Overview

The work presented in this thesis is centered on the results of the tasked fuel cell hybrid bus architecture validation, specifically with the focus of determining how the bus will perform given the specific route characteristics of a typical OSU CABS bus route. In addition, a CAN based data acquisition system was developed to be used as an on-board embedded device to log real-time data from the ECU for the duration of the technology demonstration project.

Furthermore, the work takes that which was tasked to OSU a step further through the use of software-in-the-loop (SIL) and hardware-in-the-loop (HIL) simulations to conduct real-time validation and verification of the OSU developed vehicle model simulator and proposed DesignLine bus architecture as well as the HIL validation of the developed CAN based data acquisition system with the HIL simulator.

The organization of this thesis is as follows:

- Chapter 2 – Literature Review
    - Chapter 2 consists of background information on NFCB demonstration projects, the use of standardized drive cycles versus real world drive cycles in terms of vehicle simulation, the methods employed in the modeling and simulation of vehicles, and the types of control strategies typically used to control and optimize the performance of hybrid electric vehicles.
- Chapter 3 – Simulator/Experimental Resources and Method
    - Chapter 3 details the specifics of the architecture and characteristics of the fuel cell hybrid bus, the design of the fuel cell hybrid bus simulator as originally developed by Kyle Simmons, the validation of the proposed control algorithm, and the development of the CAN DAQ system.
- Chapter 4 – Bus Hardware/Software Validation for Campus Use
    - Chapter 4 provides an in-depth validation of the proposed bus architecture and software through evaluation of the performance of the bus on the desired campus driving conditions with a focus on the development of real-world drive cycles and a new driver model, an analysis of the comparative results of the both the drive cycles and the driver models, and a weight sensitivity study, among others.
- Chapter 5 – SIL/HIL Preparation

- Chapter 5 pertains to the methodology used in the proper implementation of SIL and HIL simulations and the process used to create the SIL and HIL simulations.

- Chapter 6 – SIL/HIL Results
  - Chapter 6 goes through the results of the SIL and HIL simulations through a comparative analysis between MIL, SIL, and HIL for the standardized and developed real-world drive cycles. Conclusions based on the comparison results are then given.
- Chapter 7 – Conclusions and Future Work
  - Chapter 7 contains concluding remarks about the work presented in this thesis before discussing the possible future work that could be done with regards to the work presented.

CHAPTER 2: Literature Review

2.1 Introduction

The following sections provide background information found in relevant literature on the NFCBP demonstration projects, the use of drive cycles in vehicle simulation, modeling and simulation methods, and hybrid electric vehicle control and optimization.

2.2 NFCBP Demonstration Projects

The Department of Energy (DOE) and the FTA have been funding the National Renewable Energy Laboratory's (NREL) evaluation of fuel cell hybrid electric buses since as early as 2003. However, a number of these bus demonstration projects throughout the nation that are either planned, currently in evaluation, or done with their initial evaluation have been started as part of the NFCBP. NFCBP demonstration projects involve a partnership between the FTA and one of three non-profit industry consortia, specifically, the Northeast Advanced Vehicle Consortium (NAVC), CALSTART, and CTE.

These three consortia are responsible for the development and management of projects through collaboration with industry or academic institutions.

The primary focus of the NFCBP is the eventual commercialization of full-size, heavy-duty fuel cell transit buses through the development of commercially viable fuel cell bus technology, improved bus efficiency, and reduced petroleum consumption and bus emissions, all while establishing a U.S. industry for fuel cell bus technology and increasing public awareness and acceptance of fuel cell vehicles. The typical life cycle of a bus demonstration project can be summed up into four categories, concept development, prototype construction/build, demonstration and evaluation, and the publishing of findings. [4], [5]

2.3 Driving Cycles in Vehicle Simulation

2.3.1 Standardized Drive Cycles

The US Environmental Protection Agency (EPA) has developed a set of federal test procedures for measuring tailpipe emissions and fuel economy of passenger and heavy duty vehicles. The tests involve running a vehicle on a chassis dynamometer with a standardized driving schedule or drive cycle. These tests allow vehicle manufacturers to obtain standardized values for fuel economy on vehicles they produce.

For passenger cars, there are two main drive cycles used, one for simulated city or urban driving, known as the Urban Dynamometer Driving Schedule (UDDS), and one for simulated highway driving, known as the Highway Fuel Economy Driving Schedule

(HWFET). In addition, the US06, SC03, and Cold Cycle Supplemental Federal Test Procedures were later introduced as an attempt to simulate real-world driving and fuel economy more accurately. These three supplemental tests combine city and highway drive cycle characteristics and take factors like rapid speed fluctuations, aggressive driving behavior, air conditioning use, and cold start conditions into consideration.

In terms of heavy-duty vehicle emissions and fuel economy testing, the EPA has only developed one such federal driving schedule, the Heavy Duty Urban Dynamometer Driving Schedule (HDUDDS). However, work has been done on developing specialized drive cycles for heavy-duty vehicles ranging from refuse trucks to transit buses. A well-known simulated transit bus drive cycle is that of the Manhattan Bus Cycle. This drive cycle is used to test emissions and fuel economy of urban buses on a chassis dynamometer.

While the EPA and vehicle manufactures use these cycles to test vehicles, these standardized drive cycles are also used in the modeling and simulation of vehicles. Vehicle modeling and simulation allow engineers to test vehicle architectures and control algorithms on various simulated driving conditions without having to first build a prototype. Thus, it could possibly provide an initial estimate of the vehicle's emissions, fuel consumption, performance, and efficiency at a much faster computational time frame than dynamometer testing would allow. Since these results are obtained using standardized drive cycles, they serve as a good way to determine if the vehicle would pass the ever stringent EPA standards. [6]

2.3.2 Real-World Drive Cycles

While standardized drive cycles are useful for establishing compliance with EPA standards and obtaining performance estimates on various driving conditions, some vehicles are developed with certain specific applications in mind that require more than a simulated driving schedule. Thus, the need for real-world drive cycles is in bridging the gap between performance estimates and real time evaluated data of the vehicle in operation. Real-world drive cycles are obtained by gathering data of a typical route that the vehicle will be expected to travel. At a minimum, this data would have to include velocity and time travelled. However, in order to better characterize a real-world drive cycle, information on the road grade and distance travelled should also be gathered. The benefits of using real-world drive cycles include more accurate simulation results for emissions, efficiency, and performance because they take into consideration actual driving conditions for actual routes that the vehicle would drive. These benefits have been studied and applied in both industry and academic institutions. Researchers have studied the effects of real-world driving as compared to government standardized drive cycles on a prototype hybrid vehicle with the use of a vehicle model. [7] These benefits are not just evident when applied to vehicle modeling and simulations.

While places like the U.S. and Europe have their own standardized emissions tests, other countries have yet to develop their own and instead opt to using standardized drive cycles developed for other countries or regions that are not indicative of their local driving conditions. Such was the case in India, where vehicle

emissions testing was conducted with the use of European driving cycles. However, these drive cycles do not take into consideration the driving and traffic conditions of Indian cities. Thus, there was a need for real-world based Indian drive cycles for intra-city buses. Using a Global Positioning System (GPS), the operating characteristics of intra-city buses in Chennai, India were examined in order to develop such a driving cycle. [8] As the literature shows, the development of real-world drive cycles is critical in the proper determination of vehicle emissions and fuel consumption. Further examples of the use of real-world drive cycles can be found in [9], [10], [11], and [12].

Though little was found in the literature involving the development of real-world road grade profiles in conjunction with the use of real-world drive cycles, this thesis will provide an overview of such a task. In addition, the methods used in creating real-world road grade profiles will be highlighted. In particular, the use of Vincenty's Inverse Formula will be explained in relative detail. A more in-depth explanation of the Vincenty Formulae and its applications can found in [13].

2.4 Simulation Methods of Model-Based Control/ Systems Development

2.4.1 Introduction

By definition, model-based control and systems development requires the use of simulations. However, through the years, there has been an increase in the methods used for the modeling and simulation of differing applications. With such a vast offering of methods, it is useful to categorize them in order to better understand the underlying applicability of each method. Thus, with regard to the speed of the computation

required, simulation methods can be subdivided into three main categories, simulations without hard time limitations, real-time simulations, and simulations that are faster than real-time. [14] In terms of this work, it will be assumed that all simulations contain some sort of time limitation and thus will not be considering simulations without hard time limitations.

The remaining categories can be further subdivided into more useful subcategories centered on the several phases of a typical V-model approach to model-based control and systems development. Simulations that are faster than real-time can be equated to the initial development of model-based control systems and control/optimization strategies, referred to as model-in-the-loop (MIL) simulations or software-in-the-loop (SIL) simulations, respectively. On the other hand, real-time simulations are categorized as hardware-in-the-loop (HIL) simulations. Though at times, SIL simulations can be run in real-time, the difference is that SIL comprises both a simulated plant model and a simulated controller in real-time whereas HIL pertains to real-time simulations in which a simulated process is operated with real control hardware, typically to validate the actual controller. [14] Thus, for the most part SIL simulations are conducted at computational times that are faster than real-time.

2.4.2 V-Model Approach

Before getting into the details of the simulation methods used within a V-model approach to model-based control and systems development, it is useful to present the basic steps of such an approach. The traditional V-model approach is a graphical

representation, as shown by Figure 2, in which the left side contains several steps that can be grouped under the general term of problem definition and decomposition which includes the conceptualization of the problem, requirement definition, and design. This side of the traditional V-model ends in the eventual implementation of the system and leads to the right side of the model. The right side also contains several steps generally grouped under the integration, verification, and validation of the system.

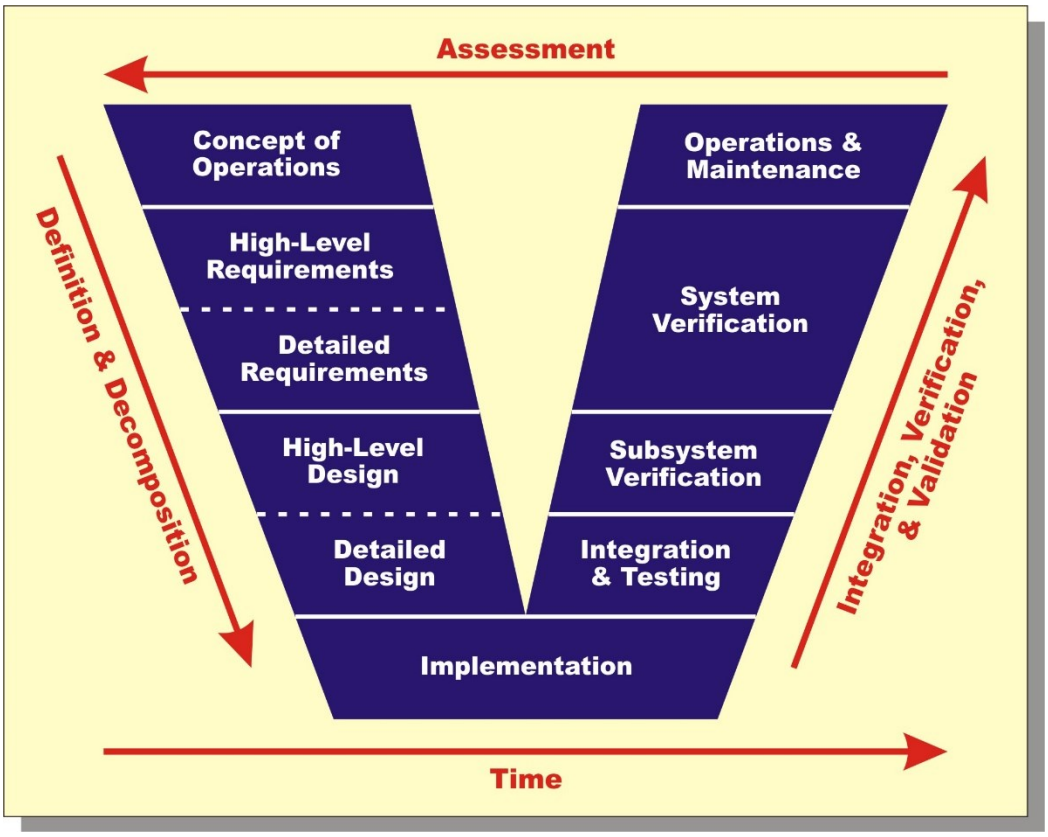

Figure 2: Typical V-Model Diagram

However, in terms of model-based control and systems development, these two sides will at times have a considerate amount of cross-over even before the implementation phase has been reached. That is, during the development of the

models and simulators of a system, several iterations of the verification and validation process are undertaken so as to derive an appropriate and sufficiently characterized model/simulator of the system. Likewise, during the post-implementation verification and validation process, it might become necessary to revisit the requirements definition or design of the system models, typically referred to as the assessment phase.

2.4.3 Verification and Validation

With such a recurrent use of the verification and validation process throughout the whole V-model, it can easily be concluded that it is one of the most crucial steps in the development of model-based control and systems. However, while some understand that these two terms have their own separate meaning, others tend to use them interchangeably within literature. Their difference can be traced down to two other terms, requirements and specifications, which also seem to be used interchangeably but have their own distinction. Requirements are used in identifying which tasks a system is meant to perform or satisfy while specifications aid in defining how the system is to function in order to perform a desired task. Therefore, validation is performed to ensure that the defined requirements are met by identifying whether or not the system carries out the desired tasks. On the other hand, verification is achieved by ensuring that the specifications have been met. [15] Still, the verification and validation process is of no use without the proper simulation methods to apply them to.

2.4.4 Model-in-the-Loop

In [15], a look at the current methods used by various entities within industry and academic research facilities for automotive model-based control is undertaken to detail the differences in methods employed, at times internally within an entity, which has led to a need for a unified approach. A detailed explanation of the several steps involved in model-based control is provided with an industry inspired perspective in order to develop a comprehensive approach to model-based control. In addition, proper MIL modeling techniques with SIL and HIL simulation and testing in mind were provided.

MIL simulations allow engineers to use a virtual environment to study the performance of the system with a certain control algorithm design. As mentioned above, these types of simulations are conducted in faster than real-time computational times. This allows for a fast, inexpensive, and easily implementable model/simulation of the system in question that can be used to run various test cases developed with an eye on the verification and validation process. These test cases aid in the analysis of the system and controller model design under several conditions that match up closely to those expected of the actual system.

While MIL simulations are very useful, their effectiveness is usually limited to the early phases of system development. This is mostly due to the fast computational times which precisely make MIL simulations valuable in early stage development. However, there is an inherent lack of information about how the system would function in real-

time, which is how an actual system would function. Thus, for higher levels of the design process, real-time simulation would have to be considered.

2.4.5 Software-in-the-Loop

As mentioned in the introduction of this chapter, SIL simulations involve the use of a simulated system with the use of a simulated controller. SIL simulations are the next step in model-based design once MIL simulations have been used. They form a bridge for the gap found between MIL and HIL simulations. Typically, building a SIL model requires the use of automatic code generation tools that compile the control model into software code. The generated control software code is then run in the same virtual environment as the plant model, which remains in the same format as that used in MIL. Thus, the code generated from a control model can be tested to verify whether or not it performs similar to the MIL simulation model. This method of simulation is mostly useful in the development of control software. The benefits of SIL simulations include an increase in simulation speed due to the use of compiled code instead of a model, the ability to run multiple versions of a control strategy in a modular manner, and, consequentially, reduced cost.

The use of SIL to test multiple control algorithms to determine which best accomplishes the desired tasks is seen in [16] where, as part of the EcoCAR 2 competition, students at The Ohio State University worked on the design and development of the architecture and control of a hybrid electric vehicle. With the use of Argonne National Lab's Autonomie software, the architecture was decided to be that of

a parallel-series plug-in hybrid electric vehicle. A detailed model of the vehicle was then created. Afterwards, software-in-the-loop simulations were used to test various control algorithms with the vehicle plant model to determine which supervisory controller would best improve the vehicle's fuel economy. Separate control algorithms based on the Equivalent Consumption Minimization Strategy (ECMS) were chosen for the charge sustaining and charge depleting modes of the vehicle. Results showed an increase in the vehicles all-electric range and fuel economy as compared to a baseline control strategy, an indication of the success of SIL simulations in developing control strategies.

2.4.6 Hardware-in-the-Loop

2.4.6.1 Introduction

Once a specific control strategy has been chosen with the use of SIL, the next step would be to evaluate how it functions along with the system in real-time. Thus, HIL simulations require the use of the compiled software code built prior to doing SIL simulations and incorporating it into actual control hardware. In addition, the plant model would then be similarly compiled into code and run in a HIL device. At this stage in the model-based control and system development process, the plant model can be thought of as being part of a dedicated real-time simulator. [15] Making changes to the model at this point would require rebuilding and recompiling the code. Apart from allowing the use of actual controller hardware, actual sensors and actuator signals can be used as part of a HIL simulator as well. Therefore, HIL simulations are particularly

useful for two general purposes, software validation and fault diagnostics. [17], [18], [19], [20]

2.4.6.2 Software Validation

While MIL and SIL simulations help in the development of the control strategy and plant model, HIL simulations are often used to further validate the control strategy or software code chosen during the SIL testing of various control strategies. The important distinction is that HIL simulations are conducted in real-time and thus allow a more accurate look at the performance of the control algorithm as it would behave in a real-world scenario. By running several simulations with a host of differing conditions and scenarios, the software control algorithm is checked for robustness, accuracy, fidelity, and repeatability of results. Results are typically compared to those of MIL or SIL simulations and checked against the system requirements through a validation process. Such was the case in [16], where students at The Ohio State University employed the use of HIL to test and validate the results they received from their SIL control strategy testing.

Validation of a vehicle control strategy is especially important because it ensures that the vehicle and vehicle systems perform as desired. Thus, as shown in [21],the development of a vehicle's ECU should involve HIL validation tests. Specifically, researchers detailed the use of a HIL system to validate the ECU of a hybrid electric vehicle. The ECU model developed contained models of the primary physical systems such as engine, transmission, and battery, sensor and input signal models, actuator

models, and models of external systems like driver and environmental interactions. Benefits of the use of a HIL system include the validation of the software and vehicle component testing as prototypes became available. Results of the validation were included as compared to actual vehicle data and showed a high level of agreement, proving the reliability of HIL in the development and validation of vehicle ECUs.

2.4.6.3 Fault Diagnostics

On the other hand, HIL simulations can also be used to perform fault diagnostics of the model-based control system along with actual sensors and actuators. Through extensive hazardous analysis for multiple conditions and scenarios, including potentially hazardous cases, an accurately designed model-based control system can be tested for faults and failures without damaging actual systems or system components. Additionally, testing of system components can be conducted before actual hardware has been built or prototyped, saving crucial development time and cost. If faults or failures are detected in this way, the control algorithm would then have to be redesigned to ensure that safety measures are in place to avoid or reduce their occurrence. This enables greater safety and reliability of the system for its eventual users and forms a critical part of the post-implementation verification and validation process. This is especially important when it concerns vehicle systems where a fault or failure could results in the injury or even death of those who use or come in contact with the vehicle or vehicle system. [22], [23], [24]

Such an analysis is performed in [14], where the history, methods, and practices of HIL simulations is presented and followed by a study focused around automotive applications. Specifically, simulation and testing were conducted for a vehicle model of a Mercedes-Benz truck with a turbocharged diesel engine. Three example HIL simulations were presented so as to show the applicability and performance of the simulator. These three examples included the effect of switching off a single injection pump valve, the full power acceleration of the truck including two gear shifts, and the experimental validation of the truck's cruise control. Analysis of all three examples shows suitable performance results, as expected. Overall, it is concluded that HIL simulation allows the reduction of development time and cost through the rapid prototyping of engine control systems while providing a platform from which to detect the presence of faults.

Similarly, in [25], an Advanced Driver Assistance System (ADAS) is developed according to the "V" diagram design process that is meant to warn the driver of a possible or impending collision. A quick look is given to the requirements/specifications, verification/validation, model-in-the-loop, software-in-the-loop, hardware-in-the-loop, and test drive design phases as they pertain to the development of the ADAS. In addition, it is argued that the test drive phase is not adequate for control system design of an ADAS due to the lack of traffic data. Thus, a new method is proposed that couples the advantages of HIL simulations with the representativeness of test drives, titled vehicle hardware-in-the-loop simulations (VeHIL). VeHIL is a multi-agent simulator in

which some of the simulated vehicles are replaced by real vehicles in an artificial HIL indoor laboratory. Finally, a case study is presented such that a driver information and warning system for safe speed and safe distance is validated and put through a fault diagnostic process with the use of VeHIL. Emphasis is given to the fact that this new approach is not meant to replace HIL simulations and test drives but rather create a link between them that can save on time and cost of ADAS's during the development process.

While software validation and fault diagnostics pertain to separate types of analysis, as shown in the literature, both are conducted through HIL simulations and are often used in conjunction. In fact, it could be said that each is merely an extension of the other and are simply required steps throughout the whole of the post-implementation verification and validation process of the V-model approach to model-based control and system development. Specifically, a complete implementation of fault diagnostic techniques would most likely involve the validation of the software before any sort of hazard analysis was conducted so as to ensure that the possible faults or failures detected are not simply due to the controller not behaving like it should. Likewise, one could hardly say that the controller software is truly validated if it has not been tested as a possible source of faults or failures.

2.5 Hybrid Electric Vehicle Control and Optimization

2.5.1 Introduction

The emergence of hybrid vehicles has brought about the need for sophisticated power management control strategies to determine the best way to provide power to the wheels from the multiple on-board power sources depending on which metrics are of concern to a specific vehicle design. For example, the control strategy could focus on the optimization of factors like fuel economy, emissions, or vehicle performance, to name a few. The way in which a hybrid vehicle's power management is handled can range from a simple constant power split to a host of more complicated approaches that attempt to reach the best possible solution through the use of complex algorithms. However, most of the control strategies used to date can be grouped under two types of strategies, rule-based or optimization based control. Each has its benefits and drawbacks, such as whether or not they are truly implementable, the amount of dependence on computational resources, and proximity to the optimal solution.

2.5.2 Rule-Based Control

The most basic types of control strategies fall under the heading of rule-based control. Rule-based control strategies are heuristic in nature, *i.e.*, they proceed to a solution through either a trial and error approach or by using a set of defined rules. These rules can be used to determine things like the mode of operation, battery state of charge (SOC), and power split method. While rule-based control is the simpler of the grouped methods, they require knowledge of appropriate threshold values for the set of

rules in order for the vehicle to meet its desired performance or fuel economy targets. Regardless of this need for extensive calibration, these strategies tend to be more easily implementable and take up less computational power and time when compared to their optimization based counterparts. Some of the more popular rule-based control strategies used within literature include thermostatic control, proportional or PID control, penalty functions, and fuzzy logic.

In [26], a team of students at The Ohio State University used lessons learned from the Buckeye Bullet 1 to design and build the Buckeye Bullet 2, a hydrogen fuel cell land speed vehicle. Land speed vehicles are used to break and set incredibly high speed records by pushing the limits of vehicle system components. The paper details the use of a supervisory control system to properly cool the fuel cell system and to determine the appropriate power management. The power management controller was based on a rule-based control strategy in which the goal was to maximize the power output to achieve the highest speed possible. Once completed, the control system of the vehicle was tested when the Buckeye Bullet 2 was run on the Bonneville Salt Flats of Utah.

As part of the analysis of a series hybrid vehicle, researchers in [27] developed a rule-based control strategy meant to split the power demand between an engine and battery pack while maintaining high operating efficiencies. The focus of the energy management strategy was in the improvement of the vehicle's fuel economy as compared to that which was obtained by a thermostatic controller. The rules used were dependent on the power demand, battery SOC, and driver acceleration command.

Simulation results of the rule-based control strategy showed a clear improvement in fuel economy over the thermostatic controller.

The literature clearly shows the effectiveness of rule-based control strategies. More examples of the use of rule-based control in the energy management of hybrid vehicles can be found in [28], [29], [30], and [31]. Nevertheless, results obtained through the use of rule-based control strategies are still suboptimal. To get optimal or near optimal solutions the use of optimization based control is needed.

2.5.3 Optimization-Based Control

2.5.3.1 Introduction

While vehicle control systems can be effectively modeled with rule-based control strategies, optimization-based control strategies determine what the optimal solution of a control problem should be and thus are more accurate. To this end, a specific payoff criterion must be identified, giving way to an associated payoff function that will aid in determining the best control for the system. Certain cost criterion could be defined so as to develop a cost function that serves as a way to measure which control actions cost the least. Which type of function is used depends on the specific desired objective for the optimization and can be generalized as an objective function. Furthermore, there are various methods of applying the optimization scheme, but in general, solutions can either be globally optimal or locally optimal. In terms of the literature, the more popular optimization-based control strategies are those of Dynamic Programming, Pontryagin's Minimum Principle (PMP), and Equivalent Consumption Minimization

Strategy (ECMS). [32] For example, work done in [33] was focused on an energetic approach to hybrid vehicle powertrain modeling and the validation of a refuse collection vehicle in which the use and comparison of multiple optimal control strategies was detailed with an emphasis on dynamic programming, PMP, and ECMS.

2.5.3.2 Dynamic Programming

Before explaining what Dynamic Programming is and how it works, it would be helpful to first understand the basic dynamic control problem. That is, given a specific initial state of a dynamic system, it is desired to determine the best way to control the system so as to end up with a set of optimal states. The aim of a basic dynamic control problem, therefore, is to find a control, called the optimal control, which either maximizes the payoff or minimizes the cost to reach each successive optimal state. Thus, it can be said that solving a dynamic control problem consists of making multiple decisions about which steps to take in order to achieve the desired objective through the best possible pathway.

Dynamic Programming is a numerical method used for solving dynamic decision problems and is based on Bellman's principle of optimality, which states that an optimal policy has the property that whatever the initial state and initial decision of a dynamic decision problem are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. [34] In other words, the entire optimal solution is made up of intermediate steps that form an optimal path starting at the initial state. Dynamic Programming describes this optimal path by determining what

the controls should be at any given state. This can be mathematically defined by considering a discrete-time system

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

in which the subscript $k = 0, 1, \ldots, N-1$, u_k denotes a control, and x_k is the system state at step k. The control policy

$$\pi = \{u_0, u_1, \ldots, u_{N-1},\} \tag{2}$$

is a sequence of controls such that $u_k \in \mathcal{U}_k(x_k)$ for all states x_k used during the optimization of the desired objective. The objective function should then be one that is dependent on the control and state variables. For hybrid electric vehicle system applications, cost functions are typically the objective function of choice, such as when seeking to minimize the consumption of fuel or greenhouse gas emissions. Thus, the total cost of a control policy starting at the initial state x_0 can be written as

$$J_\pi(x_0) = L_N(x_N) + \sum_{k=0}^{N-1} L_k(x_k, u_k) \tag{3}$$

where L_k is the instantaneous cost function at step k and L_N is the cost function at the final state. Therefore, the optimal cost function would be one that minimizes the total cost

$$J^*(x_0) = \min_\pi J_\pi(x_0). \tag{4}$$

Furthermore, the total cost of the optimal control policy

$$\pi^* = \{u_0^*, u_1^*, \ldots, u_{N-1}^*,\} \tag{5}$$

should be such that it is equivalent to the optimal cost

$$J_{\pi^*}(x_0) = J^*(x_0). \tag{6}$$

Thus, a Dynamic Programming control algorithm for a hybrid electric vehicle simulator would involve the minimization of the cost-to-go from time i to time N

$$V_i = L_N(x_N) + \sum_{k=i}^{N-1} L_k(x_k, u_k) \tag{7}$$

of a given drive cycle and would move in a backwards fashion starting at the end of the driving cycle until the optimal cost $J^*(x_0)$ at the initial step is reached. Specifically, it would provide an optimal control strategy for the power split at each time step corresponding to the chosen design objectives. [16], [33]

One can imagine that the potential solution paths of a system as complex as a HEV could be infinite. To avoid having to consider such a vast number of possible paths, the control algorithm should be developed such that it would make the power split decisions based on the current vehicle speed, the total driver power demand, and the state of the powertrain components, which includes determining the maximum and minimum power that each powertrain component can deliver. Once a discrete number of possible solutions have been selected, the optimal cost-to-go is calculated by moving backwards from the end of the drive cycle. The path with the lowest total cost is the optimal solution.

An example of this method as applied to an HEV's battery state of energy (SOE) is represented graphically in Figure 3. It can be seen that application of the algorithm requires proceeding backwards from the end point or node L to determine the "arc costs" at each node before calculating the cost-to-go. The calculation of each "arc cost" can be a computationally intensive task and completion of the backward solution for all possible paths is required before the first optimal control action can be determined. This is the reason Dynamic Programming is known as a global optimization control strategy.

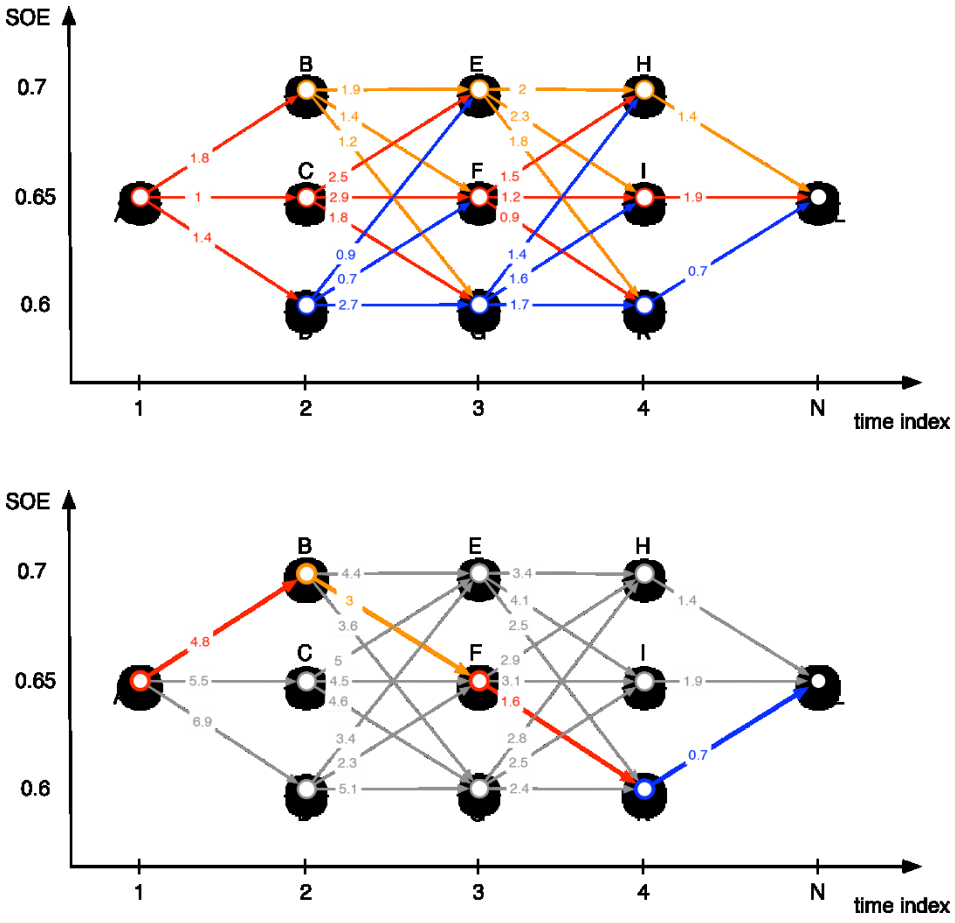

Figure 3: Example of Dynamic Programming as Applied to Battery SOE

In addition, since Dynamic Programming determines the optimal solution by working backwards from the end of the drive cycle, it requires knowledge of the entire drive cycle in advance. Therefore, while it can be applied to solve for optimal solutions of problems of varying complexity, it is not implementable in real vehicle control systems. [33] However, since Dynamic Programming provides an optimal solution, it is often used in developing rules for rule-based control strategies or simply as a benchmark for other suboptimal control algorithms. As part of an industry-academic collaboration, testing, modeling, and control design of a fuel cell hybrid vehicle is presented in [35]. System components were modeled from the test results obtained from a prototype vehicle. A complete vehicle simulator was then developed according to experimental data. The power control strategy was modeled after a stochastic dynamic programming approach in which the driver power demand at the next step depends on that of the current power demand and vehicle speed in order to optimize vehicle fuel economy while ensuring drivability. The forward-looking simulation results show a clear improvement in fuel economy from that of the gasoline counterpart of the prototype fuel cell hybrid vehicle.

Likewise, [36] showcases the optimization of a fuel cell hybrid vehicle as conducted through the development of a pseudo stochastic dynamic programming control strategy. In addition, a study of the effects on vehicle performance of component sizing was conducted. However, contrary to other literature, a strong case is made for the joint optimization of component sizing and power management. Thus, the

use of high fidelity subsystem-scaling models is used alongside the near optimal power management algorithm to determine the optimal power management and component sizing simultaneously. The overall optimized result from such an approach, as applied to fuel cell hybrid vehicles, lies in downsizing the fuel cell compressor, increasing the fuel cell stack size which decreases the battery pack size without impeding regenerative braking, and using the corresponding power management strategy.

These are just a couple of examples, as the application of dynamic control algorithms based on Dynamic Programming to hybrid electric vehicle studies or as part of a control strategy comparison can be found in a vast amount of the literature. More examples can be found in references [37], [38], [39], [40], [41], [42], and [43].

2.5.3.3 Pontryagin's Minimum Principle

While an optimal solution can be obtained through the use of Dynamic Programming, an algorithm based on it requires a large amount of computing power and time. Fortunately, DP is not the only optimization-based control strategy. Pontryagin's Minimum Principle is a very powerful tool in optimal control theory used to find the best possible control for taking a dynamical system from one state to another. Significantly, this is accomplished without using large amounts of computing time and resources, making PMP a very attractive option as an optimization-based control algorithm. The basics of optimization concerning a dynamic control problem apply here as well. Specifically, the aim is to find an optimal control which, given a specific initial state of a dynamic system, minimizes the cost to reach each successive optimal state so

as to end up with a set of optimal states. In fact, this statement is true of all optimization-based control strategies. However, the way in which the minimization is carried out is what separates them from one another. The use of PMP along a trajectory like that of a drive cycle yields a set of necessary conditions that ensure the global optimality of a constrained control problem. [44]

A full appreciation and understanding of the mechanics and necessary conditions of PMP requires knowledge of the basic concepts behind the Hamiltonian function, which will be presented below considering a dynamical system with the state equation

$$\dot{x} = f(x, u, t), \qquad x(t_0) = x_0 \tag{8}$$

with time t, states x, control input $u \in \mathcal{U}(t)$, and the initial state x_0 at the initial time t_0. Utilizing an objective function similar to that of Equation (3), we end up with

$$J(x_0, u, t_0) = h\big(x(t_f), t_f\big) + \int_{t_0}^{t_f} L(x(t), u(t), t)\, dt \tag{9}$$

for the fixed time interval $t \in [t_0, t_f]$, where h is the terminal cost and L is the instantaneous cost. Again, the optimal control $u^*(t)$ for $t \in [t_0, t_f]$ is that which minimizes Equation (9), resulting in $J^*(x_0, t)$ with the optimal state trajectory $x^*(t)$. A visualization of the optimal cost function and its corresponding state trajectory is shown in Figure 4. [44], [45]

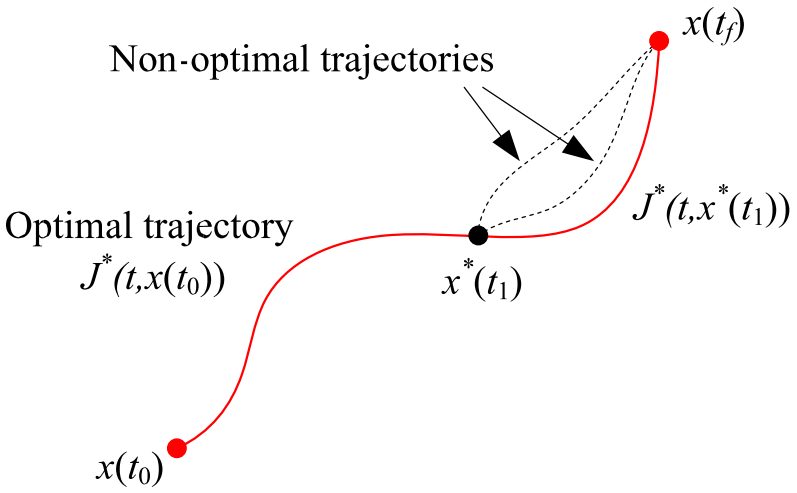

Figure 4: Optimal State Trajectory Visualization

Recalling Bellman's principle of optimality [34], one can see from Figure 4 that optimality also applies to a sub-problem of Equation (9) starting at time t_1 on the optimal trajectory. That is, for $\tau \in [t_1, t_f]$, the truncated optimal control $u^*(\tau)$ minimizes the cost function

$$J(x^*(t_1), u, t) = h\big(x(t_f), t_f\big) + \int_{t_1}^{t_f} L(x(t), u(t), t)\, dt \tag{10}$$

and results in the optimal cost function $J^*(x^*(t_1), t)$, which denotes the optimal trajectory from state $x^*(t_1)$ to the final state $x(t_f)$. Therefore, from the principle of optimality, it is possible to separate Equation (9) into two integration intervals with the time intervals $t \in [t_0, t_1] \cup [t_i, t_f]$. Ignoring the terminal cost for now, the inequality

$$\int_{t_0}^{t_1} L(x(t), u(t), t)\, dt + J^*(x^*(t_1), t) \geq J^*(x^*(t_0), t) \tag{11}$$

can be formulated in which the first interval starts at the state x_0 with an arbitrary control u, and the second interval starts at the optimal state $x^*(t_1)$ with the optimal control u^*. Thus, this inequality forms a necessary condition for optimality where the best case is that in which the first interval plus the second interval equals the optimal objective function $J^*(x^*(t_0), t)$. If this optimal objective function is moved to the left side of the inequality and all terms are divided by $t_1 - t_0$, the correlation to the Hamiltonian function

$$L(x(t), u(t), t) + \frac{\partial J^*(x(t), t)}{\partial t} + \sum_{i=1}^{n} \frac{\partial J^*(x(t), t)}{\partial x_i} \dot{x}_i \geq 0 \tag{12}$$

can be derived by allowing the interval $t_1 - t_0$ to become infinitely small such that $t_1 \rightarrow t_0$ and through use of an approximation, the definition of partial derivatives, and the chain rule. At this point, the Hamiltonian function

$$H(\lambda, x, u, t) = \lambda^T f(x, u, t) + L(x, u, t) \tag{13}$$

can be substituted into Equation (12), considering that the co-state λ is defined as a vector of Lagrange multipliers

$$\lambda_i = \left[\frac{\partial J^*}{\partial x_i}\right]^T \tag{14}$$

associated with the state equation. Therefore, Equation (12) can be rewritten as

$$\frac{\partial J^*(x, t)}{\partial t} + H(\lambda, x, u, t) \geq 0. \tag{15}$$

The derivations so far have been done to gain insight into the workings of PMP. As with any optimal control problem, the aim is to determine the optimal control input u^*. In particular, it is known that the optimal control input can be obtained by minimizing the Hamiltonian function. This fact along with the principle of optimality allows for Equation (15) to be written as an equality in the form of the Hamilton-Jacobi-Bellman equation

$$\frac{\partial J^*(x,t)}{\partial t} + \min_u H(\lambda, x, u, t) = 0 \tag{16}$$

whose solution yields the optimal control input. Furthermore, the Hamiltonian of the optimal state trajectory $x^*$and optimal co-state λ^* has a global minimum if $u = u^*$, for all $u \in \mathcal{U}(t)$ and $t \in [t_0, t_f]$. Additionally, the full set of necessary conditions for optimality on u and λ can be derived from the fact that the optimal control input minimizes the Hamiltonian. Details on how to derive a solution for the co-state and the necessary conditions, can be found in [45]. Finally, PMP states that the optimal control trajectory u^*, optimal state trajectory x^*, and optimal co-state variable λ^* minimize the Hamiltonian, such that

$$H(\lambda^*, x^*, u^*, t) \leq H(\lambda^*, x^*, u, t) \tag{17}$$

while the necessary conditions and constraints

$$\dot{x}^*(t) = \frac{\partial H}{\partial \lambda}(\lambda^*(t), x^*(t), u^*(y), t) \tag{18}$$

$$\dot{\lambda}^*(t) = \frac{\partial H}{\partial x}(\lambda^*(t), x^*(t), u^*(y), t) \tag{19}$$

$$0 = \frac{\partial H}{\partial u}(\lambda^*(t), x^*(t), u^*(y), t) \tag{20}$$

$$h\left(x^*\left(t_f\right), t_f\right) = 0 \tag{21}$$

$$x^*(t_0) = x_0 \tag{22}$$

$$x^*(t) \in \mathcal{X}(t) \tag{23}$$

$$u^*(t) \in \mathcal{U}(t) \tag{24}$$

must hold. These necessary conditions state that the optimal solution must satisfy the minimum principle. PMP ensures that if a global optimal solution exists for the control problem, it is an extremal solution. Unfortunately, as with DP, PMP is not implementable in real vehicle systems. Nevertheless, multiple studies have been conducted concerning the use of PMP as the control algorithm for hybrid electric vehicles. [44], [45], [46]

In [44], a fuel cell hybrid passenger bus simulator meant to aid in the design of an actual bus is introduced along with the characteristic equations used in the control of the model. A supervisory controller was constructed with the use of optimal control theory based on PMP. Results are presented that detail the effects of battery sizing, chemistry, and the severity of the drive cycle used, among other things. In addition, due to the fact that a PMP based controller is not implementable, an auto-regressive controller is proposed as a future work.

The optimal control problem of energy management for plug in hybrids is presented in [47] along with a traditional charge depleting – charge sustaining (CD-CS) strategy and blended strategies that are merely suboptimal at best. As an attempt to develop an implementable near optimal solution, a closer look is given to optimal control strategies like DP and PMP. Based on the optimal results, a practical implementable controller is proposed and results are compared to optimal PMP and traditional CD-CS results. It is found that the practical implementable controller provides a near optimal solution with an improved MPG rating of approximately 50%.

Furthermore, a study in [48] highlights the use of PMP as a viable real-time strategy for the power management of a HEV. The global optimality of PMP is analyzed and compared to a proposed near optimal control strategy that is based on PMP. Results of the fuel economy minimization show that this PMP based controller comes within 1% of the global optimal solution found through a DP controller. By development of a control algorithm based on PMP, it could be possible to implement a near optimal power management controller into an actual hybrid electric vehicle.

Thus, it can be seen that as an energy management control algorithm, PMP allows the desired objectives to be optimized by first deriving the Hamiltonian and then applying the necessary conditions to arrive at a globally optimal solution. These examples of vehicle control strategies based on PMP are only a small fraction of the literature dedicated to the study of PMP with hybrid electric vehicles. Yet, as seen from

[48], the literature also highlights the use of PMP as a way to formulate a near optimal control strategy known as Equivalent Consumption Minimization Strategy.

2.5.3.4 Equivalent Consumption Minimization Strategy

In general, ECMS belongs to a third family of control strategies that modify the global optimization control problem into a sequence of local instantaneous optimization problems. Originally developed at The Ohio State University as an implementable formulation of PMP, it is the most well-known of these local optimization control strategies. ECMS depends on the existence of the charge sustaining mode of operation found in hybrid vehicles. That is, the final battery SOC barely differs from the initial SOC such that the difference can be thought of as negligible with respect to the total energy used. Thus, when in charge sustaining mode, a hybrid vehicle's battery is only used as an energy buffer and, since all of the energy used for propulsion ultimately comes from the fuel consumed, can be thought of as a reversible fuel tank where the energy depleted is replenished with fuel from the engine. This replenished energy would either be in the form of a direct charging of the battery with the engine or from the indirect regeneration of energy from the wheels, which is also traced back to the engine. Then, the energy used from the battery, or reversible fuel tank, can be conceptualized as a virtual fuel consumption. [16], [33]

As with global optimization problems, the aim of local optimization is to minimize a specific objective or cost. In the case of ECMS, however, the objective is not in the form of a cost function but rather an instantaneous cost that is minimized at each

instant, called the equivalent fuel consumption. This instantaneous cost is obtained by adding a term representative of the virtual fuel consumption associated with the use of the battery $\dot{m}_{batt}(t)$ to the actual engine fuel consumption $\dot{m}_f(t)$, such that

$$\dot{m}_{eqv}(t) = \dot{m}_f(t) + \dot{m}_{batt}(t). \tag{25}$$

If the local minimization of this cost is appropriately defined, the ECMS controller would provide a near optimal solution, all while ensuring that the battery state of charge remained at the desired level. In addition, since ECMS is based on an instantaneous minimization, it is easily implementable in real-time. However, this is greatly dependent on the proper definition of the equivalent fuel consumption, which requires optimization of the tuning parameters along with advanced knowledge of the drive cycle. Nevertheless, by basing the ECMS algorithm on a globally optimal strategy like PMP, the equivalent fuel consumption can be derived appropriately and takes the form

$$\dot{m}_{eqv}(t) = \dot{m}_f(t) + \frac{s}{Q_{lhv}} P_{batt}(t) \cdot p(x) \tag{26}$$

where s is the equivalence factor used in converting electric power into an equivalent fuel consumption, Q_{lhv} is the lower heating value of the fuel, P_{batt} is the instantaneous power consumed by the battery, and $p(x)$ is an auxiliary function that accounts for the battery SOC deviation from the SOC reference value. Furthermore, taking the instantaneous fuel consumption as the instantaneous cost, the Hamiltonian function would become

$$H(\lambda, x, u, t) = \lambda^T f(x, u, t) + \dot{m}_f(x, u, t). \tag{27}$$

Additionally, if we assume that the state of the system is related to the battery SOE, whose derivate is a function of the battery power P_{batt}, we can define the system equation of the form of Equation (8) and the equivalence factor s, such that

$$\lambda^T f(x, u, t) = \frac{s}{Q_{lhv}} P_{batt}(t) \cdot p(x). \tag{28}$$

Subsequently, the Hamiltonian takes the form

$$H(\lambda, x, u, t) = \dot{m}_f(t) + \frac{s}{Q_{lhv}} P_{batt}(t) \cdot p(x) \tag{29}$$

which, when subject to the optimal control solution determined through the global optimization of PMP, not only minimizes the total fuel consumption but also that of the instantaneous equivalent fuel consumption. Therefore, as previously mentioned, an ECMS control strategy based on PMP can be implemented in real-time but it is only truly implementable under proper off-line calibration of the parameters. A more in depth derivation of ECMS can be found in [49], where a study was done to demonstrate the use of ECMS as a viable realization of PMP for HEV control. It was found that, due to the equivalence of PMP and ECMS, the global optimal solution can be found by directly implementing the ECMS.

This optimization approach was presented in [50] and compared to previous work for a fuel cell battery hybrid vehicle. The approach uses fuel mass flow rate instead of the conventional fuel mass consumption as part of the optimization cost

function. In addition, battery life is taken into consideration by taking note of the charge/discharge cycle count. Furthermore, a HIL test bench was designed so as to validate the results of the mathematical model. Results showed that the instantaneous optimization allows for the least amount of fuel consumption as compared to a fuzzy logic and PID controller. However, it did also contain a slightly higher charge/discharge cycle count. The HIL validation confirms these results. However, no direct correlation of charge/discharge cycle count to the battery life was given, except that more cycle counts would lead to a faster degradation of the battery, as this was outside the scope of the paper. In general, one would have to select the appropriate tradeoff between improved fuel consumption and improved degradation time of the battery depending on the application at hand.

Researchers in [51] have taken the ECMS control strategy a step further through the development of an adaptive ECMS. The proposed control strategy adds to the traditional ECMS framework an on-the-fly algorithm for the estimation of the equivalence factor according to the driving conditions. Thus, the resulting real-time energy management strategy is one that maintains the battery SOC and minimizes the fuel consumption according to the current road load. Through comparison of results with those of a DP controller and a traditional ECMS, it was found that the adaptive ECMS yields only slightly suboptimal results. However, the benefits lie in its validity in every driving condition, causality, and ability to be implemented in real-time.

A similar analysis can be found in [52], where the benefits of hybridization of fuel cell vehicles is addressed through the use of a vehicle simulator and a supervisory controller. The literature details the use of ECMS and a heuristic approach to ECMS as part of a comparison of global optimal and suboptimal results. The results of the analysis proved that the heuristic approach provides a very close approximation to the non-implementable ECMS control strategy while remaining causal and self-adaptive. Additionally, it was shown that fuel cell vehicles would greatly benefit from hybridization through the significant increase in fuel economy, among other things.

Other examples of the use of ECMS as a supervisory controller can be found in [53], [54], [55], and [56].

2.5.3.5 Conclusion

While most control strategies typically fall under the broad categories of rule-based control or optimization-based control, the literature shows several examples of control strategies that have aspects derived from both categories. One such example is the work done in [32] where the use of a rule-based ECMS energy management strategy is applied to a parallel hybrid electric vehicle.

CHAPTER 3: Baseline MIL Validation and CAN Data Logger Development

3.1 Introduction

A fuel cell hybrid bus is being built as part of the National Fuel Cell Bus Program as a technology demonstration project. In an attempt to reduce build time and ensure that performance requirements are met prior to the completion of the bus build, a simulator of the bus was developed. The fuel cell hybrid bus will be run as part of the OSU CABS bus fleet and should run continuously within an eight hour shift while maintaining a charge sustaining behavior of the battery and a low fuel consumption as compared to a conventional transit bus. In order to obtain the data needed for the demonstration evaluation, a controller area network (CAN) based data acquisition (DAQ) system was developed. The following sections are an overview of the performance metrics used for validation, the model/simulator architecture, and proposed control algorithm as developed by a member of the OSU team. An analysis of

the simulator performance on standardized drive cycles is provided to establish a baseline for the model and controller validation. Lastly, the development of the CAN DAQ system is discussed, which is a minor part of the work.

3.2 Performance Validation Metrics

As with any engineering project or problem, properly defining the problem and its requirements and specifications helps in ensuring that the project is completed correctly, or, in terms of the fuel cell hybrid bus, it is built to operate as desired. The same goes for the validation process undertaken for this research. Failure to do so could result in wasted time and resources, not to mention a lack of understanding of what is being validated and why the validation is being performed. In [15], the authors detail a process of defining the problem and its requirements and specifications. The following is structured after such a process.

Design Problem:

The architecture of the fuel cell bus was determined and a MIL simulator created, as detailed in [44], to determine if performance requirements of the bus are met by the selected architecture. Further validation must be obtained in the form of SIL and HIL testing to understand how the control software would actually perform in a real-time controller. The plant model used is that of a series hybrid powertrain with a fuel cell stack and large battery pack. In addition, a model of the vehicle dynamics is included. Two supervisory controller models have been tested, that of an ARMA and

PMP based model. It is desired to validate the performance of the controllers with the plant model and driver model. Also, the simulator is to be run with real-world drive cycles obtained from on campus bus routes. Thus, a driver model that tracks distance has been developed, to be discussed in greater detail in Chapter 4.

Process Requirements:

The validation process must determine whether or not the model architecture and control algorithm are sufficiently designed to allow the simulated vehicle to meet the desired performance requirements. The performance requirements are determined by the driving conditions found in the real-world drive cycles. Additionally, the energy consumption of the fuel cell hybrid bus being simulated must be minimized, and a charge sustaining behavior of the battery must be maintained.

Process Specifications:

- The vehicle will follow a velocity trace set by standardized drive cycles with no more than a 5% velocity RMS error.
- The vehicle will follow a velocity trace set by real-world drive cycles obtained from on campus bus routes with no more than a 5% velocity RMS error.
- The vehicle will follow a distance trace set by real-world drive cycles obtained from on campus bus routes with no more than a 5% distance RMS error.

- The vehicle will follow the real-world drive cycle obtained from on campus bus routes with no more than a deviation of 50 m in total distance traveled from that of the drive cycle.
- The vehicle will maintain a charge sustaining behavior within 50% to 70% state of charge (SOC).
- The average fuel economy of the vehicle will be greater than the 4 mpg achieved with a baseline diesel bus as detailed in [4].

Requirements Traceability:

Figure 5 contains a representation of the performance validation metrics defined by the specifications that are used to meet the needs or requirements of the bus.

Needs		Metrics	1	2	3	4	5	6
			Velocity trace followed with ≥5% RMS error	Distance trace followed with ≥5% RMS error	Total distance traveled within a 50 m difference	SOC at end of cycle within 50% to 70%	Fuel Economy greater than 4 mpg (diesel baseline)	SIL /HIL are in agreement with MIL in terms of trend and pattern
1	Track velocity to match up with regulatory drive cycles		●					
2	Track velocity to match up with real world drive cycles		●					
3	Track distance to match up with real world drive cycles			●	●			
4	Maintain battery charge sustaining behavior					●		
5	Minimize fuel consumption of fuel cell						●	
6	Validate MIL results with SIL and HIL simulations		●	●	●	●	●	●

Figure 5: Requirements Traceability Diagram

3.3 Vehicle Architecture and Design Characteristics

The fuel cell hybrid transit bus has a battery dominant series hybrid powertrain architecture. This architecture has the benefit of a decrease in fuel consumption as compared to a conventional transit bus or even a fuel cell dominant hybrid bus. The ECO Saver IV uses a fuel cell system as an auxiliary power unit (APU) to maintain the state of charge (SOC) of the battery at a nominally charge sustaining behavior while also providing power to any auxiliary components, such as air conditioning. In terms of tractive power, both the battery and fuel cell are used to power the bus' two rear-wheel electric traction motors. Also, a DC-DC converter is to be used to regulate the output voltage of the fuel cell system. The complete fuel cell system consists of a 75 kW Ballard FCvelocity-HD6 PEM fuel cell module and an integrated Eaton compressor for the pressurization of the fuel cell air supply. The Ballard fuel cell module uses pressurized air and gaseous hydrogen and contains an internal cooling and humidification system that is managed by a control unit that is designed for integration with bus applications. Work done in [44] involved performing simulations to determine the manufacturer, size, and chemistry of the battery used. Results led to the selection of a 600 kW GAIA Lithium Iron Phosphate (LFP) battery pack. The electric motors intended for use are two ZF transaxles with motors rated at a maximum power of 240 kW and a maximum speed of 1150 rad/s. The bus is also designed to take advantage of regenerative braking to capture some of the energy traditionally lost with friction based brake systems. A full

list of the powertrain components used is given as Table 1 and a diagram provided by DesignLine of the proposed powertrain architecture is shown as Figure 6.

Table 1: Powertrain Component Sizing and Manufacturers

Component	Component Specifications/ Details
Fuel Cell	Ballard FC velocity HD6, 75 kW max power
Fuel Cell Compressor	Eaton
Battery	GAIA Lithium Iron Phosphate, 600 kW max power
Electric Motors	2 motors, 240 kW max power, 11 50 rad/s max speed
Gearbox	Single 22.63 gear ration

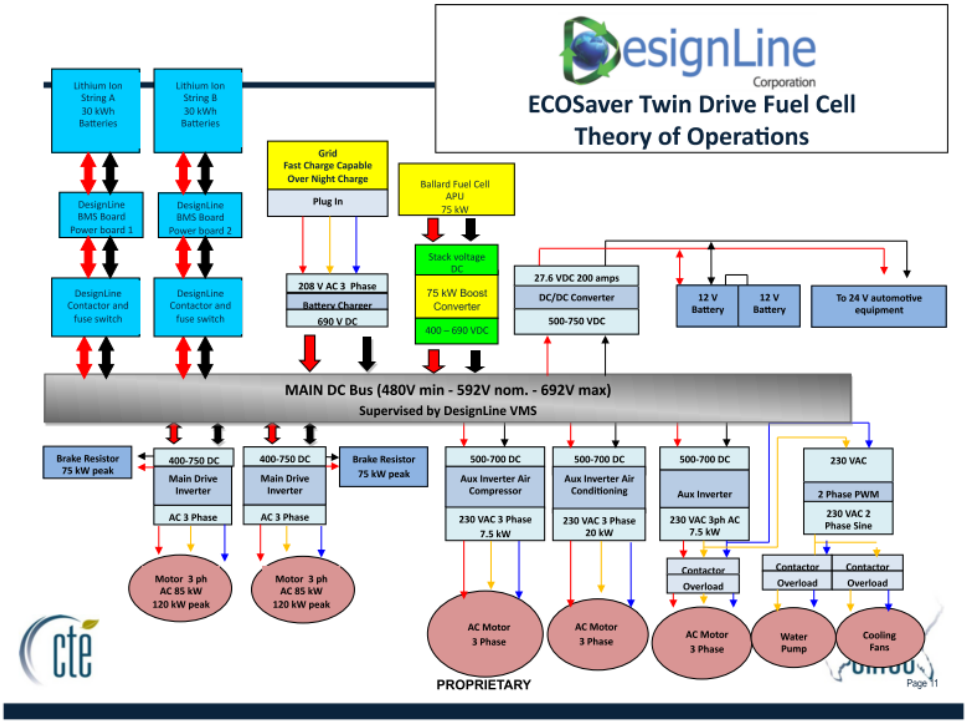

Figure 6: Powertrain Architecture as Proposed by DesignLine

In addition to establishing the proposed vehicle powertrain architecture for the fuel cell hybrid bus, DesignLine provided the various vehicle characteristics which affect the performance and dynamics of the bus such as vehicle weight and drag coefficient of the bus body design. Table 2 includes basic information about some of the vehicle characteristics as dictated by the design of the bus. Some of these characteristics are used as part of the vehicle simulator to ensure that proper calculations are made of the vehicle dynamics. Incorrect identification of these characteristics could result in the use of improperly configured powertrain components and incorrect estimations of the forces acting on the bus. For example, using a wrong value for drag coefficient would mean either an under or over estimation of the aerodynamic forces acting on the bus and would have effects on the estimated fuel consumption calculations. A full analysis of the vehicle characteristics affecting the vehicle longitudinal dynamics will be discussed in a later section dedicated to the vehicle longitudinal dynamics model. However, those listed below can be used as part of a vehicle design analysis where the effects of fluctuations in the vehicle weight due to changes in the number of passengers can be studied. This analysis will be performed and presented at the end of the following chapter.

Table 2: Vehicle Design Characteristics and Requirements

Vehicle Characteristics	Value
Weight	35000 lbs.
Seats	41

3.4 OSU Vehicle Model/Simulator

3.4.1 Simulator Architecture

In order to properly design vehicle models, the vehicle architecture and characteristics must be taken into consideration and an attempt must be made to create models that match the actual vehicle components as closely as possible. However, the degree of detail and accuracy used in a vehicle model is greatly dependent on the purpose of the simulator. The fuel cell hybrid bus simulator employs a forward-looking energetic approach with the purpose of determining the ideal power management strategy to achieve acceptable fuel economy, battery charge sustainability, and vehicle performance metrics and was designed in a modular manner with separate models for the various powertrain components. As shown in Figure 7, the forward-looking energy based vehicle simulator contains model function blocks of a driver, supervisory controller, hybrid powertrain, and vehicle dynamics.

Simulations are performed using a specific velocity profile or drive cycle. The chosen standardized drive cycles for use with the fuel cell hybrid bus simulator include the Heavy Duty Urban Dynamometer Driving Schedule (HDUDDS) and the Manhattan Bus Cycle (Manhattan), displayed as Figure 8 and Figure 9, respectively. Drive cycles are typically used by the EPA as part of their fuel economy and emissions regulations testing but are also used to obtain estimates with vehicle models and simulators. Specifically, the Manhattan and HDUDDS drive cycles are meant to be indicative of a typical velocity trace a heavy duty vehicle like a bus would be expected to accomplish despite their

great differences like average and maximum velocity. These drive cycles serve as baseline estimates of fuel economy and performance of a vehicle simulator.

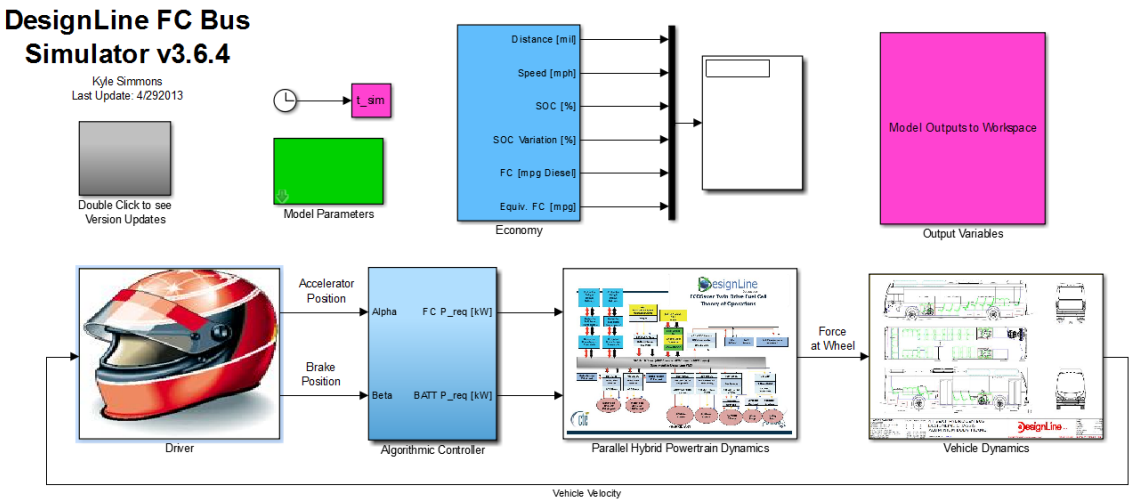

Figure 7: Fuel Cell Hybrid Bus Simulator

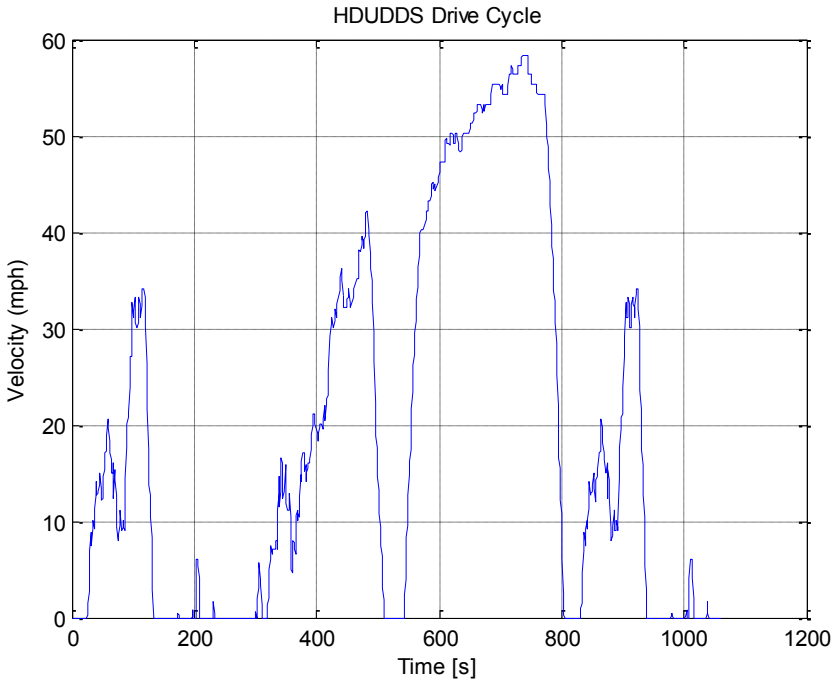

Figure 8: Heavy Duty Urban Dynamometer Driving Schedule (HDUDDS)

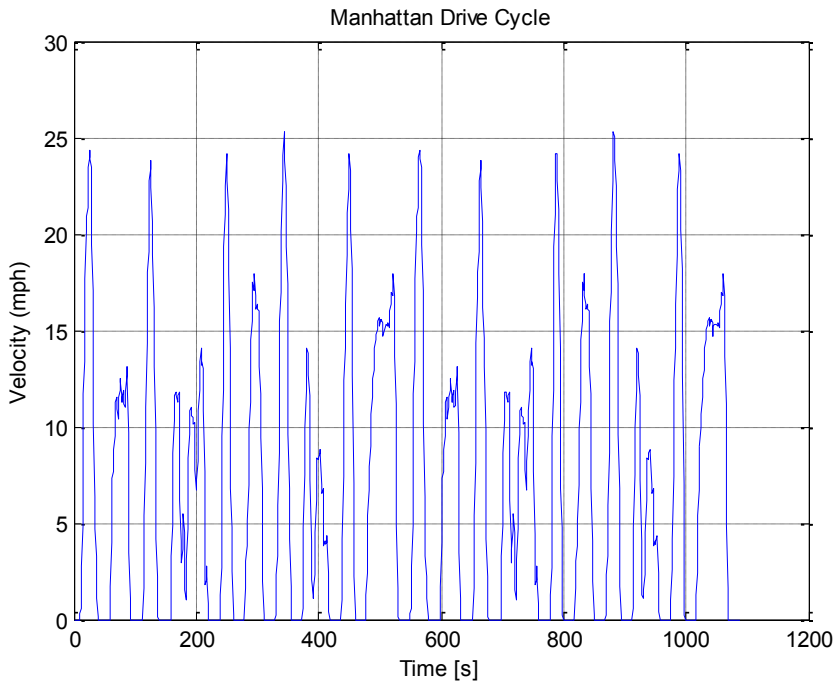

Figure 9: Manhattan Bus Driving Cycle

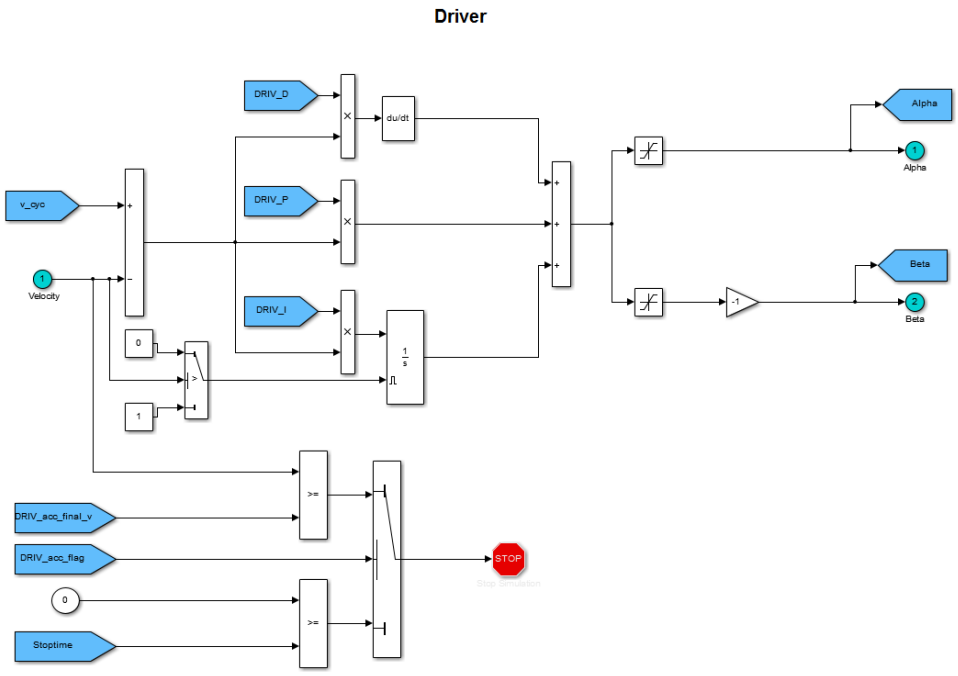

Figure 10: Driver Model - Fuel Cell Hybrid Bus Simulator

3.4.1.1 Driver Model

The driver model, Figure 10, is used in the determination of the accelerator and brake pedal positions at each time step. This is accomplished by way of a tuned PID controller that attempts to minimize the velocity error between the current simulated vehicle velocity and the desired velocity specified by the drive cycle used. The PID controller gains must be tuned properly to allow for acceptable velocity error values. The tuned controller gains used with the simulator are given in Table 3.

Table 3: Driver Model PID Parameter Gains – Simulator

Parameter	Value
$\boldsymbol{K_P}$	2.3
$\boldsymbol{K_I}$	0.6
$\boldsymbol{K_D}$	0

The output of the PID controller can either be negative or positive. A positive PID controller output indicates that the current calculated velocity was slower than that of the desired velocity. Thus, the driver model outputs a zero brake command and the nonzero acceleration command,

$$\alpha = K_P\left(v_{cyc} - v_{veh}\right) + K_I \int \left(v_{cyc} - v_{veh}\right)dt + K_D \frac{d}{dt}\left(v_{cyc} - v_{veh}\right) \tag{30}$$

where K_P, K_I, and K_D are the respective proportional, integral, and derivative gain constants, v_{cyc} is the desired cycle velocity, and v_{veh} is the actual calculated velocity. Likewise, a negative PID controller output is given when the calculated velocity is higher

than that of the drive cycle, resulting in a zero acceleration command and the nonzero brake command,

$$\beta = -\left(K_P\left(v_{cyc} - v_{veh}\right) + K_I \int \left(v_{cyc} - v_{veh}\right)dt + K_D \frac{d}{dt}\left(v_{cyc} - v_{veh}\right)\right). \quad (31)$$

Respective values of alpha and beta lie between zero and plus or minus one, where a zero value means that the respective pedal is not being used, and a value of one means that the respective pedal is being fully compressed. [44]

3.4.1.2 Supervisory Controller Model

The supervisory controller model is meant to determine how best to meet the calculated power demand related to the driver command received. How this is accomplished depends greatly on the control algorithm used. In a general sense, the controller model takes the driver command and calculates either the maximum acceleration or maximum braking power, and based on the power calculated, the battery SOC, and any other parameters used by the algorithm, outputs a battery and fuel cell power demand that the powertrain model must meet. Information on the specific control strategies used will be discussed in a later section.

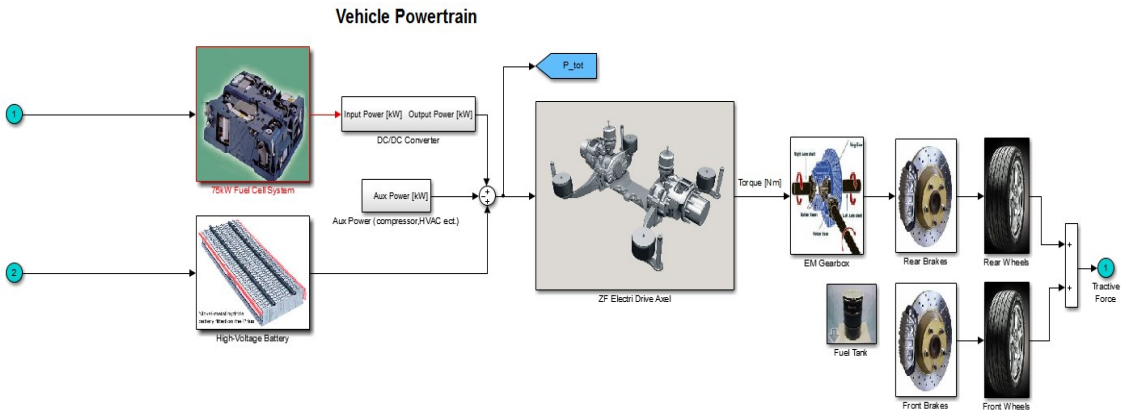

Figure 11: Powertrain Model - Fuel Cell Hybrid Bus Simulator

3.4.1.3 Powertrain Model

The powertrain model consists of several sub-models of the fuel cell stack, battery pack, DC-DC converter, compressor and auxiliary loads, duel electric motors, gearbox, fuel tank, front and rear brakes, and front and rear wheels, as shown by Figure 11. Having separate models for each component of the powertrain allows for a modular design that can be changed by simply bringing in new models of the respective component being changed without affecting the rest of the powertrain model. Additionally, this allows one to adequately analyze the energy flow and consumption of the fuel cell hybrid bus. [44] Most of the component sub-models were represented as a static model and thus neglect any dynamic behavior. The dynamic behavior was not needed for the purpose of an energy analysis and fuel consumption estimation. For example, while temperature, air pressure, and humidification are critical for the accurate modelling of a fuel cell, the effects of these were neglected as it was assumed

that the manufacturer's controller would adequately maintain them at their appropriate levels for proper optimal operation of the fuel cell.

The fuel cell model was based on a table interpolation of the power-current relationship as shown in experimental data provided by the manufacturer as well as a curve fit for improved model resolution. While the electric motors are also based on a static model, they use experimentally obtained efficiency and speed-power maps instead of a curve fit of the power-current relationship. Due to the need of an accurate dynamic model for the proper estimation of battery SOC, it was not possible to model the battery as a static model. The dynamic behavior of batteries is attributed to factors like electrochemistry and ion diffusion and as such are highly complex and difficult to predict. Fortunately, these dynamic behaviors can be predicted through approximation by an equivalent electrical circuit of reduced order. Thus, the battery was modeled as an equivalent electrical circuit model.

Since the work presented in this thesis did not include the development of the vehicle simulator and its component models, a detailed study of the components used in the powertrain model is beyond the scope of this thesis. For a more detailed explanation of the models and experimental data used as a part of the powertrain model, including those of components not discussed, see [44]. Where appropriate, some model equations related to the power management and fuel consumption estimation will be presented but should not be taken as a full representation of the

component models that make up the complete powertrain model. The powertrain model output is in the form of a total force supplied to the wheels.

3.4.1.4 Vehicle Dynamics Model

Since the simulator was designed to perform an energetic analysis the effects of lateral dynamics can be ignored. On the other hand, longitudinal dynamics are still very much important to the proper modeling of the vehicle dynamics and are implemented by simple dynamic equations of aerodynamic drag, rolling resistance, and grade resistance forces. The total tractive force at the wheels F_{tot} is then combined with these environmental forces that act against the bus, and applying Newton's second law gives

$$m\dot{v}_{veh} = F_{tot} - F_a - F_r - F_g \tag{32}$$

where m is the mass of the bus, $\dot{v}_{veh}$ is the longitudinal acceleration of the bus, F_a is the aerodynamic force, F_r is the rolling resistance force, and F_g is the force due to road grade. The environmental forces acting against the bus are respectively calculated by

$$F_a = \frac{1}{2}\rho_{air} c_d A_f v_{veh}^2 \tag{33}$$

$$F_r = mg\cos\theta\, c_r v_{veh} \tag{34}$$

$$F_g = mg\sin\theta \tag{35}$$

where ρ_{air} is the density of air, c_d is the aerodynamic drag coefficient, A_f is the frontal area of the bus, v_{veh} is the longitudinal velocity of the bus, g is gravity, θ is the road grade angle, and c_r is the rolling resistance coefficient. Equation (32) is then used to

determine the velocity of the bus by integration of the longitudinal acceleration with respect to time. This calculated velocity is then fed back to the driver so that it may compare with the desired drive cycle. Values for the parameters used within the vehicle dynamics model can be found in Table 4.

Table 4: Vehicle Dynamics Parameters

Parameter	**Value**
Vehicle Weight	35000 lbs.
Gravitational Acceleration	9.81 m/s^2
Rolling Resistance Coefficient	0.006
Density of Air	1.29 kg/m
Aerodynamic Drag Coefficient	0.54
Frontal Surface Area of Bus	7.5 m^2

3.4.2 Supervisory Control Strategies Used

The supervisory controller is what communicates with the plant model in order to operate the simulated vehicle as desired. Specifically, the supervisory controller takes in pedal positions as inputs and then outputs the power demands to the battery and fuel cell system. For the work discussed in this thesis, two different supervisory controllers were used. The first was based on Pontryagin's Minimum Principle and is one of many optimization-based control strategies used to find the global optimal control solution. Detailed information concerning the PMP control strategy algorithm can be found in Chapter 2. A simplified diagram of the PMP supervisory controller is given as Figure 12.

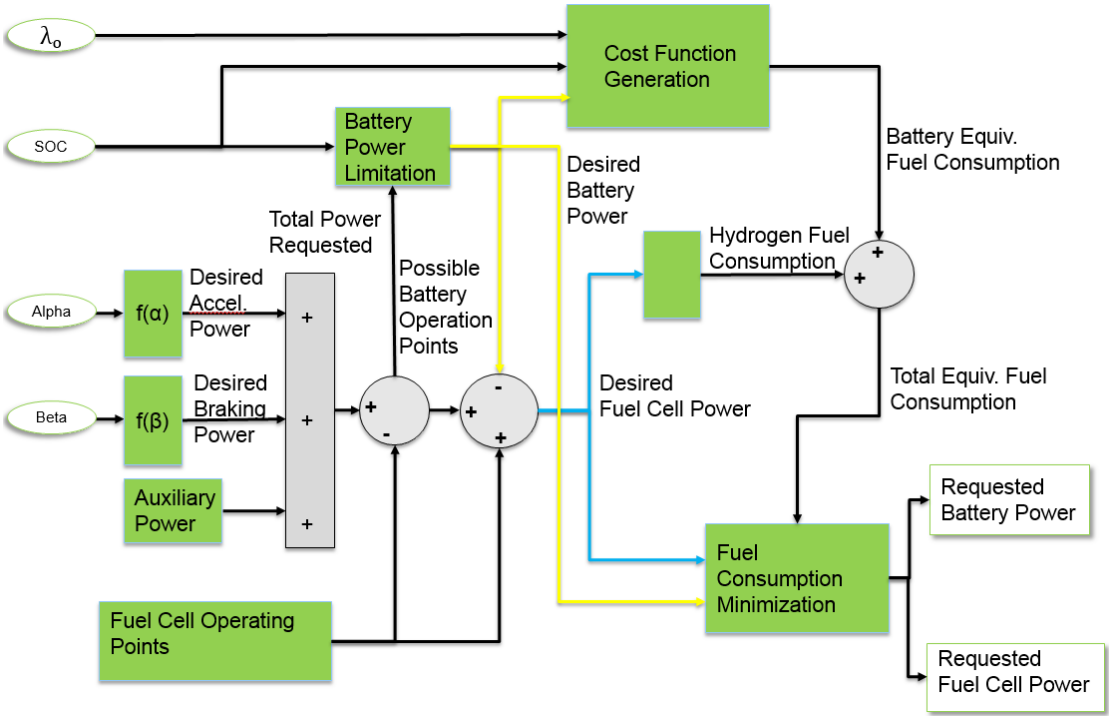

Figure 12: PMP Supervisory Controller Diagram

It should be noted that the PMP controller utilizes a constant optimal co-state variable λ_0^* for each drive cycle as determined by initial simulation work. Table 5 shows the respective values of the co-state variable used for the HDUDDS and Manhattan driving cycles.

Table 5: Co-state Variable Value for HDUDDS and Manhattan

	λ_o Values
HDUDDS	2038860
Manhattan	1945260

The full development of the PMP controller model for the simulator is explained in greater length in [44]. For this thesis, the inclusion of the cost function, Hamiltonian function, and the necessary conditions for optimality will suffice. Therefore, the performance criterion of PMP

$$J = \int_0^{t_f} \dot{m}_H(P_{batt}(t), t)dt, \tag{36}$$

where $\dot{m}_H$ is the mass flow rate of hydrogen and P_{batt} is the battery power demand, leads directly to the formulation of the Hamiltonian function

$$H = \lambda(t)S\dot{O}C(SOC(t), P_{batt}(t), t) + \dot{m}_H(P_{batt}(t), t) \tag{37}$$

in which $S\dot{O}C$ is known as the battery SOE. The necessary conditions are then,

$$\dot{x}^*(t) = \frac{\partial H}{\partial \lambda} = S\dot{O}C \tag{38}$$

$$\dot{\lambda}^*(t) = -\frac{\partial H}{\partial x} = -\frac{\lambda}{Q_{cell}} \frac{I_{cell}}{\sqrt{(V_{OC} - \sum_{i=1}^{2} V_{Ci} - V_h) - 4P_{cell}R_0}} \frac{\partial V_{OC}}{\partial SOC} \tag{39}$$

$$SOC^*(0) = SOC_0 \tag{40}$$

$$SOC^*(t_f) = SOC_0 \tag{41}$$

$$SOC^*(t) \in \mathcal{X}(t) \tag{42}$$

$$P_{batt}{}^*(t) \in \mathcal{U}(t) \tag{43}$$

where Q_{cell} is the total charge capacity of a battery cell, I_{cell} is the current through the cell, V_{OC} is the open circuit voltage, V_{Ci} is the voltage across the i^{th} capacitor, V_h is the voltage due to hysteresis, P_{cell} is the battery cell power, and R_0 is the internal resistance

of the battery. Recall, however, that PMP is not implementable in many real-time control systems because it requires full knowledge of the drive cycle being used before determining the optimal solution. Still, it can be used as a standard to which other control algorithms can be compared. In addition, optimal results obtained from PMP can be used to develop suboptimal, but implementable control algorithms.

As part of the demonstration project, OSU developed a control algorithm that would possibly perform better than a more traditional approach yet not require precise knowledge of the drive cycle. The proposed control algorithm was developed through analysis of the results given by the PMP controller and is based on an autoregressive moving average (ARMA) model. The ARMA model is a statistical analysis tool that is used in the prediction of future values in a time series of a data variable and consists of two parts, an autoregressive part and a moving average part. Thus, the controller was modeled such that the fuel cell stack power demand is determined by

$$P_{FC,k+1} = \sum_{i=1}^{N} \alpha_i P_{FC,k+1-i} + \sum_{j=1}^{M} \beta_j \left(SOC_{k+1-j} - SOC_{ref} \right) \tag{44}$$

where the first term is the autoregressive average of N past values of the fuel cell stack power P_{FC} with weighted variable α_i, and the second term is the moving average of M past values of the SOC deviation from the target SOC with weighted variable β_j. The ARMA control algorithm therefore works by determining the best fuel cell stack power for the current time step based on the weighted average of the past values of fuel cell

stack power and SOC deviation. Figure 13 contains a diagram of the ARMA supervisory controller.

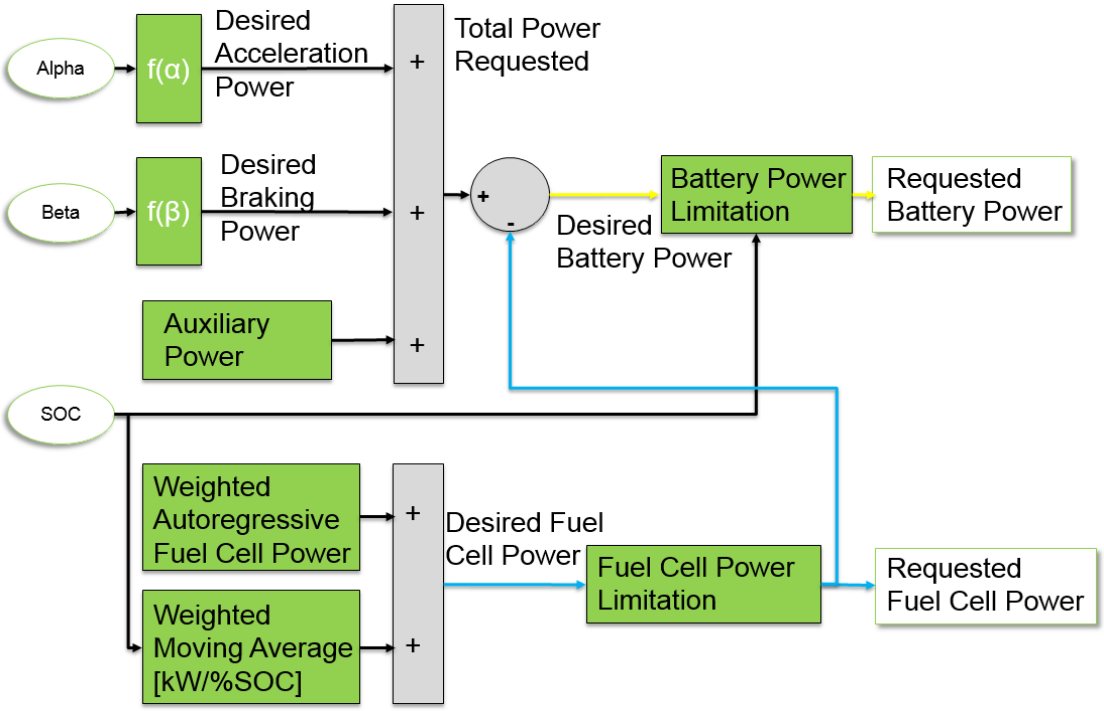

Figure 13: ARMA Supervisory Controller Diagram

By setting the sum of the weighted variable α_i equal to one and interpreting β_j as the gains of an extended PI controller, the control algorithm has been shown to exhibit stability. However, for a specific desired performance of the controller, the period of adaption would have to be tuned, where k is an index of adaptation time.

A definite benefit of the proposed ARMA control strategy is that it contains inherent low pass filtering which limits the fuel cell transients and prolongs the life of the fuel cell stack. In fact, the fuel cell is seen to operate in a drastically smoothed out manner to that of the PMP controller. Though initial inspection of the ARMA control

strategy in [44] indicates good performance under various driving conditions, a full analysis and validation of its performance was left as future work. This validation and analysis will be presented in the bulk of this thesis.

3.5 Proposed Controller Validation - Performance on Standardized Drive Cycles

As part of OSU's role in the ECO Saver IV fuel cell hybrid electric bus project, the proposed algorithm, detailed in the previous section, would have to be tested and validated to show that it could achieve the desired goals of reduced fuel consumption, charge sustaining behavior, and appropriate transient behavior of the battery and fuel cell stack. In order to obtain a set of baseline results for the validation and comparison of the proposed control strategy, the simulator was run on the chosen standardized drive cycles using both supervisory controllers. The calcualted relative RMS error for the HDUDDS drive cycle under the ARMA and PMP controller are given in Table 6. The values indicate that the ARMA and PMP controllers allow for a velocity error that is less than the valiadtion perfromance metric of 5%. Thus, both control strategies provide adequate performance in terms of being able to match up the calculated velocity to the desired velocity trace.

Table 6: Comparison of ARMA and PMP - HDUDDS Velocity RMS Error

	Velocity RMS Error (%)
HDUDDS – ARMA MIL	1.05
HDUDDS – PMP MIL	1.20

However, a proper analysis of the results requires more than checking if the driving conditions were met. It is necessary to see what the main powertrain components do throughout the duration of the drive cycles as well as determine whether the driving characteristics were appropriate in terms of the behavior of the accelerator and brake pedal positions. Figure 14 contains a plot of the battery SOC on the HDUDDS drive cycle for the ARMA and PMP controllers. It can be seen that the ARMA controller is very closely related to the optimal SOC trajectory. More importantly, the ARMA controller is able to maintain the bus' charge sustaining behavior within the desired 50% - 70% SOC. However, as the algorithms differ, there is naturally a difference in the way the power demands are determined.

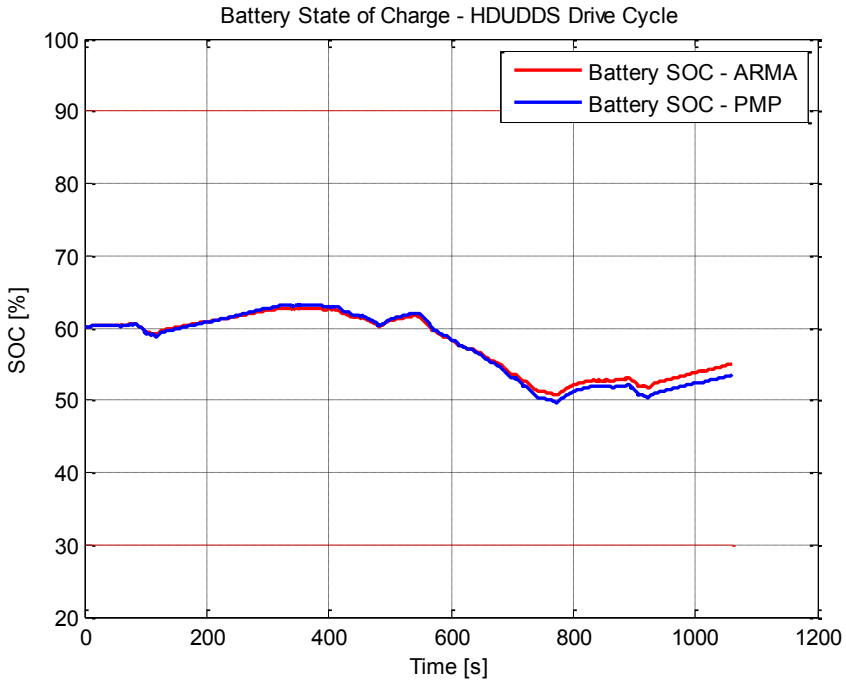

Figure 14: Comparison of ARMA and PMP – HDUDDS Battery SOC

While the PMP controller works by minimizing the mass flow rate of hydrogen with regards to the control of battery power demand and with necessary conditions focused on maintaining the battery charge sustaining behavior, the ARMA controller, on the other hand, works by determining the proper fuel cell power demand with respect to the average of the past fuel cell power values and the SOC variation. The PMP controller uses the battery power as part of its control parameter.

Due to the fuel cell being used as an APU, any energy from hydrogen used would either be used to charge the batteries, power auxiliaries, provide power to the traction motors, or regenerated from the wheels. This creates a difference in how the fuel cell power demand is treated in respect to its dynamics. PMP's focus on the minimization of fuel flow rate with respect to the control of battery power leads to a trajectory of fuel cell power that contains several transients. In contrast, the ARMA controller focuses on determining a smooth fuel cell power trajectory. This analysis of the structure of the algorithms and differences in how power demands are determined is confirmed by Figure 15 where the PMP results clearly show a significant amount of fuel cell transients that could negatively impact the life of the fuel cell stack. Thus, it can be said that the proposed algorithm not only meets the battery SOC and driving conditions of standard drive cycles, but also ensures the fuel cell is transient free.

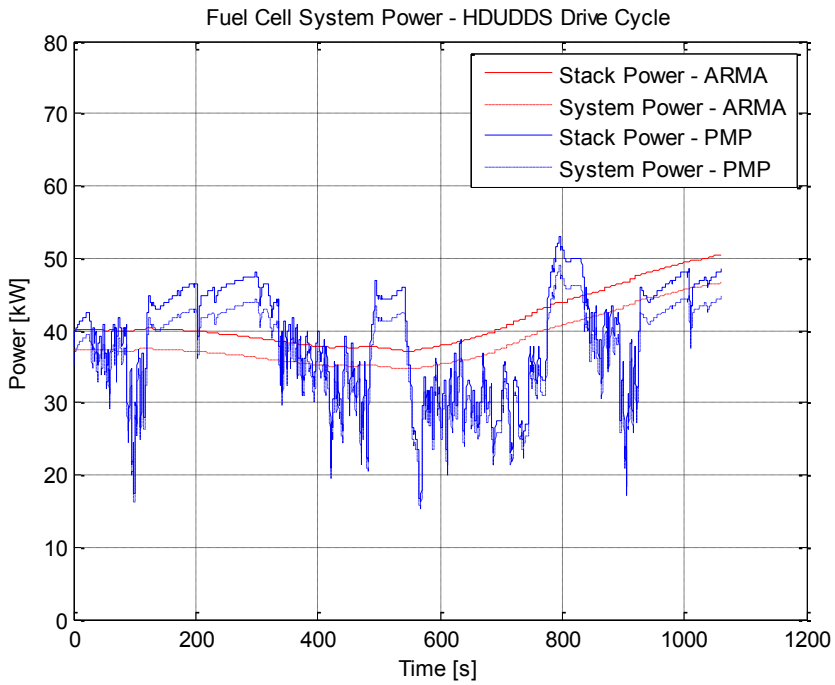

Figure 15: Comparison of ARMA and PMP – HDUDDS Fuel Cell Power Demand

Finally, as the concern of the automotive industry has rightly been on the improvement of fuel economy, it is of special interest to determine how the proposed controllers fair in terms of fuel consumption and its diesel equivalent mileage. Table 7 displays the hydrogen fuel consumption and diesel equivalent fuel economy of the fuel cell hybrid bus on the HDUDDS drive cycle for both the ARMA and PMP controllers. While the mpg values shown do not seem all that impressive in view of the typical values of passenger hybrid vehicles, one must keep in mind that the bus weighs nearly 40 tons.

Table 7: Fuel Consumption and Diesel Equivalent Fuel Economy - HDUDDS

	Fuel Consumption (kg)	**Diesel Equivalent Mileage (mpg)**
HDUDDS – ARMA MIL	0.566	8.28
HDUDDS – PMP MIL	0.672	8.41

Thus, considering that a conventional diesel transit bus, as determined in [4], achieves an estimated mileage of 4 mpg, the value of the diesel equivalent fuel economy obtained with the ARMA controller indicates an approximate improvement in mileage by a factor of two. Furthermore, comparison to the optimal fuel consumption and diesel equivalent fuel economy of the PMP controller, respectively, shows that the ARMA controller provides a solution that is very close to the optimal solution. Thus, it is said that the proposed ARMA control strategy is a near-optimal solution. Furthermore, it is concluded that the ARMA controller provides an implementable control algorithm that displays a limited amount of fuel cell transients and a battery charge sustaining behavior.

However, in order to verify that the conclusions reached concerning the performance of the ARMA controller are indeed valid for varying driving conditions, simulation results using the Manhattan bus driving cycle were also obtained and compared to their respective PMP optimal counterparts. Table 8 verifies that the velocity RMS values for the Manhattan are well within the desired range.

Table 8: Comparison of ARMA and PMP - Manhattan Velocity RMS Error

	Velocity RMS Error (%)
Manhattan – ARMA MIL	2.29
Manhattan – PMP MIL	2.09

Comparison of the battery SOC and fuel cell power demand show a clear difference between the ARMA and PMP. While the battery SOC of both controllers is relatively charge sustaining, Figure 16 shows that they differ in that the ARMA battery SOC has an upward trend, and the PMP battery SOC has a downward trend. In addition, it is seen that the PMP SOC trajectory for the Manhattan drive cycle did not meet the necessary condition that requires the final SOC to be equal to the reference SOC. In order to ensure that the final SOC equals the reference SOC, the co-state variable would need to be changed. However, the co-state value used was chosen as it allowed the SOC trajectory to be the closest it could get to a true charge sustaining behavior while still providing a reasonable overall performance of the bus. Even more surprising than the differences in SOC trend for the Manhattan drive cycle are the differences seen in the fuel cell power trajectory, shown in Figure 17. The ARMA fuel cell power demand has a smooth trajectory as expected from the ARMA algorithm. However, while the PMP controller stays rather steady with respect to an average fuel cell power demand, the ARMA controller has a downward trend.

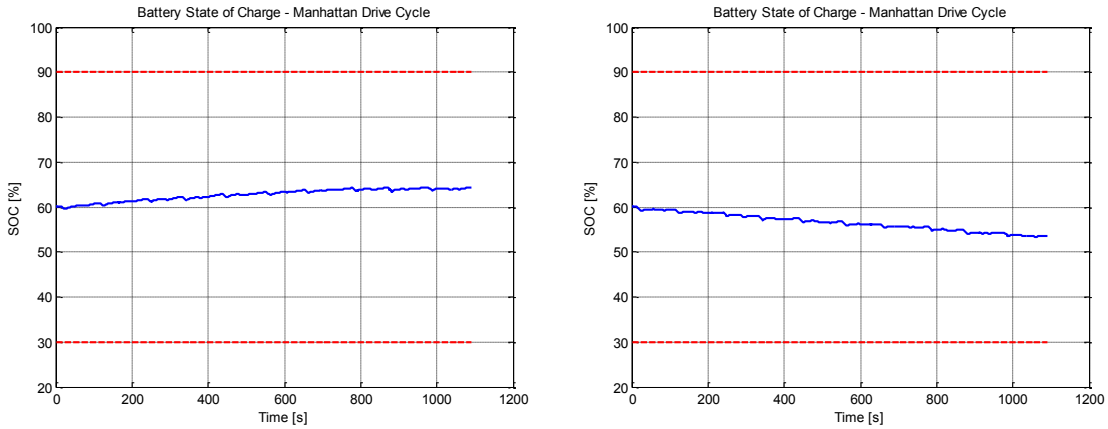

Figure 16: Comparison of ARMA (left) and PMP (right) Battery SOC– Manhattan Battery SOC

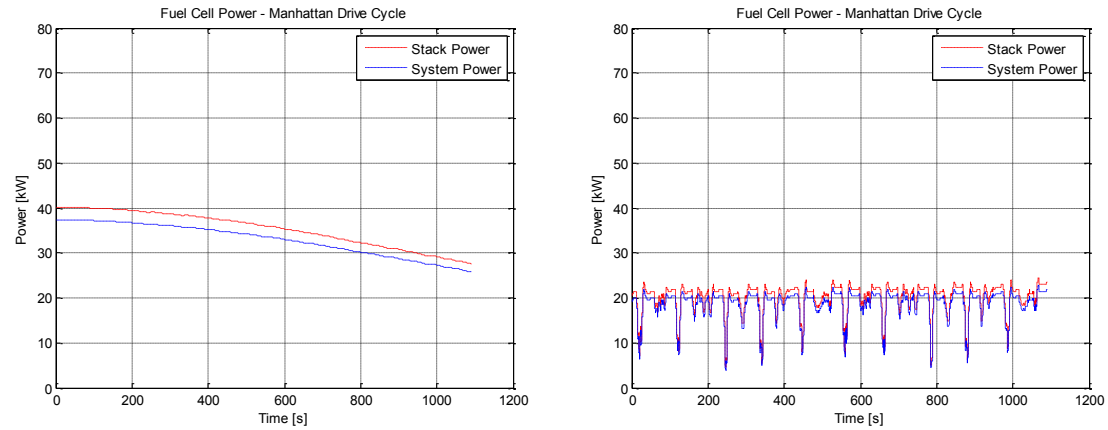

Figure 17: Comparison of ARMA (left) and PMP (right) Fuel Cell Power Demand – Manhattan

Furthermore, while the optimal solution has an initial fuel cell power demand of about 20 kW, the ARMA controller starts off at about 40 kW. It is assumed that the original developer of the control strategy set the initial fuel cell power of 40 kW as this coincides with the initial fuel cell power found in the PMP HDUDDS results. A look at

the diesel equivalent fuel economy of both controllers for the Manhattan drive cycle in Table 9 shows that this difference in initial fuel cell power has a drastic effect. Though the optimal diesel equivalent fuel economy is approximately twice that of a conventional diesel bus, the average fuel economy of the ARMA controller is practically that of a diesel bus. As shown by the table below, decreasing the initial fuel cell power used in the ARMA controller allows for almost twice the fuel economy to be achieved.

Table 9: Fuel Economy to Initial Fuel Cell Power Relation

Initial Fuel Cell Power	**Diesel Equivalent Fuel Economy**
40 kW	4.21 mpg
20 kW	7.73 mpg

By taking a closer look at the fuel cell power demand of the ARMA controller, it is clear that, as the drive cycle progressed, the ARMA controller continually decreased the fuel cell power demand as the fuel cell hybrid bus clearly did not need it to achieve the lower velocities and greater start-stop events of the Manhattan drive cycle, hence the downward trend. Thus, the initial fuel cell power demand set within the ARMA controller was changed to 20, as dictated by the optimal PMP results, and the simulator was run. Results showed that for the ARMA controller, starting at an initial fuel cell power of 20 kW still allowed the simulator to achieve the desired driving conditions. The battery SOC in Figure 18 is charge sustaining and follows a trend similar as the optimal battery SOC.

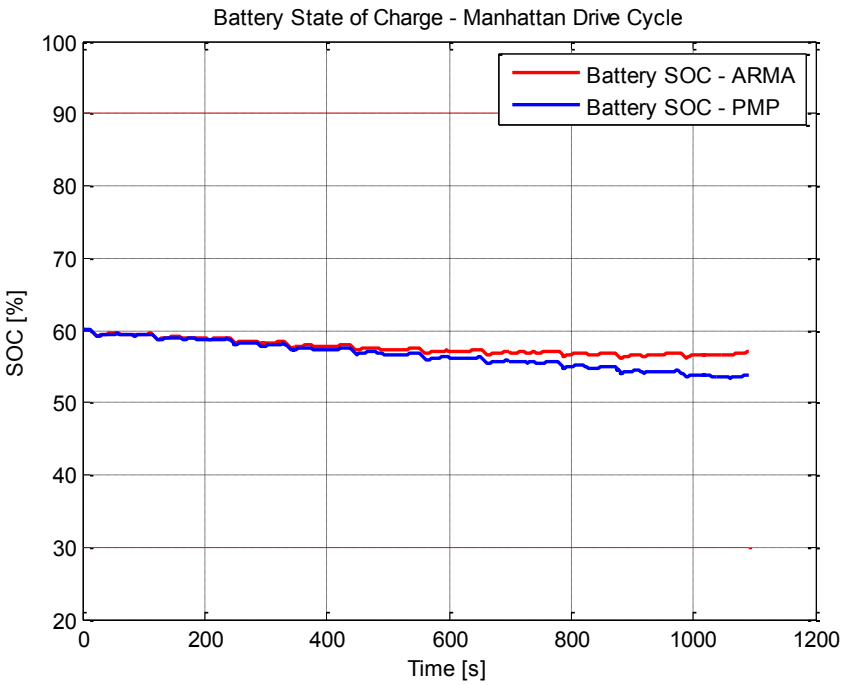

Figure 18: PMP and ARMA Battery SOC for 20 kW Initial Fuel Cell Power - Manhattan

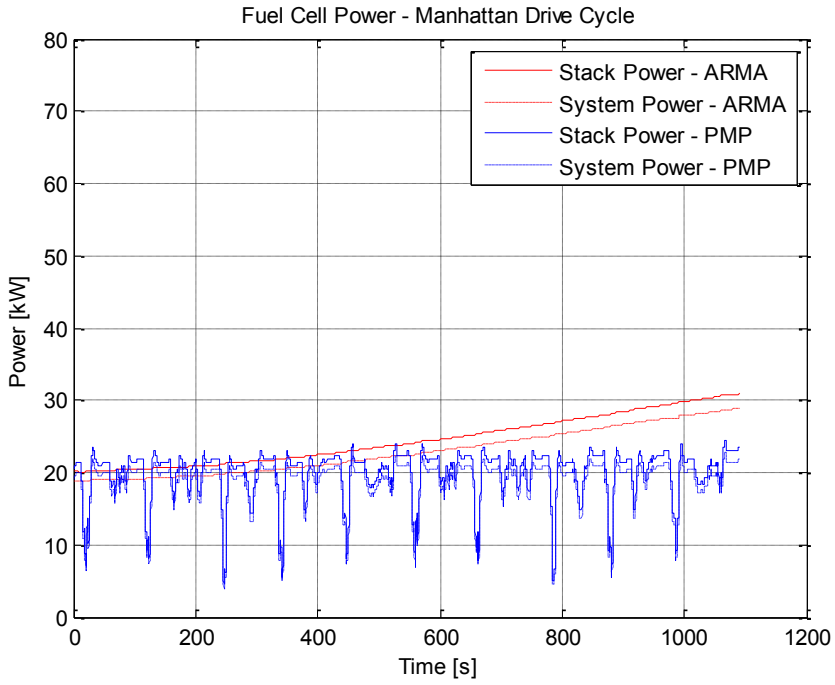

Figure 19: PMP and ARMA Fuel Cell Power Demand with 20 kW Initial Fuel Cell Power – Manhattan

Table 10: Manhattan - Fuel Consumption and Diesel Equivalent Fuel Economy

	Fuel Consumption (kg)	**Diesel Equivalent Mileage (mpg)**
Manhattan – ARMA MIL	0.336	7.69
Manhattan – PMP MIL	0.276	8.51

Also, the ARMA fuel cell power demand, Figure 19, now starts at 20 kW and increases towards about the same power level seen at the end of the 40 kW initial fuel cell power trajectory. A look at the new fuel consumption and diesel equivalent fuel economy given in Table 10 shows that this change in initial fuel cell power makes a significant difference. In fact, the new fuel economy is now closer to the optimal PMP mileage than one that resembles a diesel bus. The ARMA diesel equivalent fuel economy for an initial fuel cell power similar to that given by the PMP controller is approximately twice that of a conventional diesel bus. This leads to an important lesson in that the initial power of the fuel cell plays a crucial role in the fuel economy of the bus. For the remainder of the results, initial fuel cell power will be set to that which is given by the optimal solution.

3.6 CAN DAQ System Development

Per the initial task given as part of the ECO Saver IV bus project, a data logger was to be designed and developed to capture data while the bus would be in operation. The data recorded would later be used to analyze the performance and success of the fuel cell demonstration bus. In addition, DesignLine had plans to use the data as

research for possible future buses. The data would also be of use in the educational and industry advancement goals of OSU CAR.

3.6.1 CAN DAQ Hardware Development

As with any product design and development, design requirements must be first outlined so as to ensure the end product effectively performs the task desired. Upon initial conception of the DAQ system, a desire was identified to use hardware that would readily be able to communicate with the various electronic control modules contained within a vehicle. At one point in time, the only electronic device found on a vehicle was the radio. However, with the onset of advanced system control, on-board diagnostics and on-board electronic devices, there has been a multitude of electronic subsystems and components, such as the ECU, transmission control unit (TCU), anti-lock braking system (ABS), and the many control units used to control hybrid powertrain components, falling under the term of powertrain control module (PCM), that have been added as vital parts of a vehicle. Many of these devices have allowed for vehicles to achieve many of the emissions standards set out by the EPA while also contributed greatly to improvements in vehicle performance, driver and passenger comfort, ease of manufacture, and cost effectiveness. These electronic control modules receive sensor inputs like vehicle speed, temperature, and pressure that are used to control various actuators that carry out tasks determined by the control module. For example, the ECU must rely on information from sensors to know what velocity the vehicle is travelling,

which in turn would have to be communicated to the TCU so that it could determine when to shift gears.

Thus, there was a need to be able to control and connect all of these devices with one form of communication and without having to directly wire every component to each other. To meet this need, development of a vehicle network for the exchange of data was started and led to the creation of the CAN bus protocol in 1983. CAN bus, or CAN, is a protocol of a vehicle bus, that is, a specialized internal communications network that interconnects components inside a vehicle. Thus, using CAN as a way to communicate with the vehicle during the data logging process seemed like a natural choice. While vehicles have been known to use other vehicle bus protocols like local interconnect networks (LIN), CAN allows microcontrollers and devices to communicate with each other and with sensors and actuators without the need of a host computer. Instead, data is passed through CAN controllers and processors which are relatively inexpensive. Furthermore, each control module makes up a node within the CAN bus network and is able to send and receive messages that are pertinent to its functions through an arbitration system that depends on the priority of the message as determined by its message identifier or ID.

With this in mind, it was decided that the DAQ system should be able to communicate via CAN and be able to identify CAN message frames being sent and received through the network. In addition, it was decided that the hardware would need to have the ability to store data within an internal or external hard drive until it

was extracted and also allow for real-time data streaming allowing for analysis of data while the vehicle was in operation. This meant that the hardware selected for the DAQ system would also need to communicate with a host computer. Furthermore, the hardware used would have to be modular and rugged enough to withstand typical conditions within the confines of the vehicle's hood.

While all these constraints on the hardware design played an integral role in the selection of the hardware device, the intended use and functionality with the host computer during data extraction and analysis was perhaps the greater of the contributing factors in terms of shortening the list of possible devices. That is, since DesignLine would be given the data logger to use with the bus after the demonstration project was completed, the DAQ system had to be designed with a user friendly interface. Therefore, as is often the case, the design of the DAQ hardware was influenced by the design of the DAQ software. Specifically, the necessity for a user interface meant that the programming language used would need to allow for the creation of one. This led to the decision of developing the CAN DAQ software through the National Instruments (NI) visual programming language LabVIEW, which is a system design platform and development environment. Thus, with this decision made, the hardware device used would have to be able to readily interface with LabVIEW as well as meet all of the aforementioned constraints and requirements. Fortunately, NI is also a developer of LabVIEW compatible instruments for all sorts of applications, including the acquisition of data. However, while this greatly aided the selection process, there

was still a question of whether any devices that NI produced could meet the system design constraints.

With the aid of NI application engineers, the decision was made to use a NI compact reconfigurable input/output (cRIO) embedded real-time control and acquisition system which consists of an embedded controller for communication and processing that is attached to a reconfigurable chassis that houses the user-programmable FPGA and into which several I/O modules can be plugged in. Additionally, the system is programmed with LabVIEW and, depending on what type of I/O modules are used, can be used for a variety of embedded control and monitoring/acquisition applications. Particularly, the selected CAN DAQ system consists of a NI cRIO-9024 real-time controller, NI cRIO-9111 chassis, and a NI-9853 high speed CAN module. The system is powered by a NI PS-15 power supply.

The NI cRIO-9024 features an 800 MHz real-time processor that runs LabVIEW Real-Time for deterministic, reliable real-time control, data logging, and analysis. It contains 512 MB of DDR2 memory and 4 GB of storage for storing programs and logged data. An available high speed USB port allows the use of external USB-based flash and memory devices which could be used to expand the data storage capabilities of the DAQ system. The need for system to host computer connectivity is met by its dual Ethernet ports for programmatic communication over a network and built-in HTTP web and FTP file servers that allow for remote user interfacing. Also, the cRIO-9024 meets the

demands for ruggedness and reliability due to its operating temperature range of -20 to 55 °C and fault-tolerant file system for data-logging applications.

The NI cRIO-9111 is a reconfigurable embedded chassis with four slots that accept any cRIO I/O module, allowing for a modular design of the DAQ system which could later be expanded to include other I/O modules, such as one that would interface with certain vehicle components that use LIN instead of CAN. It employs a Xilinx Virtex-5 reconfigurable I/O FPGA core for high processing power and programmability of an integrated circuit containing programmable logic components called logic blocks that are programmed through the use of LabVIEW to create custom hardware for various embedded applications. The design of the FPGA is dependent on the I/O modules used with the chassis. The NI 9853 high speed CAN module selected for the CAN DAQ system enables the communication with CAN processors and controllers and contains 11-bit and 29-bit CAN message arbitration ID support. Thus, the FPGA program would reflect the use of the NI 9853. Finally, further ruggedness is ensured with an operating temperature range of -40 to 70 °C for the NI cRIO-9111.

While the system architecture selected ensured that all of the design constraints were met, the CAN DAQ system still lacked the ability to transfer data wirelessly, which was later decided on as an additional desired design requirement. Without wireless access capabilities, one would have to periodically connect a host computer to the on-board embedded CAN DAQ system to extract the data logged and stored within its hard drive. Thus, by adding wireless connectivity, one could simply ensure that a host

computer and the system were on the same wireless network and take advantage of the already present built-in HTTP server capability of the NI cRIO-9024 real-time controller. Originally, it was hoped that the NI 9795 C Series Wireless Sensor Network (WSN) Gateway could meet this need as it is a cRIO compatible I/O module that could easy be inserted into one of the available slots on the chassis. However, it was learned that the NI 9795 was meant as a way to wirelessly connect to measurement nodes that act as sensors for various applications and not a means to transfer data wirelessly.

Eventually, it was determined that the best solution would be to purchase a Moxa AWK-3121 industrial wireless access point and network bridge in order to incorporate wireless data streaming during actual vehicle operation. Even though this device was not developed by NI, extensive testing had been done that proved it could provide reliable wireless connectivity and long distance data streaming performance when used with NI cRIO hardware. As such, it was sold as an accessory for the NI cRIO-9024 by NI and could be readily powered by the NI PS-15 power supply already obtained. More importantly, the AWK-3121 can also be used as an IEEE 802.11 client to add wireless connectivity to any device with Ethernet capabilities and contains network security protocols to ensure that the wireless network used for the CAN DAQ system is secure and free from undesired traffic. To further ensure the network used with the system was secure and free from undesired traffic, a Verizon Jetpack 4G LTE Mobile Hotspot MiFi 4620LE was added to establish a dedicated wireless network for the CAN DAQ system.

In summary, the CAN DAQ system hardware was made up of an embedded NI cRIO-9024 real-time controller, NI 9111 reconfigurable FPGA chassis, NI 9853 high speed CAN module, Moxa AWK-3121 wireless access point, Verizon MiFi 4620LE mobile hotspot, and NI PS-15 power supply. The ruggedly designed hardware selected allows for the reliable acquisition of data from the fuel cell hybrid bus via CAN, storage of logged data within its internal hard drive, wireless extraction of data from the HTTP web server through a host computer, and development of the user interface software program with the use of NI LabVIEW. Figure 20 shows a picture of the CAN DAQ system architecture during the software development phase. Detailed specifications can be found in [57].

Figure 20: CAN Based Data Acquisition System – Software Development Phase

3.6.2 CAN DAQ Software Development

As mentioned in the section above, the software for the CAN DAQ system was developed using the NI LabVIEW visual programming language. The development of the software was completed as a two part program where the first part consisted of the FPGA program and the second is that of the real-time controller program. The FPGA program, displayed as Figure 21, serves as a means to read in any CAN frames that are being sent to the NI 9853 CAN module. Without this FPGA program, the real-time controller program would not have any way to bring in the CAN input signals to achieve the programmed control tasks. Thus, it is important to create an FPGA program that works as desired. While this can be a bit more complicated for other applications, the CAN DAQ system only requires a way to read in the CAN signals. The rest of the program simply ensures that a valid CAN frame is received and that there is not an over flow of frames that could cause the data logger to timeout or slow down, thus losing frames.

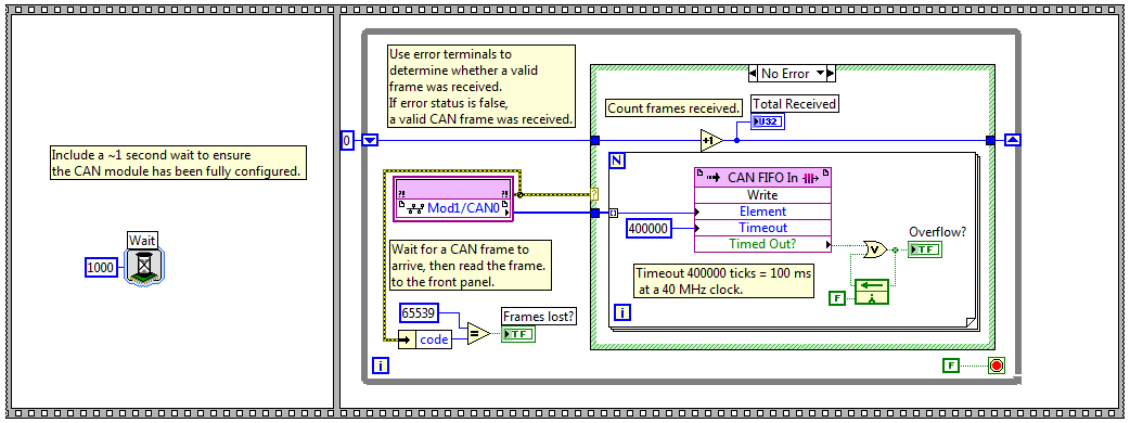

Figure 21: CAN DAQ FPGA Program

While development of the FPGA program was rather straight forward, the controller program required some work and several iterations of the block diagram to get right. The general idea behind the real-time controller software was that it would read in CAN frames from the bus' CAN vehicle network and record them to a file for future analysis of the operating performance of the fuel cell hybrid bus. Initial attempts at creating the data logger software were made with the use of the NI-CAN Driver Software which contains a CAN Frame application programming interface (API) that enables the reading of CAN frames and, according to the NI website, would simplify the creation of a data acquisition application. However, simply logging a CAN signal in the frame format read in from a vehicle network would not be of much use unless it was converted from its hexadecimal representation to engineering units. Furthermore, in order to know how exactly to convert a specific frame to engineering units, a set of rules must be associated to each message. This set of rules is defined within a CAN database file that contains scaling information for each channel within a frame. A CAN channel contains the individual data in engineering units of each signal within a frame that is associated to a specific message ID. While the NI-CAN Driver Software includes a CAN Channel API, it is not compatible with the NI 9853 CAN module. While conversion by hand is possible, it would make more sense to have the CAN DAQ system automatically convert the CAN frame before it logged the data. Thus, the CAN DAQ software had to be able to convert CAN frames to CAN channels. Fortunately, NI has developed a

collection of virtual instruments (VIs), called the Frame Channel Conversion Library, which allow for fast conversion from frames to channels.

Therefore, the development of the CAN DAQ software program was furthered by the use of this library of useful VIs, with which the CAN frames read in could be converted to their respective channels and logged to a file. The file format chosen was that of a Technical Data Management Streaming (TDMS) file because it offers high-speed streaming, a small disk footprint, LabVIEW support, and exchangeability. In addition, TDMS files are saved in a structured hierarchy where a file can contain several channel groups that in turn hold an unlimited number of channels with their very own headers. This file structure seemed ideal for logging CAN channels into separate TDMS channel arrays within a file. However, while the general idea behind the data logger was correct, in practice, simply feeding the whole CAN frame signal into the Frame to Channel VI was causing the data to not be logged properly in the TDMS file. Specifically, the CAN channels were all being saved under one TDMS channel, making the log file one long array of data that did not make any sense. Solving this problem required taking a closer look at the structure of a CAN frame, how the Frame to Channel VI converts the data, and how TDMS files are written. Through further investigation and with the use of a PEAK PCAN-USB adapter, which allowed for “dummy” CAN signals to be used during the development process, it was learned that the TDMS file required each channel to be logged separately. Considering that each CAN message can contain several channels,

each frame would have to be fed through the converter as many times as the number of channels it has so that each channel could be logged in a successive manner.

While this did allow for channels to be logged in their own separate array, at times the data would be inconsistent to that of the "dummy" signal being used. This could be remedied by ensuring that the data logging process only occurred when an actual frame was being sent in the correct order. By using the fact that a CAN frame consists of an arbitration ID, time stamp, and a data field, among other things, the Frame to Channel VI could be fed the data field without the rest of the CAN frame so that there was a reduced chance of logging random converted hexadecimal values. Moreover, through use of a Case Structure, a block within LabVIEW that allows for the creation of multiple cases with unique case identifiers, a case could be created for each CAN message where the message ID was used as the case identifier. Then, using a For Loop, the data field could be fed in as many times as needed to log each channel of a respective message case.

Though this block diagram design was able to achieve separate CAN channel logging, the program was receiving frames faster than it could log them, leading to frames being lost and the program getting stopped up due to an overflow of CAN frames. This meant that the program would have to be analyzed for any inefficiencies of operation. In addition, considering that signals within vehicle networks do not have transients as fast as the rate in which CAN frames were being read from the FPGA, it was decided to incorporate a sample and hold feature where data would only be logged

every tenth iteration of the While Loop encasing the controller program. This would additionally guarantee that an equal amount of data was logged for each CAN channel. The indicators that were being used for the sample and hold feature were then cleared out after each use of the CAN DAQ program so as to prevent the logging of old data that could cause an overflow problem.

With all of these changes and improvements, and after ensuring that all wiring inefficiencies had been removed per the help found on the NI forums, the development of the CAN DAQ software program was finally complete. The final architecture of the CAN DAQ software is shown in Figure 22. The resulting real-time controller program incorporates the program enhancements by first initializing an array of channel names and message IDs based on those found in any database file of choice, creates a TDMS file with a file name corresponding to the day of creation, and creates a queue that will be used as a means to apply the sample and hold feature. Then, the CAN frames are brought in from the FPGA and sent to a While Loop that reads in the frames and places them into the queue. Next, in a separate While Loop, the queued CAN frames are brought in and separated into arrays of message ID, time stamp, and data field. The data field array is sent to the Frame to Channel VI while the message ID is used to search through the initialized array of channel names. Finally, on every tenth iteration of the conversion loop, an array containing data for all of the channels of every message and their respective time stamps is logged to the created TDMS file.

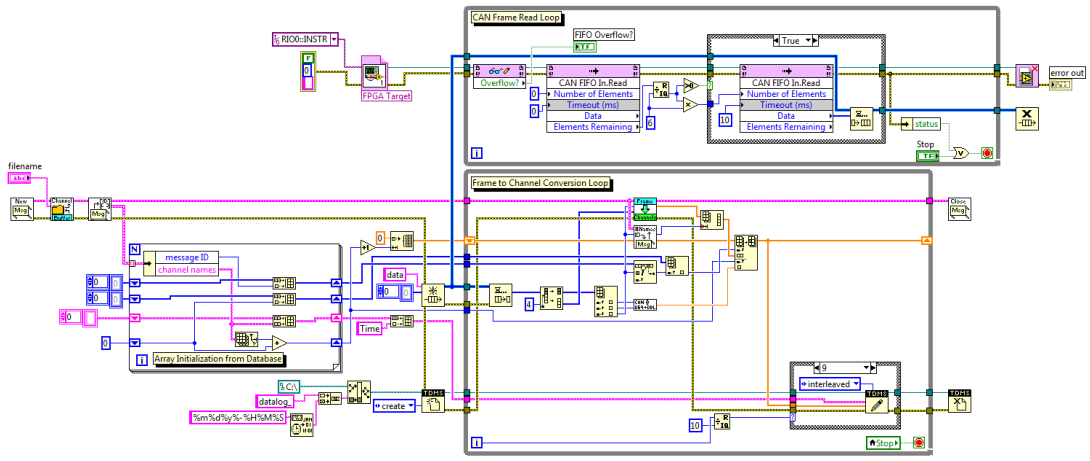

Figure 22: CAN DAQ Real-Time Controller Program

The final step in the development of the CAN DAQ system involved creating a software application of the program that would run upon start-up of the CAN DAQ hardware. When embedded onto the bus, the CAN DAQ system would be able to log data and temporarily store it in the internal storage in TDMS files with a structured format that would be easy to view and analyze. Furthermore, the system created can be used on various projects by simply changing the database file used and specifying how many channels the TDMS file should expect.

3.7 Conclusion

In the chapter, the development of a CAN based DAQ system for the real-time logging of data from the fuel cell hybrid bus CAN network was detailed and shown to be adaptable to various other projects, which was an interest of both DesignLine and OSU. The fuel cell hybrid bus simulator was then presented along with information pertaining to the proposed controller. The proposed controller was then extensively tested along

with the simulator and validated through comparison with an optimal PMP controller. Results for the HDUDDS and Manhattan drive cycles show that the proposed ARMA control algorithm is a near-optimal solution for varying driving conditions as compared to the optimal PMP control solution. In addition, the proposed ARMA controller displays a limited amount of fuel cell transients and a battery charge sustaining behavior that is desirable for the operation of the proposed fuel cell hybrid bus powertrain architecture. Therefore, the proposed ARMA controller provides an implementable control algorithm that has proven to perform well on the selected standardized drive cycles.

CHAPTER 4: Bus Hardware/Software Validation for Campus Use

4.1 Introduction

In the previous chapter, it was shown that the chosen fuel cell hybrid bus powertrain architecture and proposed ARMA control algorithm were designed to allow for desirable performance on varying driving conditions as established by the standardized HDUDDS and Manhattan drive cycles. While these results offer a valid look into the simulated performance of the fuel cell hybrid bus, they lack any real indication of how the ECO Saver IV bus would actually perform at OSU. This chapter is focused on answering this question through a detailed look into the process used to develop real-world drive cycles based on typical on-campus bus routes at OSU, an analysis of simulation results using the developed real-world drive cycles, the development and analysis of a new driver model that addresses potential problems of the current driver model, and an in depth analysis of the simulators sensitivity to changes in the vehicle architecture and design.

4.2 Development of Drive Cycles Using Real World Routes

4.2.1 Motivation for the Use of Real World Drive Cycles

Standardized drive cycles are used in industry during vehicle development to ensure that they meet emissions standards and fuel economy specifications. While it is a good practice to have a standardized set of drive cycles to keep every vehicle manufacturer accountable under the same regulations, at times they are not sufficient for vehicles designed to run under very specific driving conditions. Such is the case for the ECO Saver IV, which is meant to operate on a set of specific bus routes. Thus, it was determined that a set of real world velocity profiles would have to be developed to properly simulate and validate the performance of the bus. The advantage of using real world drive cycles is having prior knowledge of the actual expected driving conditions, such as maximum velocity, acceleration, and frequency of start-stop events, enabling one to better develop the vehicle architecture and control algorithm. Moreover, unlike standardized drive cycles, real world drive cycles contain the inherent presence of a road grade profile, which significantly affects vehicle performance results.

4.2.2 Acquisition of GPS Data for Real World Routes

A proper study of the fuel cell hybrid bus would undoubtedly require knowledge of its application, *i.e.* information on a typical route the bus would be expected to take. Thus, data had to be collected concerning the current on-campus bus routes. An attempt was made to collect such data from OSU CABS but the information provided required more detail. Therefore, it was determined that route data would have to be

gathered, which required boarding several CABS buses with different routes and using a GPS device to record the data. Initial attempts were undertaken with the use of a GARMIN USB GPS device plugged into a laptop but the data was not usable due to numerous and lengthy moments in which GPS signal was lost. A second attempt was done using a smartphone with the Google MyTracks app. The data file recorded and extracted from the MyTracks app included the latitude, longitude, altitude, bearing, accuracy, speed, and time, of which only latitude, longitude, speed, and time were used to analyze the route. Once the GPS data had been collected and visually verified to be a good representation of the respective bus routes recorded, a usable drive cycle, consisting of velocity and grade profiles, could be developed.

4.2.3 Velocity and Grade Profile Development - GPS Data Post-Processing

However, as with most data collected, the initial MyTracks data required some filtering and post-processing due to noise and outliers. In addition, as most anyone would attest to, GPS signal strength can at times be extremely variable, leading to intervals of missing data. To remedy this, the data was first interpolated to extract the missing gaps in the route with as much fidelity as possible. The resulting interpolated latitude and longitude data was then compared to a map of the route via Google Maps and found to be a satisfactory match. Then, the interpolated GPS data pertaining to the bus' speed and time on the route was used for the development of a real-world velocity trace. The resulting velocity trace contained several instances of speeds that were clearly outliers, such as data points signifying that the bus had been travelling about 60

mph in an urban area for a single data point. Thus, the time and velocity data was filtered through the use of the MATLAB functions *butter* and *filtfilt* to eliminate any outliers and smoothen the noise from the trajectory. The filtered interpolated velocity was then plotted against time, providing a final usable velocity trace.

While the recorded GPS data pertaining to the bus' speed and time on the route was successfully used for the creation of a real-world velocity trace, proper characterization of real-world drive cycles and a thorough study of the simulators performance for the intended driving conditions should include a road grade profile and information on the distance travelled. Typically, GPS data does not include road grade data and the distance provided is usually only that of the final total distance travelled rather than an instantaneous distance travelled. Thus, development of a road grade profile for a real-world drive cycle required calculating the grade from the elevation and instantaneous distance travelled, such that

$$grade = \frac{elevation}{distance}. \tag{45}$$

In order to check the validity of the less accurate GPS altitude data, it was compared to data found in The National Map, a resource of U.S. topographical information complied by the U.S. Geological Survey (USGS), in which data on elevation, known as the National Elevation Dataset (NED), is among eight primary data types. Through [58], a look-up of the elevation data within NED, with respect to the latitude and longitude data collected, was performed. As expected, the comparison showed

that the values did not coincide perfectly. Thus, the more accurate NED elevation data was taken as the data of choice to use with the road grade equation.

The road grade was then calculated using distance obtained through [56] and was found to be within an approximate ±6% grade for the on-campus bus routes. However, to ensure that this road grade data was in fact correct, the distance was again calculated using the available data by taking the integral of the vehicle speed.

$$x_{dist} = \int_0^{t_f} v_{veh} dt \tag{46}$$

However, results from the application of Equation (46) to the road grade equation contained road grade percentages that were clearly incorrect. Seeing as how both road grade calculations used the same elevation data, the difference in maximum road grade percentages had to be due to differences in the distance data. Furthermore, since the recorded velocity data used in the calculation of distance was assumed to be correct, the problem had to lie in the initial assumption that distance should be obtained through the integration of velocity.

A closer look at the problem led to the understanding that the road grade equation requires the use of the curved distance between two points in order to be truly accurate, whereas the integration of velocity yields the linear distance between two points. While it is common knowledge that the shortest linear path between two points is a straight line, this rule does not apply when considering a curved space or path, in which case the shortest path between two points is defined as a geodesic.

Originally, being derived from geodesy, or the science of measuring the size and shape of the Earth, the term geodesic was used in describing the shortest route between two points on the Earth's surface. While Pythagoras first postulated that the Earth was a spherical body, we now know that the Earth is more closely approximated by an ellipsoid of revolution, also known as an oblate spheroid. Thus, the study of geodesics on an ellipsoid provides a more accurate path between two points on Earth's surface. However, this more accurate and complex approximation of the shape of the Earth led to various problems. Fortunately, through the work of many of the great mathematicians of the past, the problems in geodesy can be reduced to two main cases, the direct and inverse problem. The direct problem involves finding the position along a geodesic after travelling a certain distance, given a starting point and an initial heading. On the other hand, the inverse problem involves finding the geodesic of two given points on an ellipsoid.

While many have developed algorithms that would solve the two geodesic problems, most approaches require large amounts of computing resources and time. However, work done by Thaddeus Vincenty in 1957 led to the formulation of what are now known as the Direct and Inverse Vincenty Formulae. Thus, the curved distance needed for the correct calculation of the road grade can be determined through solving the inverse geodesic problem with the use of the Vincenty Inverse Formulas:

$$\sin^2\sigma = (\cos U_2 \sin\lambda)^2 + (\cos U_1 \sin U_2 - \sin U_1 \cos U_2 \cos\lambda)^2 \quad (47)$$

$$\cos\sigma = \sin U_1 \sin U_2 + \cos U_1 \cos U_2 \cos\lambda \quad (48)$$

$$\tan\sigma = \frac{\sin\sigma}{\cos\sigma} \quad (49)$$

$$\sin\alpha = \cos U_1 \cos U_2 \frac{\sin\lambda}{\sin\sigma} \quad (50)$$

$$\cos 2\sigma_m = \cos\sigma - 2\sin U_1 \frac{\sin U_2}{\cos^2\alpha} \quad (51)$$

$$C = \frac{f}{16}\cos^2\alpha\,[4 + f(4 - 3\cos^2\alpha)] \quad (52)$$

$$\lambda = L + (1 - C)f \sin\alpha\,\{\sigma + C\sin\sigma\,[\cos 2\sigma_m + C\cos\sigma\,(-1 + 2\cos^2 2\sigma_m)]\} \quad (53)$$

where σ is the angular distance between two chosen points on an auxiliary sphere, U_1 and U_2 are reduced latitudes on the sphere, λ is the difference in longitude on an auxiliary sphere, α is the azimuth of the geodesic at the equator, σ_m is the angular distance on an auxiliary sphere from the equator to the midpoint of the line formed between the two chosen points, C is the geodesic curve between the two chosen points, L is the difference in longitude between the two chosen points, and f is the flattening of an ellipsoid, determined from

$$f = \frac{(a - b)}{a} \quad (54)$$

where a and b are the major and minor semi-axes of the ellipsoid, respectively. Furthermore, the reduced latitudes are obtained from the geodetic latitudes, ϕ_1 and ϕ_2, of the two chosen points, such that

$$U_1 = \tan^{-1}[(1-f)\tan\phi_1] \tag{55}$$

$$U_2 = \tan^{-1}[(1-f)\tan\phi_2]. \tag{56}$$

Having set λ to an initial value of L, the goal is to evaluate Equations (47) to (53) through an iterative process until the change in λ is negligible, at which point an evaluation of the change in angular distance between the chosen points is carried out, such that

$$u^2 = \cos^2\alpha\frac{(a^2-b^2)}{b^2} \tag{57}$$

$$A = 1 + \frac{u^2}{16384}\{256 + u^2[-768 + u^2(320 - 175u^2)]\} \tag{58}$$

$$B = \frac{u^2}{1024}\{256 + u^2[-128 + u^2(74 - 47u^2)]\} \tag{59}$$

$$\begin{aligned}\Delta\sigma = B\sin\sigma\Big\{&\cos 2\sigma_m \\ &+ \frac{1}{4}B\Big[\cos\sigma\,(-1 + 2\cos^2 2\sigma_m) \\ &- \frac{1}{6}B\cos 2\sigma_m\,(-3 + 4\sin^2\sigma)(-3 + 4\cos^2 2\sigma_m)\Big]\Big\}\end{aligned} \tag{60}$$

where A and B are the two chosen points of the ellipsoid, as shown in Figure 23 .

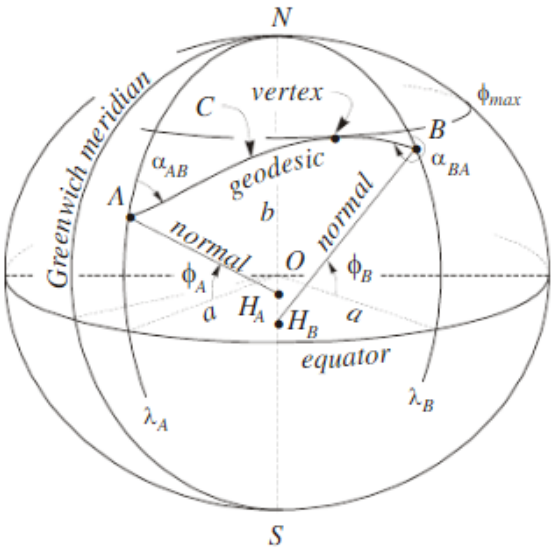

Figure 23: Geodesic Curve on an Ellipsoid

Finally, the length of the geodesic between the chosen points is determined from

$$s = bA(\sigma - \Delta\sigma). \tag{61}$$

Though not needed for the application at hand, the azimuths of the geodesic, α_1 and α_2, can be determined from

$$\tan \alpha_1 = \frac{\cos U_2 \sin \lambda}{\cos U_1 \sin U_2 - \sin U_1 \cos U_2 \cos \lambda} \tag{62}$$

$$\tan \alpha_2 = \frac{\cos U_1 \sin \lambda}{-\sin U_1 \cos U_2 + \cos U_1 \sin U_2 \cos \lambda} \tag{63}$$

Thus, through use of Vincenty's Inverse Formulas, the ellipsoidal distance between every two GPS points was determined and then plugged into the grade equation along with the elevation data previously obtained. While the calculated total distance data was now in agreement with the website distance, the road grade

calculated now contained values upwards of 20%. Fortunately, it was found that this could be remedied with data filtering, so the use of statistical analysis was incorporated to eliminate any outliers in the grade data. Specifically, the standard score, or z-score,

$$z = \frac{x_g - \mu}{\sigma} \tag{64}$$

of each raw road grade value x_g was calculated, where μ is the mean and σ is the standard deviation of the entire road grade data set. The z-score denotes the number of standard deviations that a specific data point is away from the mean, where a positive z-score means the data point is above the average and a negative z-score means it is below the average. The criterion used to remove outliers was based on the statistical three-sigma rule that states that three standard deviations will theoretical cover 99.73% of a given distribution. Thus, any data points with z-scores higher than +3 and lower than -3 were removed. What remained was a road grade profile that had a maximum and minimum grade percentage with a relatively realistic magnitude as expected. The developed velocity and road grade profiles for a specific OSU CABS bus route would then make up a complete real-world drive cycle.

4.2.4 Real-World Drive Cycles

While GPS data had been recorded for several CABS bus routes, the real-world drive cycles developed were those of the OSU on-campus bus routes, Campus Loop – North (CLN) and Campus Central (CC). The methodology detailed in the previous section was used in making the velocity and grade profiles of these real-world drive cycles.

Figure 24 and Figure 25 contain plots of the final form of the real world CLN drive cycle developed for use with the fuel cell hybrid bus simulator including its respective road grade profile. While at first glance the road grade profile may seem unrealistic as it contains many seemingly quick transitions, Figure 26, which displays a close up look at the CLN grade profile with respect to distance, proves that these profiles are in fact reasonable considering the distance that is covered.

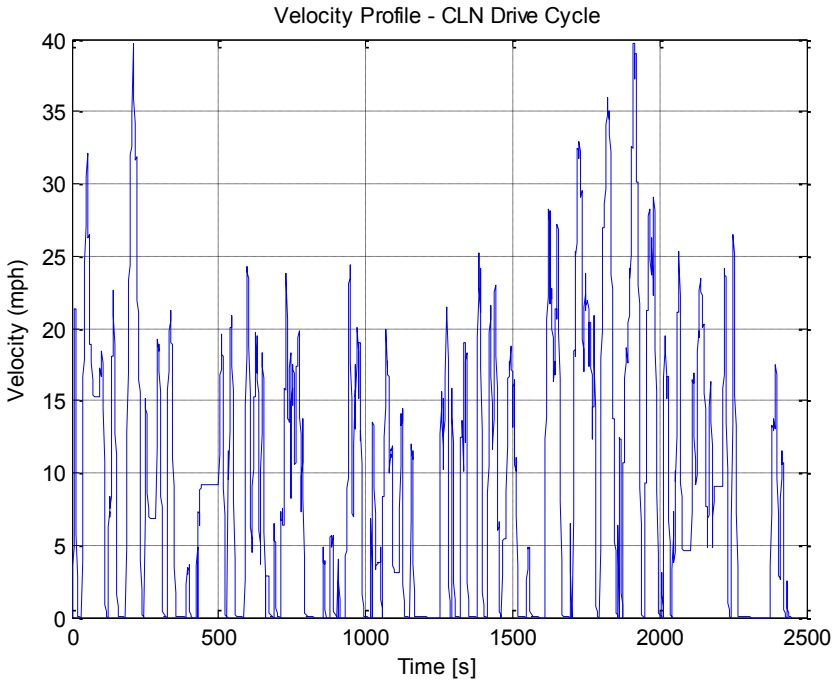

Figure 24: Velocity Profile Developed from CABS CLN Bus Route

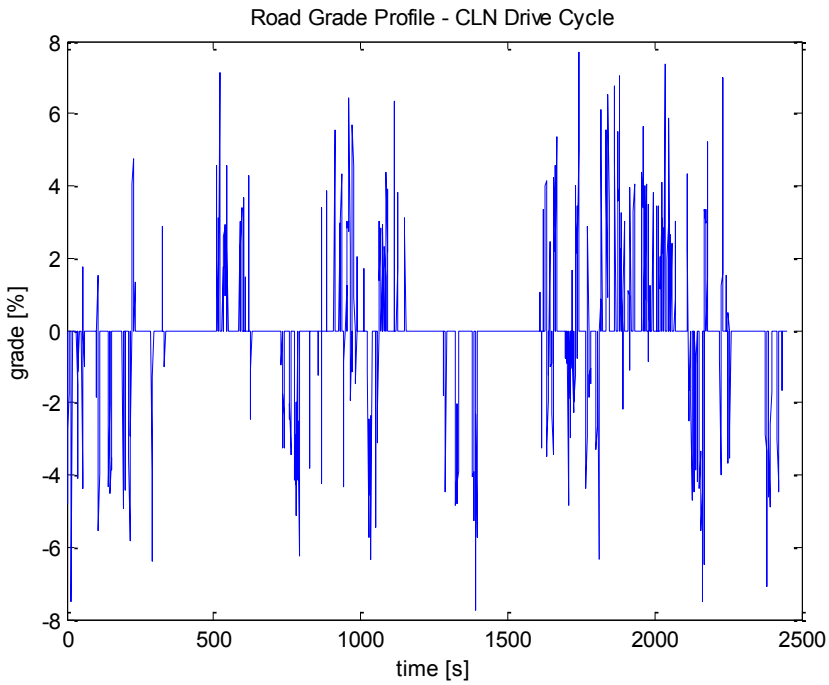

Figure 25: Road Grade Profile Developed from CABS CLN Bus Route

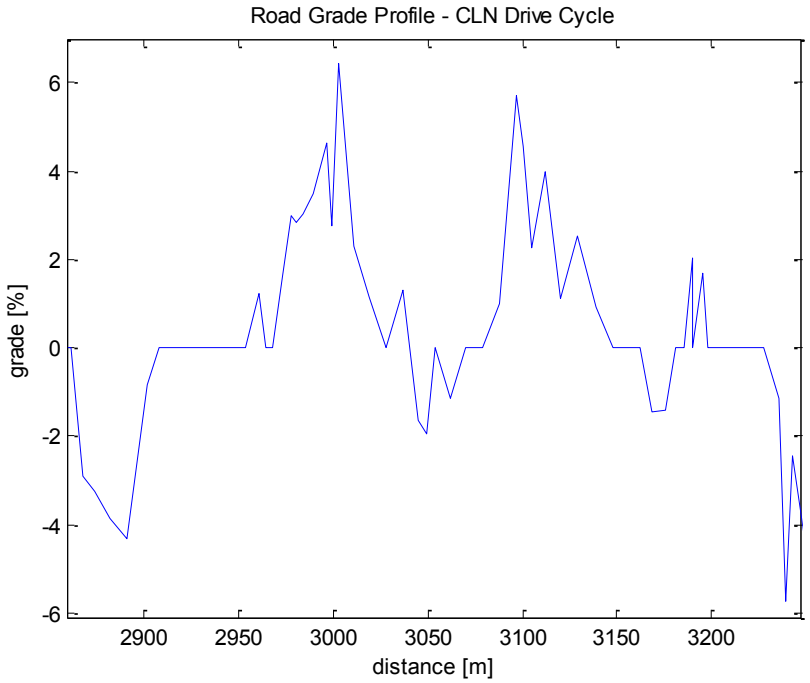

Figure 26: Close-up View of Road Grade vs. Distance - CLN Drive Cycle

The velocity and road grade profiles of the developed real-world CC drive cycle are shown as Figure 27 and Figure 28. Again, a close up look at the corresponding road grade profile, as given by Figure 29, reveals that it is in fact a realistic representation of the grade with respect to distance traveled. These drive cycles were used to determine the performance of the simulator with real-world route characteristics prior to integration into the CABS bus fleet for the demonstration.

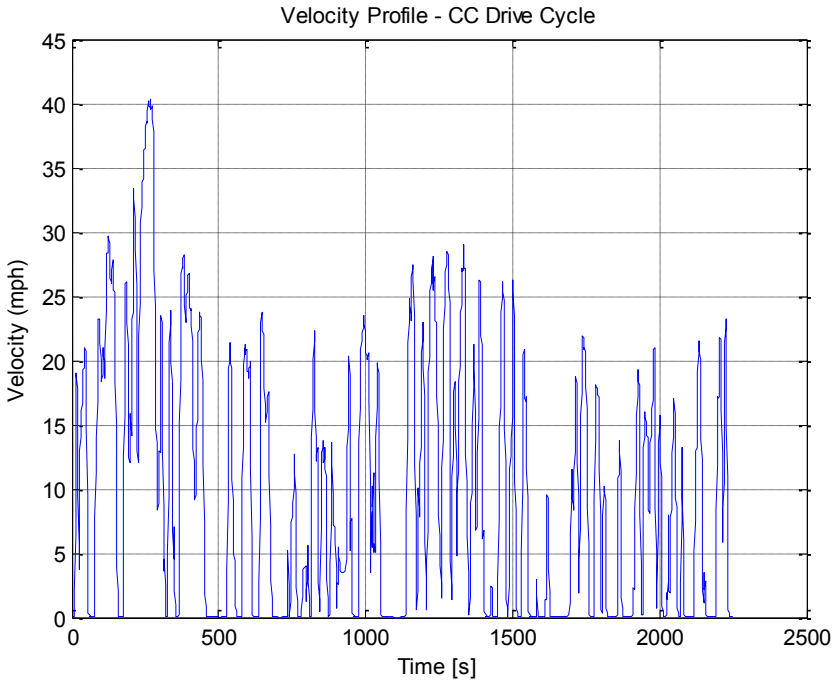

Figure 27: Velocity Profile Developed from CABS CC Bus Route

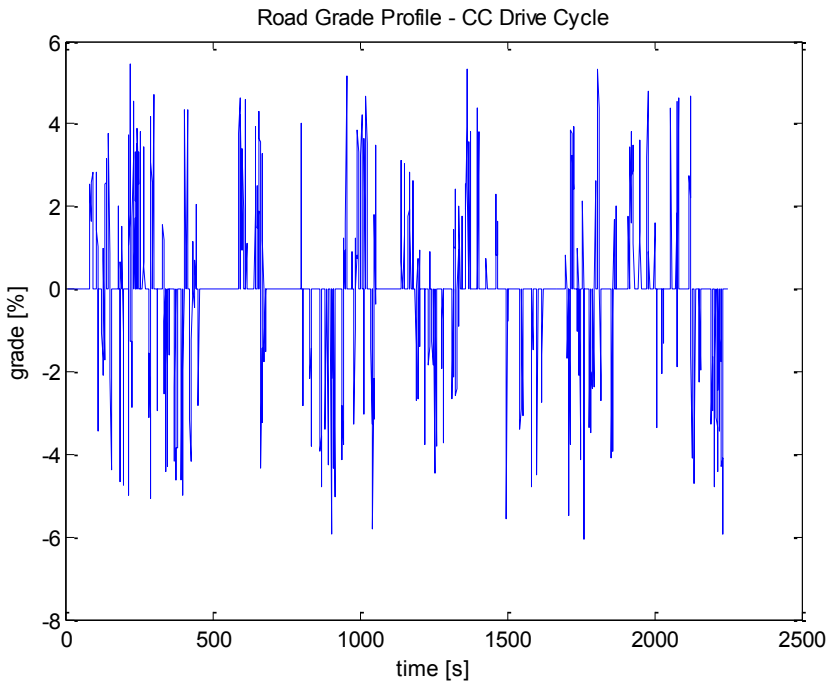

Figure 28: Road Grade Profile Developed from CABS CC Bus Route

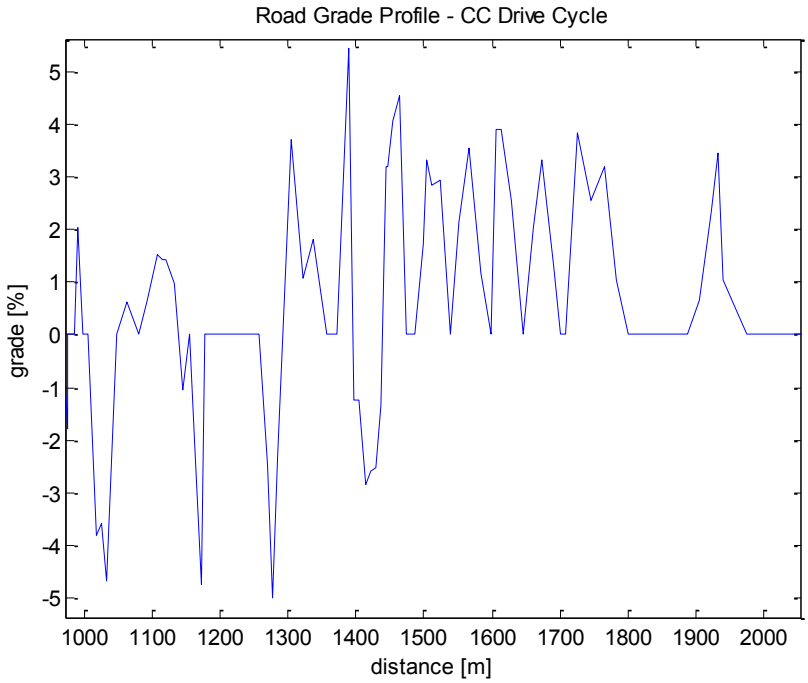

Figure 29: Close-up View of Road Grade vs. Distance - CC Drive Cycle

In order to properly determine if the vehicle architecture and the proposed ARMA control algorithm had been well defined, an analysis of the fuel cell hybrid bus' performance during operation on these real-world drive cycles was conducted; similar to what was done with the validation of the control algorithm with standardized drive cycles. Thus, the following section details the simulation results with the use of the OSU CABS CLN and CC real-world drive cycles as well as a comparison of the performance of the simulator and controllers when operated with the real-world drive cycles versus the HDUDDS and Manhattan standardized drive cycles. In addition, potential problems with the use of the real-world drive cycles are discussed.

4.2.4.1 Potential Mismatch between Velocity and Grade Profiles

While the use of real-world drive cycles is encouraged to properly simulate the expected driving conditions of the bus, their use brought up questions of the validity of the driver model. The driver model, as explained in Chapter 3, tracks the velocity error between the desired and actual calculated velocity before computing the accelerator and brake pedal positions through use of a PID controller. While this method has been proven to work successfully for the standardized drive cycles, it was concluded that there could be a potential problem caused by the fact that the real-world drive cycles were based on actual on-campus bus routes. Specifically, it was thought that the bus might get "behind" in terms of the schedule the bus would be expected to keep when travelling from stop to stop due to a mismatch between the velocity profile and road grade profile that would become more and more apparent as time went by. If this were

to happen, the controller would essentially be providing the incorrect power demands for the given time and location on the drive cycle. The following section offers a solution that would ensure the potential problem is avoided.

4.3 Development of Distance Based Driver Model

4.3.1 Motivation for the Need of a Distance Based Driver Model

Traditionally, a driver model for an energy management model simply applies a PID controller to attempt to match the simulated vehicle velocity with that of a chosen drive cycle. The simulator had just such a driver model. This driver model could sufficiently match up the simulated vehicle velocity for several drive cycles. However, it was agreed upon that simply tracking velocity was not appropriate for the real world drive cycles. As previously mentioned, this conclusion was centered on the fact that these drive cycles were based on an actual bus route in which the bus would have to be a certain distance into the route at any given time such that the bus could keep up with the routes bus schedule. The problem was the eventual mismatch between the velocity and road grade profiles. Thus, the distance traveled by the bus would have to be taken into consideration, which led to the need of a distance based driver model. Such a model would have to match the linear distance traveled according to the real world drive cycles developed. The respective linear distance profiles for the CLN and CC drive cycles are given as Figure 30 and Figure 31.

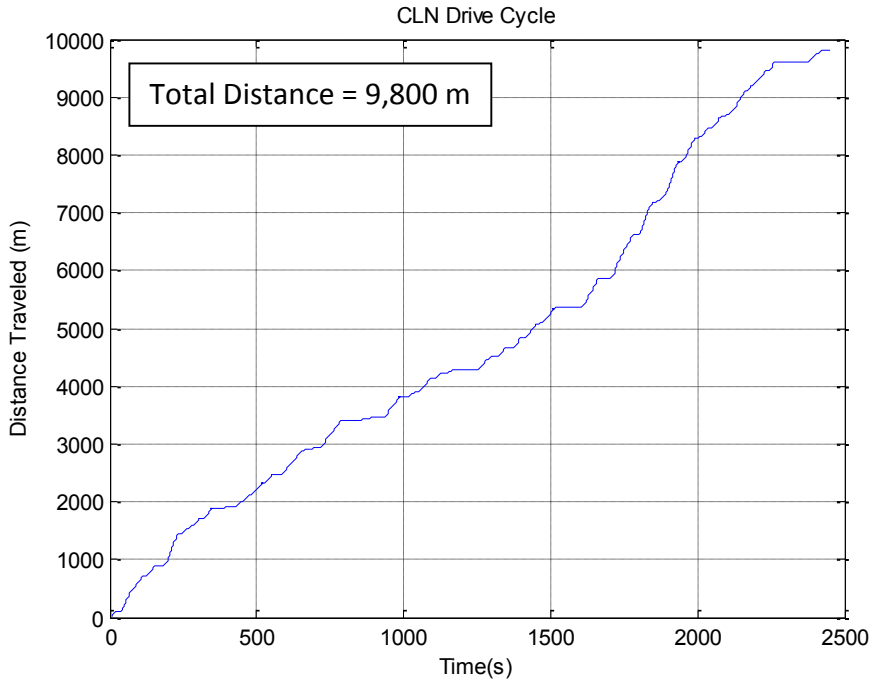

Figure 30: OSU CLN Distance Profile

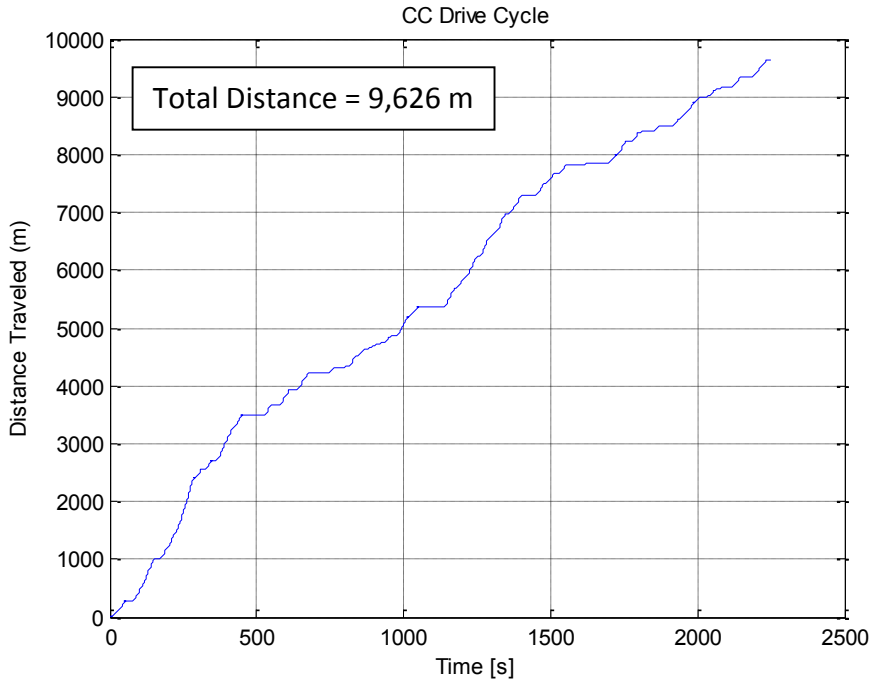

Figure 31: OSU CC Distance Profile

4.3.2 Heuristic Design Method for Distance Based Driver Model

Though not as efficient, a heuristic design method was used to develop the distance based driver model. This was mostly due to the lack of previous work in undertaking such a task and thus could not have any information or initial idea of how to properly track distance in a way that would ensure that the vehicle met its power requirements as well as kept up with the distance profile. As an initial attempt, the vehicle velocity was simply integrated and the resulting linear distance was passed through the PID controller, after which the PID gains were retuned. A single set of PID gains for all the drive cycles was desired so as to eliminate the need to change gains every time the drive cycle was changed. While this method worked for some drive cycles, others required retuning the PID gains with varying success in terms of reducing the distance error to a satisfactory level. As the tracking of distance worked at times, it was thought that the solution would be to try to track both distance and velocity.

The next attempt involved incorporating a sort of feedback controller in which velocity was tracked with a PID controller as before. The resulting signal was then integrated so as to obtain distance and then sent through another PID controller. The final resulting signal was used to determine pedal positions, alpha and beta. This method was definitely more complex since it required tuning two PID controllers. An algorithm was developed in MATLAB to determine the optimal gain values for the PID controllers but the results proved to be unsatisfactory. At times, the distance profile

would be matched up very well but the velocity profile would hardly match, and vice versa.

Another problem identified was the presence of extreme transients in the accelerator and brake pedal position signals. This caused transients to be present in the power demands calculated and ultimately caused the velocity and distance traces to be off by large margins at times. Although more complex, this feedback like controller resulted in even less desirable performance than the previous attempt.

4.3.3 Chosen Model Architecture

Finally, after several attempts at fine tuning the previously mentioned driver models, a simple yet elegant solution was used. The solution came in the form of a weighted function as shown by the red circle in Figure 32. First, the vehicle velocity was integrated to obtain linear distance traveled by the bus. Then, the velocity error and distance error were calculated and passed through the weighted function such that

$$(v_{desired} - v_{actual})w_{vel} + \left(\int v_{desired}dt - \int v_{actual}dt\right)w_{dist} = E_{tot} \tag{65}$$

where $v_{desired}$ and v_{actual} are the respective desired and actual velocity whose difference and integrated difference are the respective velocity error the distance error, w_{vel} and w_{dist} are the respective velocity and distance weights, and E_{tot} is the total error signal sent to the PID controller to determine the pedal positions.

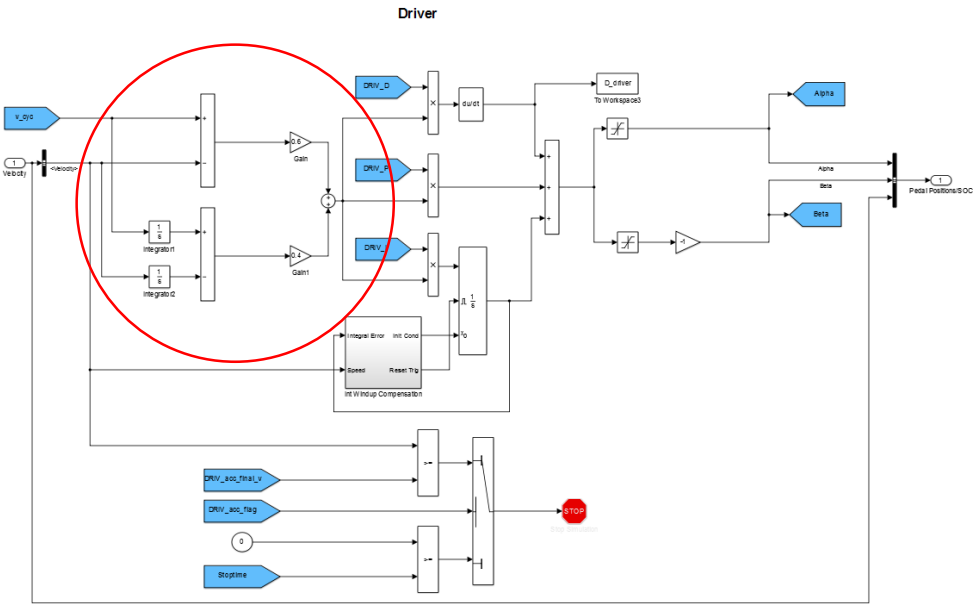

Figure 32: Velocity and Distance Weighted Function Driver Model

The PID gains were tuned along with determining the best weights to apply to velocity and distance. It was determined that best results could be obtained by giving velocity a slightly higher weight than distance. This weighted function approach for tracking distance and velocity will simply be referred to as the distance based driver model whereas the driver model initially obtained will be referred to as the original driver model. Table 11 contains the values of the newly tuned PID gains and the weighted values used in the distance based driver model.

Table 11: Parameter Values Used in the Distance Based Driver Model

Parameter	Value
K_P	6.5
K_I	2
K_D	0
w_{vel}	0.6
w_{dist}	0.4

4.4 Performance on Real-World Drive Cycles

4.4.1 Real-World Drive Cycle Results

In order to truly gain an understanding of how the bus would work at OSU, the use of actual driving conditions at OSU had to be taken into consideration. To this end, the simulator was run on the developed real-world drive cycles. Specifically, it was of interest to know if the bus' chosen powertrain architecture would be able to meet the specific driving conditions of the bus routes considering that a non-zero grade profile was now being used. Table 12 shows that the simulated vehicle completes the OSU CLN and CC drive cycles with velocity errors that met the 5% performance metric threshold. For both the CLN and CC drive cycles the baseline results indicate that the distance error was very small. In fact, the total distance traveled at the end of the cycle matched up with the desired total distances of the CLN and CC drive cycles to the order of three significant digits.

Table 12: Velocity RMS Error - OSU Drive Cycles

	Velocity RMS Error (%)	Distance RMS Error (%)	Distance Traveled
CLN – ARMA MIL	1.27	2.00E-3	9.80E+3
CLN – PMP MIL	1.59	2.60E-3	9.80E+3
CC – ARMA MIL	1.05	1.70E-3	9.63E+3
CC – PMP MIL	1.01	1.60E-3	9.63E+3

It should be noted that, as with the standardized drive cycles, the co-state variable values for the OSU CLN and CC drive cycles were recalculated for use within the optimal PMP controller. Their values, along with the associated approximate initial fuel cell power, are found in Table 13.

Table 13: Co-State Variable and Initial Fuel Cell Power Values for the OSU CLN and CC Drive Cycles

	λ_o Values	Approximate Initial Fuel Cell Power
CLN	1991440	29 kW
CC	2019560	34 kW

A look at the battery SOC trajectories for both the CLN and CC drive cycles, Figure 33, shows that, under both the ARMA and PMP controllers, the bus simulator is able to maintain the charge sustaining behavior desired for the battery.

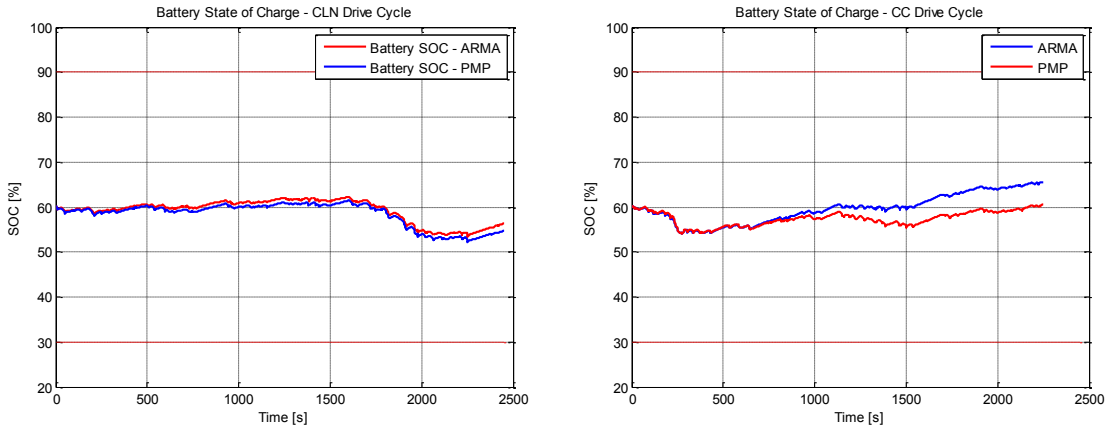

Figure 33: Comparison of ARMA and PMP Battery SOC - OSU CLN and CC Drive Cycles

As for the performance of the fuel cell, Figure 34 shows that the fuel cell power outputs of the ARMA and PMP controller for the OSU CLN drive cycle are comparable to what was seen for the standardized cycles.

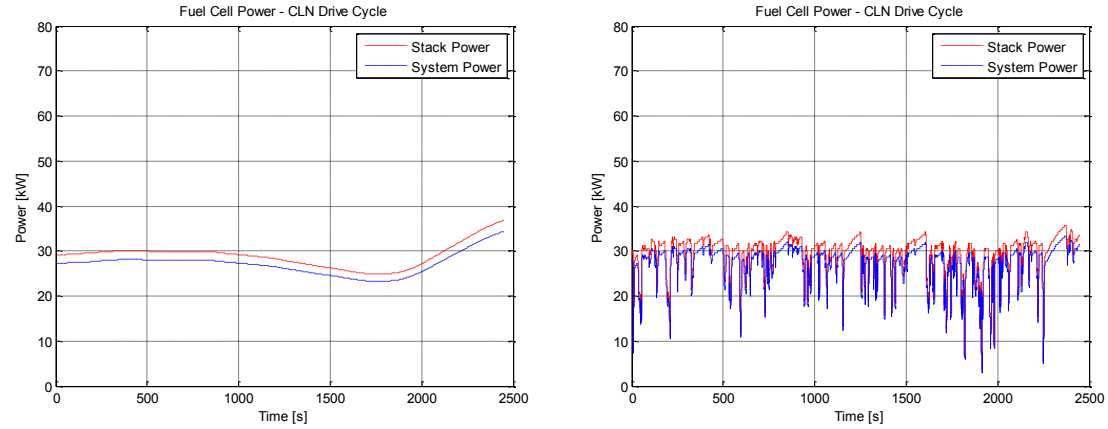

Figure 34: Comparison of ARMA and PMP Fuel Cell Power - OSU CLN Drive Cycle

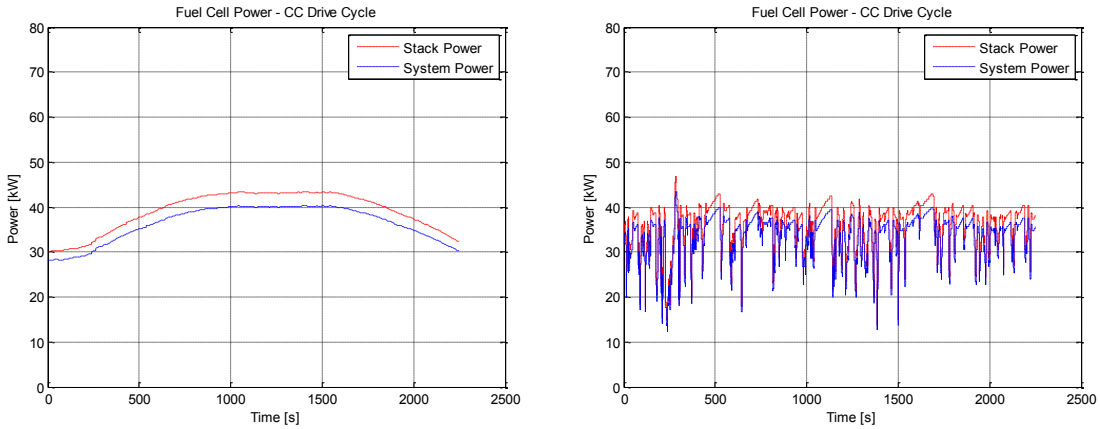

Figure 35: Comparison of ARMA and PMP Fuel Cell Power - OSU CC Drive Cycle

The same is seen for the fuel cell power of the OSU CC drive cycle, shown as Figure 35. Based on these fuel cell and SOC results, it could be concluded that the power outputs are sufficient to meet the power requirements of the on-campus driving conditions. It should be noted that, as before, the initial fuel cell power has been modified to match what is given by the PMP controller for both the CLN and CC drive cycle. Neglecting to do so would have tremendous effects on the fuel consumption and fuel economy obtained for the operation of the simulator on the real world drive cycles.

Having taken the necessary precautions, Table 14, shows that for the real-world drive cycles, as with the standardized drive cycles, the bus is able to achieve a mileage that is about two times that of a conventional diesel bus. As expected, the optimal PMP results provide a higher fuel economy than that of the ARMA controller. However, the difference is very slight as the ARMA controller gives a near optimal solution. Again, this is dependent on setting the initial fuel cell power to that which is determined by the optimal PMP controller.

Table 14: OSU Drive Cycles - Fuel Consumption and Diesel Equivalent Fuel Economy

	Fuel Consumption (kg)	Diesel Equivalent Mileage (mpg)
CLN – ARMA MIL	0.910	7.81
CLN – PMP MIL	0.920	7.85
CC – ARMA MIL	1.05	7.92
CC – PMP MIL	1.03	8.58

The results above confirm that the chosen bus powertrain architecture is more than suitable to meet the power demands indicative of the real-world driving conditions as provided by the use of the OSU CLN and CC real-world drive cycles.

A sensitivity analysis of the co-state variable of the optimal PMP solution and an analysis of the sensitivity of the near-optimal solution provided by the developed ARMA controller to the initial fuel cell power selected will be presented in the following section along with other relevant performance analyses. The complete validation process involved the correct selection of the co-state variable and initial fuel cell power at each successive step. Changes were only made where necessary.

4.4.2 Performance Analysis

4.4.2.1 Weight Sensitivity Analysis

Fuel economy and vehicle performance is greatly dependent on the weight of a vehicle. In the case of a transit bus, the total weight during operation varies due to the number of passengers riding the bus. A weight sensitivity analysis was performed to

understand how the fuel cell hybrid bus would perform in terms of the fuel economy, consumption, and SOC deviation. An analysis was performed for both the ARMA and PMP controllers for the OSU CC drive cycle. In the case of the optimal PMP controller, the co-state variable was tuned during each simulation as the weight of the bus was increased due to an increase of passengers on-board. Similarly, the initial fuel cell power was changed as needed for each weight case to better approximate the optimal solution. An average passenger weight of 185 lbs. was used in the analysis. The results for the optimal PMP controller, shown in Table 15, indicates a decrease in fuel economy as the number of passengers increased.

Table 15: Weight Sensitivity Analysis on OSU CC Drive Cycle - PMP

Number of Passengers	Fuel Cell Bus Weight (lbs.)	Co-State Variable	SOC Deviation	Hydrogen Fuel Consumption (kg)	Diesel Equivalent Fuel Economy (mpge)	Passenger Diesel Equivalent Fuel Economy (mpge)
0	35000	2019560	-0.923	1.03	8.58	8.58
7	36295	2025560	-0.731	1.07	8.28	58.0
14	37590	2031560	-0.674	1.10	8.01	112
21	38885	2037560	-0.815	1.14	7.76	163
28	40180	2043560	-0.950	1.18	7.53	211
35	41475	2050560	-0.829	1.22	7.27	254
41	42770	2056560	-0.795	1.26	7.06	290
49	44065	2063560	-0.873	1.30	6.83	335

Due to the change in co-state variable for each case, the SOC deviation was kept to a minimum. It should be noted that the passenger fuel economy, which takes into account the total fuel economy associated with transporting a certain number of

people, increased as the number of passengers increased. The same trend was seen with the ARMA controller with results shown in Table 16. However, a decrease in diesel equivalent fuel economy of nearly 1 mpg from that of the optimal solution was obtained. The SOC deviation was also seen to be higher than with the optimal PMP controller but was still within the performance metric range. Overall, the results show that the difference between running the bus with no passengers and running it with more passengers than available seats causes a decrease in fuel economy of nearly 2 mpg. However, the more passengers are on-board, the higher the total passenger fuel economy gets. At full capacity, the bus approaches 300 passenger mpg, making the increase in weight a benefit rather than a hindrance for the fuel cell hybrid transit bus.

Table 16: Weight Sensitivity Analysis on OSU CC Drive Cycle - ARMA

Number of Passengers	Fuel Cell Bus Weight (lbs.)	Approx. Initial Fuel Cell Power	SOC Deviation	Hydrogen Fuel Consumption (kg)	Diesel Equivalent Fuel Economy (mpge)	Passenger Diesel Equivalent Fuel Economy (mpge)
0	35000	34	5.751	1.14	7.68	7.68
7	36295	35	5.944	1.17	7.45	52.1
14	37590	37	6.036	1.21	7.16	100
21	38885	38	6.414	1.29	6.86	144
28	40180	39	6.743	1.34	6.64	186
35	41475	40	7.093	1.39	6.37	223
41	42770	41	7.463	1.43	6.19	254
49	44065	42	7.947	1.49	5.99	293

4.4.2.2 Co-State Variable and Initial Fuel Cell Power Sensitivity Analysis

As has been seen throughout the MIL simulations, the co-state variable used within the PMP algorithm for ensuring proper charge sustainability must be tuned for each drive cycle. Changes to the powertrain architecture and the use of a grade profile have required the re-tuning of the co-state variable. To determine how sensitive the PMP controller is to changes in the co-state variable, a sensitivity analysis was performed in which the co-state variable used with the PMP controller for the OSU CC drive cycle was varied. Figure 36 shows that the SOC deviates by nearly 5% for every change of 20,000 of the co-state variable with zero deviation occurring at a value of about 2020000 for the OSU CC drive cycle.

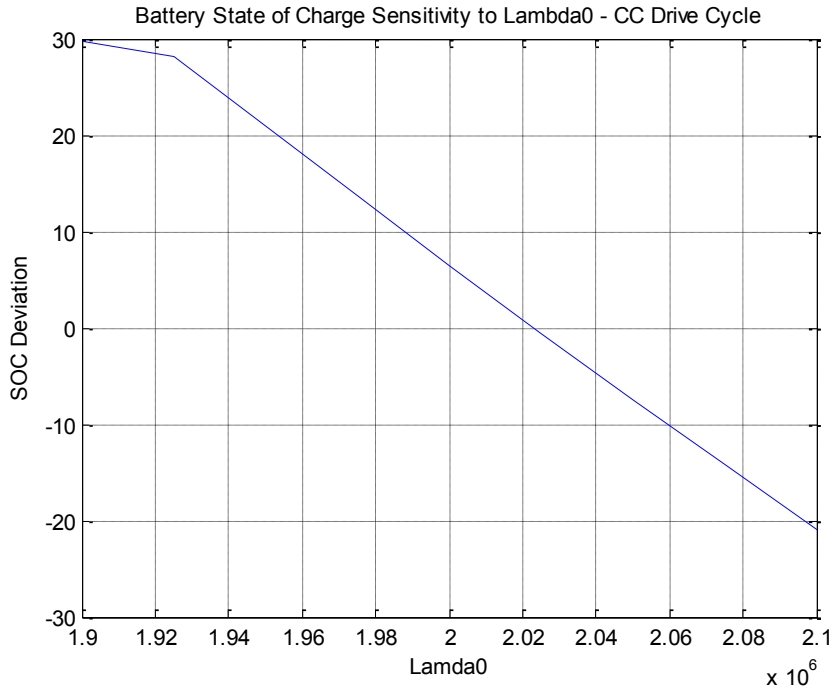

Figure 36: Co-State Variable Sensitivity Analysis - OSU CC Drive Cycle

The initial fuel cell power values corresponding to the co-state variable values used for the above analysis were used to determine how sensitive the resulting fuel economy was to changes in the initial fuel cell power used with the ARMA controller. The analysis was performed on the OSU CC drive cycle. As indicated by Figure 37, the diesel equivalent fuel economy decreased as the initial fuel cell power increased.

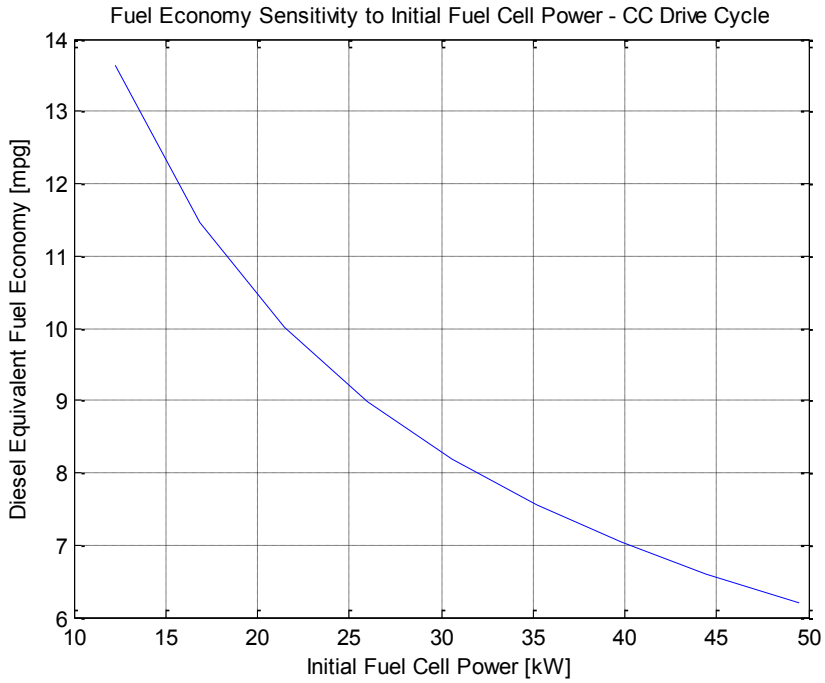

Figure 37: Initial Fuel Cell Power Sensitivity Analysis - ARMA

4.5 Conclusion

A set of real-world drive cycles were developed from GPS data of two OSU CABS bus routes. These drive cycles contained a velocity and road grade profile that were representative of the on-campus driving conditions. A distance based driver model was then developed to ensure proper synchronization between the velocity and road grade

profiles through the tracking of distance. However, the fuel cell hybrid bus was shown to be lacking in torque output by the current design of the powertrain architecture, which led to incorrect tracking of the velocity trace set by the real-world OSU drive cycles. A gain was used on the torque output to ensure that the driving conditions could be met and thus allow for validation of the control algorithm for on-campus use.

CHAPTER 5: SIL/HIL Preparation

5.1 Introduction

The work presented thus far has been that of simulation results obtained from a MIL simulator. While MIL simulations might suffice for certain applications, it was desired to further validate the Fuel Cell Hybrid Bus Simulator through the use of SIL and HIL simulations in order to achieve two overall objectives. (1) Develop the ARMA control algorithm into potentially usable control software through the automatic generation of compiled software code. (2) Determine how well the simulator and control algorithm would perform on real-time hardware. Completion of these objectives would then provide a better understanding of the way in which the proposed control algorithm would perform on the actual demonstration bus if it were used as part of the on-board vehicle control. The following is a detailed account of the steps taken to achieve the aforementioned objectives during the conversion of the MIL simulator to SIL and HIL, where special attention is given to the proper model implementation

techniques for SIL and HIL simulations as learned throughout the process and from the literature.

5.2 Proper Model Implementation Techniques for SIL and HIL Simulations

In terms of model function, a vehicle model is composed of two main components, the controller model and the plant model. The plant model consists of the dynamics associated with the fuel cell hybrid bus as a system, *i.e.,* the driver model, powertrain model, and vehicle dynamics model. The controller model consists of the software used to implement the control strategy. MIL simulations are performed with both the controller and plant model in the same virtual environment and operate using the same software language. SIL and HIL, on the other hand, involve completely separating the controller and plant model, albeit in different manners, while still allowing them to communicate with each other. While SIL and HIL simulation can be run without ensuring that the controller and plant model are their own separate entities, usually, the results obtained are not guaranteed to be correct.

The fuel cell hybrid bus simulator model was not properly configured for SIL and HIL simulations and thus required restructuring certain aspects of the model architecture. The first step in restructuring the model required separating the simulator into the controller and plant models to ensure proper implementation in the SIL and HIL simulators. Since the simulator was initially constructed in a modular manner, this process was thought to be as simple as just moving the driver model, powertrain model, and vehicle dynamics model into a plant model block and rewiring as necessary to

communicate with the controller. The controller model was then coded into a C-language code using MATLAB's automatic code generator. For the SIL model, the control software was used in the same SIMULINK virtual environment as the plant model, whereas, for the HIL model, the compiled controller software was transferred to a microcontroller. Preliminary results from the SIL simulator indicated the existence of lag in the signals involved in the calculation of the power demands. As described in Chapter 3, both controllers use the calculated pedal positions and SOC of the battery as feedback from the plant to generate the battery and fuel cell power demands.

However, it was found that there were more signals passing back and forth between the plant and the controller. Specifically, for the ARMA controller the signals for the force at the wheels, force from the electric motor, and auxiliary power were being sent from the plant model to the controller. These were needed by the controller to accomplish its calculation of the power demands. These signals were not measurable quantities but they could be estimated in the controller based on the measureable vehicle speed parameter. The same is seen with the PMP controller only with the addition of other signals that are used in the determination of the battery equivalent fuel consumption. These signals include the open circuit voltage, resistance, and derivative of SOC.

Similarly, the signals for total brake torque, total brake power, rear brake power, and maximum electric motor braking power were being computed by the controller and sent to the plant. Although these signals were not translated into a control effort,

because they were not controlling any actuators, they had a direct effect on the dynamics of the plant. These vehicle dynamics parameters were originally being estimated in the controller because they were dependent on the brake pedal position as well as the force at the wheels and force from the electric motor signals. Figure 38 provides a schematic view of the problem.

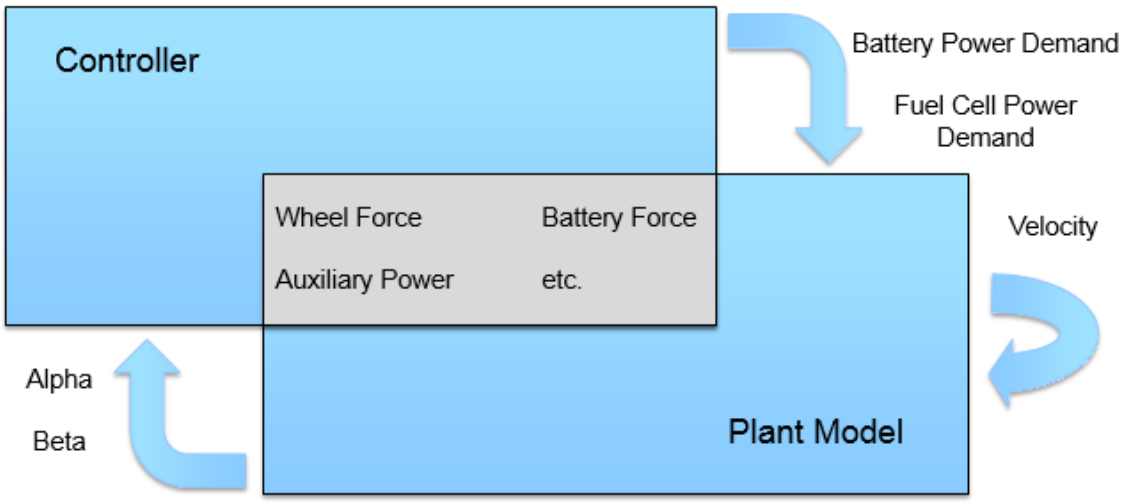

Figure 38: Schematic of Problem with Simulator Structure

As mentioned in [15], HIL testing requires the architecture of the model being used to be properly configured into independent plant and controller models. In view of this, the problem with the structure of the model for the controller and the plant was that the interconnected signals identified above were being calculated on different physical components of the HIL system and communicated with one another through a CAN Network. Due to the inherent delay and asynchronous behavior of the CAN, the dynamics of the plant was seen to be erratic at times. For example, the battery SOC was

not behaving in the desired charge sustaining manner. Furthermore, it would not be realistic to have feedback signals which do not have sensors to be measured physically and to have control signals which cannot affect the dynamics because of lack of actuators. Therefore, proper MIL to SIL/HIL conversion required estimating the above signals independently in the plant model and controller model, eliminating the need for communicating the mentioned signals and the behavior that was due to the delay in CAN communication. A schematic of the proper implementation is given in Figure 39.

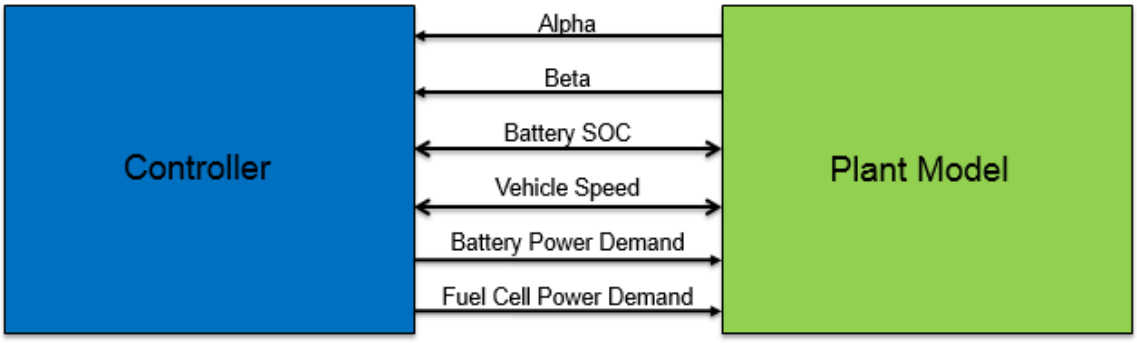

Figure 39: Schematic of Proper Model Structure for SIL/HIL Implementation

5.3 Software-in-the-Loop Simulator Development

The motivation behind the building of a SIL model is to validate how the control algorithm would work when used as software code for the bus' control unit. This requires converting the controller model into C-code. As with the MIL simulator, the SIL model was developed on MATLAB/ SIMULINK. Specifically, SIMULINK contains a block, titled S- function block, which allowed the control algorithm to be converted into C-code. By specifying the properly defined supervisory controller, the S- function block automatically generates the necessary code per the defined conversion rules set by

SIMULINK. One downside to this method is not readily knowing the exact code being generated for the controller model.

As explained in the above section, the main components of a vehicle model are the controller and plant models. Once the ARMA and PMP controllers were structured according to the proper implementation techniques, the controller model for the fuel cell hybrid SIL simulator was completed through use of the S-function block automatic code generator. The resulting control software code was then run alongside the correctly structured plant model in the SIMULINK virtual environment. The completed SIL simulator allowed for the quick interchanging of controllers for use with the fuel cell hybrid bus plant model. The use of software code instead of a model caused the simulator to complete simulations at a much faster rate. However, since the controller was no longer in the form of a model, the control algorithm could no longer be modified without having to generate the C-code for the controller each time. To remedy this, code was generated for each initial fuel cell value used.

5.3.1 Use of Discretization to Approximate the Behavior of a Real Controller

As is common with control hardware, signals are processed as discrete data and thus contain visible effects of quantization. However, initial SIL simulations revealed the lack of these discretization and quantization effects that are indicative of real controllers. This is due to the SIL control software being run on a virtual environment thus producing a digital approximation of an analog signal in continuous time. While an analog signal is continuously varying, a discrete signal can only contain one of a set of

finite values in time. In view of the SIL/HIL comparison in the next chapter, these quantization effects were artificially added to the SIL model. This allowed the SIL controller to better approximate a real controller. In doing so, a better comparison with the HIL simulator which uses real controller hardware, was possible.

The quantization effects were added through use of the SIMULINK Quantizer block which required defining a quantization interval. The input signal is essentially passed through a stair-step function, forcing neighboring points on the input axis to be mapped to one point on the output axis. Thus, a smooth signal is quantized into a stair-step output, computed using the round-to-nearest method such that

$$y = q * round(u/q) \tag{66}$$

where y is the output, u is the input, and q is the quantization interval. The value used for the quantization interval at which the input control signals were discretized was determined by first deciding on a rough estimate of how many bits the signal would contain when sent over a CAN network. The total number of possible quantization levels l associated with the respective number of bits was then calculated as

$$l = 2^{(\#\ of\ Data\ Bits)}. \tag{67}$$

The interval was then calculated using an estimated range of values r for the signal and the number of quantization levels such that

$$q = \frac{r}{l}. \tag{68}$$

This calculated interval value was then passed through the ceiling function to obtain the quantization interval value used within the Quantizer block. The values used for the determination of the quantitation interval for the four control inputs are found in Table 17. In addition to the above, a memory block was used to introduce a time delay that would be representative of the behavior seen in the HIL simulator.

Table 17: Quantization Effects for SIL/HIL Comparison

Control Input Signal	Signal Bits	Quant. Levels	Est. Data Range	Calculated Quant. Int.	Used Quant. Interval
Vehicle Velocity (mph)	13	8192	35	0.004272	0.01
Battery SOC (%)	10	1024	1	0.000977	0.001
Brake Pedal Position (%)	10	1024	1	0.000977	0.001
Accel. Pedal Position (%)	10	1024	1	0.000977	0.001

5.4 Hardware in the Loop Simulator Development

As indicated by the second objective for the SIL/HIL validation of the fuel cell hybrid passenger bus, it was desired to determine how well the simulator and control algorithm would perform on real-time hardware. This was accomplished through the development of the HIL simulator and test bench system. The HIL simulator, as with the SIL simulator, was made up of separate plant and controller models. In contrast to the SIL simulator, the controller model was to be used on real control hardware and the plant model was implemented as part of a dedicated HIL system module. The HIL test

bench system used consisted of a dSPACE HIL system, which acted as the real-time hardware for the plant model, and a MicroAutobox (MABX) microcontroller, which was used as the controller hardware for the controller software code. In addition, a PC and laptop with the dSPACE ControlDesk software installed were used to run the plant and controller models on their respective hardware devices as shown in Figure 40, where the two systems are seen to communicate over a CAN network.

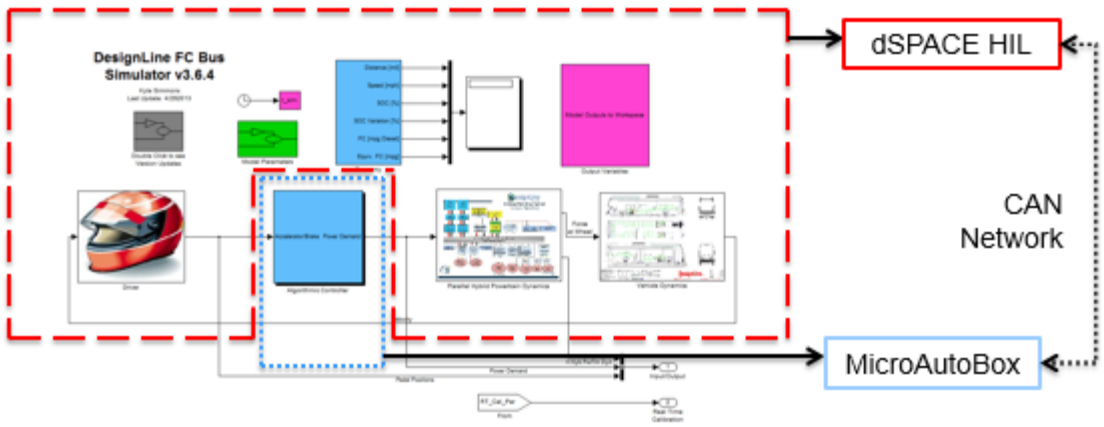

Figure 40: Interaction between Plant/Controller Model and HIL System Components

Having properly defined the controller for the SIL simulator and followed proper model structure to maintain separate independent plant and controller models, the HIL simulator development process was much easier. Specifically, the development of the HIL simulator involved taking the properly structured main vehicle model components and running the Real-Time Interface (RTI) MATLAB/SIMULINK add-on provided as part of the dSPACE ControlDesk software on each separate plant and controller model. RTI forms a link between the dSPACE hardware and a SIMULINK model and works by implementing the respective SIMULINK model into a graphical I/O configuration that is

then automatically generated into code that can be readily used in ControlDesk. Therefore, the controller and plant models are turned into software code that is then flashed onto the MABX microcontroller and dSPACE HIL system, respectively.

5.4.1 Validation of CAN DAQ System Using HIL Test Bench

As seen from

Figure 41, the developed CAN DAQ system was then added to the HIL test bench to incorporate the validation of the data logger hardware and software with that of the controller and simulator vehicle architecture validation. Integration of the CAN DAQ system was facilitated due to the HIL system already having an established CAN network over which signals were sent. Per the design of the CAN DAQ system software, any signal that is sent to the high speed CAN module that is also defined within a database file should be logged to its respective array of data. Thus, by creating a database file that contained several messages that pertained to signals being used within the HIL simulator the CAN DAQ system could be tested on actual real-time signals. This process was done several times during the running of HIL simulations. Each time the resulting TDMS file was inspected and compared to the signals that were sent over the CAN network. After several runs, it was concluded that the CAN DAQ system was capable of consistent successful logging of data over a CAN network.

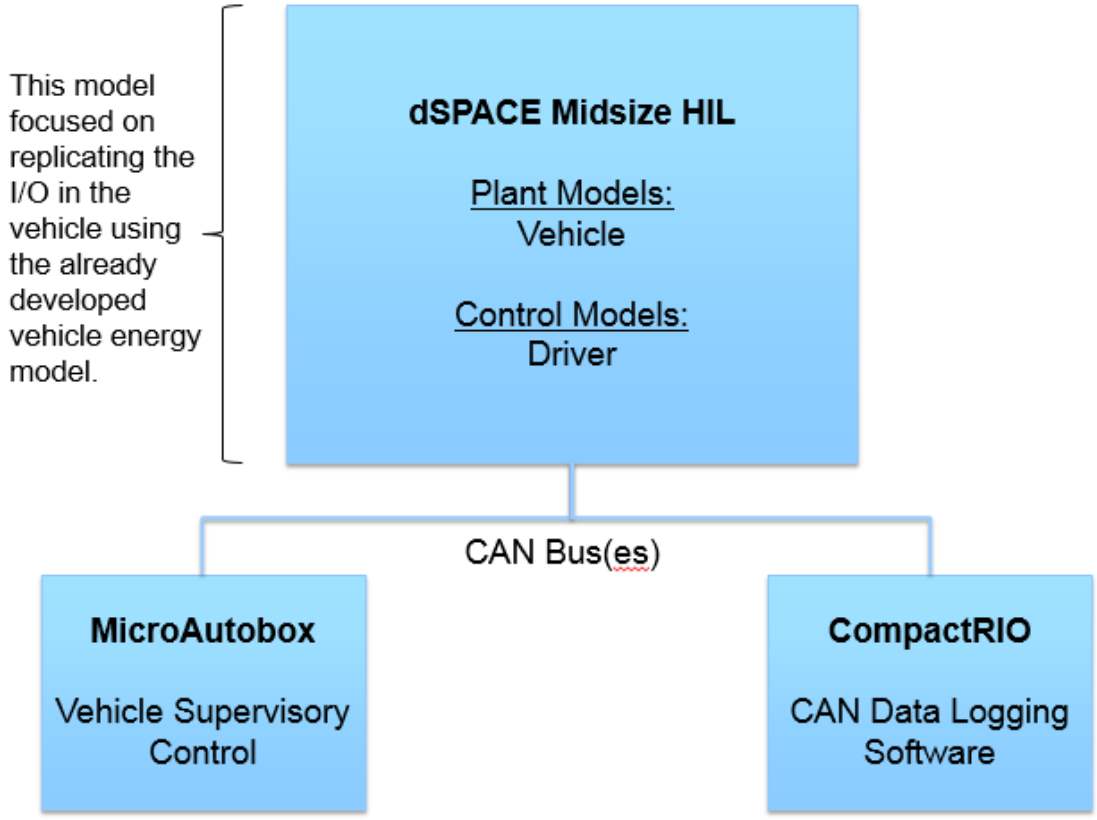

Figure 41: Diagram of HIL Test Bench with Integrated CAN DAQ System

5.5 Conclusion

The described processes in this chapter were used in building the SIL and HIL simulators for various cases involving the ARMA and PMP controllers, the original and distance based driver model, and the standardized and OSU real-world drive cycles. The following chapter contains a comprehensive comparison between the MIL, SIL, and HIL simulators including a sensitivity analysis of the co-state variable and initial fuel cell power for the SIL and HIL simulators.

CHAPTER 6: SIL/HIL Validation Results

6.1 Introduction

The SIL and HIL simulator were developed as described in the previous chapter. In addition, artificial quantization effects were added to approximate the behavior of a real controller while also creating a better comparison between the SIL and HIL validation results. The following sections pertain to said validation results for the standardized and real-world drive cycle in which three simulators were used with both the ARMA and PMP controllers, as shown in Table 18.

Table 18: Validation Test Cases

Test Case	Description
SIL	Software-in-the-Loop Analog Signal Simulator
SILQ	Software-in-the-Loop with Quantization Simulator
HIL	Hardware-in-the-Loop Simulator

6.2 SIL/HIL Validation on Standardized Drive Cycles

The validation process so far has proven to be successful for the MIL fuel cell hybrid bus simulator when run on the chosen standardized drive cycles. The HDUDDS and Manhattan drive cycles were run on the SIL and HIL simulators for both the ARMA and PMP controller. The results showed that the SIL model was able to function almost exactly like the MIL with no need of re-tuning of the co-state variable. However, for both the HDUDDS and Manhattan cycles, the co-state value had to be changed for the SIL with quantization simulator. This change in co-state value had an associated change in the initial fuel cell power, leading to a decrease in the diesel equivalent fuel consumption as seen by the baseline SIL simulator. The co-state variable and initial fuel cell power values used with the SIL and HIL simulators for the HDUDDS and Manhattan drive cycles are given in Table 19. Use of these supervisory controller parameters led to the results found in *Table 20 and*

Table 21 for the standardized drive cycles.

Table 19: Co-State and Initial Fuel Cell Power Values - Standardized Drive Cycles

	Co-State Variable	Initial Fuel Cell Power
HDUDDS – SIL	2038860	38 kW
HDUDDS – SILQ	2070610	44 kW
HDUDDS – HIL	2070610	44 kW
Manhattan – SIL	1945260	20 kW
Manhattan – SILQ	1995260	30 kW
Manhattan – HIL	1995260	30 kW

Table 20: SIL and HIL Validation Results - HDUDDS Drive Cycle

	Velocity RMS Error (%)	Fuel Consumption (kg)	Diesel Equivalent Mileage (mpg)
HDUDDS – ARMA SIL	1.11	0.550	8.67
HDUDDS – PMP SIL	1.06	0.542	8.45
HDUDDS – ARMA SILQ	1.27	0.615	7.55
HDUDDS – PMP SILQ	1.59	0.617	7.36
HDUDDS – ARMA HIL	1.18	0.568	6.79
HDUDDS – PMP HIL	1.36	0.627	6.22

Table 21: SIL and HIL Validation Results - Manhattan Drive Cycle

	Velocity RMS Error (%)	Fuel Consumption (kg)	Diesel Equivalent Mileage (mpg)
Manhattan – ARMA SIL	2.38	0.338	7.72
Manhattan – PMP SIL	2.30	0.275	8.52
Manhattan – ARMA SILQ	2.88	0.431	5.39
Manhattan – PMP SILQ	3.89	0.412	5.58
Manhattan – ARMA HIL	2.41	0.314	5.93
Manhattan – PMP HIL	2.86	0.254	6.17

A representative plot of the comparison of HDUDDS battery SOC trajectories achieved by the MIL baseline and SIL simulators running the ARMA control strategy in Figure 42 shows that the SIL simulator was a near exact match to the baseline results. The SIL with quantization simulator, on the other hand, causes a slight deviation from the baseline results, as expected. In general, the battery SOC trajectories for the HDUDDS and Manhattan drive cycles were seen to fall within the performance metric boundaries of 50% and 70% for the SIL, SIL with quantization, and HIL simulators, with

the SIL software code of the controller providing a trajectory that is almost the exact match of the baseline SOC trajectory. Thus, it was verified that the code generated from the MIL controller model functions in a relatively identical fashion to the model.

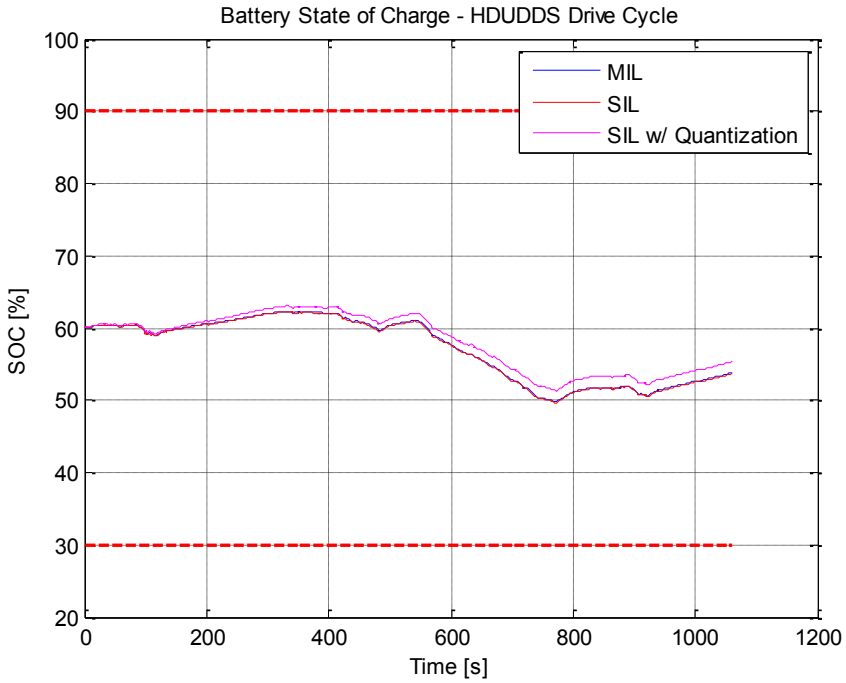

Figure 42: Representative MIL/SIL Battery SOC Comparison Using the ARMA Controller - HDUDDS Drive Cycle

Additionally, since the results of the SIL simulator with quantization effects approximates those of the HIL simulator more than do the baseline SIL results, it can be concluded that the addition of quantization achieved the desired effect. Figure 43 contains a graphical representation of the SIL and HIL validation results for the standardized HDUDDS and Manhattan drive cycles where emphasis was given to the performance metrics of the velocity traceability and fuel economy as compared to the baseline MIL performance results

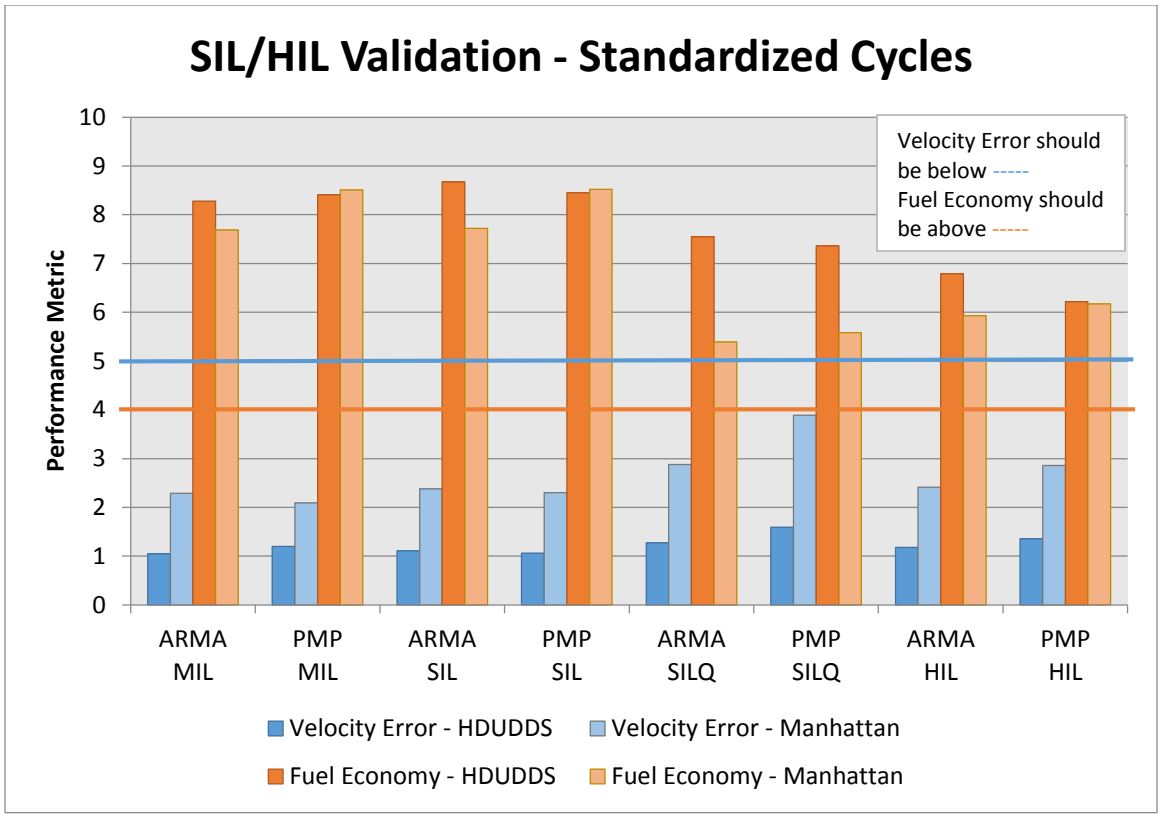

Figure 43: SIL/HIL Validation Bar Graph - Standardized Drive Cycles

The bar graph above helps highlight the differences in velocity error and fuel economy as obtained from the MIL, SIL, and HIL simulators for the HDUDDS and Manhattan cycles. A color coded line has been added to the graph that indicates the performance metric. In the case of the velocity error, the blue line is the threshold that must not be passed in order for the performance metric to be met. On the other hand, the orange line depicts the 4 mpg diesel transit bus baseline used as the fuel economy performance metric that should be passed to indicate a good performance. Thus, the velocity error and fuel economy metrics have been met for the standardized drive cycles throughout the validation process by the ARMA and PMP controllers.

6.3 SIL/HIL Validation on Real-World OSU Drive Cycles

The MIL results for the real-world drive cycles of the CLN and CC bus route revealed that the bus powertrain was adequate enough to meet the road loads of such a drive cycle per the performance metrics chosen. A similar analysis was then carried out in the SIL and HIL simulators in order to validate the control algorithm using the real-world OSU drive cycles. While the SIL simulator with quantization effects required the retuning of the co-state variable for the standardized drive cycles, the same was not the case with the OSU drive cycles, *i.e.*, the same MIL co-state value also allowed for a proper SOC charge sustaining behavior with the SIL and HIL simulators for the PMP controller. This meant that the same initial fuel cell power was also used throughout the SIL and HIL validation for the ARMA controller. The values of the OSU drive cycle co-state and initial fuel cell used through the SIL and HIL validation for the ARMA and PMP controllers are given in Table 22.

Table 22: Co-State, Initial Fuel Cell Power, and Torque Output Gain Values – OSU CC Drive Cycles

	Co-State Variable	**Initial Fuel Cell Power**
OSU CLN – SIL	1991440	29 kW
OSU CLN – SILQ	1991440	29 kW
OSU CLN – HIL	1991440	29 kW
OSU CC – SIL	2019560	34 kW
OSU CC – SILQ	2019560	34 kW
OSU CC – HIL	2019560	34 kW

The SIL and HIL simulators were then run using these controller parameters for the respective supervisory controllers with the OSU drive cycles in order to validate the results received with the MIL baseline for on-campus use. As seen from Table 23, the velocity error for the SIL with quantization simulator for the CLN drive cycle is seen to be slightly larger than the SIL and MIL baseline results when using both the developed ARMA and optimal PMP controllers. This is due to the effects of discretization that are applied to the velocity control input signal as described in the previous chapter. Regardless of this increase in RMS velocity error for the SIL with quantization simulator, the validation results indicate that the velocity performance metric was met for all of the SIL and HIL simulators. In addition, the RMS distance error achieved with the MIL, SIL, and HIL simulators are seen to be within the distance performance metrics as shown in Table 24. However, while the total distance for the MIL and SIL results are a near match, the total distance obtained with the HIL simulator is off by nearly 400 and 100 meters for the ARMA and PMP controller, respectively. This is explained by the inherent accumulation of error in the HIL simulator and to the competing tracking of distance and velocity with the real-world drive cycles.

Table 23: SIL and HIL Validation Results - CLN Drive Cycle

	Velocity RMS Error (%)	Fuel Consumption (kg)	Diesel Equivalent Mileage (mpg)
CLN – ARMA SIL	1.32	0.993	7.39
CLN – PMP SIL	1.59	0.920	7.85
CLN – ARMA SILQ	3.16	1.03	7.26
CLN – PMP SILQ	3.76	0.931	7.74
CLN – ARMA HIL	4.42	1.01	6.04
CLN – PMP HIL	4.58	0.922	6.57

Table 24: Distance Error Validation – CLN Drive Cycle

	Distance RMS Error (%)	Distance Travelled (m)
CLN – ARMA SIL	2.00E-3	9.80E+3
CLN – PMP SIL	2.60E-3	9.80E+3
CLN – ARMA SILQ	5.70E-3	9.80E+3
CLN – PMP SILQ	7.00E-3	9.80E+3
CLN – ARMA HIL	2.32E-3	9.40E+3
CLN – PMP HIL	2.24E-3	9.69E+3

The results for the CC drive cycle, given as Table 25, also show the effect of the quantization on the velocity RMS error, i.e., there is a clear increase in the velocity error from that of the SIL simulator without quantization. Again, this is due to the effects of discretization that were added to the velocity control input signal. As seen from

Table 26, the RMS distance error of the CC drive cycle is small and the total distance travelled is an exact match for all of the simulators except for the HIL simulator, which produces a total distance that is off by 300 meters for the ARMA controller but only by 30 meters for the PMP controller. As with the CLN drive cycles, the PMP controller allows for better matching of the desired total distance traveled.

Table 25: SIL and HIL Validation Results - CC Drive Cycle

	Velocity RMS Error (%)	Fuel Consumption (kg)	Diesel Equivalent Mileage (mpg)
CC – ARMA SIL	1.45	1.15	7.76
CC – PMP SIL	1.01	1.03	8.48
CC – ARMA SILQ	3.67	1.19	7.65
CC – PMP SILQ	4.09	1.04	8.49
CC – ARMA HIL	1.86	1.21	6.05
CC – PMP HIL	4.55	1.04	6.80

Table 26: Distance Error Validation - CC Drive Cycle

	Distance RMS Error (%)	Distance Travelled (m)
CC – ARMA SIL	2.00E-3	9.63E+3
CC – PMP SIL	1.60E-3	9.63E+3
CC – ARMA SILQ	5.90E-3	9.63E+3
CC – PMP SILQ	6.80E-3	9.63E+3
CC – ARMA HIL	3.60E-3	9.34E+3
CC – PMP HIL	1.44E-3	9.60E+3

The resulting battery SOC trajectories showed a good match between the SIL with quantization simulator and the HIL simulator as proven by the representative SOC plot of the OSU CC drive cycle given in Figure 44, where it is seen that the battery SOC performance metric is met.

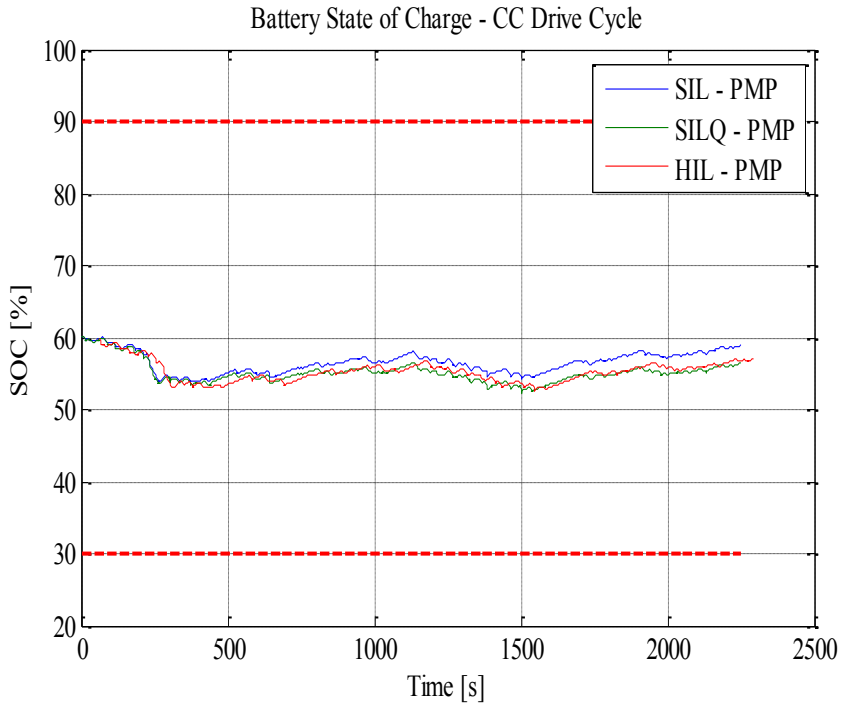

Figure 44: Representative SIL/HIL Battery SOC Comparison Using the PMP Controller - CC Drive Cycle

The SIL and HIL validation results for the OSU CLN and CC drive cycles are displayed in a graphical representation in Figure 45 where emphasis was given to the performance metrics of the velocity traceability and fuel economy as compared to the baseline MIL performance results. Similar to the validation bar graph of the standardized drive cycle results, a color coded line has been added to the graph that indicates the performance metric.

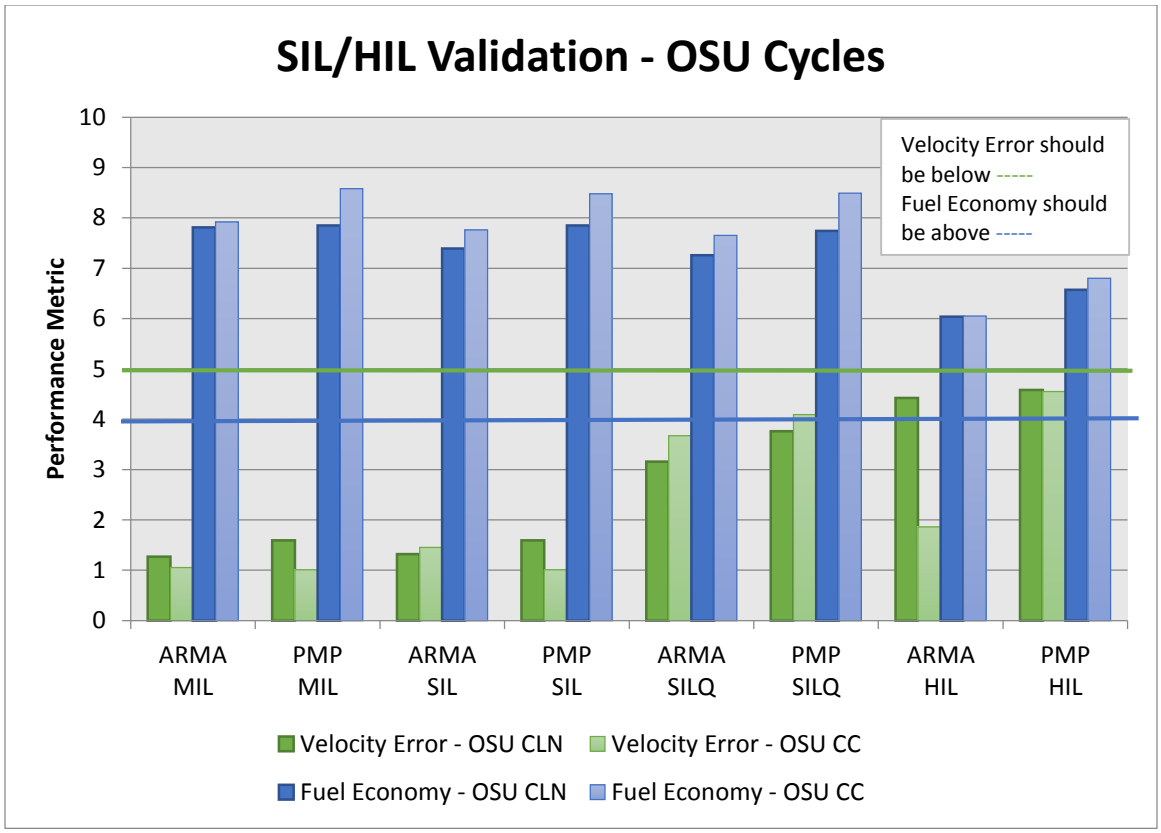

Figure 45: SIL/HIL Validation Bar Graph - OSU Drive Cycles

The blue line in this case is the threshold defined by the 4 mpg diesel transit bus baseline used as the fuel economy performance metric that should be passed to indicate a good performance. The green line is defined as the velocity error percentage that must not be passed in order for the performance metric to be met. Overall, the bar graph shows that the velocity error and fuel economy metrics were met for the developed real-world OSU drive cycles throughout the validation process by the developed ARMA and optimal PMP supervisory controllers.

6.4 Conclusion

The results above were obtained through the SIL and HIL validation of the developed ARMA supervisory controller and proposed powertrain architecture of the fuel cell hybrid bus MIL simulator. A set of results were obtained using the standardized HDUDDS and Manhattan drive cycles, which showed that the ARMA controller could approximate the optimal solution and meet the performance requirements when run in real-time. In addition, the set of results for the developed OSU drive cycles, CLN and CC, showed good performance of the ARMA controller on real-time hardware. Inspection of Figure 43 and Figure 45 reveals a trend in the results*, i.e.,* the level of velocity error increases while the fuel economy generally decreases as the use of real-time simulations are approached in the validation process. These differences in the real-time simulation results of the HIL simulator are due to an inevitable execution time delay associated with HIL simulations. [59] Other delays could be present due to the HIL system interface with the CAN DAQ system or to improper synchronization between the controller and plant model. These delays could explain the large variation between the desired total distance traveled and that which was obtained with the HIL simulator.

In order to reduce some of the delays present due to improper model synchronization, switches were added to the HIL simulator via the ControlDesk interface. However, these are dependent on human interaction and could contain delays as well. The effects of delays could be removed from the HIL system through application of countermeasures similar to those in [60]. Yet, as seen in the results

presented in this chapter, the presence of quantization also causes a decrease in the fuel economy obtained. Thus, through the validation process it was learned that the HIL simulator did not only contain effects of time delays but also of discretization that caused its values of fuel economy to be lower than those of the baseline results.

The results indicate that re-tuning for the standardized drive cycles was only needed once the effects of quantization were introduced. Yet, the presence of quantization is only seen to require re-tuning of the co-state variable for the standardized drive cycles but not for the real-world drive cycles. Regardless, the optimal co-state value used for the MIL and SIL simulators differs from that used for the SIL with quantization and HIL simulators by only an approximate 2% for the standardized drive cycles. Therefore, it can be concluded that the co-state variable, and coincidentally the initial fuel cell power for a specific drive cycle, is not very sensitive to the differences in the MIL, SIL, and HIL simulators. This fact is favorable in that, in theory, one could create a controller for a MIL simulator and, assuming that it does not require prior knowledge that would make it non-implementable in real control hardware, use it on the desired application immediately after a MIL validation.

Furthermore, while an optimal PMP controller is typically not implementable, measures could be taken to prove otherwise, *i.e.*, a set of acceptable co-state variable values could be determined for various driving conditions and used within the controller along with a pattern detection algorithm to pair up certain driving conditions with an

acceptable co-state variable. This would effectively make the PMP implementable though, arguably it would not be optimal at all times.

As for the results of the ARMA controller, it was seen that they closely approach the optimal solution. As was seen during the validation process, the initial fuel cell power used plays a large part in the ARMA controller's proximity to the optimal solution. Specifically, it was seen that the initial fuel cell power had to be set beforehand for each drive cycle. This slight requirement for near optimality with the ARMA controller could be thought of as grounds for a classification of a non-implementable controller, as setting the initial fuel cell power on the actual bus would seem to require prior knowledge of the driving conditions. However, as the application at hand involves running the fuel cell hybrid transit bus on specific bus routes, the bus' ECU could overcome this by sending information to the controller concerning the planned bus route for the bus. This information is readily available in several buses which currently contain a system that alerts its passengers of the current and following bus stop. The controller would then use this information to determine how much fuel cell power it should expect to use.

These additions to the ARMA and PMP controllers, though not necessary, would enable both to truly be implementable while allowing for near-optimal operation of the fuel cell hybrid transit bus. Overall, the results have proven that the SIL and HIL validation of the developed ARMA control strategy for use with the fuel cell hybrid transit bus simulator was successful.

CHAPTER 7: Conclusions and Future Work

7.1 Conclusions

As a sub-contractor of CTE for the NFCBP's ECO Saver IV demonstration bus, OSU has developed a fuel cell hybrid bus simulator and control algorithm. The work presented in this thesis was focused on the process applied to the validation of the simulator and control algorithm as well as the development of a CAN based DAQ system that is meant to be deployed on the prototype bus. The Validation process was initiated through obtaining a set of baseline results with the MIL fuel cell hybrid bus simulator using an optimal PMP controller and the developed ARMA controller. Modifications of the MIL simulator for appropriate SIL/HIL implementation were carried out before developing a SIL simulator. Artificial quantization effects that were added to the SIL simulator highlighted the expected behavior of the HIL simulator. The results indicated that the developed ARMA controller can approximate the optimal PMP solution while also allowing the simulated fuel cell hybrid bus to meet certain performance metrics.

Real-world drive cycles were developed that were representative of the driving condition found at the OSU campus. A driver model was developed for tracking both distance and velocity for use with the real-world drive cycles that ensured that the velocity and road grade profiles were synchronized. Initial on-campus validation suggested that the currently designed powertrain architecture was adequate enough to meet the demands of the OSU CLN and CC drive cycles. SIL results using the OSU drive cycles were then obtained for the on-campus driving conditions.

A HIL Test Bench was developed that allowed for the proposed ARMA control algorithm to be validated in real-time. Overall, it was shown that the ARMA supervisory controller was able to perform well on real-time hardware, an important step towards the eventual use on the bus' control hardware. In addition, the developed CAN DAQ system was proven to successfully log data from a real signal when tested alongside the HIL Test Bench.

7.2 Future Work

The developed distance based driver model for use with the bus route based OSU drive cycles tracked the linear velocity and distance. It is suggested that the driver model be modified to track the curved distance between each data point in the drive cycle as this would provide a more accurate portrayal of the distance travelled when considering a non-zero road grade profile. The Vincenty Inverse Formula used for determining the road grade profile would be able to achieve this.

As the HIL simulator was seen to contain time delays, future work could be done to apply countermeasures that compensate for the time delays. Doing so would allow for more accurate results and a better comparison between the various simulator results and the actual data that is collected.

Also, as mentioned in the final remarks concerning the validation results, the ARMA and PMP controllers could be made to be implementable on the actual fuel cell hybrid bus through the use of a pattern recognition algorithm that would respectively modify the fuel cell power or co-state variable in real-time. While not necessary for the implementation of the ARMA controller, this work could be done in the future as a way to demonstrate the application of an optimal control strategy in real control hardware.

Finally, as the ECO Saver IV demonstration bus was not completed at this time, the deployment of the developed CAN DAQ system is left as a future work, pending the completion of the prototype fuel cell hybrid transit bus.

References

[1] "Study of the Potential Impacts of Hydraulic Fracturing on Drinking Water Resources - Progress Report," US Environmental Protection Agency Office of Research and Development, EPA/601/R-12/011, 2012.

[2] S. G. Osborn, A. Vengosh, N. R. Warner, and R. B. Jackson, "Methane contamination of drinking water accompanying gas-well drilling and hydraulic fracturing," *Proceedings of the National Academy of Sciences*, vol. 108, no. 20, pp. 8172–8176, 2011.

[3] J. Ogden, J. M. Cunningham, and M. A. Nicholas, "Roadmap for Hydrogen and Fuel Cell Vehicles in California: A Transition Strategy through 2017," 2010.

[4] L. Eudy, K. Chandler, and C. Gikakis, "Fuel Cell Busses in U.S. Transit Fleets: Current Status 2012," National Renewable Energy Laboratory, Technical Report TP-5600-56406, 2012.

[5] "National Fuel Cell Bus Program," 2012. [Online]. Available: www.fta.dot.gov.

[6] "Emissions Standards Reference Guide," 2012. [Online]. Available: www.epa.gov.

[7] D. F. Opila, X. Wang, R. McGee, J. A. Cook, and J. W. Grizzle, "Performance comparison of hybrid vehicle energy management controllers on real-world drive cycle data," in *American Control Conference, 2009. ACC'09.*, 2009, pp. 4618–4625.

[8] K. S. Nesamani and K. P. Subramanian, "Development of a driving cycle for intra-city buses in Chennai, India," *Atmospheric Environment*, vol. 45, no. 31, pp. 5469–5476, Oct. 2011.

[9] S. H. Kamble, T. V. Mathew, and G. K. Sharma, "Development of real-world driving cycle: Case study of Pune, India," *Transportation Research Part D: Transport and Environment*, vol. 14, no. 2, pp. 132–140, Mar. 2009.

[10] B. M. Marshall, J. C. Kelly, T.-K. Lee, G. A. Keoleian, and Z. Filipi, "Environmental assessment of plug-in hybrid electric vehicles using naturalistic drive cycles and vehicle travel patterns: A Michigan case study," *Energy Policy*, vol. 58, pp. 358–370, Jul. 2013.

[11] J. Gonder, T. Markel, M. Thornton, and A. Simpson, "Using Global Positioning System Travel Data to Assess Real-World Energy Use of Plug-In Hybrid Electric Vehicles," *Transportation Research Record*, vol. 2017, no. -1, pp. 26–32, Dec. 2007.

[12] M. André, R. Joumard, R. Vidon, P. Tassel, and P. Perret, "Real-world European driving cycles, for measuring pollutant emissions from high- and low-powered cars," *Atmospheric Environment*, vol. 40, no. 31, pp. 5944–5953, Oct. 2006.

[13] T. Vincenty, “Direct and Inverse Solutions of Geodesics on the Ellipsoid with Application of Nested Equations,” *Directorate of Overseas Surveys of the Ministry of Overseas Development - Survey Review*, vol. 23, no. 176, p. 6, 1975.
[14] R. Isermann, J. Schaffnit, and S. Sinsel, “Hardware-in-the-loop simulation for the design and testing of engine-control systems,” *Control Engineering Practice*, vol. 7, no. 5, pp. 643–653, 1999.
[15] S. Gurusubramanian, “A Comprehensive Process for Autmotive Model-Based Control,” The Ohio State University, 2013.
[16] K. M. Bovee, “Design of the Architecture and Supervisory Control Strategy for a Parallel-Series Plug-in Hybrid Electric Vehicle,” Ohio State University, 2012.
[17] H. Dai, X. Zhang, X. Wei, Z. Sun, J. Wang, and F. Hu, “Cell-BMS validation with a hardware-in-the-loop simulation of lithium-ion battery cells for electric vehicles,” *International Journal of Electrical Power & Energy Systems*, vol. 52, pp. 174–184, Nov. 2013.
[18] M. Short and M. J. Pont, “Assessment of high-integrity embedded automotive control systems using hardware in the loop simulation,” *Journal of Systems and Software*, vol. 81, no. 7, pp. 1163–1183, Jul. 2008.
[19] R. M. Moore, K. H. Hauer, G. Randolf, and M. Virji, “Fuel cell hardware-in-loop,” *Journal of Power Sources*, vol. 162, no. 1, pp. 302–308, Nov. 2006.
[20] E. Wilhelm, “Model-based validation of fuel cell hybrid vehicle control systems,” 2007.
[21] D. Ramaswamy, R. McGee, S. Sivashankar, A. Deshpande, J. Allen, K. Rzemien, and W. Stuart, “A case study in hardware-in-the-loop testing: Development of an ECU for a hybrid electric vehicle,” *SAE SP*, pp. 33–42, 2004.
[22] S. Varigonda and M. Kamat, “Control of stationary and transportation fuel cell systems: Progress and opportunities,” *Computers & Chemical Engineering*, vol. 30, no. 10–12, pp. 1735–1748, Sep. 2006.
[23] L. Xu, J. Li, M. Ouyang, J. Hua, and X. Li, “Active fault tolerance control system of fuel cell hybrid city bus,” *International Journal of Hydrogen Energy*, vol. 35, no. 22, pp. 12510–12520, Nov. 2010.
[24] J. Kruckenberg, “Fault Diagnosis and Hardware in the Loop Simulation for the EcoCAR Project,” The Ohio State University, 2011.
[25] O. J. Gietelink, J. Ploeg, B. De Schutter, and M. Verhaegen, “Development of a driver information and warning system with vehicle hardware-in-the-loop simulations,” *Mechatronics*, vol. 19, no. 7, pp. 1091–1104, Oct. 2009.
[26] K. Ponziani, “Control System Design and Optimization for the Fuel Cell Powered Buckeye Bullet 2 Land Speed Vehicle,” The Ohio State University, 2008.
[27] N. Jalil, N. A. Kheir, and M. Salman, “A rule-based energy management strategy for a series hybrid vehicle,” in *American Control Conference, 1997. Proceedings of the 1997*, 1997, vol. 1, pp. 689–693.

[28] H. Banvait, S. Anwar, and Y. Chen, “A rule-based energy management strategy for plug-in hybrid electric vehicle (PHEV),” in *American Control Conference, 2009. ACC’09.*, 2009, pp. 3938–3943.
[29] M. Sorrentino, G. Rizzo, and I. Arsie, “Analysis of a rule-based control strategy for on-board energy management of series hybrid vehicles,” *Control Engineering Practice*, vol. 19, no. 12, pp. 1433–1441, Dec. 2011.
[30] T. Hofman, M. Steinbuch, R. Van Druten, and A. Serrarens, “Rule-based energy management strategies for hybrid vehicles,” *International Journal of Electric and Hybrid Vehicles*, vol. 1, no. 1, pp. 71–94, 2007.
[31] Y. Eren, O. Erdinc, H. Gorgun, M. Uzunoglu, and B. Vural, “A fuzzy logic based supervisory controller for an FC/UC hybrid vehicular power system,” *International Journal of Hydrogen Energy*, vol. 34, no. 20, pp. 8681–8694, Oct. 2009.
[32] B. Sampathnarayanan, “Analysis and Design of Stable and Optimal Energy Management Strategies for Hybrid Electric Vehicles,” The Ohio State University, 2012.
[33] L. Serrao, “A Comparative Analysis of Energy Management Strategies for Hybrid Electric Vehicles,” The Ohio State University, 2009.
[34] R. E. Bellman, *Dynamic Programming*. Princeton University Press, 1957.
[35] M. J. Kim, H. Peng, C.-C. Lin, E. Stamos, and D. Tran, “Testing, modeling, and control of a fuel cell hybrid vehicle,” in *American Control Conference, 2005. Proceedings of the 2005*, 2005, pp. 3859–3864.
[36] M.-J. Kim and H. Peng, “Power management and design optimization of fuel cell/battery hybrid vehicles,” *Journal of Power Sources*, vol. 165, no. 2, pp. 819–832, Mar. 2007.
[37] L. V. Pérez, G. R. Bossio, D. Moitre, and G. O. García, “Optimization of power management in an hybrid electric vehicle using dynamic programming,” *Mathematics and Computers in Simulation*, vol. 73, no. 1–4, pp. 244–254, Nov. 2006.
[38] R. Wang and S. M. Lukic, “Dynamic programming technique in hybrid electric vehicle optimization,” in *Electric Vehicle Conference (IEVC), 2012 IEEE International*, 2012, pp. 1–8.
[39] Y.-B. Yu, “Control strategy optimization using dynamic programming method for synergic electric system on hybrid electric vehicle,” *Natural Science*, vol. 01, no. 03, pp. 222–228, 2009.
[40] C.-C. Lin, H. Peng, and J. W. Grizzle, “A stochastic control strategy for hybrid electric vehicles,” in *American Control Conference, 2004. Proceedings of the 2004*, 2004, vol. 5, pp. 4710–4715.
[41] G.-E. Katsargyri, “Optimally Controlling Hybrid Electric Vehicles Using Path Forecasting,” Massechusetts Institute of Technology, 2008.
[42] X. Lin, A. Ivanco, and Z. Filipi, “Optimization of Rule-Based Control Strategy for a Hydraulic-Electric Hybrid Light Urban Vehicle Based on Dynamic Programming,” *SAE International Journal of Alternative Power*, vol. 1, no. 1, pp. 249–259, 2012.

[43] M. P. O'Keefe and T. Markel, "Dynamic programming applied to investigate energy management strategies for a plug-in HEV," National Renewable Energy Laboratory, 2006.
[44] K. S. Simmons, "Modeling and Optimal Supervisory Controller Design for a Hybrid Fuel Cell Passenger Bus," The Ohio State University, 2013.
[45] B. Jager, T. Keulen, and J. Kessels, *Optimal Control of Hybrid Vehicles*. Springer London, 2013.
[46] J. Bernard, S. Delprat, T. M. Guerra, and F. N. Büchi, "Fuel efficient power management strategy for fuel cell hybrid powertrains," *Control Engineering Practice*, vol. 18, no. 4, pp. 408–417, Apr. 2010.
[47] O. P. Sharma, "A practical implementation of a near optimal energy management strategy based on the Pontryagin's minimum principle in a PHEV," The Ohio State University, 2012.
[48] Namwook Kim, Sukwon Cha, and Huei Peng, "Optimal Control of Hybrid Electric Vehicles Based on Pontryagin's Minimum Principle," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 5, pp. 1279–1287, Sep. 2011.
[49] L. Serrao, S. Onori, and G. Rizzoni, "ECMS as a realization of Pontryagin's minimum principle for HEV control," in *American Control Conference, 2009. ACC'09.*, 2009, pp. 3964–3969.
[50] A. Fadel and B. Zhou, "An experimental and analytical comparison study of power management methodologies of fuel cell–battery hybrid vehicles," *Journal of Power Sources*, vol. 196, no. 6, pp. 3271–3279, Mar. 2011.
[51] C. Musardo, G. Rizzoni, Y. Guezennec, and B. Staccia, "A-ECMS: An adaptive algorithm for hybrid electric vehicle energy management," *European Journal of Control*, vol. 11, no. 4, pp. 509–524, 2005.
[52] Y. Guezennec, T.-Y. Choi, G. Paganelli, and G. Rizzoni, "Supervisory control of fuel cell vehicles and its link to overall system efficiency and low-level control requirements," in *American Control Conference, 2003. Proceedings of the 2003*, 2003, vol. 3, pp. 2055–2061.
[53] C. H. Zheng, C. E. Oh, Y. I. Park, and S. W. Cha, "Fuel economy evaluation of fuel cell hybrid vehicles based on equivalent fuel consumption," *International Journal of Hydrogen Energy*, vol. 37, no. 2, pp. 1790–1796, Jan. 2012.
[54] L. Xu, J. Li, J. Hua, X. Li, and M. Ouyang, "Optimal vehicle control strategy of a fuel cell/battery hybrid city bus," *International Journal of Hydrogen Energy*, vol. 34, no. 17, pp. 7323–7333, Sep. 2009.
[55] L. Xu, J. Li, J. Hua, X. Li, and M. Ouyang, "Adaptive supervisory control strategy of a fuel cell/battery-powered city bus," *Journal of Power Sources*, vol. 194, no. 1, pp. 360–368, Oct. 2009.
[56] J. Morbitzer, "High-Level Modeling, Supervisory Control Strategy Development, and Validation for a Proposed Power-Split Hybrid-Electric Vehicle Design," The Ohio State University, 2005.
[57] "National Instruments," 2013. [Online]. Available: www.ni.com.

[58] A. Schneider, “GPS Visualzier,” 2013. [Online]. Available: www.gpsvisualizer.com.
[59] C. A. G. Carrillo, J. V. Ferreira, and P. S. Meirelles, “THE CHARACTERIZATION OF THE TIME DELAY PROBLEM IN HARDWARE IN THE LOOP SYSTEM APPLICATIONS,” *Conference Proceedings of the Society for Experimental Mechanics Series*, vol. 7, pp. 661–671, 2013.
[60] Chinchul Choi and Wootaik Lee, “Analysis and Compensation of Time Delay Effects in Hardware-in-the-Loop Simulation for Automotive PMSM Drive System,” *IEEE Transactions on Industrial Electronics*, vol. 59, no. 9, pp. 3403–3410, Sep. 2012.