# SYSTEM MODELING AND ENERGY MANAGEMENT STRATEGY DEVELOPMENT FOR SERIES HYBRID VEHICLES

A Thesis
Presented to
The Academic Faculty

by

Patrick Wilson Cross

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
George W. Woodruff School of Mechanical Engineering

Georgia Institute of Technology
August 2008

# SYSTEM MODELING AND ENERGY MANAGEMENT STRATEGY DEVELOPMENT FOR SERIES HYBRID VEHICLES

Approved by:

Dr. Nader Sadegh, Advisor
George W. Woodruff School of Mechanical Engineering
*Georgia Institute of Technology*

Dr. Wayne Book
George W. Woodruff School of Mechanical Engineering
*Georgia Institute of Technology*

Dr. William Singhose
George W. Woodruff School of Mechanical Engineering
*Georgia Institute of Technology*

Date Approved: May 14, 2008

# ACKNOWLEDGEMENTS

I would first of all like to thank my advisor, Dr. Nader Sadegh, for guiding me through the research process, both academic and administrative. In addition to helping with my research, I developed by interest for control theory and practice in courses that he taught as an undergraduate and graduate student.

I would also like to thank my wife for enduring the late nights of work, supporting us during school, and always giving me a push when needed.

Lastly, I would like to thank John Deere and the members of the Advanced Energy Systems group in Charlotte, NC. Not only did they fund my research, but they were always ready to immediately provide information and guidance when it was needed.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# NOMENCLATURE

| Symbol | Description | Unit |
|---|---|---|
| $A$ | Area | $m^2$ |
| $b_c$ | Linear regression offset for charging | W |
| $b_d$ | Linear regression offset for discharging | W |
| $BDC$ | Bi-directional converter | N/A |
| $b_x$ | General BDC linear regression offset | W |
| $C$ | Capacitance due to charge double layers in porous electrodes | F |
| $C_b$ | Battery effective capacitance (linear model) | F |
| $c_p$ | Specific heat at constant pressure | kJ/(kg·K) |
| $EA$ | Engine/Alternator unit | N/A |
| $E_{in}$ | Energy input | J |
| $E_{out}$ | Energy output | J |
| $ESS$ | Energy Storage System | N/A |
| $h$ | Heat transfer coefficient | $W/(m^2 \cdot K)$ |
| $i_{alt}$ | Alternator output current | A |
| $i_{batt}$ | Battery current | A |
| $i_{charge,max}$ | Battery current limit, charge | A |
| $i_{discharge,max}$ | Battery current limit, discharge | A |
| $i_{motors}$ | Traction motor current | A |
| $J$ | Rotational inertia | $kg \cdot m^2$ |
| $k$ | Steady state gain | various |
| $LHV_{diesel}$ | Average lower heating value of diesel fuel | J/kg |
| $m$ | Mass | kg |
| $m_c$ | Linear regression slope for charging | unitless |
| $m_d$ | Linear regression slope for discharging | unitless |
| $m_x$ | General BDC linear regression slope | unitless |
| $N_{alt}$ | Alternator speed | rpm |
| $N_{cells}$ | Number of cells in the battery pack | unitless |
| $N_{eng}$ | Engine speed | rpm |
| $N_{motors}$ | Motor speed | rpm |
| $N_{set}$ | Engine speed setpoint | rpm |
| $P_{alt}$ | Alternator output electrical power | W |
| $P_{batt}$ | Battery power at terminals (low side of converter) | W |
| $P_{batt,chem}$ | Rate of change of battery chemical energy storage | W |
| $P_{EA}$ | Engine/alternator output power | W |
| $P_{eng}$ | Engine output power | W |
| $P_{ESS}$ | Energy storage system output power | W |
| $P_{fuel}$ | Chemical input fuel power | W |
| $P_{HS}$ | High-side (bus) power ($P_{ESS}$) | W |
| $P_{LS}$ | Low-side (battery) power ($P_{batt}$) | W |

| Symbol | Description | Unit |
|---|---|---|
| $P_M$ | Traction system power at the bus | W |
| $P_{motors,traction}$ | Traction motor mechanical power output | W |
| $Q$ | Actual battery capacity | C (A·s) |
| $Q_{max}$ | Rated battery capacity | C (A·s) |
| $R$ | Total internal battery resistance | Ω |
| $r$ | radius | m |
| $R_1$ | Battery element resistance 1 | Ω |
| $R_2$ | Battery element resistance 2 | Ω |
| $SOC$ | Battery state of charge | unitless |
| $T$ | Battery temperature | °C |
| $T_a$ | Ambient temperature | °C |
| $T_{eng}$ | Engine output torque (at crankshaft) | N·m |
| $T_{eng,raw}$ | Engine cylinder torque (before delay) | N·m |
| $T_{load}$ | Load torque on engine from alternator | N·m |
| $T_{motors}$ | Actual motor torque | N·m |
| $T_{motors,com}$ | Motor torque command (from controller to motors/traction controller) | N·m |
| $T_{motors,req}$ | Motor torque request (from user to controller) | N·m |
| $T_{rel}$ | Relative temperature | °C |
| $V_1$ | Voltage of first RC battery element | V |
| $V_{batt}$ | Battery pack terminal voltage | V |
| $V_{bus}$ | DC bus voltage | V |
| $V_{cell}$ | Battery cell terminal voltage | V |
| $V_{fuel}$ | Volume of diesel fuel consumed | $m^3$ |
| $V_{OC}$ | Battery open circuit voltage | V |
| $Z_c$ | Complex impedance due to a capacitor | Ω |
| $\eta_{alt}$ | Alternator efficiency | unitless |
| $\eta_{batt}$ | Battery efficiency | unitless |
| $\eta_{BDC}$ | Bi-directional converter efficiency | unitless |
| $\eta_{cycle}$ | Total system cycle efficiency | unitless |
| $\eta_{EA}$ | Engine/alternator combined efficiency | unitless |
| $\eta_{eng}$ | Engine efficiency | unitless |
| $\eta_{ESS}$ | Energy storage system efficiency | unitless |
| $\eta_M$ | Traction system efficiency | unitless |
| $\eta_{motors}$ | Traction motor efficiency (electrical to mechanical) | unitless |
| $\eta_{PE}$ | Power electronics efficiency | unitless |
| $\eta_{PE}$ | Power electronics efficiency | unitless |
| $\rho_{diesel}$ | Mass density of diesel fuel | $kg/m^3$ |
| $\tau$ | First order system time constant | s |
| $\omega_{eng}$ | Engine speed | rad/s |
| $\omega_{motors}$ | Motor speed | rad/s |

# SUMMARY

A series hybrid electric vehicle is a vehicle that is powered by both an engine and a battery pack. An electric motor provides all of the mechanical motive power to the transmission. Engine power is decoupled from the transmission by converting engine power into electricity which powers the electric motor. The mechanical decoupling of the engine from the transmission allows the engine to be run at any operating point (including off) during vehicle operation while the battery back supplies or consumes the remaining power. Therefore, the engine can be operated at its most efficient operating point or in a high-efficiency operating region.

The first objective of this research is to develop a dynamic model of a series hybrid diesel-electric powertrain for implementation in Simulink. The vehicle of interest is a John Deere M-Gator utility vehicle. This model serves primarily to test energy management strategies, but it can also be used for component sizing given known load profiles for a vehicle.

The second objective of this research is to develop and implement multiple energy management strategies of varying complexity from simple thermostat control to an optimal control law derived using dynamic programming. These energy management strategies are then tested and compared over the criteria of overall fuel efficiency, power availability, battery life, and complexity of implementation. Complexity of implementation is a critical metric for control designers and project managers.

The results show that simple point-based control logic can improve upon thermostat control if engine efficiency maps are known. All control method results depend on the load profile being used for a specific application..

# CHAPTER 1

# INTRODUCTION

## 1.1 Background

As fuel costs and environmental concerns continue to rise, interest in methods for decreasing fossil fuel consumption has increased. One promising method of reducing energy consumption is through the use of hybrid vehicles. A hybrid vehicle powertrain is defined as having two sources that provide energy for an end use. Commonly one energy source is rechargeable (such as batteries) for energy recovery. For a diesel-electric hybrid vehicle, the two energy sources are the diesel fuel and the batteries and/or ultracapacitors. Since hybrid vehicles utilize two energy sources, they can potentially deliver that energy more efficiently than a non-hybrid vehicle. A diesel-electric hybrid can operate the engine in its high-efficiency region while allowing the battery pack to supply or consume the excess energy, thus increasing overall efficiency.

There are two major types of hybrid vehicles: series and parallel. For the purpose of explanation, the two energy sources for a hybrid vehicle will be an engine and a battery pack. In a series hybrid, the engine is not directly linked to the vehicle's transmission. Instead, the engine power is converted into electricity which powers an electric motor. This configuration allows the engine to be run at any operating point (including off) during vehicle operation. In a parallel hybrid, both the engine and the battery pack (via the electric motor) are independently linked to the transmission. Since both engine and battery/motor power can be delivered in parallel, the battery and motor can be undersized compared to a series hybrid and still deliver the same acceleration. However, since the engine is directly linked to the transmission, it cannot always be operated in its most efficient zone. Some vehicles are designed in a combination series/parallel (or powersplit) configuration. This configuration uses a power diverting

device (typically a planetary gearset and clutch configuration) so that power can be delivered in series or in parallel.

## 1.2 Motivation for Utility Vehicle Hybridization and Intelligent Energy Management

A utility vehicle is a small work vehicle that is used for a wide array of tasks including transport (brush, feed, tools, etc.), towing, blading (smoothing dirt), and stump pulling. Utility vehicles are used in many locations including farms, landscaping sites, athletic facilities, and military bases. Commercial and governmental users of utility vehicles are highly interested in decreasing fuel costs, so hybridization can make sense. In addition, many utility vehicle task load profiles are characterized by frequent peaks and valleys (stopping and starting while moving brush piles), which suggests significant fuel economy improvements can be made through hybridization. The use of a series hybrid also adds the feature of engine-off operation (battery only), which means the vehicle can be operated indoors.

If a series hybrid architecture is chosen, one of the ensuing tasks is determining how to control the power flows in the vehicle. When should the engine be turned on? Where should the engine be operated (speed and torque output)? These choices of how the engine and battery power flows are defined affect vehicle efficiency, available power, and user acceptability.

## 1.3 Past Work

While the concept of hybrid vehicles has existed for over a century, research into computerized control of hybrid vehicle power flows is much more recent. Research has been conducted including model based cost function minimization [1,2,3], rule-based control [4], and sliding mode control [5]. Methods for achieving optimal control via dynamic programming have also been presented [6,7,8], and other various energy

management strategies for both series and parallel hybrids are presented in [9]-[11]. Methods for hybrid vehicle modeling are presented in [12]-[17].

Wang and Sadegh presented a method for modeling and controlling a series hybrid powertrain using optimal control methods [1]. The modeling methods are purely linear which severely limits the accuracy of the model since hybrid powertrains are highly nonlinear. However, this linear modeling was necessary for implementation of linear optimal control methods. It is likely that these purely linear methods would not perform well on a real-world system.

Barsali, Ceraolo, and Possenti look at minimizing fuel consumption for a given power demand for the cases of neglecting battery losses and including battery losses [2]. They discuss losses associated with engine startup cycles and analytically model how these losses shift the optimal operating point of the engine given a forecasted load cycle. Their method of load forecasting is simple first-order low-pass filtering of the past load. A cost function is developed that relies on engine efficiency data, simplified battery performance data, and forecasted load. The proposed control method actively chooses and updates an engine power and minimum battery state-of-charge that minimizes the developed cost function. This method's reliance on an accurate load forecasting method makes it potentially difficult to implement.

Barsali, Miulli, and Possenti later improved upon the aforementioned previous work of Barsali, Ceraolo, and Possenti by developing a more detailed control algorithm and performing further case studies [3]. The new algorithm utilizes a new type of load forecasting that characterizes the load profile by an average power level and an average power ripple. The new algorithm uses these load profile parameters and a new cost function formulization to determine the proper engine operation that will minimize fuel consumption. This approach relies on both engine, generator/alternator, and battery system models, and it seems well formulated.

Jalil, Kheir, and Salman presented a rule based energy management strategy for series hybrid vehicles [4]. This approach is similar to standard thermostat (on/off with hysteresis) control, but with rules that override the basic thermostat control. There is little discussion of modeling, but good discussion of the purpose and goals of energy management strategies. An altered version of this method is discussed and implemented later.

Gokasan, Bogosyan, and Goering proposed a sliding mode based powertrain control for series hybrid vehicles [5]. The root of their proposed control algorithm is modified thermostat control, but two sliding mode controllers are used to ensure that the engine operates within its maximum efficiency region. While normal thermostat control typically turns the engine on to its maximum efficiency operating point (torque and speed couple), the method proposed by Gokasan, Bogosyan, and Goering turns the engine on to a torque value that depends on battery state-of-charge error (via a PID loop using a state-of-charge reference and the calculated state-of-charge). The requested engine torque is saturated to be kept within a high efficiency zone as opposed to the maximum efficiency point. Two sliding mode controllers are then used to keep the engine at the desired torque and speed couple. While the energy management is basic, the use of chatter-free sliding mode controllers (as opposed to a simple PI controller or similar) helps to ensure robust engine control.

Brahma, Guezennec, and Rizzoni presented a method for using dynamic programming to achieve optimal control of a series hybrid vehicle [6]. While they present a clear and concise system description, the algorithm that they use does not ensure charge sustainability. They use a weighting factor that must be tuned via trial and error to achieve charge sustainability. The use of dynamic programming also requires that the entire vehicle load profile be known which is highly unlikely in most applications.

Sciarretta and Guzzella presented a nice article on the optimal control of hybrid vehicles using dynamic programming [7]. The article is in reference to parallel hybrid

vehicles, but the dynamic programming methods and real-time implementation techniques are valid for series hybrid vehicles as well. They compiled a list of many of the research groups active in hybrid vehicle energy management and their various approaches.

Johannesson, Asbogard, and Egardt present a method for modeling load cycles as stochastic processes based on measured data [8]. They then present three controllers, each based on varying levels of information about the modeled load cycle. Dynamic programming was used for each of the controllers using different load cycle knowledge. The case with full knowledge of the entire load cycle gave the truly optimal solution, while the other solutions (based on less knowledge) were necessarily sub-optimal.

Kleimaier and Schroder present a method for online energy management optimization for parallel vehicles [9]. They develop a cost function utilizing weighting factors that penalize battery usage and engine cycling frequency (to prevent frequent on-off switching). These weighting factors must be determined experimentally.

Pisu and Rizzoni present a very good explanation of the energy control problem for series hybrid vehicles [10]. They discuss how the overall objective of minimizing fuel consumption for a load cycle is a global problem (solvable by dynamic programming), but real-time implementation requires a local problem formulation. Instead of minimizing total fuel consumption for the entire load cycle, they minimize an equivalent fuel rate at every time step. This equivalent fuel rate is the sum of the actual fuel rate and the corresponding estimated future fuel rate that will be required to return the battery to its present state-of-charge. This method is generally known as an instantaneous Equivalent Consumption Minimization Strategy (ECMS). While this paper was geared specifically for series hybrid electric vehicles with two energy storage systems (battery pack and ultracapacitors), the ECMS is general and can be applied to various hybrid architectures. One downside of this method is that the calculation of the

estimated future fuel rate relies on a guess of the future charge and discharge efficiencies (likely determined experimentally).

Pisu and Rizzoni later performed a comparative study of energy management systems, but for parallel hybrid electric vehicles [11]. While this study was for parallel vehicles in particular, the energy management systems and comparison methods could be adapted to the series hybrid case. One of the methods they use for the comparison study is an adaptive ECMS, or A-ECMS. This method uses an adaptive algorithm to adjust the estimates for future charge and discharge efficiencies online (as opposed to guessing or determining experimentally as mentioned in their previous paper). Rule-based and H-infinity control methods are also compared against the optimal solution derived using dynamic programming.

In relation to modeling, Rizzoni, Guzzella, and Baumann presented a general method for modeling hybrid vehicles [12]. Their modeling methods are directed towards modeling accurate energy and power flows between common hybrid components and subsystems, specifically internal combustion engines and induction type electric machines. A key aspect to their modeling efforts was to make all models scalable. This way these models could be easily applied to different sized hybrid vehicles for various applications. This method is appropriate for design optimization exercises, but when modeling a specific vehicle, it may be easier (and more accurate) to use specific lookup tables for the various components in the hybrid vehicle.

For battery performance modeling, Kumaresan, Sikha, and White presented a method for developing a thermal model of a lithium-ion cell for use in predicting the discharge performance at various operating temperatures [13]. Mathematical models are developed, and the parameters of the models were fit using test data. It was shown that this approach could predict battery voltage for different discharge rates at various temperatures very well. While these models work well, they rely on extensive test data to correctly fit the model parameters. For the model of a specific battery type, it may be

simpler to use the test data in lookup table form directly. However, mathematical representation may be good for model linearization.

Verbrugge, Frisch, and Koch presented a method to estimate battery state-of-charge and state-of-health using an equivalent circuit model and adaptive filtering [14]. State-of-charge is determined by combining estimates based on open circuit voltage measurements and coulomb counting (battery current integration with respect to time). The use of a simplified circuit model enables viable real-time implementation.

Abraham, Kawauchi, and Dees presented a method for modeling the impedance versus voltage characteristics for a specific lithium-ion cell chemistry [15]. Through complex models and parameter fitting to test data, they were able to describe cell impedance as a function of cell voltage. These models would be useful if highly accurate battery modeling is needed, but if simpler models are needed (for faster simulation for instance), a lumped-parameter equivalent circuit model may be more appropriate.

Gao, Liu, and Dougal, presented a more simplified version of lithium ion battery modeling developed specifically for simulation [16]. This method uses a lumped-parameter equivalent circuit model to approximate the high-level dynamics of a lithium ion cell. This approach of using simple resistances, capacitances, and voltage sources yields a model with enough accuracy for energy management system simulation while remaining simple enough for efficient computation.

Cho and Hedrick presented a method for modeling an automotive powertrain including engine and transmission dynamics [17]. The method for modeling the delay between fuel injection and torque output from the engine is easily implementable in a model for a hybrid drivetrain, while the transmission model is not needed for a fixed gear ratio utility vehicle.

## 1.4 Purpose

The objective of this research was twofold: first, develop a dynamic model of a series hybrid vehicle for simulation; second, develop various energy management strategies and implement them in simulation for a given load profile. The purpose of the dynamic model is to accurately simulate all of the components of the hybrid vehicle for energy simulation. The model could also be used in design for component sizing given known load profiles. By simulating various energy management strategies, they can be compared over various criteria.

While many energy management methods exist, a control designer must choose what level of complexity is appropriate for the vehicle of interest. Some methods require extensive modeling, while others require load forecasting. This project sought to develop and implement several energy management strategies of varying complexity so that control designers can weigh the benefits of complexity against the cost of development.

# CHAPTER 2

# SYSTEM DESCRIPTION

The target vehicle for the energy management system is a diesel-electric series hybrid version of the John Deere M-Gator [18]. The commercial non-hybrid version is shown in Figure 2.1. The M-Gator is a multi-purpose utility vehicle is used for material transport, injured personnel transport, towing, and other work related tasks. The hybrid vehicle is being currently developed by the Advanced Energy Systems group of John Deere for a military contract. A standard Gator diesel engine (approximately 15 kW) drives a high voltage alternator (700 VDC) providing power to the DC vehicle bus. Power conditioning units transfer power from the DC vehicle bus to dual AC traction motors. The dual traction motors give the ability to independently apply traction power to each side of the vehicle. Hybridization is achieved by a 270 VDC lithium ion battery pack which is connected to the 700 VDC vehicle bus through a bidirectional converter.

**Figure 2.1: John Deere M-Gator**

The system is defined as having one power producing unit (engine/alternator) and two power producing/consuming units (battery/converter and traction motors). A basic power flow diagram of these components is shown in Figure 2.2. This figure shows that power into the DC bus is defined as positive for the engine/alternator and battery/converter. Conversely, power out of the DC bus is defined as positive for the traction motors. For the sake of brevity, the engine/alternator unit will be denoted EA, and the energy storage system (battery and converter) will be denoted ESS. As shown by the system control boundary, ESS input and output power is considered internal to the system. Figure 2.3 shows system efficiencies and power flows with the associated nomenclature.

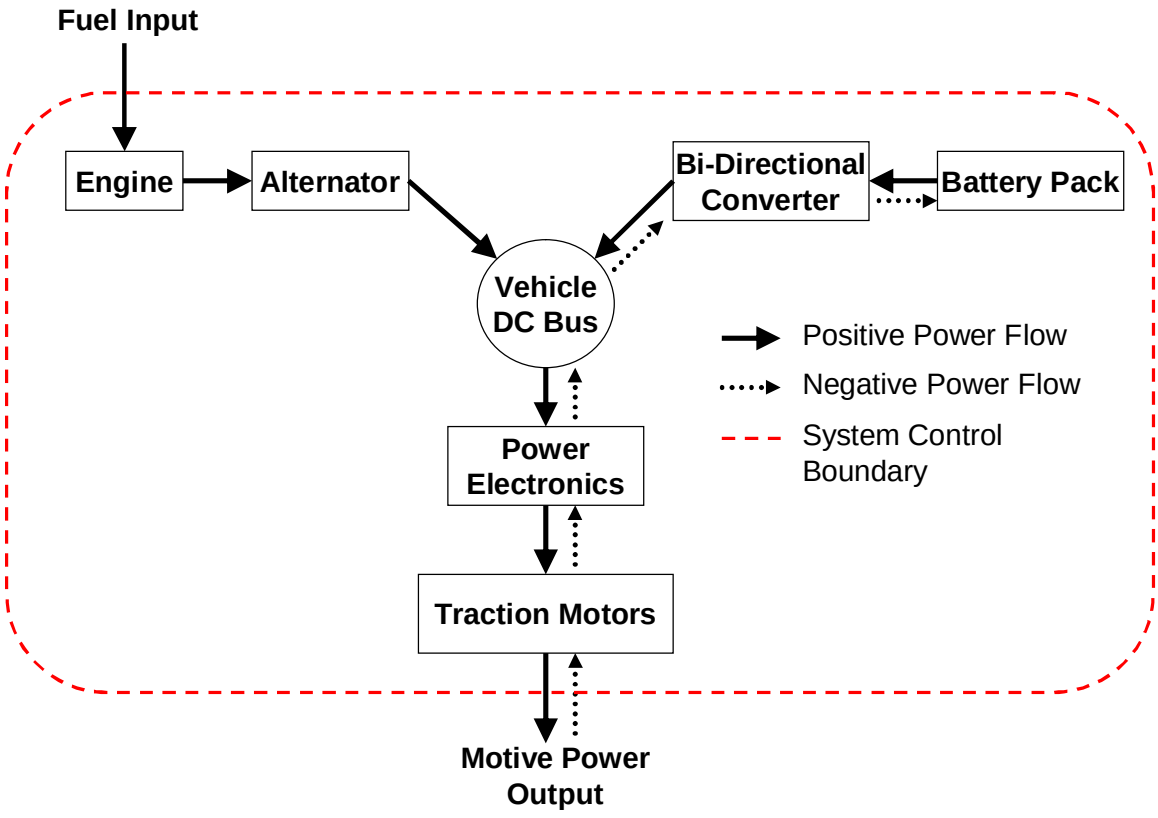


**Figure 2.2: System power conversion elements showing system level power flow sign conventions**

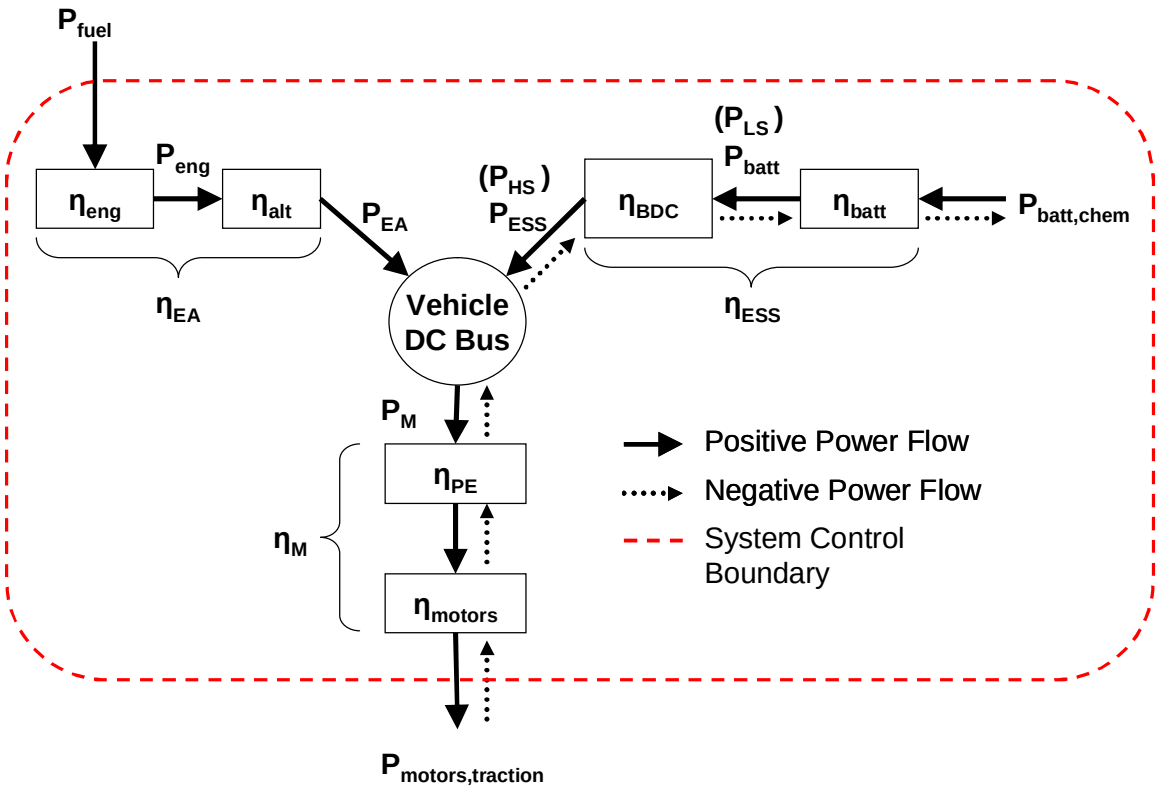


**Figure 2.3: System efficiency and power flow layout and nomenclature (see nomenclature section for details)**

Total cycle efficiency of the system is defined as the ratio of output to input energy. If the battery is not charged to the level where it started, the energy required to charge the battery must be added to the total energy input. Output energy is the integral of motor power, and input energy is proportional to the total amount of fuel consumed including fuel needed to recharge the battery if charge was not sustained. Equation 2.1 shows this efficiency relationship where $LHV_{diesel}$ is the lower heating value of diesel fuel.

$$\eta_{cycle} = \frac{E_{out}}{E_{in}} = \frac{\int T_{motors}\omega_{motors}}{V_{fuel}\rho_{fuel}LHV_{diesel}} \tag{2.1}$$

# CHAPTER 3

# MODEL DEVELOPMENT

A simplified model of each system component was developed in Matlab/Simulink so that simulations could be performed to evaluate the effectiveness of various energy management strategies. All component models were developed using manufacturer and Deere test data for an accurate system representation.

## 3.1 Engine

The goal of the engine model was to dynamically determine the engine speed given a desired engine speed setpoint and a load torque from the alternator. These inputs and output are shown schematically in Figure 3.4. A closed loop speed controller was also developed as shown in Figure 3.5. The fidelity of detailed thermodynamic and mass-transport models was not necessary for the required high level efficiency calculations.


**Figure 3.4: Engine model inputs and output**

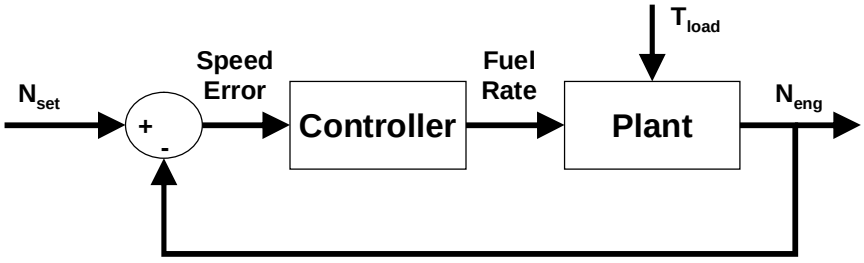


**Figure 3.5: Basic engine model configuration**

The plant of the engine model is comprised of three components: an engine map, engine torque dynamics, and inertial dynamics. A diesel internal combustion engine is a

highly non-linear system, so a critical component of the engine model was the engine map. The engine map was implemented as a two-dimensional lookup table with fuel delivery rate (g/s) and engine speed (rpm) as inputs and steady-state engine torque (N·m) as the output. Since fuel rate is the power input and engine torque and speed define the power output, engine efficiency is implicit in this engine map configuration. The data for this engine map was compiled from Deere test data, but it is available for many engines from their manufacturers.

The test data used to develop this engine map only included data points with the engine under load. However, the engine model must also be able to operate with no load. A natural approach would be to apply a damping torque through the use of a transfer function. During engine operation, this external transfer function would add extra damping losses on top of the losses that are implicit in the engine map. Therefore, the engine map was expanded to include damping in the "engine-off" region in the form of negative torque that is proportional to speed. At all points outside of the normal engine operating range (especially at zero fuel rate), there should be a negative torque output that is proportional to the engine speed. Figure 3.6 shows test data for engine output torque versus engine speed for various constant input fuel rates. As this figure shows, for a constant fuel rate, engine torque output decreases linearly with engine speed. This is due to the engine pumping losses and damping. Therefore, linear regressions were performed for each fuel rate, and the average of all of the slopes was taken to be the constant damping coefficient for the engine. An engine off line was added to the engine map for a fuel rate of zero. Figure 3.6 shows this added damping line for a zero fuel rate, where the slope of this line is the damping coefficient. Figure 3.7 shows the same data, but with the independent axes showing engine map inputs and the contours showing the engine map output (torque). This figure also shows the operating limits.

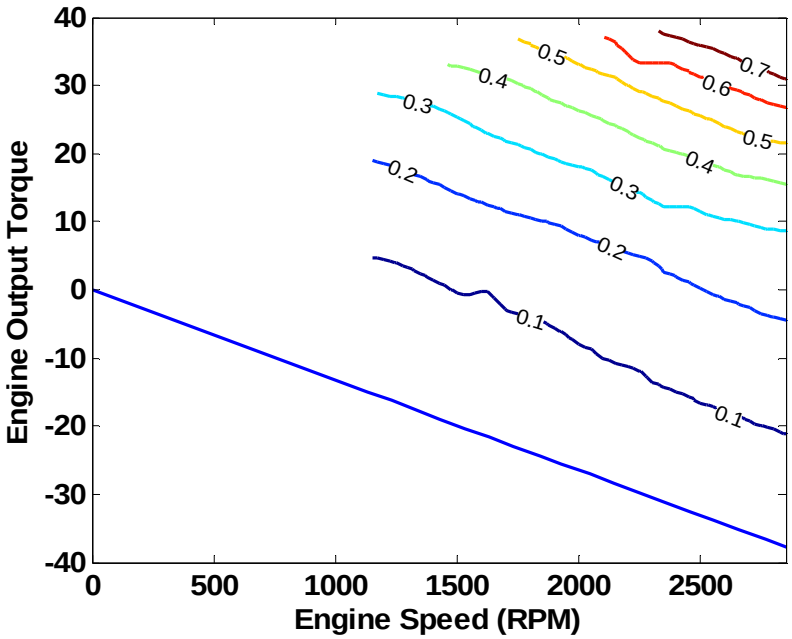


**Figure 3.6: Engine output torque vs. engine speed for various input fuel rates (g/s)**

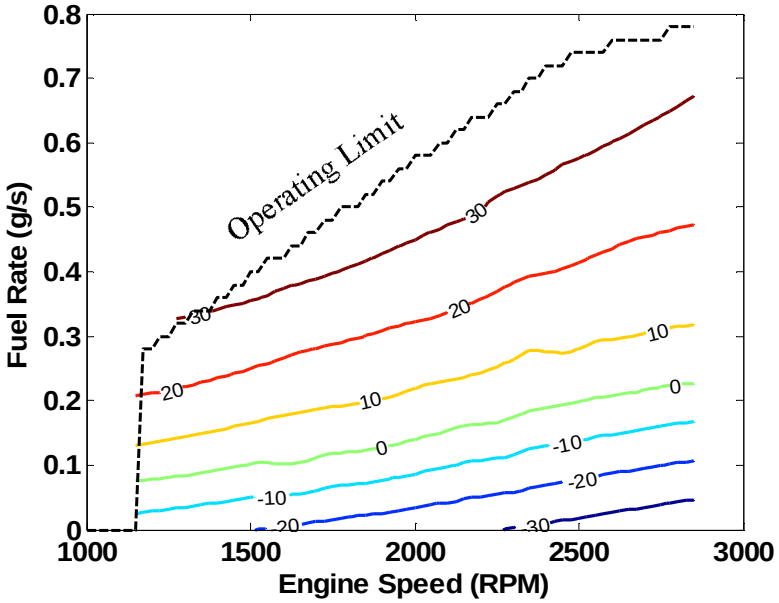


**Figure 3.7: Engine torque output as a function of engine speed and fuel rate also showing operating limits**

The conversion from fuel input to torque output is not instantaneous, so this small delay was modeled as a fast first order filter with unity gain. This usage of a first order lag to represent the dynamics of fuel injection is similar to the method used in [17]. The

implemented continuous-time transfer function for the engine torque dynamics is shown in Equation 3.2, where $T_{eng}$ is the output engine torque (at the crankshaft before the flywheel) and $T_{eng,raw}$ is the raw output of the engine map. The value for the time constant is not experimentally determined, but simply assumed to be a reasonable delay from actual fuel injection to torque measured at the crankshaft.

$$\frac{T_{eng}(s)}{T_{eng,raw}(s)} = \frac{1}{0.01s+1} \tag{3.2}$$

The inertial characteristics of the engine were modeled using Equation 3.3, where $J$ is the engine/flywheel/shaft inertia, $\omega_{eng}$ is the angular velocity of the engine crankshaft, and $T_{load}$ is the load torque (in this case from the alternator). The common dot notation is used to represent the time derivative. Damping is not included in this equation since it is already accounted for in the engine map.

$$J\dot{\omega}_{eng} = T_{eng} - T_{load} \tag{3.3}$$

The rotational inertia was calculated by assuming the majority of the engine inertia is in the flywheel. Therefore, Equation 3.4 was used which yields moment of inertia for a cylinder rotating about its central axis. Dimensions for the flywheel were estimated and the density of steel was used to calculate the mass from the estimated volume.

$$J = \frac{1}{2}mr^2 \tag{3.4}$$

On the actual Deere hybrid Gator, a speed controller is used to adjust fuel lever angle to maintain engine speed. On diesel engines with mechanical fuel injection, the fuel lever controls the amount of fuel being injected into the cylinders which then maps to output torque. To model this speed controller, a PID controller was used with engine speed error as the input and desired fuel rate as the output. Controller gains were tuned

manually to achieve fast response. The fuel rate output is limited to stay within the operable range of the engine using a saturation limit that varies with speed. This dynamic saturation limit was derived from the boundaries of the engine map and it is shown as the black dashed line in Figure 3.7.

To bring the engine from rest to within its operating zone, an engine starter was modeled. The dynamics of an actual starter motor were not of concern; rather, a simple method was needed to bring the engine into its operating zone for simulation. Therefore, the starter was modeled as a simple constant torque input to the engine crankshaft. Hysteresis logic was employed so that the starter torque would only be applied starting at zero engine speed and only stop once engine speed was within the operating zone of the engine map. Using this hysteresis logic allows the engine to fully come to rest if a large external load torque spike "kills" the engine. Starter torque and speed are monitored to calculate starter energy consumed for total system efficiency purposes. Monitoring the energy consumed by the starter shows the energy penalty incurred from frequently cycling the engine on and off, which is critical for energy management strategy development. While starter energy is small, it does make a difference when comparing energy management strategies.

## 3.2 Alternator

The alternator model uses engine speed, battery current, battery voltage, and motor power as inputs, and the single output is load torque on the engine. The inputs and outputs are shown schematically in Figure 3.8.

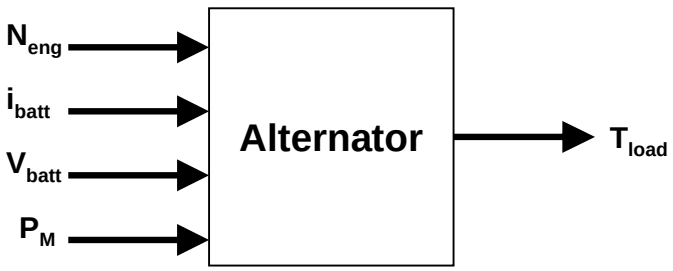


**Figure 3.8: Alternator model inputs and output**

Since the mechanical response of an engine is much slower than the electrical response of an alternator, the alternator was modeled as a static system. Any delays in the alternator were assumed to be dwarfed by engine delays rendering them negligible. Like the engine, the alternator exhibits very nonlinear behavior. Therefore, the alternator was modeled using a two-dimensional lookup table with engine speed and electrical bus power as the inputs and load torque as the output. Engine speed and alternator speed are related by a belt ratio of approximately 1:1.95.

The alternator power output is not directly controlled. Motor power is demanded by the user and the energy management controller commands a current from the bi-directional converter. Due to Kirchoff's current law, the sum of the motor, battery, and alternator current must sum to zero since they are on the same bus. As bus loads change, a voltage regulator on the alternator adjusts field current (and hence power output) to maintain a constant bus voltage. The electrical power output of the alternator is determined from battery and motor power at the bus as shown in Equation 3.5.

$$P_{EA} = P_M - P_{ESS} \tag{3.5}$$

Battery power at the bus is calculated using Equations 3.6 and 3.7, where either the buck (charging) or boost (discharging) efficiency of the bi-directional converter is used (shown as $\eta_{BDC}$), as will be described in detail in Section 3.4.3. Even though $\eta_{BDC}$ is a function of power, a piecewise function must be used for Equation 3.7 since $\eta_{BDC}$ is always less than one.

$$P_{batt} = V_{batt} i_{batt} \tag{3.6}$$

$$P_{ESS} = \begin{cases} \dfrac{P_{batt}}{\eta_{BDC}}, & P_{batt} < 0 \\ P_{batt}\eta_{BDC}, & P_{batt} \geq 0 \end{cases} \tag{3.7}$$

Motor power at the bus is calculated using Equations 3.8 and 3.9, where $\eta_{motors}$ describes electrical losses in the motors and $\eta_{PE}$ describes electrical losses in the power

electronics. These losses are assumed to be constant regardless of the direction of current flow. The actual values for motor efficiencies are not critical for energy management strategy development. Assuming the user gets whatever power they demand (within the limits of the system), the energy management controller has no control over motor power (and hence motor efficiencies). The only controllable efficiencies relate to the engine/alternator and the battery/converter. Since motor and power electronics losses are assumed to be constant, Equation 3.9 is piecewise to account for change in power flow direction.

$$P_{motors,traction} = T_{motors}\omega_{motors} \tag{3.8}$$

$$P_M = \begin{cases} P_{motors,traction}\eta_{motors}\eta_{PE}, & P_{motors,traction} < 0 \\ \dfrac{P_{motors,traction}}{\eta_{motors}\eta_{PE}}, & P_{motors,traction} \geq 0 \end{cases} \tag{3.9}$$

Similar to the engine lookup table, alternator efficiency is implicit since speed and load torque describe mechanical power input and electrical power output is used directly. The alternator lookup table is shown in Figure 3.9. Once again, this test data came from Deere. The relationship between power input and output is governed by the nonlinear alternator efficiency as shown in Equation 3.10 where $\eta_{alt}$ is the alternator efficiency as a function of alternator speed and power output, $V_{bus}$ is the voltage of the DC bus to which the alternator is connected, and $i_{alt}$ is the current output of the alternator.

$$\eta_{alt}\left(N_{alt}, P_{alt}\right) = \frac{V_{bus} i_{alt}}{N_{alt} T_{load}} \tag{3.10}$$

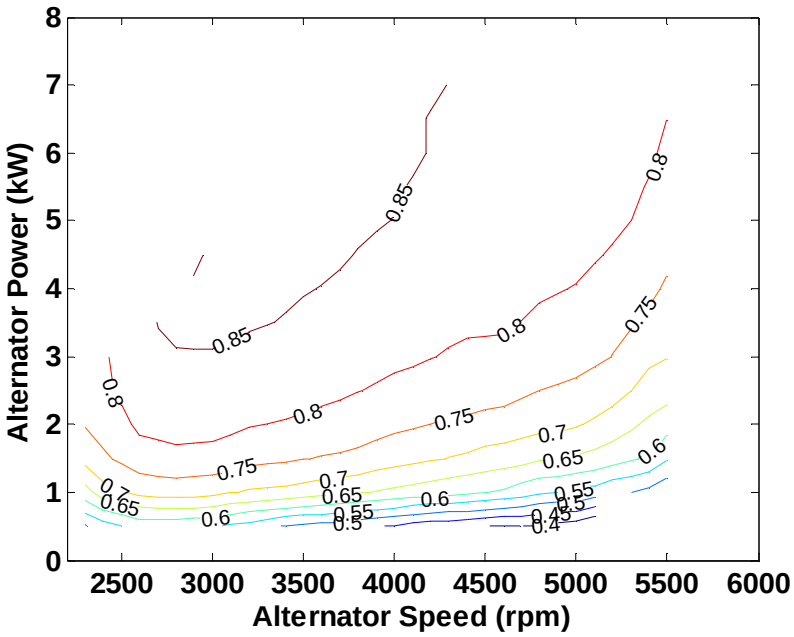


**Figure 3.9: Alternator efficiency as a function of alternator speed and power output**

## 3.3 Engine/Alternator Efficiency

Knowledge of efficiency is critical for energy management, so a lookup table for the combined engine/alternator efficiency was calculated as shown in Figure 3.10. The optimal engine/alternator operating point corresponding to a combined efficiency of 28.33% is shown as a red star, and it occurs at a power output of 5.1 kW and an engine speed of 1780 rpm. The engine speed that maximizes efficiency for a given power output is shown as the blue line. Tighter contours near the efficiency peak are shown in Figure 3.11.

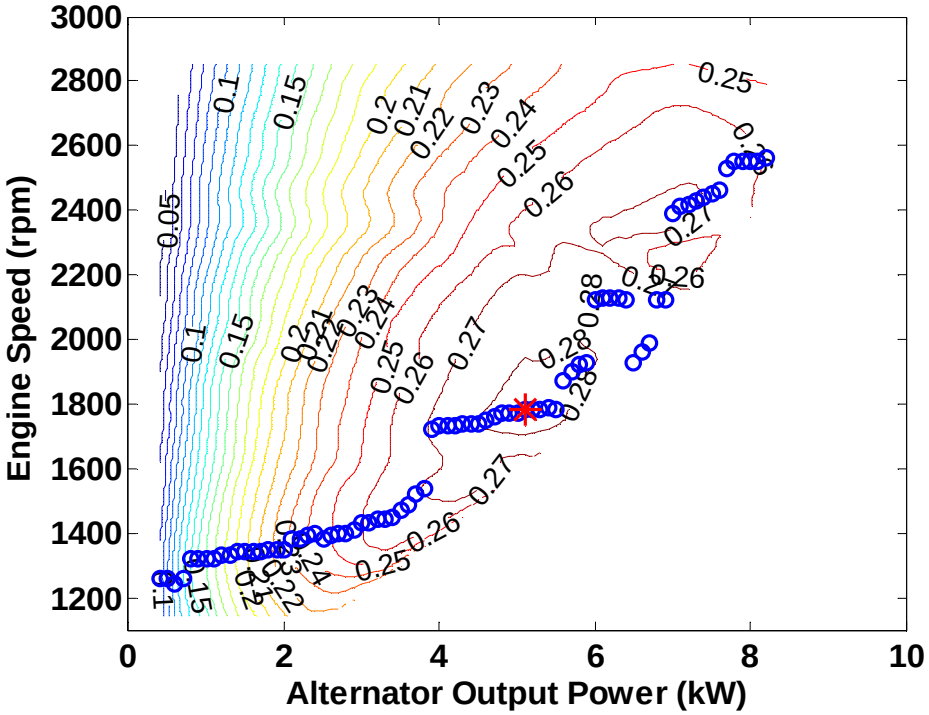


**Figure 3.10: Combined engine/alternator efficiency map with optimal points overlaid**

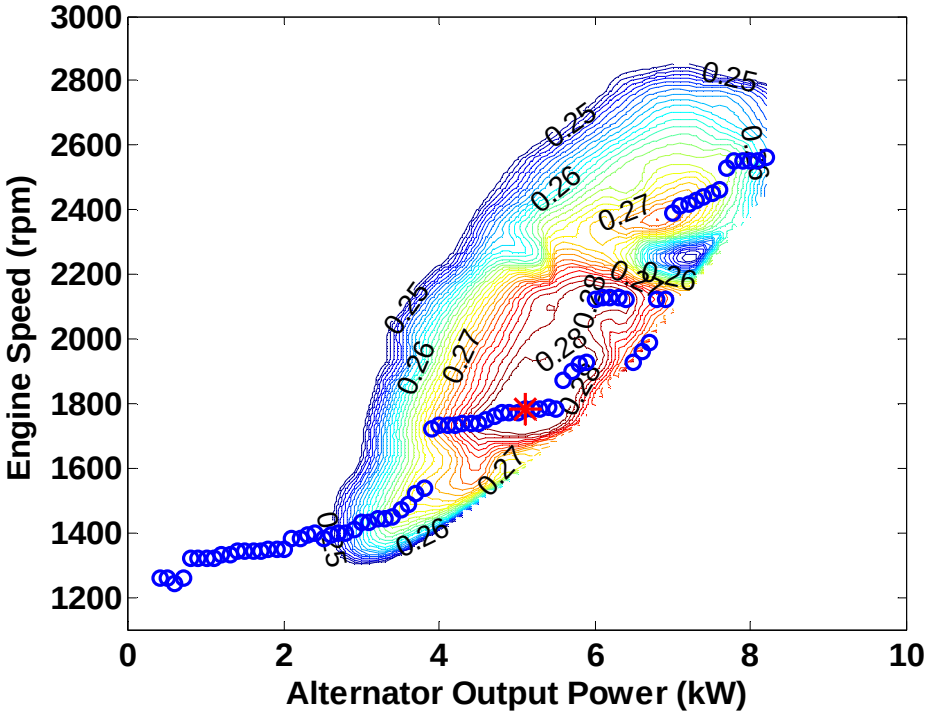


**Figure 3.11: Engine/alternator efficiency map with tighter contours showing overall optimal operating state**

## 3.4 Batteries

A simplified battery model was created that uses battery current as the input and outputs battery voltage, battery state-of-charge, and variable charge and discharge current limits. For simulation only, the controller also requires battery model outputs of open circuit voltage and the voltage of an RC circuit element, denoted $V_1$. The battery input and outputs are shown schematically in Figure 3.12. The battery pack being modeled is 72 Kokam lithium-polymer 11 A·hr cells (model SLPB 55205130H) connected in series yielding a nominal pack voltage of approximately 270V.

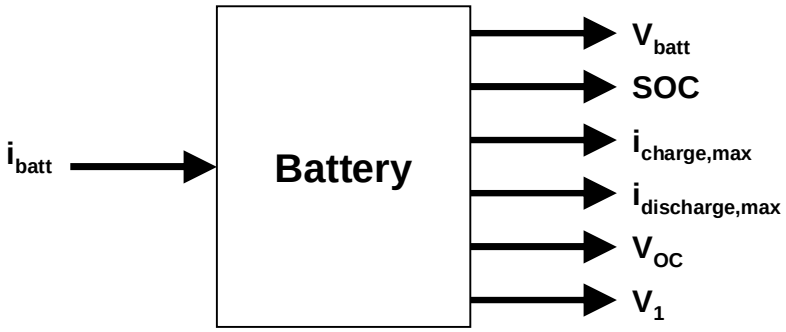


**Figure 3.12: Battery model input and outputs**

For the sake of all battery calculations the passive sign convention was used (current into the battery is positive), which is opposite from the chosen system sign convention (current out of the battery is positive). When considering the hybrid system as a whole, the battery is seen as a power source, but when considering the battery as a single component (for the present purposes of modeling), the battery is seen as a power consumer.

### 3.4.1 Electrical Battery Modeling

Battery SOC is defined as the fraction of charge that is available, as shown in Equation 3.11, where $Q$ is the present charge of the battery and $Q_{max}$ is the maximum rated charge (capacity) of the battery. Battery capacity is measured in ampere-hours

(A·hr) and is a direct representation of the amount of charge a battery contains. One A·hr is equal to 3600 coulombs (C), the SI unit for charge. Capacity is calculated by integrating current from time zero when the battery is at minimum voltage to the present time. In practice the integration is performed from an arbitrary initial time with an initial condition for the battery capacity, *Q*. This process of integrating current to determine capacity is commonly known as "coulomb counting." While coulomb counting can be difficult in practice (it relies on precise current sensing), it works well for simulation.

$$SOC = \frac{Q}{Q_{\max}} = \frac{1}{Q_{\max}} \int_0^t i \cdot dt \tag{3.11}$$

A suitable simplified battery circuit model and corresponding transfer function were chosen to produce a response similar to available manufacturer test data. Pulsed charge scheme data for 1C charge, 2C charge, 1C discharge, and 10C discharge were all used. The manufacturer test data includes current, terminal voltage, temperature, and SOC. Graphs of this data are not shown for proprietary reasons. For each current pulse, voltage immediately jumps when current is applied. The voltage rounds off to a steady linear increase until the current supply is removed. At this point, the voltage drops back to a level higher than before. The at-rest voltage levels correspond to the open circuit voltage of the battery. The immediate jump in voltage is due to internal resistance, and the smoothing effect is due to the capacitive effect of charge double layers in porous battery electrodes.

The circuit chosen to model the battery is shown in Figure 3.13 which is adapted from [16], where the sum of $R_1$ and $R_2$ represent total internal resistance, $C$ represents an effective capacitance due to charge double layers in the porous battery electrodes, $C_b$ is the effective battery capacitance, and $V_{OC}$ is the battery open circuit voltage. The battery terminal voltage for a single battery cell is denoted by $V_{cell}$. The battery pack voltage, $V_{batt}$, can be calculated by multiplying $V_{cell}$ by the number of cells since they are all

arranged in series. It should be noted that by modeling the battery as a capacitor, the capacitance is very large.

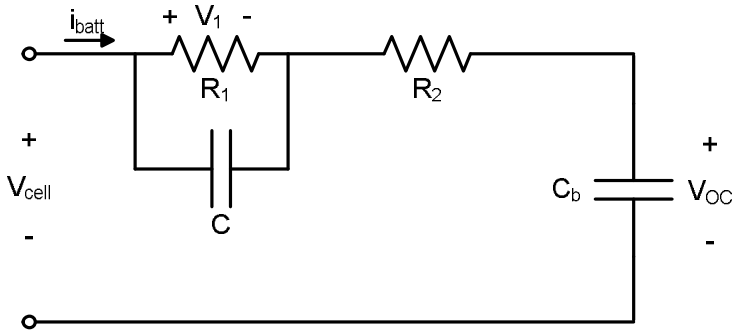


**Figure 3.13: Battery model electrical circuit.**

Batteries are typically charged with a voltage source and the current profile is a result of the voltage input. However, in the Simulink model, desired battery current is the known value and desired battery voltage is unknown. Therefore the battery current is defined as the input and the battery terminal voltage is defined as the output. The circuit equation in the time domain is given by Equations 3.12 and 3.13 and the corresponding continuous-time transfer function in the s-domain is given by Equation 3.14.

$$V_1(t) = i(t)\left(\frac{1}{R_1} + \frac{1}{Z_c}\right)^{-1} = i(t)\frac{R_1 Z_c}{R_1 + Z_c} = i(t)\frac{R_1}{R_1\dfrac{1}{Z_c} + 1} \tag{3.12}$$

$$V_{cell}(t) = V_1(t) + R_2 i(t) + \frac{1}{C_b}\int i(t)dt \tag{3.13}$$

$$\frac{V_{cell}(s)}{i_{batt}(s)} = \frac{(R_1 R_2 C C_b)s^2 + (R_1 C_b + R_2 C_b + R_1 C)s + 1}{(R_1 C C_b)s^2 + C_b s} \tag{3.14}$$

The effective battery capacitance, $C_b$, represents the major charge storing ability of the battery. An ideal capacitor exhibits a current-voltage relationship given by Equation 3.15, where $Q$ is once again the present battery capacity. Substituting Equation

3.11 into Equation 3.15 and rearranging the terms yields Equation 3.16, a linear approximation of the open circuit voltage and SOC relationship.

$$V = \frac{1}{C}\int i \cdot dt = \frac{Q}{C} \tag{3.15}$$

$$V_{OC} = \frac{Q_{\max}}{C_b} SOC \tag{3.16}$$

If the battery were a linear system, there would be a simple linear relationship between SOC and open circuit voltage, defined by the slope $Q_{max}/C_b$. Actual test data for open circuit voltage and battery capacity is available from the battery manufacturer, but the manufacturer-supplied plots cannot be shown for proprietary reasons. However, the open circuit voltage and SOC data was compiled and averaged as shown in Figure 3.14.

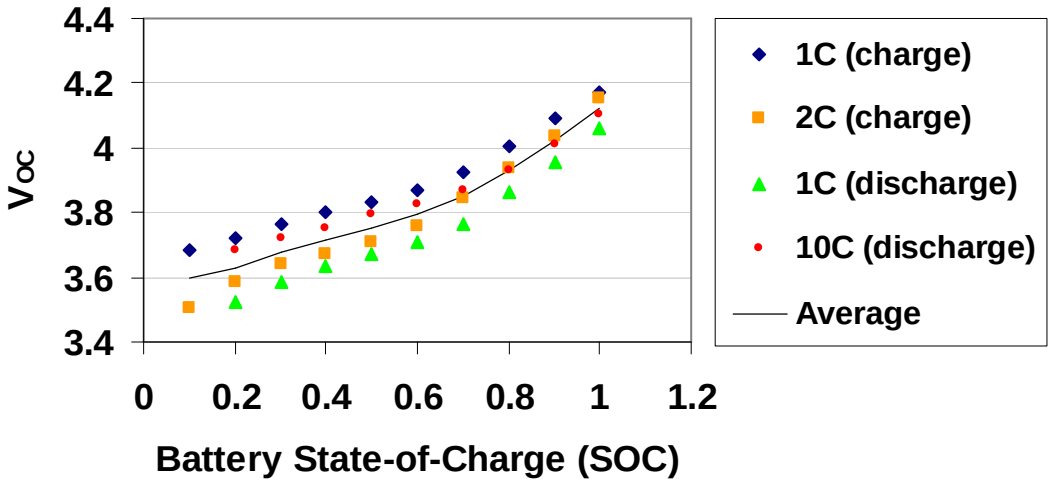


**Figure 3.14: Battery open circuit voltage versus SOC for various charge/discharge rates (data from Deere)**

A linear regression could be performed to calculate a value for $C_b$ ($Q_{max}$ is provided by the manufacturer), but this approximation is not necessary. For the sake of simulation, it is more accurate to use the averaged data from Figure 3.14 as a one-dimensional lookup table to determine open circuit voltage for any given SOC. With $V_{oc}$

defined as a function of SOC, a new circuit model can be used, as shown in Figure 3.15 and a new continuous-time transfer function can be written as shown in Equation 3.17.

$$\frac{(V_{cell} - V_{oc})(s)}{i_{batt}(s)} = \frac{(R_1 R_2 C)s + (R_1 + R_2)}{(R_1 C)s + 1} \tag{3.17}$$

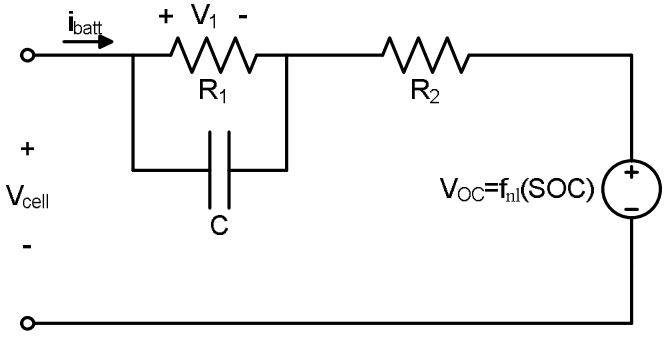


**Figure 3.15: Non-linear battery circuit model**

Once Equation 3.17 is used to calculate ($V_{cell} - V_{OC}$) and Figure 3.14 is used to determine $V_{OC}$, the battery cell terminal voltage can be easily calculated by summing the two voltages to solve for $V_{cell}$. The battery pack voltage is then calculated by multiplying by the number of battery cells since they are all in series, as shown in Equation 3.18.

$$V_{batt} = N_{cells} V_{cell} \tag{3.18}$$

The voltage model was verified for the 2C pulse charge scheme from manufacturer data as shown in Figure 3.16. The modeled voltage is slightly higher for this pulse charge scheme mainly because of errors in the SOC lookup table (hence the higher at-rest voltages). As seen in Figure 3.14, the averaged data used for the lookup table lies above the 2C test data for open circuit voltage, which corresponds with the positive modeling errors specifically for the 2C charge scheme. For other charge pulse schemes such as the 1C charge scheme, the model data would fall below the test data.

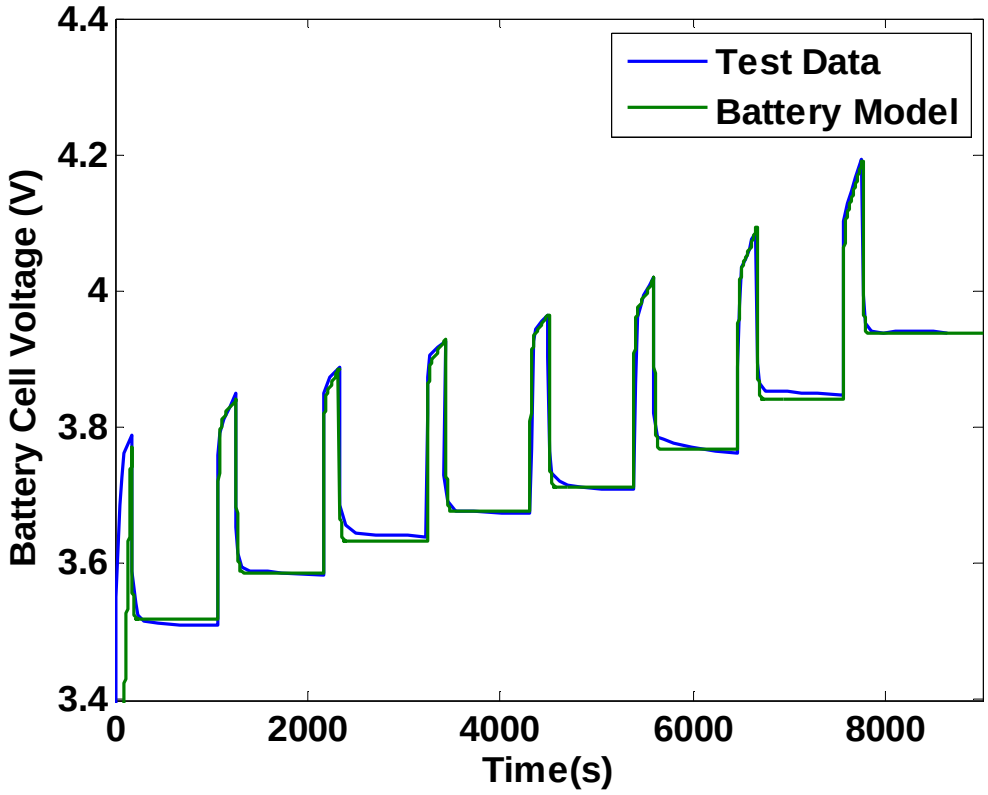


**Figure 3.16: 2C (22 A) charge pulse scheme battery voltage model verification**

### 3.4.2 Thermal Battery Modeling

In addition to an electrical battery model, a thermal battery model is needed to ensure temperature limits are not exceeded and to determine charge and discharge current limits. To create a thermal model, a heat balance was formed as shown in Equation 3.19, where $m$ is the mass of the battery, $c_p$ is the specific heat at constant pressure, $T$ is the temperature of the battery, $R$ is the total internal resistance of the battery, $h$ is the equivalent heat transfer coefficient for both conduction and convection, $A$ is the surface area of the battery, and $T_a$ is the ambient temperature. Radiative heat transfer is assumed to be negligible so that a linear model can be employed.

$$mc_p\dot{T} = Ri_{batt}^{\ 2} - hA(T - T_a) \tag{3.19}$$

Assuming $T_a$ to be constant, $(T\text{-}T_a)$ can become $T_{rel}$, or the relative temperature. Since $T_a$ is constant, the rate of change of $T$ is the same as the rate of change of $T_{rel}$. Substituting $T_{rel}$ into Equation 3.19 and rearranging the terms yields Equation 3.20.

$$\frac{mc_p}{hA}\dot{T}_{rel} + T_{rel} = \frac{R}{mc_p} i_{batt}^{\ 2} \tag{3.20}$$

Equation 3.20 describes a first order dynamic system with $i_{batt}^{\ 2}$ as the input, $T_{rel}$ as the output, a time constant of $\tau = \frac{mc_p}{hA}$, and a steady state gain constant of $k = \frac{R}{mc_p}$. It would be difficult to attempt estimates of all of the individual parameters that constitute the time constant and gain values, so it is more appropriate to use system identification methods to fit a first order system model to the temperature data supplied by the battery manufacturer. The temperature and current data were digitized and imported into Matlab, where system identification tools were used to estimate values for $\tau$ and $k$. The resulting first-order continuous-time transfer function is shown in Equation 3.21. As evidenced by the large time constant, the thermal response of the battery is very slow.

$$\frac{T_{rel}(s)}{i_{batt}^{\ 2}(s)} = \frac{k}{\tau s + 1} = \frac{0.0052}{628.57s + 1} \tag{3.21}$$

This transfer function would of course change for different cooling environments for the battery (varying $hA$), but this configuration will be used for simulation.

The modeled temperature of the battery pack is used to limit current into and out of the batteries. A lookup table is used to determine charge and discharge current limits for a given battery pack temperature. For temperatures within the normal operating thermal range stated by the battery manufacturer, current limits are set to the manufacturer's limits. For temperatures nearing the safe limits of the battery, the current limit is gradually decreased to zero. The charge and discharge limit lookup table plots are shown in Figure 3.17 and Figure 3.18.

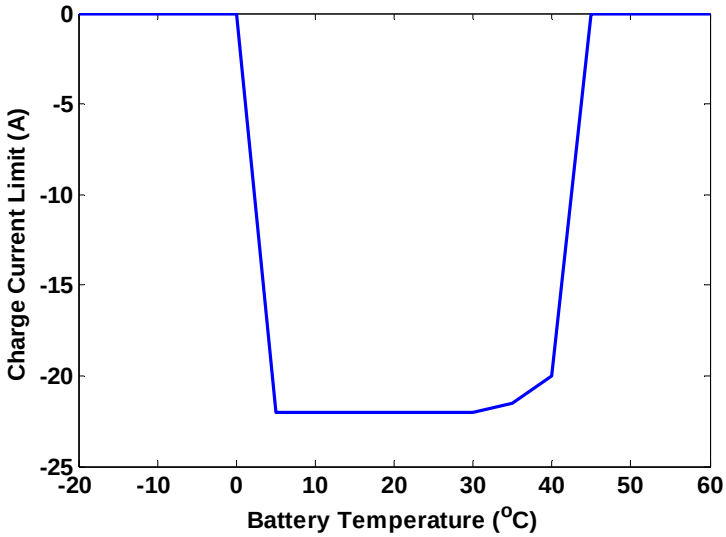


**Figure 3.17: Charge current limits as a function of battery temperature**

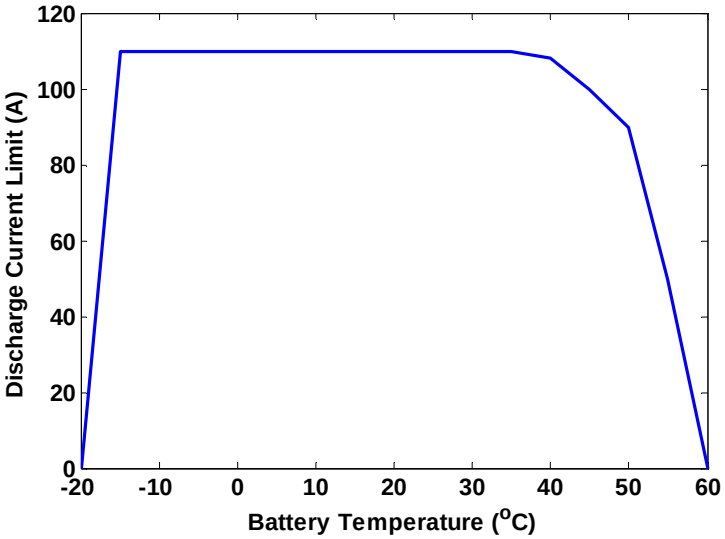


**Figure 3.18: Discharge current limit as a function of temperature**

### 3.4.3 Energy Storage System Related Efficiencies

The energy storage system has two main efficiencies: battery efficiency and bi-directional converter efficiency. For batteries, both electron and ion transport are the dominant loss mechanisms. These losses show up in the equivalent resistance of the battery as ohmic losses. Battery resistance varies mainly as a function of temperature and SOC, and without temperature control, temperature is a function of battery current.

Battery resistance for the lithium polymer batteries being modeled is shown in Figure 3.19. The data was compiled from manufacturer test data. While resistance is variable, it was assumed to be a constant value for simulation and efficiency calculations.

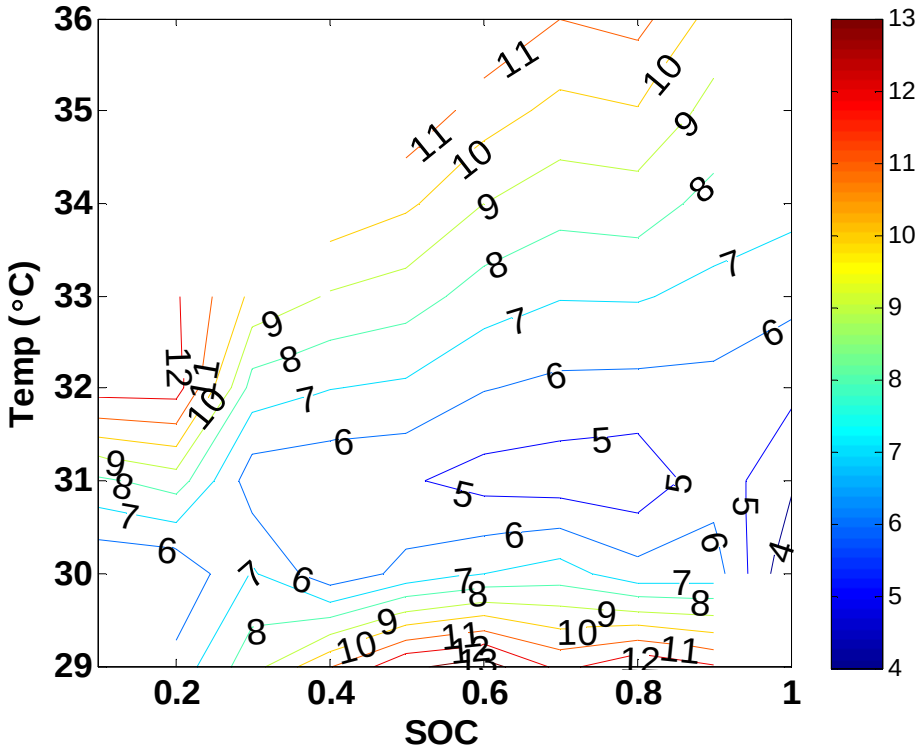


**Figure 3.19: Battery cell resistance (mΩ) as a function of temperature and SOC**

At the root level, battery efficiency is a function of current and SOC (which is consequently a function of the integral of current). The derivation of the efficiency equation can be approached in multiple ways. The simplest way is to begin with the ratio of output and input power as shown in Equation 3.22. This can be verified by examining steady state power loss as shown in Equation 3.23.

$$\eta_{batt,charge} = \frac{P_{out}}{P_{in}} = \frac{V_{oc} i_{batt}}{V_{cell} i_{batt}} = \frac{V_{oc}}{V_{cell}}$$

$$\eta_{batt,discharge} = \frac{P_{out}}{P_{in}} = \frac{V_{cell} i_{batt}}{V_{oc} i_{batt}} = \frac{V_{cell}}{V_{oc}} \quad (3.22)$$

$$\eta_{batt,charge} = 1 - \frac{P_{loss}}{P_{in}} = 1 - \frac{i_{batt}{}^2 (R_1 + R_2)}{V_{cell} i_{batt}} = \frac{V_{cell} - i_{batt}(R_1 + R_2)}{V_{cell}} = \frac{V_{oc}}{V_{cell}}$$

$$\eta_{batt,discharge} = 1 - \frac{P_{loss}}{P_{in}} = 1 - \frac{i_{batt}{}^2 (R_1 + R_2)}{V_{oc} i_{batt}} = \frac{V_{oc} - i_{batt}(R_1 + R_2)}{V_{oc}} = \frac{V_{cell}}{V_{oc}} \quad (3.23)$$

Since $V_{cell}$ is a function of current and $V_{oc}$, the battery efficiency is a function of current and SOC, as shown in Equation 3.24, where $f_{nl}(SOC)$ represents a non-linear function of SOC.

$$\eta_{batt,charge} = \frac{V_{oc}}{V_{cell}} = \frac{V_{oc}}{V_{oc} + i_{batt}(R_1 + R_2)} = \frac{f_{nl}(SOC)}{f_{nl}(SOC) + i_{batt}(R_1 + R_2)}$$

$$\eta_{batt,discharge} = \frac{V_{cell}}{V_{oc}} = \frac{V_{oc} + i_{batt}(R_1 + R_2)}{V_{oc}} = \frac{f_{nl}(SOC) + i_{batt}(R_1 + R_2)}{f_{nl}(SOC)} \quad (3.24)$$

Since the efficiencies of the bi-directional converter, engine, alternator, and motor are all modeled as functions of power, it is useful to also examine battery efficiency as a function of power. Multiple simulations were run to obtain instantaneous efficiency curves as functions of cell terminal power and SOC, as shown in Figure 3.20.

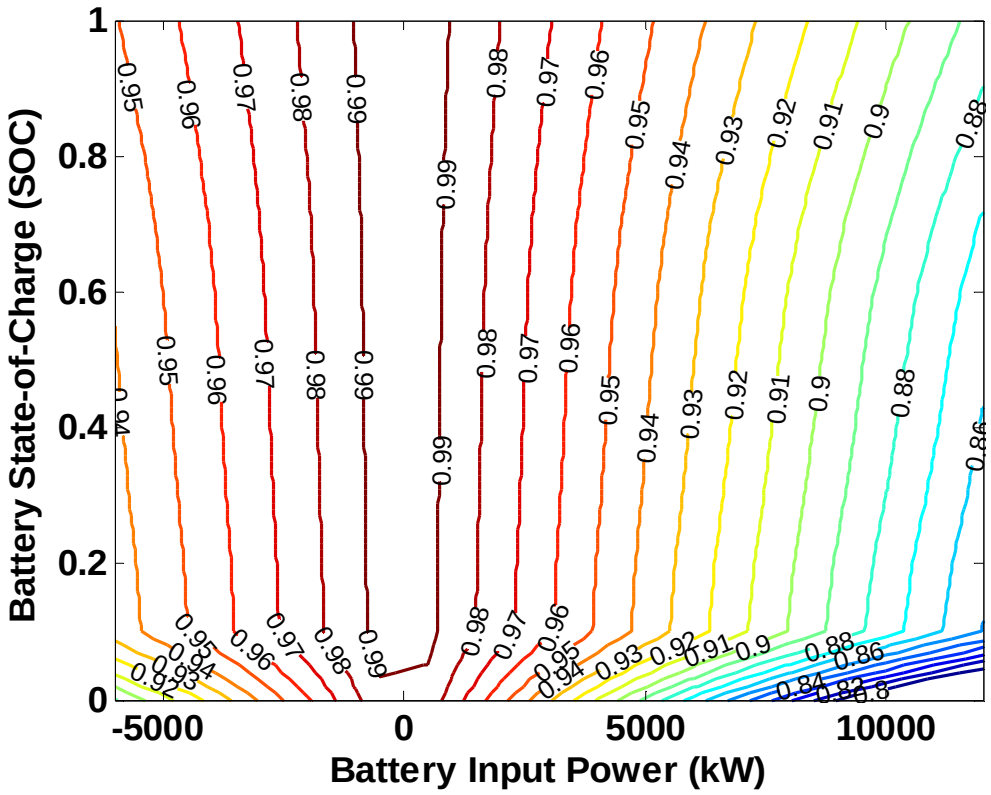


**Figure 3.20: Battery instantaneous efficiency**

The second energy storage system efficiency is due to losses in the bi-directional converter. The dynamics of the actual converter were neglected due to the small time constants of power electronics systems, so the converter was modeled as simply an efficiency loss gain that is a function of power. John Deere performed multiple tests on the bi-directional converter being used on the hybrid M-Gator utility vehicle. Output powers were recorded over a range of input powers for both buck (battery charge) and boost (battery discharge) operations. There is a non-linear relationship between power and efficiency, however, a highly linear relationship exists between input and output power as shown in Figure 3.21 for boost operation and Figure 3.22 for buck operation. In the model, it is simpler and more computationally efficient to use a slope and offset to convert from input power to output power than using a lookup table to determine efficiency to use as a gain on input power.

For simulation the linear relationship is useful, but for energy management strategy development, the actual power-to-efficiency relationship is more useful. The linear power-to-power curve fit coefficients were used to develop a non-linear power-to-efficiency curve fit as shown in Equations 3.25-3.27.

$$P_{LS} = m_x P_{HS} + b_x \tag{3.25}$$

$$\eta_{BDC,charge} = \frac{P_{LS}}{P_{HS}} = m_c + \frac{b_c}{P_{HS}} \tag{3.26}$$

$$\eta_{BDC,discharge} = \frac{P_{HS}}{P_{LS}} = \frac{P_{HS}}{m_d P_{HS} + b_d} \tag{3.27}$$

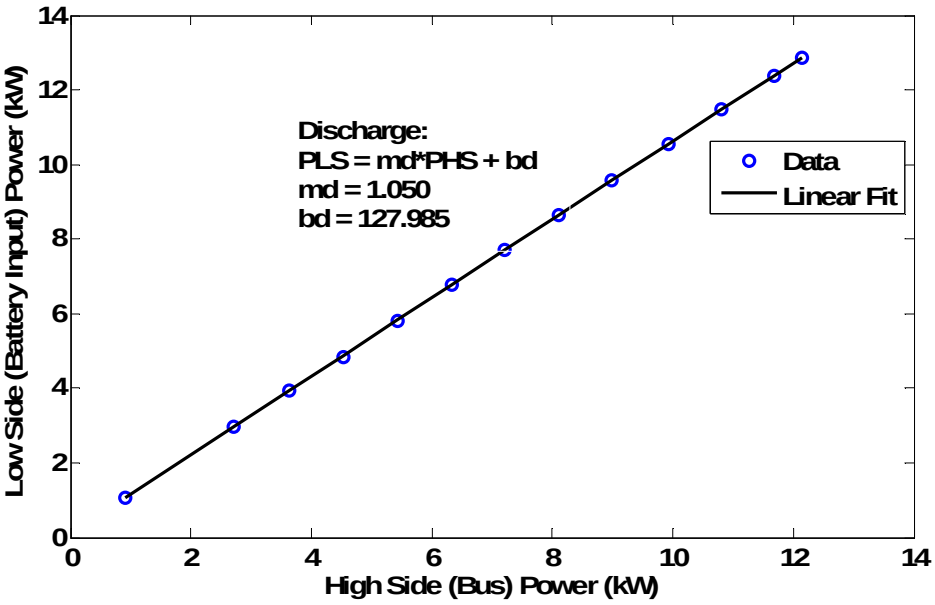


**Figure 3.21: Bi-directional Converter linear discharge power relationship**

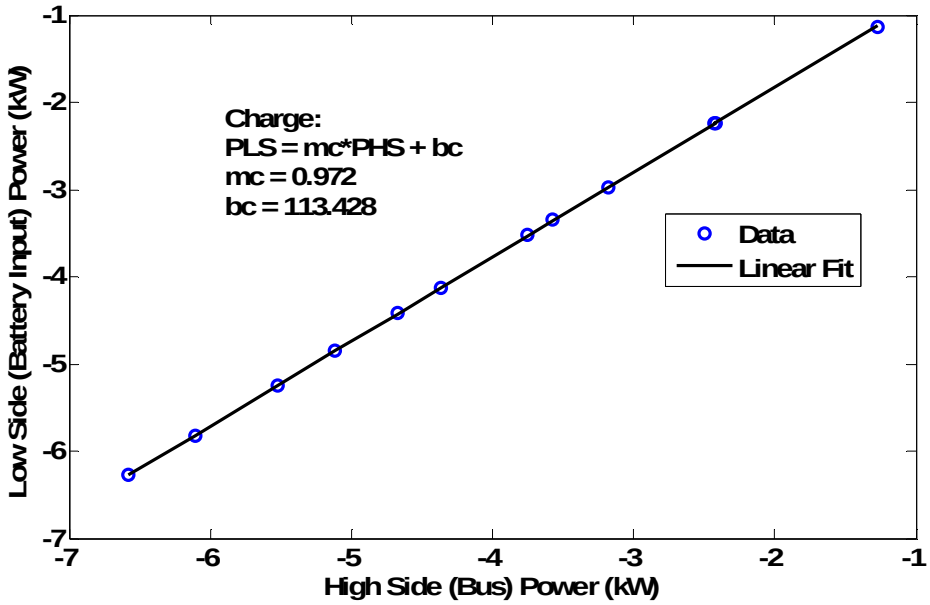


**Figure 3.22: Bi-directional Converter linear charge power relationship**

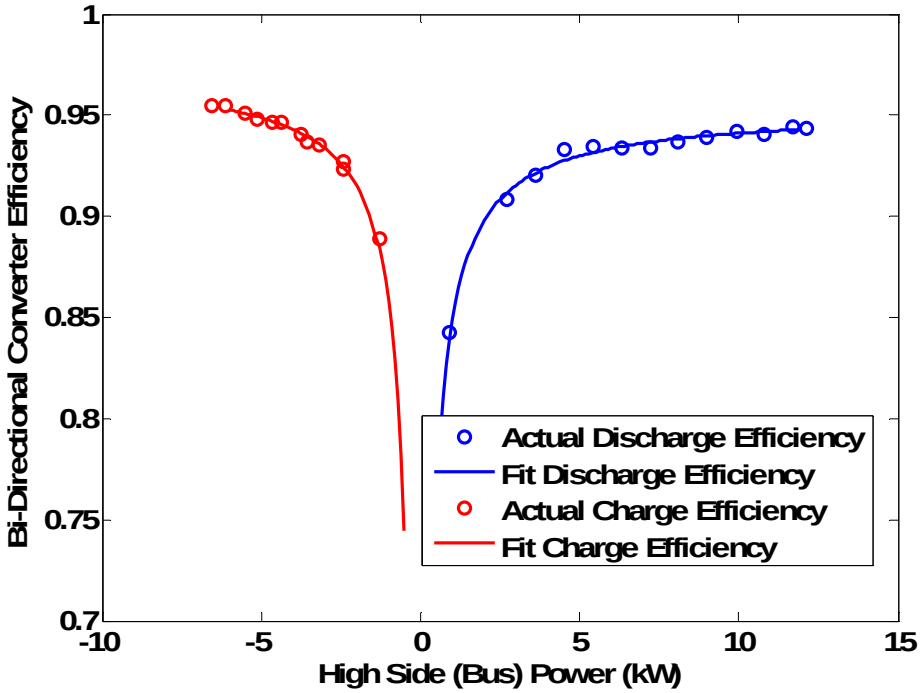


**Figure 3.23: Bi-directional converter efficiencies**

With battery efficiency characterized as a function of terminal (low side) power and SOC, $\eta_{batt}=f(P_{batt}, SOC)$, and bidirectional converter efficiency characterized as a

function of bus power, $\eta_{BDC}=f(P_{BDC})$, the two efficiencies can be combined to yield an overall energy storage system efficiency as a function of bus power. The battery efficiency is transformed to a function of bus power simply by performing the linear transformations given by Equation 3.28, where $m_x$ and $b_x$ are the slope and offset coefficients given in Figure 3.21 and Figure 3.22. The energy storage system efficiency is given by Equation 3.29. Since battery efficiency decreases with increasing power magnitude and BDC efficiency increases with increasing power magnitude, it is apparent that maxima will exist for overall efficiency. A contour plot for overall efficiencies is shown in Figure 3.24 where charge and discharge efficiency maxima can be observed. While efficiency increases only slightly with increasing SOC, power level has a large impact. Operating at high SOC range around 3.5 kW for charge and discharge would be optimal use of this energy storage system as modeled.

$$P_{ESS} = \frac{P_{batt} - b_x}{m_x} \tag{3.28}$$

$$\eta_{ESS}(P_{ESS}) = \eta_{batt}(P_{ESS})\eta_{BDC}(P_{ESS}) \tag{3.29}$$

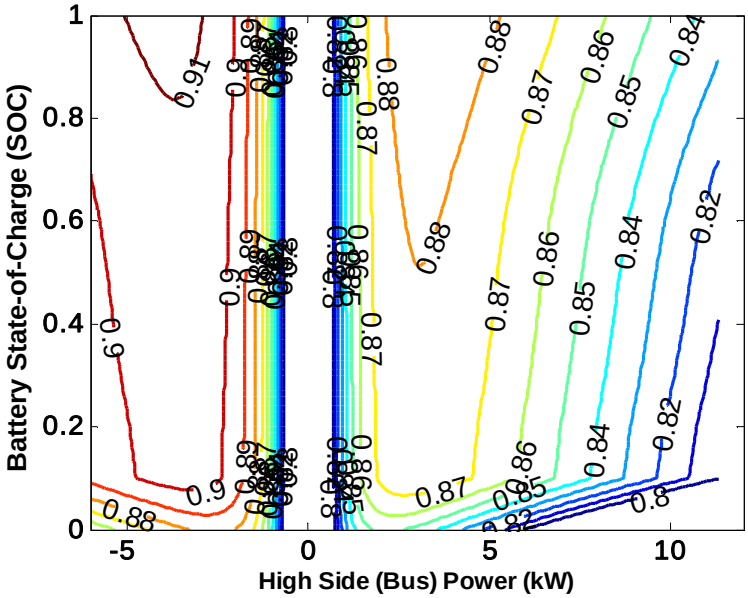


**Figure 3.24: Total energy storage system instantaneous efficiency**

# CHAPTER 4

# ENERGY MANAGEMENT CONTROLLER DEVELOPMENT

## 4.1 Controller Input/Output Definitions

With modeling complete, the next step was to define inputs and outputs for an energy management controller. For the actual controller, the only necessary inputs are SOC, battery charge and discharge current limits, actual engine speed, actual motor torque and speed, and user requested motor torque. For simulation only, the modeled battery voltages $V_{OC}$ and $V_1$ are also needed as controller inputs. These two voltages are needed to avoid an algebraic loop in simulation which will be explained later. Necessary controller outputs are engine speed setpoint, battery current command, and motor torque command. For the model only, the controller also outputs the calculated power draw of the motors for use by the alternator model. In the actual controller this output will not be necessary. The controller inputs and outputs are shown schematically in Figure 4.25, where the red I/O are for simulation purposes only.

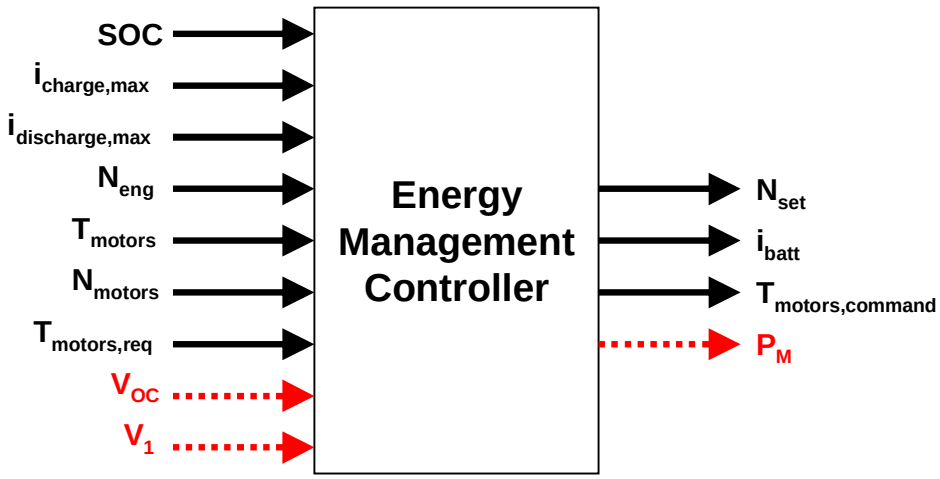


**Figure 4.25: Energy management controller inputs and ouputs (red denotes for simulation only)**

The objective of any hybrid energy management controller is to determine how power should be routed through the available power consumers and producers to achieve a stated goal or set of goals. For this project, the stated goals in order of importance are:

1) Control power flow between the vehicle power producers and consumers so that the vehicle is operable;
2) Improve roundtrip fuel efficiency while at least maintaining standard vehicle performance;
3) Improve vehicle performance (defined by acceleration and hill climbing/towing ability both with and without load);
4) Have long battery life.

Since the first requirement is quite basic, the majority of this research is devoted to the second requirement: improving fuel efficiency.

## 4.2 Engine Speed Calculation

When the engine is turned on, the desired engine speed is set immediately, but the desired power output of the alternator delayed. If a large load torque was placed on the engine immediately, it would either respond very slowly or bog down completely, depending on the magnitude of the load torque from the alternator. Therefore, the engine is allowed to come up to speed before any load torque is applied. Once the engine is up to speed, the alternator power is increased according to a slew rate. This rate limiting is common to prevent step changes in a power system. While all of this delays alternator power output by a couple of seconds, it allows fast and consistent engine response. This operation is shown in Figure 4.26.

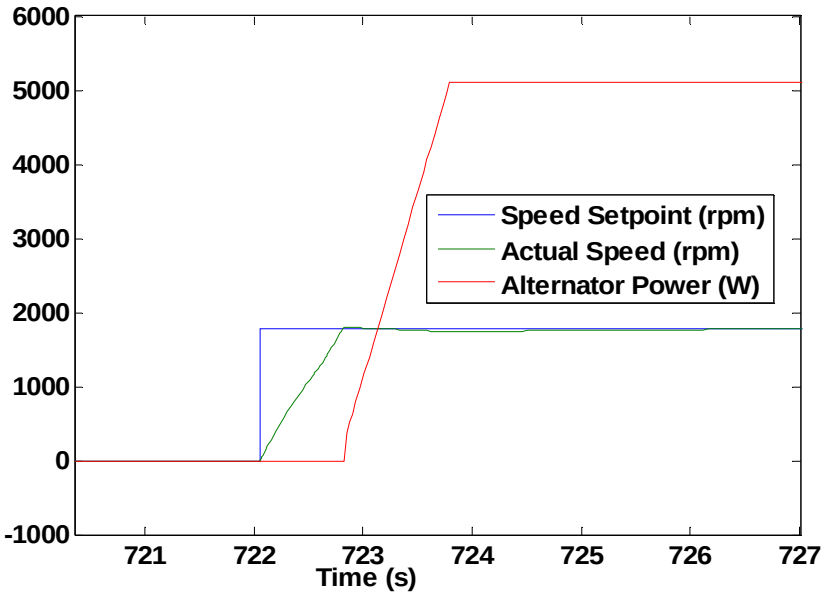


**Figure 4.26: Engine startup and alternator power draw startup**

## 4.3 Battery Current Calculation

To calculate the desired battery current, desired alternator power is subtracted from desired motor power to yield a desired battery power. This high side battery power is converted to low side power using the linear converter efficiency functions discussed previously. In the actual controller, this value could simply be divided by measured battery voltage to yield the desired battery current. However, in the simulation the battery voltage transfer function has direct feedthrough of the battery current. Therefore, attempting to use battery voltage to calculate battery current forms an algebraic loop and the simulation cannot function. To avoid this problem, a power balance is used to calculate battery current. The power balance is set up to correspond to the battery circuit shown in Figure 3.13 as shown in Equation 4.30.

$$V_{batt} i_{batt} = R_2 i_{batt}^{\ 2} + V_1 i_{R1} + V_1 i_c + V_{OC} i_{batt} \tag{4.30}$$

Due to Kirchoff's current law, $i_{R1} + i_C = i_{batt}$. Also, since the left-hand side of Equation 4.30 is the same as desired battery power, it can be rewritten as Equation 4.31, where $P_{batt,des}$ is the desired battery power.

$$0 = R_2 i_{batt}^{\ 2} + (V_1 + V_{OC}) i_{batt} - P_{batt,des} \tag{4.31}$$

Battery current can be solved using the quadratic formula as shown in Equation 4.32. Using the negative root would only result in a negative current (which would not correspond to a positive power). The positive root yields currents that correctly match the sign of the power.

$$i_{batt}^{+} = \frac{-(V_1 + V_{OC}) + \sqrt{(V_1 + V_{OC})^2 + 4R_2 P_{batt,des}}}{2R_2} \tag{4.32}$$

It should be noted that in order for this solution to be real, the discriminant must remain positive. Therefore, the desired battery power must obey the inequality shown in Equation 4.33.

$$P_{batt,des} \geq \frac{-(V_1 + V_{OC})^2}{4R_2} \tag{4.33}$$

For the target vehicle the worst case scenario would correspond to the minimum expected values of $V_1$ and $V_{OC}$ which are zero and 2.7 V, respectively. With an estimated average value for $R_2$ of 5.5 mΩ, the minimum allowable battery power is approximately -331 W per cell. This corresponds to a maximum battery charging current of approximately 245 A, which is more than an order of magnitude larger than the manufacturer specified charge current limit of 22 A. The physical limits of the system occur before the mathematical limits of the calculation.

## 4.4 Motor Torque Command Calculation

For the most part, the motor torque command will be the same as the motor torque request from the user. However, the motor torque command must be limited for cases when the battery has reached a maximum SOC. When maximum SOC is reached, current can be pulled out of the battery, but none can be allowed in. Therefore, no regenerative motor torque is allowed when maximum SOC is reached.

## 4.5 System Model Results Verification

### 4.5.1 Engine/Alternator Results

Looking at just the first three seconds shown in Figure 4.27 reveals the details of the engine/alternator operation. Initially the engine starter comes on providing a constant torque of 40 N-m (red line, $T_{starter}$). As the engine speeds up it exerts a growing negative torque due to friction (aqua line, $T_{eng}$). Once the engine speed reaches into the zone of operation, the starter turns off and fuel delivery to the engine begins (yellow line, *Fuel Rate*). Initially, fuel delivery is saturated, meaning the engine is producing as much torque as possible for its given speed (black and yellow lines). Also, load torque begins to grow corresponding to the rate limited desired alternator power (magenta line, $T_{load}$). The difference between the engine torque and load torque defines the acceleration of the engine mass. Once the engine speed reaches the setpoint, fuel delivery gradually settles to a steady state value. The settling characteristics of the engine speed and fuel rate are defined by the PID engine speed controller gains. Any steady state error (as observed between 1 and 2 seconds) is eliminated by the integral control of the speed controller. Within two seconds, all states have settled to their steady state values.

Figure 4.28 shows transient operation when the engine turns off. As expected, all torques and fuel delivery go to zero, and the engine exerts a frictional torque that is proportional to engine speed until the engine comes to rest.

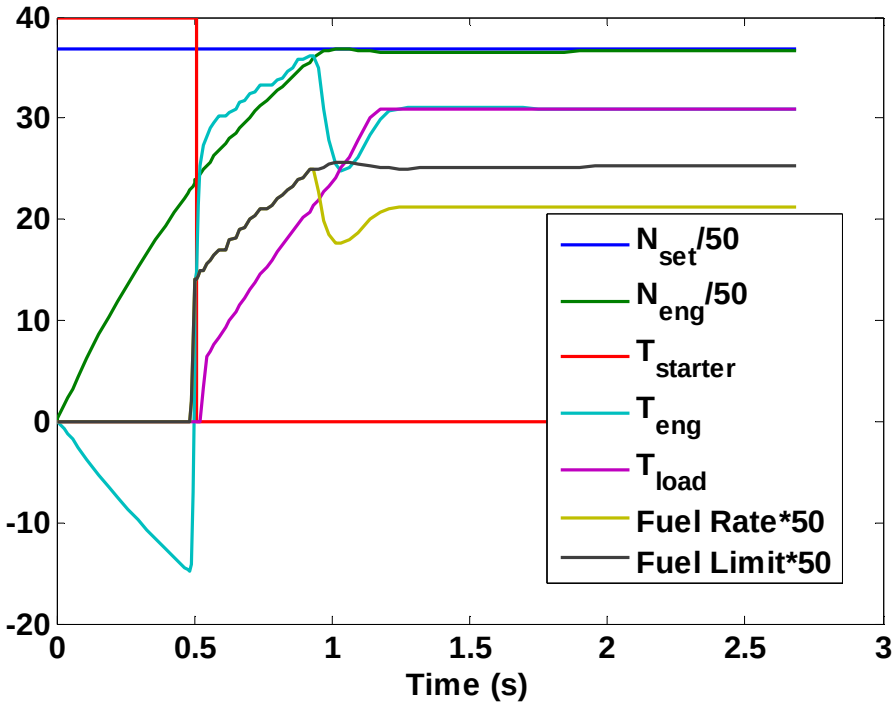


**Figure 4.27: Various engine related variables showing engine startup.**

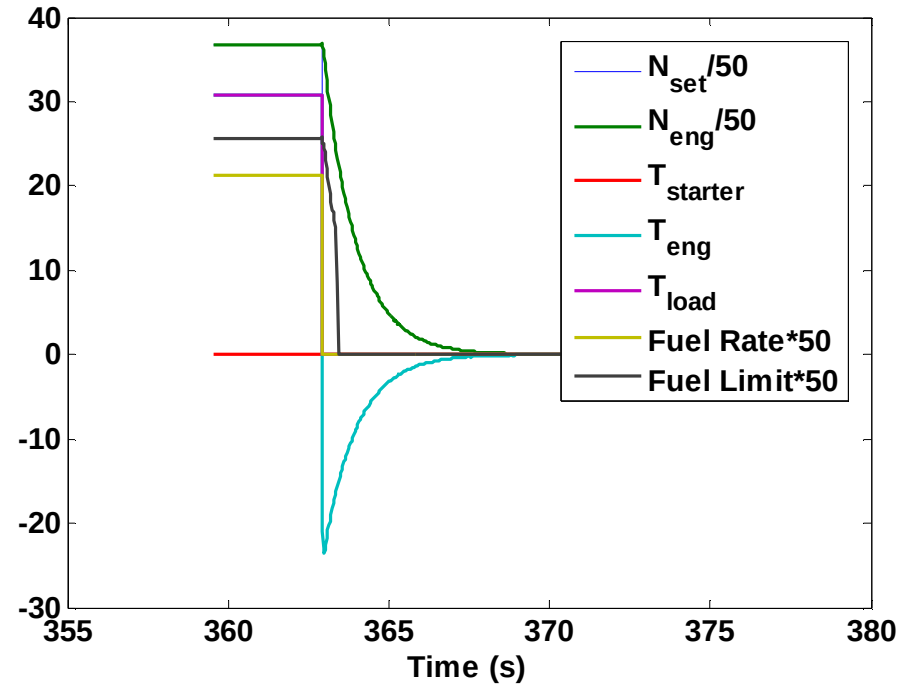


**Figure 4.28: Various engine related variables showing engine shutdown.**

### 4.5.2 Battery Results

Figure 4.29 shows battery SOC, current, and single cell voltage over an hour long simulation with several engine on/off cycles. With a constant motor power demand of 2 kW, SOC varies between a minimum and maximum with a constant slope. For a variable motor power demand, this slope would not be constant. When the engine is on, battery current falls to a negative value, meaning the batteries are being charged. As battery voltage rises, the magnitude of battery current falls to maintain a constant battery power. When the engine is off, battery current is positive, meaning all motor power is being supplied by the battery pack.

Figure 4.30 shows a close-up of battery cell voltage on a step change in battery current sign. The voltage performs similarly to the test data used to create the battery model as discussed previously. At the step change, voltage jumps due to the internal resistance of the battery ($R_1$ and $R_2$). This jump is smoothed due to the capacitance from porous electrodes ($C$). The voltage then linearly increases due to the overall effective battery capacitance ($C_b$) which was modeled as a lookup table as previously discussed.

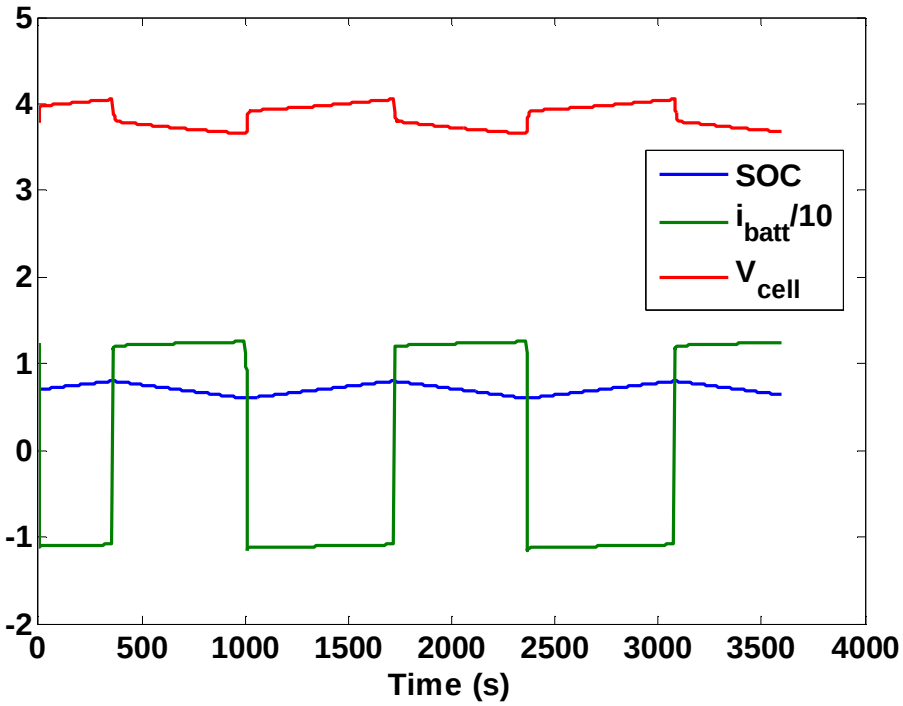


**Figure 4.29: SOC, battery current, and single cell battery voltage.**

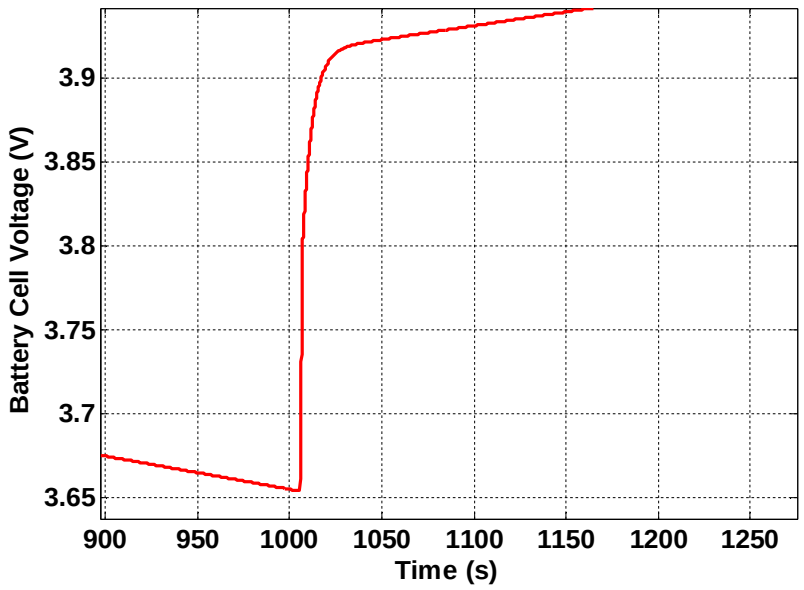


**Figure 4.30: Battery cell voltage on step change in battery current.**

## 4.6 Simulation Setup for Energy Management Strategy Evaluation

A time-varying load profile was developed to use as the desired motor power as shown in Figure 4.31. This load profile was chosen to have a range of powers, including regenerative braking (negative motor power). The simulation was chosen to run for 2 hours, which is sufficient time to achieve a good average efficiency. For thermostat and other energy management strategies that require a SOC operation bandwidth, a lower limit of 60% and an upper limit of 80% SOC were chosen. At the end of the 2 hours, the batteries are charged back to their initial SOC using the engine/alternator at its most efficient operating point to ensure a charge sustaining system. By enforcing a charge sustaining system, all control strategies can be equally compared.

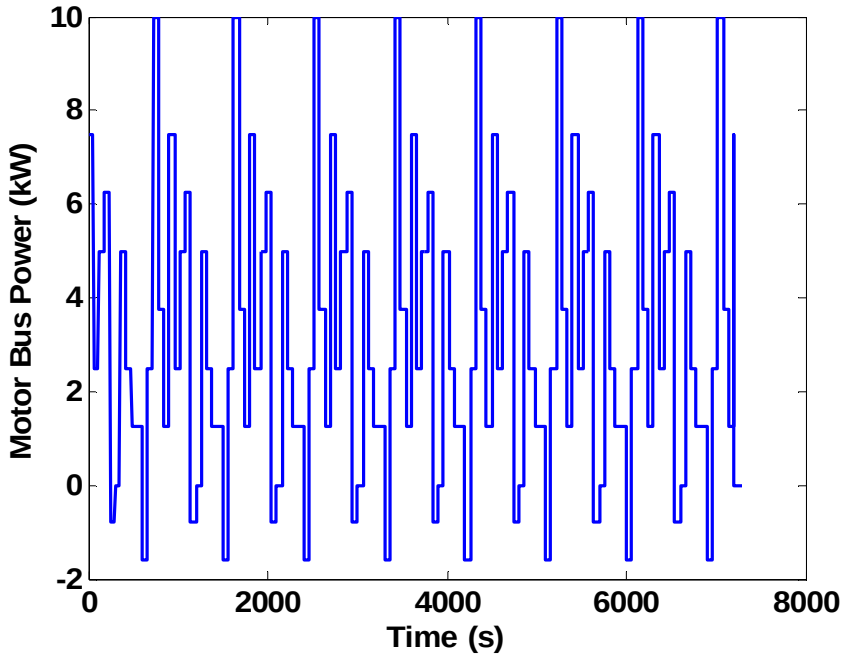


**Figure 4.31: Load profile used for all simulations**

# CHAPTER 5

# NON-OPTIMAL CONTROL METHODS

## 5.1 Thermostat control

### 5.1.1 Algorithm

The most straightforward control method for this application is what is commonly known as thermostat control. As its name implies, thermostat control functions the same way a common household thermostat or an air compressor operates (on/off with hysteresis) as shown in Figure 5.32. The control law is defined to allow the battery SOC to vary between upper and lower limits. When the SOC reaches the upper limit, the engine is turned off, and all motor power is provided by the batteries. When the SOC reaches the lower limit, the engine turns on and the controller sets the engine speed and battery current so that the engine runs at its most efficient operating state. As previously discussed, this operating state is at 5.1 kW power output and an engine speed of 1780 rpm. Motor power is supplied by engine/alternator power, and any excess/deficit is consumed/supplied by the battery pack. Due to its simplicity, this control strategy was used as a baseline for comparison to more sophisticated control strategies.

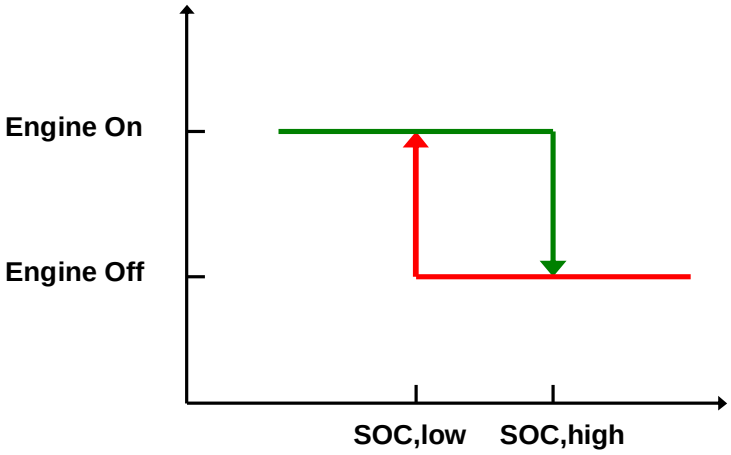


**Figure 5.32: Thermostat control law**

One limitation of basic thermostat control is described by Equation 5.34. Basic thermostat control only works if the average demanded power is less than the engine-on output power (in this case 5.1 kW).

$$\left( P_{motor,average} = \frac{1}{t_f} \int_0^{t_f} \left( P_{motor} \cdot dt \right) \right) \leq P_{engine,ON} \tag{5.34}$$

In steady state, thermostat control is loosely analogous to pulse width modulation (PWM) of the engine to supply an average power. If the average motor power were 5.1 kW, the engine would have to run all of the time (100% duty cycle). If the vehicle were used in a high average power application, additional control logic would have to be implemented to supply this sustained high power. This could easily be achieved by defining a secondary low SOC threshold that is below the normal low SOC limit; if it is reached, the engine is commanded to supply maximum power until the upper SOC limit is reached.

### 5.1.2 Results

The thermostat control law performed as expected and achieved a total efficiency of 25.86%. The engine/alternator output power and battery SOC are shown in Figure 5.33. As expected, the engine turns on when the SOC reached 0.6 and turned off when the SOC reached 0.8. At the first engine turn-on event, the SOC actually drops below 0.6. This is because the demanded motor power was larger than 5.1 kW (the engine/alternator output power), so the batteries had to continue supplying power until motor power dropped below 5.1 kW. At the end of the two hours (denoted by the vertical dashed line), the batteries were recharged back to a SOC of 0.8 to maintain charge.

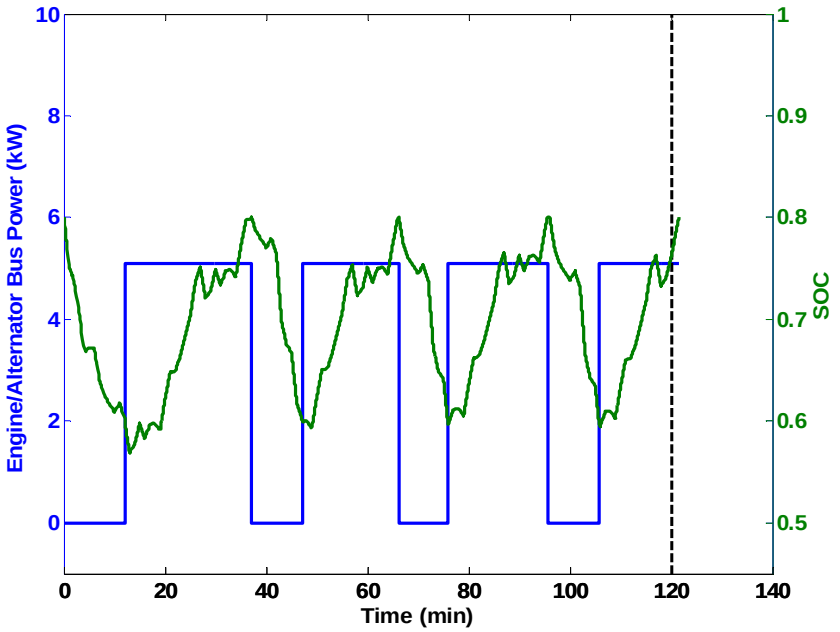


**Figure 5.33: Engine/alternator power output and SOC for thermostat control**

The choice of SOC bandwidth is critical in thermostat control. It is desirable to keep the average battery SOC in a range that maximizes efficiency, but a narrow SOC bandwidth would cause frequent engine cycling. Other SOC limits were tested and compared to the default limits of 0.6 and 0.8 as shown in Table 5.1.

**Table 5.1: Thermostat control efficiency results for various SOC ranges**

| SOC Limits | Efficiency |
|---|---|
| 0.6 to 0.8 | 25.86% |
| 0.4 to 0.8 | 25.64% |
| 0.75 to 0.8 | 25.72% |
| 0.7 to 0.9 | 25.89% |

As seen in Figure 3.24, the battery is most efficient at higher states of charge. In the second thermostat test, lowering the SOC lower limit to 0.4 from 0.6 reduced the average SOC. This reduction in average SOC is likely the main contributor to the lower total efficiency for the SOC range of 0.4 to 0.8. In the third thermostat test, the SOC lower limit was raised to 0.75. By raising the average SOC, one might assume that total efficiency would improve due to the higher battery efficiency, but it did not (compared to

the 0.6 to 0.8 baseline). This is most likely due to the increased engine cycling. The engine state transitions cause excess losses, so frequent cycling will hurt efficiency. In the final thermostat test, the SOC bandwidth was maintained so that engine cycling would remain nearly the same, but the average SOC was raised. As expected, this raised the total efficiency.

While it would be tempting to run the thermostat control with an upper limit near 100% in order to increase efficiency, any charge current into the battery at a high SOC has the potential of raising the battery voltage above a safe level. Therefore the power acceptance of the battery is greatly reduced at high states of charge.

## 5.2 Point Control

### 5.2.1 Motivation

A further step in complexity of control is to modify thermostat control in an attempt to increase direct energy usage. The battery pack in a series hybrid acts as an energy buffer, consuming and supplying energy excesses and deficiencies when needed. However, battery usage involves inherent battery losses. The benefit of battery usage often outweighs the cost (hence the original purpose of hybridization), but unnecessary battery usage blocks the hybrid system from reaching its efficiency potential. A strategy that maximizes on-time of the engine in its most efficient operating region while minimizing unnecessary battery usage will increase system efficiency.

### 5.2.2 Algorithm

The overall control scheme is identical to thermostat control, but a rule is made to act as an override to the thermostat control. The point based rule states that whenever demanded power at the DC bus exceeds the max-efficiency engine/alternator power output (5.1 kW in this case), the engine should be turned on. This is termed the point

based rule because the engine/alternator only operates at its most efficient operating point shown as the red star in Figure 3.10.

An additional logic check must be made to protect against battery overvoltage in high power load profile applications. If the power demand frequently exceeds the max-efficiency engine/alternator power output, the point control rule will frequently override thermostat control and turn the engine on. This can cause the SOC to rise too high which is very dangerous. A supervisory hysteresis loop must also be used to protect against this. Point control operates normally unless a high state of charge (90% for example) is reached. If this high state of charge is reached, the engine will be kept off until SOC falls back to the standard thermostat lower limit.

### 5.2.3 Results

The total efficiency of point control was 26.07%, which is higher than thermostat control. Figure 5.34 shows the engine/alternator power output and battery SOC as a function of time. Once again, the vertical dashed line represents the end of the 2 hour load profile and the start of the battery recharge to its original SOC. The figure shows how the engine turns on during the typical "off" portion of the thermostat cycle. The engine-on overrides correspond to power demands exceeding 5.1 kW. Increasing direct energy usage (when produced optimally) increases total efficiency.

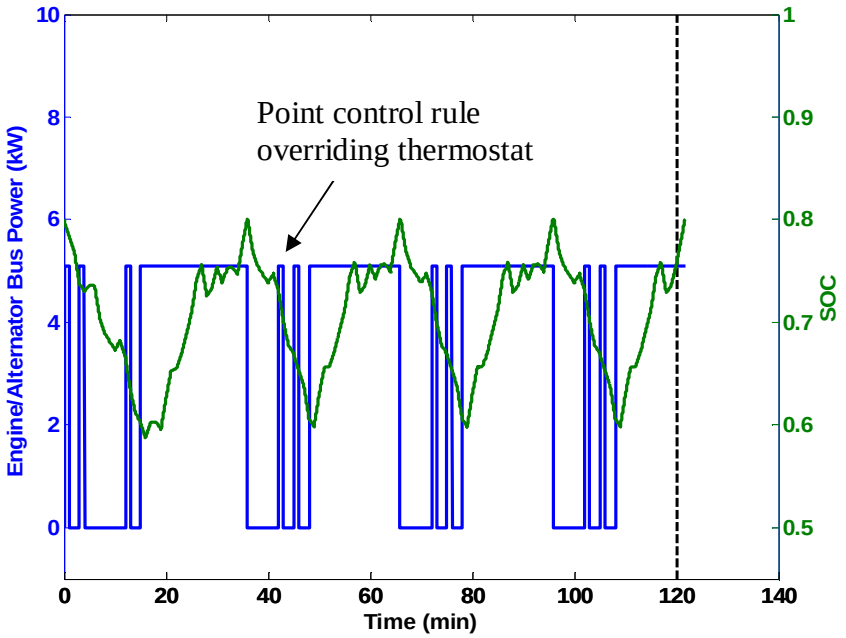


**Figure 5.34: Engine/alternator power output and SOC for Point Control**

## 5.3 Engine Optimized Line Control

### 5.3.1 Motivation

Point control operates on the basis of decreasing unnecessary battery usage to increase efficiency. Point control achieves this by only turning on to the engine/alternator's optimal power production point when the demanded power exceeds that point. However, a small deviation from that optimal point would only slightly reduce engine/alternator efficiency, but it would prevent battery usage losses. It is possible that using the engine at non-optimal points will increase total efficiency. The small efficiency decrease caused by moving from the optimal engine/alternator point can outweigh the efficiency decrease cause by using the batteries.

### 5.3.2 Algorithm

The line control rule states that whenever demanded power at the DC bus falls within a predetermined high-efficiency engine/alternator output zone, turn the engine on

at the power that matches power demand and the speed that maximizes engine/alternator efficiency for that power. For a visual explanation, if the engine efficiency is viewed as a hill with speed and power as the *x-y* axes, instead of operating only at the peak of the hill (point control), the engine operates above a chosen elevation but only along the ridgeline. This ridgeline is shown as the blue line in Figure 3.10. The turn-on elevation can be chosen to be any efficiency deemed sufficiently high, but a more intelligent method is to choose a turn-on elevation that corresponds with battery losses.

The break-even point for using the batteries can be approximated by Equation 5.35, where $\eta_{EA}$ is the engine/alternator efficiency as a function of output power, $\eta_{ESS}$ is the energy storage system efficiency (battery and converter) as a function of output power, $P_{charge}$ is whatever power is used to charge the batteries, and $P_{motor}$ is the motor power demand. Whenever the inequality is true, the engine/alternator alone should be used to supply the demanded motor power. The negative sign appears to maintain the sign convention that charge power is negative for the ESS.

$$\eta_{EA}\left(P_{charge}\right)\eta_{ESS}\left(-P_{charge}\right)\eta_{ESS}\left(P_{motor}\right) \le \eta_{EA}\left(P_{motor}\right) \tag{5.35}$$

This equation is based on the logic that any energy taken out of the battery must also be put back in. Therefore, the actual efficiency related to using the batteries is the product of the battery discharge efficiency at the present motor demand, the battery charge efficiency and the engine/alternator efficiency at whatever power is used to charge the batteries. Use of this equation can only be an approximation because the power that will be used to charge the batteries at some time in the future can only be a guess. Therefore, it is assumed that the batteries will be charged in a best case scenario at whatever power optimizes efficiency. This power level is determined by maximizing the product of engine/alternator efficiency and battery charge efficiency as functions of bus power as shown in Figure 5.35. This figure assumes a 70% SOC for the ESS efficiency. Since the EA efficiency has a steeper drop-off than the ESS efficiency, maximum

combined efficiency remains at 5.1 kW, the same location as the maximum EA efficiency. It should be noted that the charging efficiency is plotted versus positive powers simply for easier visualization (charge power is negative).

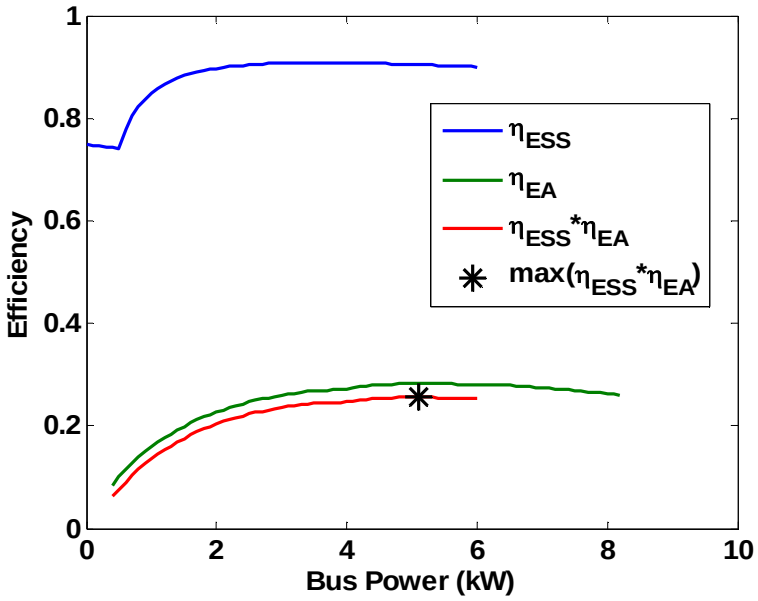


**Figure 5.35: Maximum combined system charge efficiency is also at 5.1 kW, with a value of 25.62%**

The two sides of the inequality in Equation 5.35 are shown in Figure 5.36 for the best case charging scenario. For the case shown, it is apparent that the engine/alternator should be used alone for all motor power demand levels above approximately 2 kW. At demands below 2 kW, the batteries should be allowed to discharge alone.

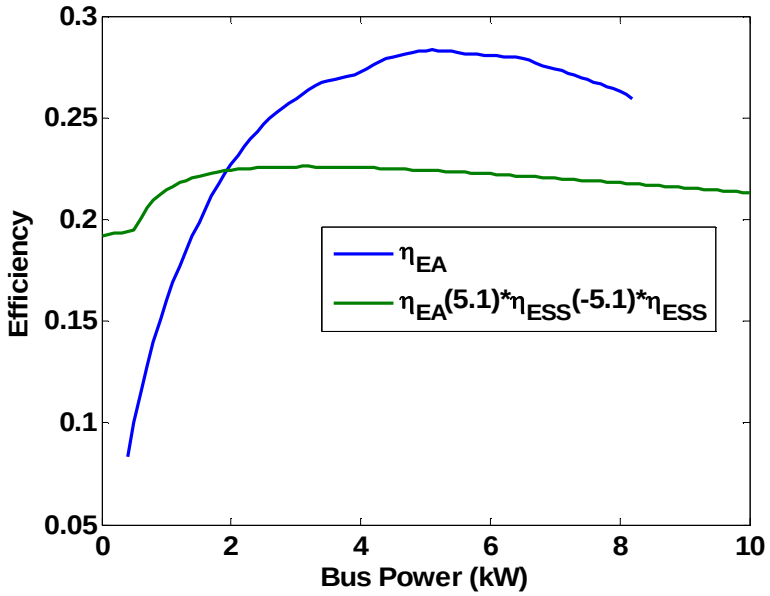


**Figure 5.36: Engine based line control ridgeline cutoff occurs at the shown intersection**

### 5.3.3 Results

Allowing the engine to provide all of the power between 2 kW and 8 kW on the discharge portion of the thermostat cycle resulted in a total efficiency of 26.18%. As expected, this efficiency exceeds the point control method, but only slightly. Since the power level cutoffs for the engine override were determined through assumptions, other power level combination levels were tested via trial and error. Using a lower limit of 3 kW and an upper limit of 7 kW, a total efficiency of 26.48% was achieved, showing that the method used in Figure 5.36 is not necessarily ideal. The EA power output and battery SOC for this improved operation are shown in Figure 5.37. It should be noted that since line control deviates from simple on-off control, efficiencies obtained from even an optimal bang-bang control law can be exceeded.

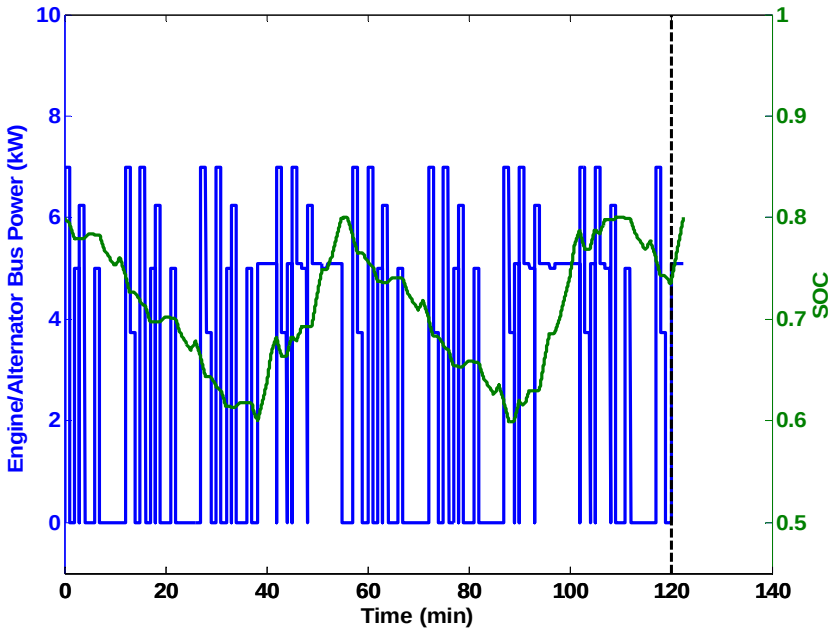


**Figure 5.37: Engine/alternator power output and SOC for Engine Optimized Line Control**

## 5.4 System Optimized Line Control

### 5.4.1 Motivation

The previous algorithm (engine optimized line control) seeks to maximize direct energy usage by operating the engine/alternator within a defined high-efficiency power output band that matches motor demanded power. When outside of this band, normal thermostat rules are obeyed. However, following normal thermostat rules when outside of the high-efficiency band is not necessarily ideal. When operating outside of the high-efficiency band, the batteries are being used by definition (unless demanded power is zero). If, for instance, the motors are regenerating and typical thermostat rules say that the engine should be on at 5.1 kW, it may be more efficient so slightly vary the engine output to maximize the combined EA and ESS efficiency. We have previously shown in Figure 5.35 that for the Deere hybrid M-Gator, optimal combined efficiency occurs at 5.1 kW for *zero* motor power demand. This optimal combined efficiency can be defined for all power demands and followed when direct energy usage is not being utilized.

### 5.4.2 Algorithm

The combined system efficiency (EA and ESS) as a function of motor power demand and EA power output is shown in Figure 5.38. As stated previously for engine optimized line control, the ideal high-efficiency power band for direct energy usage is between 3 kW and 7 kW (shown as vertical black dashed lines). For all power demands outside of this band, total system efficiency should be optimized.

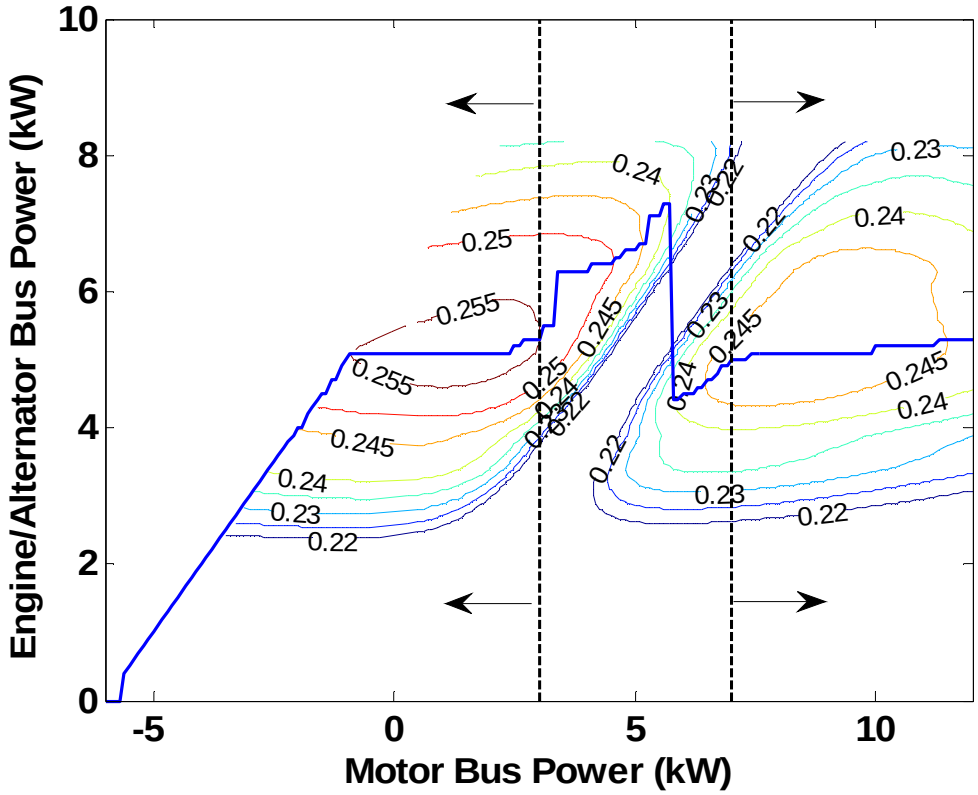


**Figure 5.38: EA power to be used for a given motor power demand to optimize EA/ESS system efficiency**

This algorithm has the potential to improve total efficiency for some systems. However, for this specific system the EA output power that corresponds with optimal system efficiencies outside of 3 to 7 kW is very close to the EA output power for normal thermostat rules (5.1 kW). Therefore, the difference in performance of the engine optimized control algorithm and the system optimized control algorithm is negligible for

the case of the Deere hybrid M-Gator. The purpose of the engine power derating during regenerative braking is to limit ESS charge power to 6 kW or less which is maximum rated charge power as stated by the manufacturer.

### 5.4.3 Results

The system optimized line control algorithm achieved identical results to the engine optimized line control algorithm (26.48% total efficiency). As previously stated, this is due to the similarity in engine optimized efficiencies and system optimized efficiencies outside of the high-efficiency power band. For systems other than the Deere hybrid M-Gator that have different efficiency profiles, the system optimized line control will possibly offer some marginal improvement. The results of the algorithm are shown in Figure 5.39. Figure 5.40 shows the details of how the EA output matches power demand within the 3 to 7 kW band, and outside of this band, system efficiencies are optimized (typically at 0 or 5.1 kW for this case). The spikes in ESS power coincide with high power demands due to the engine response time. While the engine is getting up to speed to produce power, the battery supplies the demanded power, creating the brief ESS power spike.

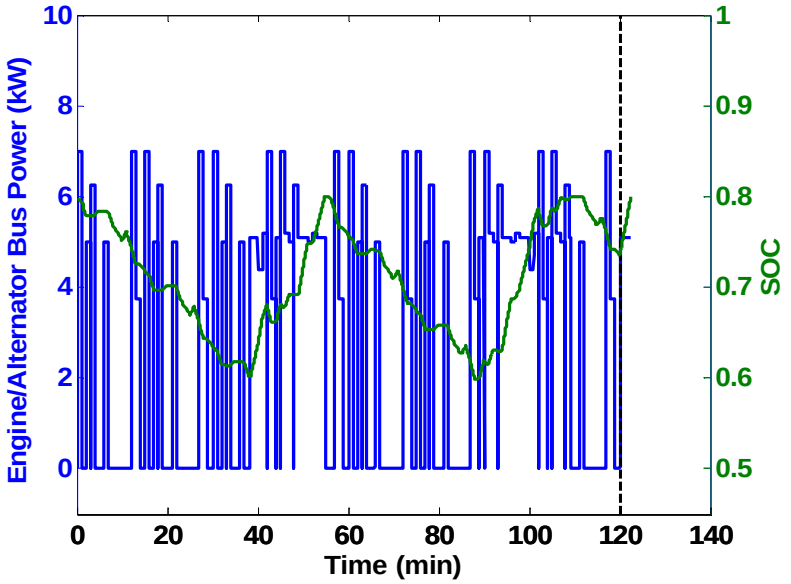


**Figure 5.39: EA power output and SOC for system optimized line control**

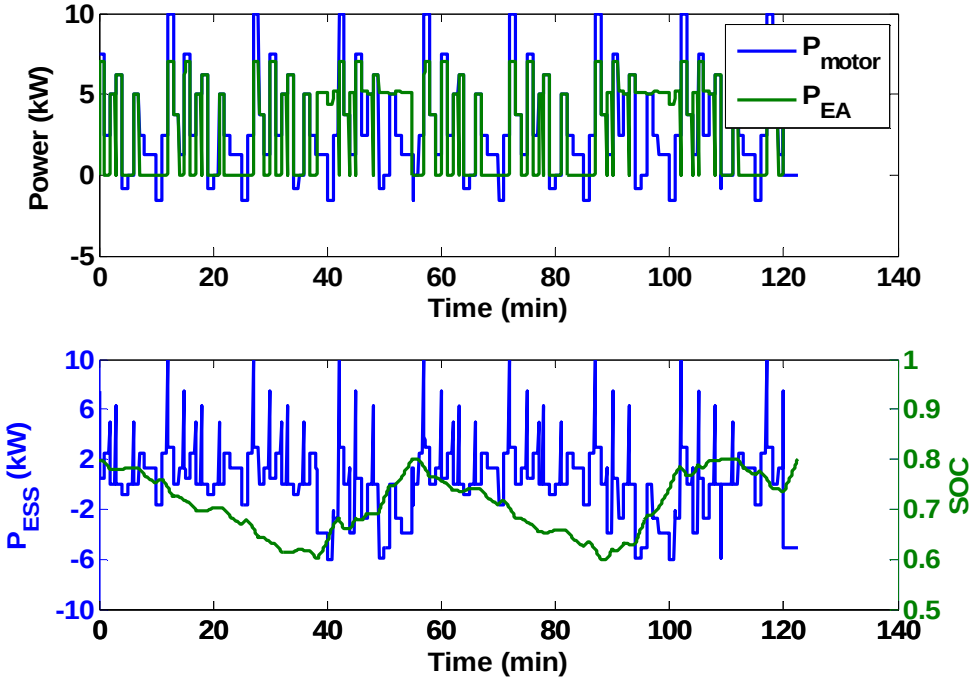


**Figure 5.40: Sytem optimized line control details**

# CHAPTER 6

# OPTIMAL CONTROL

## 6.1 Motivation

If the entire load profile for a vehicle is known, it is possible to determine the optimal energy management strategy. Dynamic programming was the method used for determining this optimal strategy.

## 6.2 Introduction to Dynamic Programming

Dynamic programming is a backwards recursion method that finds a path from point to point that minimizes the cost required to travel that path. A modified version of the traveling salesman problem can be used to illustrate the methods of dynamic programming.

Suppose a person wishes to travel from city *A* to city *J* as shown in Figure 6.41. Multiple paths of varying distances exist. The traveler must choose which cities to visit along the way that will minimize the cost (could be distance, time, fuel, etc.) from *A* to *J*. For the sake of this problem, the choices are broken up into five unique stages. If all of the distances between the cities are known, dynamic programming can be used to find the optimal path (or paths) from beginning to end. In dynamic programming parlance, each city is a node, and each portion of the journey is a stage. In this problem, there are five stages and ten total nodes.

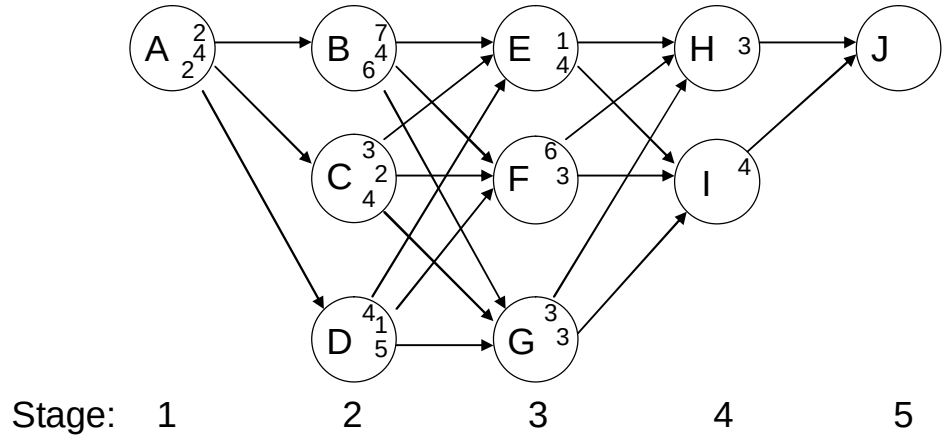


**Figure 6.41: Sample dynamic programming problem**

The map shown in Figure 6.41 is placed in an *m*-by-*n* grid, where *m* is the maximum number of choices in a stage (the maximum number of rows), and *n* is the total number of stages. The couple (*i*,*j*) represents a node's location in the grid. Let **C** be an *m*-by-*n*-by-*m* matrix where **C***(i,j,k)* represents the cost to travel from node *(i,j)* to node *(k,j+1)*. This cost from one stage to the next will be known as the step cost. Referring to Figure 6.41, the *i*-*j* plane of the cost matrix is the same as the visual node map, and the *k* dimension (into the page) refers to the step costs shown in each node circle. For example, the cost required to travel from node *A* to node *D* would be **C**(1,1,3) = 2. Let the elements of **C** where no possible cost exists remain null. This null cost exists both where nodes do not connect (there is no road between cities) and where a node does not exist (a city does not exist at that location). For example, if node *G* did not connect to node *I* as it does in this example, the cost would be **C**(3,3,2) = *NaN*. Also, the cost from node *G* to the location directly to the right of node *G* is **C**(3,3,3)= *NaN* since no node exists at the map location (4,3). Theoretically, impossible paths have an infinite cost, so it would also be practically valid to set null costs to very large values.

Let **F** be an *m*-by-*n* matrix where **F***(i,j)* represents the minimal (optimal) cost to travel from node *(i,j)* to the final destination node (in this example node J). This minimal cost from one stage to the destination will be known as the optimal node cost. The

optimal node cost for the beginning node (**F**(1,1) for node *A* in this example) is the same as the overall optimal path cost. In mathematical terms, the optimal node cost is calculated using Equation 6.36. Optimal node costs are calculated starting at the final stage and working backward to the beginning stage. When the beginning stage is reached, the overall optimal path cost is known.

$$\mathbf{F}(i,\ j) = \min_{k=1:m}\left(\mathbf{C}(i,\ j, k) + \mathbf{F}(k,\ j+1)\right) \tag{6.36}$$

Multiple optimal paths can exist, and the choice of optimal paths is arbitrary unless additional optimization preferences or costs are introduced. After backwards recursion is completed and an optimal path is determined, a path matrix is used to trace the path back from the beginning node to the destination node. The path matrix is analogous to a roadmap telling the driver where to go to achieve a minimum cost. Let **P** be an *m*-by-*n* matrix where **P***(i,j)* is the row index of the next node in the optimal path. For instance, at node *(i,j)*, the next node in the optimal path would be *(***P***(i,j),j+1)*. The path matrix is determined simultaneously with the **F** matrix. As shown in Equation 6.36, a minimization is performed as *k* varies over vector of length *m*. The *k* index where the minimum occurs is the optimal path index, **P***(i,j)*. If a minimum exists for multiple *k* indices, further arbitration is required to determine which to choose as previously discussed.

To clarify the algorithm, the simple example in Figure 6.41 will be worked through. The algorithm begins at node *J* in stage 5, where all costs are set to zero. The optimal cost for nodes in stage 4 is simple since there is only one choice for each node.

$$\mathbf{F}(H) = \mathbf{F}(1,4) = 3 \tag{6.37}$$

$$\mathbf{F}(I) = \mathbf{F}(2,4) = 4 \tag{6.38}$$

For each node in stage 3 there are more choices. For node *E*, the step cost to node *H* is **C**(*E*,*H*)=**C**(1,3,1)=1, and the following cost from *H* to *J* is **F**(*H*)=**F**(1,4)=3 as

previously determined, yielding a total cost of 4. The step cost from *E* to *I* is **C**(*E*,*I*)=**C**(1,3,2)=4, and the following cost from *I* to *J* is **F**(*I*)=**F(**2,4) = 4 as previously determined, yielding a total cost of 8. Therefore the minimum cost from node *E* to the destination node *J* is 4, with a path *E-H-J*. These calculations for stage three and the preceding stages are shown in Table 6.2 through Table 6.4. The completed optimal node cost matrix is shown in Equation 6.39.

**Table 6.2: Stage 3 calculations**

| ***j=3*** | **C**(*i,j,k*) + **F**(*k,j+1*) | | **F**(*i,j*) | **P**(*i,j*) |
|---|---|---|---|---|
| | H | I | | (go to) |
| E | 4 | 8 | 4 | H |
| F | 9 | 7 | 7 | I |
| G | 6 | 7 | 6 | H |

**Table 6.3: Stage 2 calculations**

| ***j=2*** | **C**(*i,j,k*) + **F**(*k,j+1*) | | | **F**(*i,j*) | **P**(*i,j*) |
|---|---|---|---|---|---|
| | E | F | G | | (go to) |
| B | 11 | 11 | 12 | 11 | E or F |
| C | 7 | 9 | 10 | 7 | E |
| D | 8 | 8 | 11 | 8 | E or F |

**Table 6.4: Stage 1 calculations**

| ***j=1*** | **C**(*i,j,k*) + **F**(*k,j+1*) | | | **F**(*i,j*) | **P**(*i,j*) |
|---|---|---|---|---|---|
| | B | C | D | | (go to) |
| A | 13 | 11 | 10 | 10 | D |

$$\mathbf{F} = \begin{bmatrix} 10 & 11 & 4 & 3 & 0 \\ NaN & 7 & 7 & 4 & NaN \\ NaN & 8 & 6 & NaN & NaN \end{bmatrix} \tag{6.39}$$

For this example, the optimal cost to travel from node *A* to node *J* is **F**(*A*)=**F**(1,1)=10, and the optimal path is either *A-D-E-H-J* or *A-D-F-I-J*. Since no secondary costs or preferences were specified for this example, the choice between the two optimal paths is arbitrary.

## 6.3 Development of Transition Maps

For the purpose of energy management, a state map analogous to the map shown in Figure 6.41 had to be developed. Every node in the map represents a specific system operating state. There are two states of interest: battery state of charge (SOC) and the engine state. The SOC was quantized into 200 bins, each 0.5% wide. For simplification, it was assumed that the engine would only be run in two states: off or on at its most efficient operating point. This on-off control restricts the optimality of the derived control law, but it allows for a more manageable dynamic programming problem formulation. The effect of allowing more allowable engine states is currently unknown, and a sensitivity study is left for future work. This formulation of 200 SOC states and 2 engine states yields 400 possible system states. This yields a state map that has 400 rows and the same number of columns as time steps. If a simulation is run for 2 hours sampled every 60 seconds, the state map would be 400-by-120. It is apparent how more engine states would quickly make the dynamic programming computationally difficult. Additional states were attempted through the use of finer SOC resolution, but Matlab memory restrictions prevented this finer resolution.

With a state grid created, it was necessary to determine how the hybrid energy system transitions from one system state to another. In other words, what will the change in SOC be from one stage to the next for a given power demand, time step duration, engine state, and engine history. For simpler problems it is possible to run a model simulation for every state transition during the dynamic programming backwards recursion. However, simulating the hybrid system for every state transition at runtime of the backwards recursion would be very computationally intensive. Therefore, it was decided to develop a state transition map. This state transition map would be a multi-input lookup table that shows how the system state changes given the aforementioned system inputs of power demand, time step duration, engine state, and engine history. The power demand was quantized from -6 to 12 kW for every kilowatt yielding 19 power

demand bins. The time step was chosen to be 60 seconds. The time scale of system simulations is in hours, so a time step of 60 seconds is relatively small without becoming computationally unreasonable. It is important to note that by choosing a 60 second time step, it is being assumed that the power demand is constant over those 60 seconds. While power demand will very likely not be constant for that long in real load cycles, this assumption is necessary to make the problem computationally reasonable. It is therefore important that the input load cycle reflect average power demands at every 60 second time step.

It is not enough to know if the engine is simply on or off, because the engine consumes more power during startup than if it was already running. Therefore the engine operating history also must be known. There are four possibilities: *on-on*, *on-off*, *off-on*, and *off-off*. These engine transitions are defined to describe the engine state in past stages. For instance, engine *off-on* means that the engine was previously off and is now on (as opposed to currently off and about to be turned on). The *on-off* transition is redundant since no extra energy is consumed turning off the engine. Therefore, the same amount of energy is consumed for an on-off transition as an off-off transition (zero energy).

The developed Simulink system model was used to run simulations for all possible combinations of power demand, engine transitions, and starting SOC for the chosen time step. The change in SOC and the energy consumed were recorded. The collected data was fit to four lookup tables (one for each engine transition type) with power demand and final SOC as the inputs. These lookup tables (or transition maps) can be implemented efficiently at runtime of the backwards recursion to determine system state transitions from one stage to the next. Since dynamic programming is backwards recursion, the input to the transition maps was final SOC and the output was initial SOC (given where we are now, where did we come from given these inputs). Total energy consumption was also calculated for every possible state transition. The basic input/output relationship for the state transition maps is shown in Figure 6.42

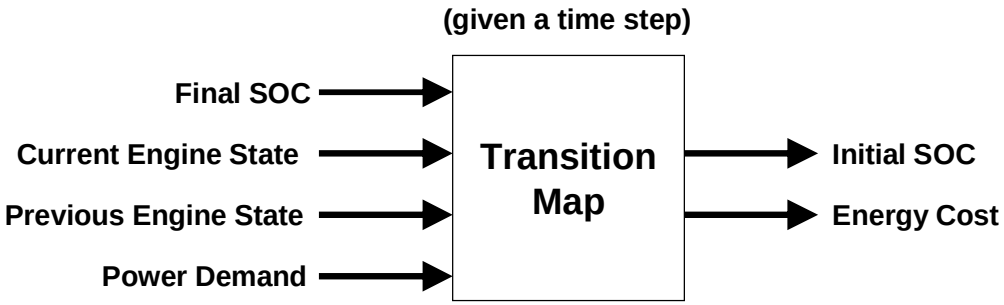


**Figure 6.42: Basic input/output relationship for state transition maps**

It should be noted that the use of transition maps yields a large advantage over runtime simulation besides computational efficiency and simplicity. If no system model is available or model errors are believed to be large, a state transition map can be easily developed experimentally using the actual physical system. The test procedure is the same as with a model (multiple runs for the various power demands and engine transitions while recording state of charge and energy consumption). This allows dynamic programming to be used on a system with no time or effort spent in model creation, and no error contribution from modeling. However, there will be measurement and noise error.

## 6.4 Cost Structuring and Development

In addition to the transition maps, a cost matrix also had to be created to determine costs required to transition from one state to another. The primary cost for this system optimization was chosen to be energy consumption. By minimizing energy consumption for a given load cycle, overall fuel efficiency is maximized. As stated before, the simulation data-based transition maps yielded total energy consumption for a transition from one state to another. A summary of the chosen costs is given in Table 6.5.

**Table 6.5: Step costs chosen for state transitions**

| Engine State | Cost | | |
|---|---|---|---|
| (Previous Stage - Current Stage) | Primary | Secondary | Total |
| on-on | $E_{fuel}$ | 0 | $E_{fuel}$ |
| on-off | 0 | 1 | 1 |
| off-on | $E_{fuel}+E_{starter}$ | 1 | $E_{fuel}+E_{starter}+1$ |
| off-off | 0 | 0 | 0 |

The primary costs are related to energy consumption which is the primary optimization parameter. Zero energy is consumed when the engine is off, so a zero cost is assigned for those stage durations. When the engine is on, the incurred cost is equal to the total energy consumed. For the case of engine startup, this total energy includes both fuel energy and the energy required to start the engine. Secondary costs were chosen to minimize cycling of the engine. This secondary cost has no bearing on energy consumption due to engine cycling which is already accounted for in the primary cost. However, if more than one optimal path choice exists, the secondary cost is intended to narrow the optimal path choices to those that are less appealing to the user (frequent engine cycling is assumed to be less appealing). The relative size of the secondary cost is determined by Equations 6.40 and 6.41, where $C_2$ is the secondary cost, $C_1$ is the primary cost, and $T$ is the sampling period (stage duration). By using a secondary cost that is an appropriate order of magnitude less than the primary cost, the secondary cost can never interfere with the primary cost.

$$\#stages = T * load\_cycle\_time \tag{6.40}$$

$$\max(C_2) * (\#stages) < \min(C_1) \tag{6.41}$$

## 6.5 Implementation of the Algorithm

Dynamic programming for energy management was implemented in Matlab. The optimization was run for a variable power demand for a load cycle two hours long. Desired initial, final, minimum, and maximum SOCs were chosen. The script operates in two main sections: cost matrix creation and backwards recursion.

Starting at the final stage and working backwards, the cost matrix is filled in. A given node represents the unique combination of an engine state and an SOC state for a unique overall system state. With the system state and the input power for the given stage, a transition map is used to calculate what old SOC (from the previous stage) would result in the current SOC. Lookup tables exist for all relevant engine transition possibilities (*off-off*, *off-on*, *on-off*, and *on-on*). The appropriate cost is assigned for every possible node transition depending on the engine transition type. One node can only reach a maximum of four nodes in the next stage (one for each engine transition type), so all costs from the originating node to unreachable nodes in the next stage are set to invalid (*NaN* in Matlab).

After the cost matrix is constructed, the backwards recursion is executed using Equation 6.36 to create the optimal node cost matrix and the path matrix. When multiple optimal paths exist, the last path is chosen which keeps SOC high. Keeping SOC high generally decreases depth-of-discharge (assuming SOC begins high) and increases battery life. The actual code implemented in Matlab is given in the appendix.

## 6.6 Results of the Algorithm

An example result of the dynamic programming optimization algorithm is shown in Figure 6.43. The power demand varies between -2 and 8 kW, and the load cycle was forced to be charge sustaining (end at the same SOC as it started). For demonstration purposes, a binary cost was assigned for the engine-on penalty as opposed to an individualized fuel consumption cost. As a result, multiple optimal paths were found. Figure 6.44 shows a result using the fuel consumption costs. Since the energy costs were highly variable between node transitions, a single optimal solution was found.

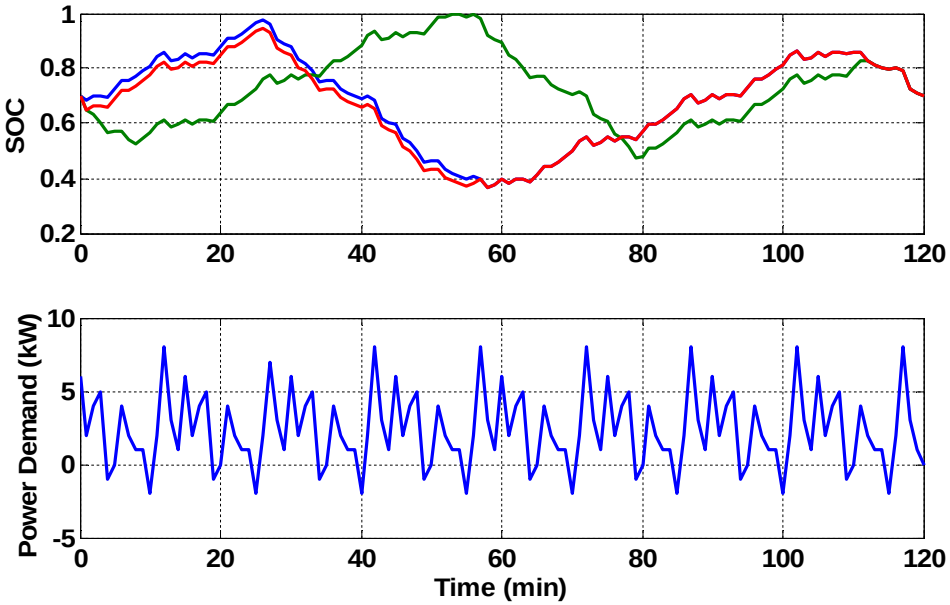


**Figure 6.43: Multiple optimal solution paths using binary engine costs (1 or 0)**

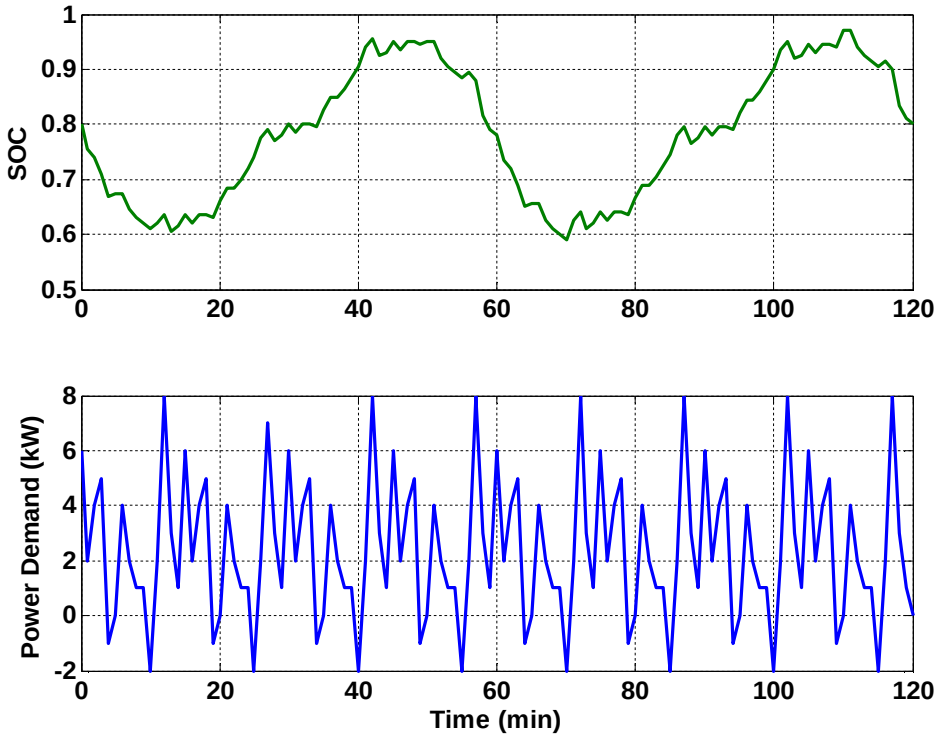


**Figure 6.44: Single optimal solution path using individualized fuel costs**

## 6.7 Results used in Simulink Model and Comparison to Expected Results

A single optimal path was chosen from the results of the dynamic programming algorithm and the engine cycling commands were defined to be the optimal control law for the given power demand. The optimal control law and corresponding power demand were used as inputs to the developed Simulink system model. The comparison between the expected dynamic programming results and the results of the Simulink model is shown in Figure 6.45. It is apparent that the path shape is correct, but some slight drift error occurs. This error is most likely due to the quantization of SOC. For this simulation, SOC was quantized into 0.5% bins, so any rounding error gradually builds over the simulation. Since rounding error occurs in both directions, the results are good, but not ideal. This drift actually causes the final SOC to be higher than the beginning SOC, so slightly more energy was expended than needed to maintain charge.

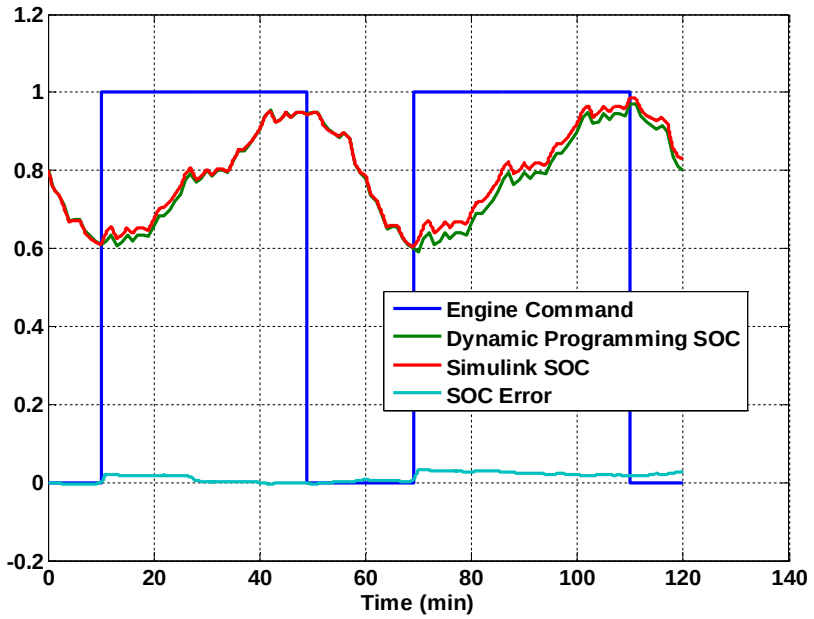


**Figure 6.45: Simulink model using optimal control law comparison to expected results**

The SOC and EA output power are shown in Figure 6.46. The calculated total efficiency was only 25.75%, 0.11% worse than regular thermostat control. However, this

calculated efficiency does not take into account the excess charge remaining in the battery. After discounting the consumed energy by the amount of fuel energy it would take to charge the battery from 80% SOC to its final SOC of 82.86% (1.201 MJ), the calculated efficiency is 26.11%. This calculation is shown in Equation 6.42, where $E_{excess}$ is the 1.201 MJ of excess energy used to charge the battery an extra 2.86% SOC.

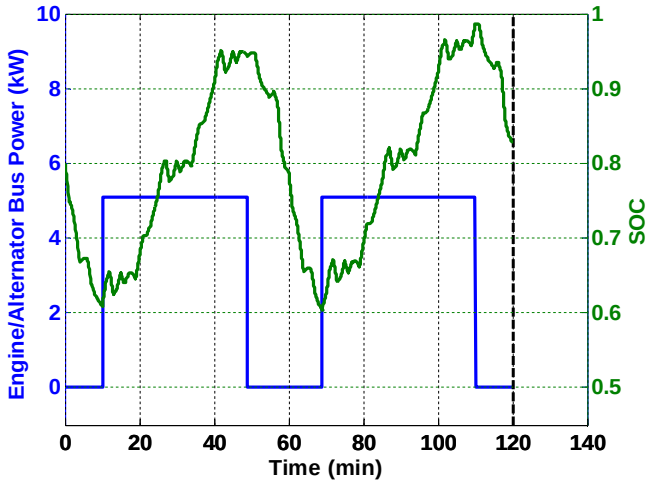


**Figure 6.46: SOC and EA output power for the "optimal" control law**

$$\eta_{optimal} = \frac{E_{out}}{\left(E_{in} - E_{excess}\right)} \tag{6.42}$$

It is interesting to note that by imposing the constraint of on-off control, the dynamic programming algorithm gives a good idea of an ideal thermostat hysteresis bandwidth. For this specific case, allowing SOC to range between approximately 60% and 95% is optimal. Of course the optimal bandwidth can change for a given vehicle based on load cycle. For real world use, dynamic programming can be used to determine an optimal SOC hysteresis bandwidth for a given vehicle with a typical usage load profile. The production vehicle can then be programmed with thermostat control and an SOC bandwidth that optimizes efficiency for the projected typical usage patterns. For instance, if a vehicle is typically used for on-road transport, an optimal thermostat control law can be calculated for a mostly flat load profile.

As verification, SOC limits of 60% and 95% were used for thermostat control and a total efficiency of 26.07% was achieved. This high and wide SOC bandwidth keeps SOC high where the ESS is most efficient and also reduces engine cycling losses. This thermostat implementation of the near optimal on-off control law is shown in Figure 6.47.

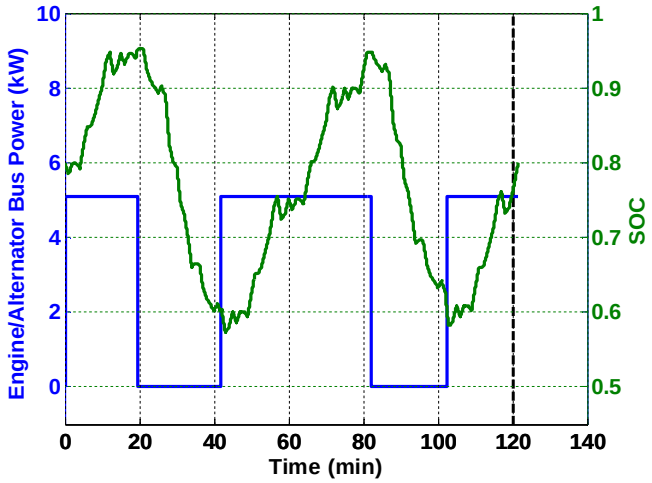


**Figure 6.47: Thermostat implementation of near optimal on-off control law**

# CHAPTER 7

# COMPARISON OF ENERGY MANAGEMENT STRATEGIES

## 7.1 Comparison for Single Test Case

All of the energy management strategies were tested for a single load profile, and their total cycle efficiency results are shown in Table 7.6 and Figure 7.48. Three of the energy management strategies are considered on-off methods since they limit use of the engine to "on" (at its most efficient operating state) or "off". The other two methods release the on-off constraint and allow the engine/alternator unit to vary its power output over a specified range.

**Table 7.6: Comparison of efficiencies to thermostat control**

| | Strategy | Total Cycle Efficiency | % Improvement from Thermostat |
|---|---|---|---|
| On-Off Methods | Thermostat | 25.86% | 0.00% |
| | Point Control | 26.07% | 0.81% |
| | Optimal On-Off | 26.11% | 0.97% |
| Line Methods | Line Control (Engine Optimized) | 26.48% | 2.40% |
| | Line Control (System Optimized) | 26.48% | 2.40% |

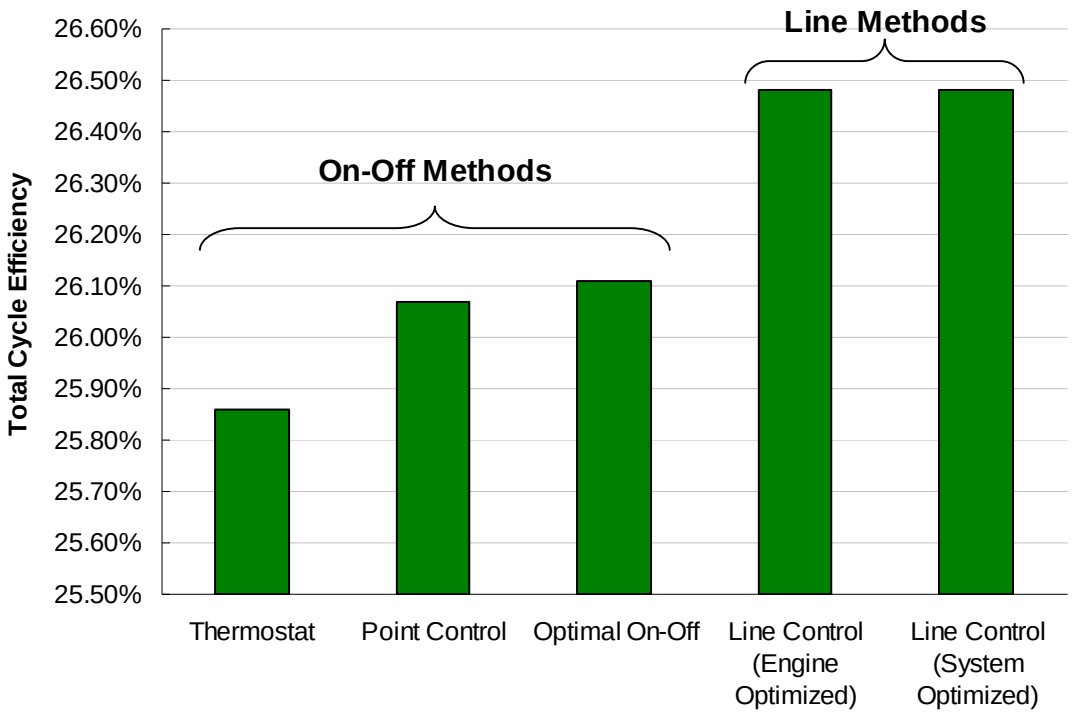


**Figure 7.48: Efficiency comparison of energy manangement strategies (note non-zero ordinate origin for emphasis of differences)**

With only knowledge of the EA's most efficient operating state and a method for calculating battery SOC, both thermostat and point control can be easily implemented. Point control shows a slight advantage since it increases direct energy usage and decreases unnecessary ESS usage (and the associated losses). The dynamic programming algorithm results in an optimal on-off control law for a given load cycle. If a vehicle is used in a common way, this typical load cycle can be used to determine an optimal on-off control law and it can be implemented through the use of thermostat control. Through either simulation or physical experimentation, a near optimal control law could be found by trial and error and knowledge of ESS efficiency trends.

When the constraint of on-off control is released, great improvements in efficiency are observed. Both line control methods seek to further increase direct energy usage by allowing the engine/alternator to provide power within a high-efficiency region. These methods require more detailed knowledge of the system to keep efficiency high.

Engine and alternator efficiencies as a function of power and speed need to be known to define the high-efficiency ridgeline of operation shown as the blue line between 3 and 7 kW in Figure 3.10. This efficiency data may be available from the manufacturer or it may be determined through physical experimentation. The actual ridgeline power limits that maximize total efficiency vary with load cycle, so power limits must be chosen that work for the most common vehicle usage load profiles.

Besides the primary optimization parameter of fuel efficiency, the secondary consideration of battery life must be considered. By choice, all non-optimal control methods limited battery depth of discharge to only 20% (SOC limits of 60% and 80%). The optimal on-off method allowed a 35% depth of discharge (SOC limits of 60% and 95%). A larger depth of discharge results in shorter battery life, but total amount of battery usage also affects battery life. The control methods that maximize direct energy usage from the engine/alternator by definition reduce battery usage and hence increase battery life. Not only do the line-based methods increase efficiency, they also increase battery life. Even though the line methods increase engine wear, engine maintenance is typically cheaper than battery replacement.

## 7.2 Effect of Different Load Profiles

It is important to note that the load profile has the potential to greatly affect the outcome of different energy management strategies. For instance, if the power demand never exceeds the optimal power output of the engine/alternator (in this case 5.1 kW), point control will have the same control law (and hence outcome) as thermostat. This is because the condition for the override rule that point control uses is never satisfied. The state-of-charge, engine/alternator output power, and motor power demand for thermostat, point and line control methods are all shown in Figure 7.49. Optimal control results are shown in Figure 7.50. The comparison of total energy consumption between the various energy management strategies is shown in Table 7.7. From this comparison, it is

apparent that only optimal control has any advantage over thermostat control. This is because the conditions for the override rules of each of the other strategies are never met for a consistently low power demand.

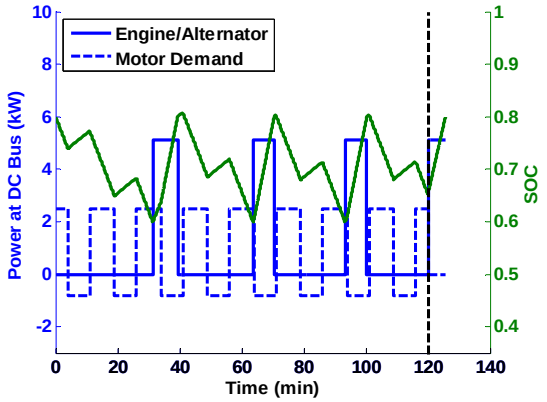


**Figure 7.49: Identical thermostat, point, and line control results for low power load profile**

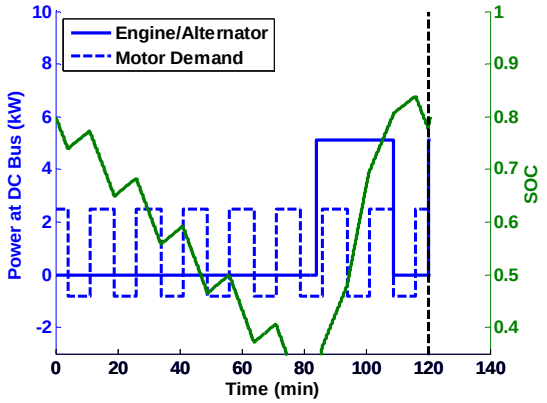


**Figure 7.50: Optimal control results for low power load profile**

**Table 7.7: Comparison of EMS for low power load profile**

| Low Power | Strategy | Energy Input (MJ) | %Improvement Over Thermostat |
|---|---|---|---|
| On-Off Methods | Thermostat | 29.82 | 0.00% |
| | Point Control | 29.82 | 0.00% |
| | Optimal On-Off | 27.93 | 6.34% |
| Line Methods | Line Control (Engine Optimized) | 29.82 | 0.00% |
| | Line Control (System Optimized) | 29.82 | 0.00% |

Maintaining a low power demand, results can change based on the amount of regenerative braking (negative power demand) for a load profile. By increasing the regenerative braking of the previous load profile by only 1 kW, the results change as shown in Figure 7.51 and Table 7.8. It is apparent that total energy consumption goes down since more energy is returned to the batteries via regenerative braking, but percent improvement of optimal control over the other strategies has decreased. This relative improvement is dependent on load profile, not just energy management strategy selection.

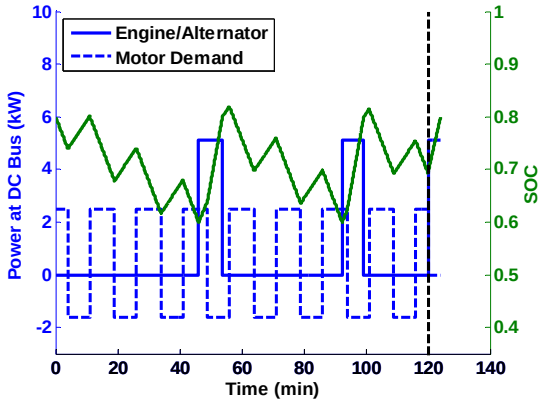


**Figure 7.51: Thermostat control for low power load profile with slightly more negative power**

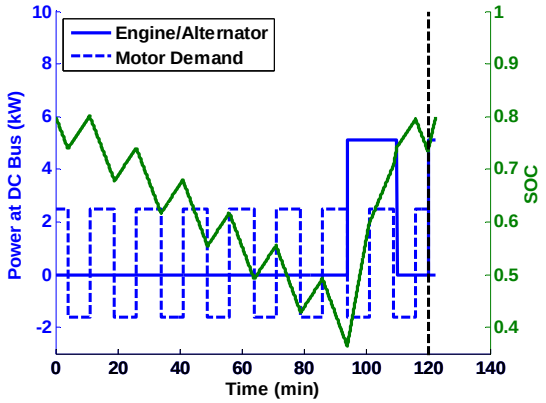


**Figure 7.52: Optimal control for low power load profile with slightly more negative power**

**Table 7.8: Comparison of EMS for low power load profile with more regenerative braking**

| Low Power (more regen) | Strategy | Energy Input (MJ) | %Improvement Over Thermostat |
|---|---|---|---|
| On-Off Methods | Thermostat | 20.06 | 0.00% |
| | Point Control | 20.06 | 0.00% |
| | Optimal On-Off | 19.63 | 2.13% |
| Line Methods | Line Control (Engine Optimized) | 20.06 | 0.00% |
| | Line Control (System Optimized) | 20.06 | 0.00% |

If a mid-level power demand is selected, the point control rule is still not engaged, but the line control rules are. For this type of load profile line control and optimal control can yield an advantage over thermostat control, but point control still does not. The results and comparison are shown in Figure 7.53, Figure 7.54, Figure 7.55, and Table 7.9.

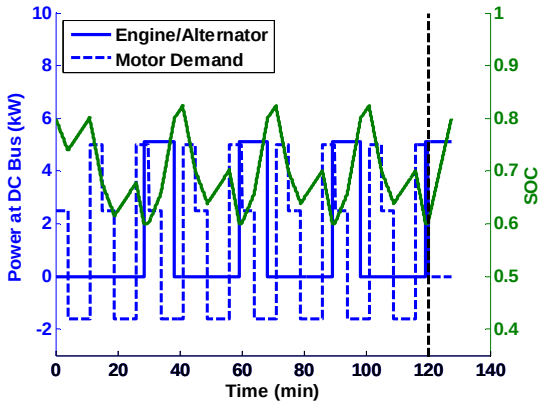


**Figure 7.53: Identical thermostat and point control for mid-power load profile**

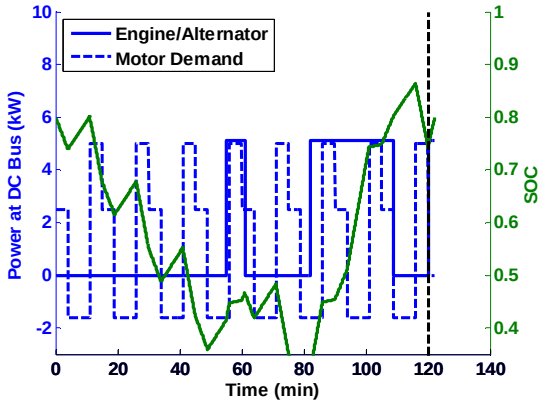


**Figure 7.54: Optimal control for mid-power load profile**

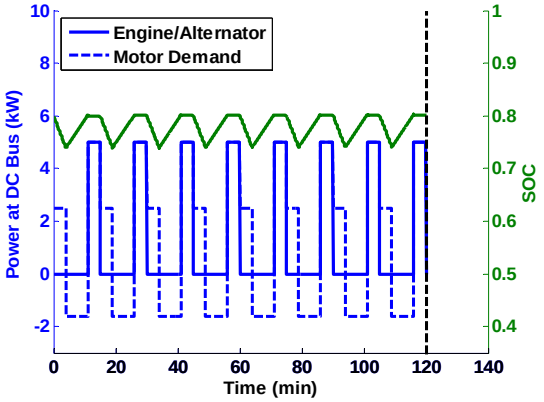


**Figure 7.55: Line control for mid-power load profile**

**Table 7.9: Comparison of EMS for mid-power load profile**

| Mid Power | Strategy | Energy Input (MJ) | %Improvement Over Thermostat |
|---|---|---|---|
| On-Off Methods | Thermostat | 38.94 | 0.00% |
| | Point Control | 38.94 | 0.00% |
| | Optimal On-Off | 37.48 | 3.76% |
| Line Methods | Line Control (Engine Optimized) | 33.89 | 12.97% |
| | Line Control (System Optimized) | 33.89 | 12.97% |

Conversely, if the power demand frequently exceeds the optimal power output of the engine/alternator, point control may greatly improve upon thermostat control. This was demonstrated before with the single test case, but here the load profile is simplified to a single high power level alternating with a single regenerative braking power level. Thermostat, point control, optimal control, and line control results are shown in Figure 7.56 through Figure 7.59, respectively. The comparison of energy consumption results are shown in Table 7.10. Now, since the criteria for the override rules of point and line control are met for the high power load profile, both improve upon thermostat control. It is interesting to note that the optimal profile is similar to point control in that it minimizes unnecessary battery usage by turning the engine on during high power demand.

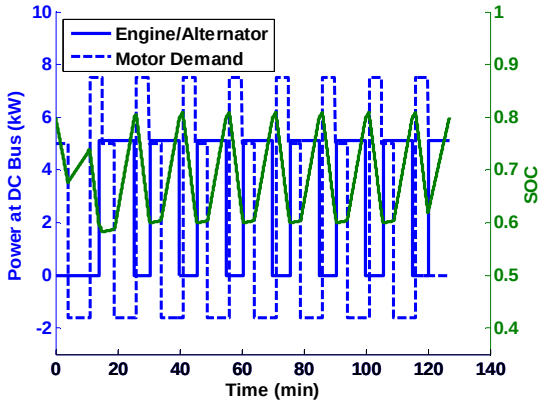


**Figure 7.56: Thermostat control for high power load profile**

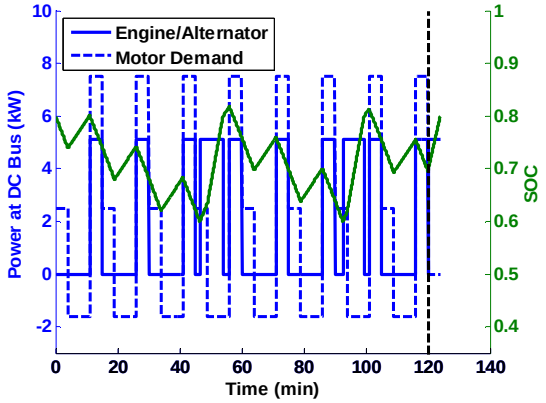


**Figure 7.57: Point control for high power load profile**

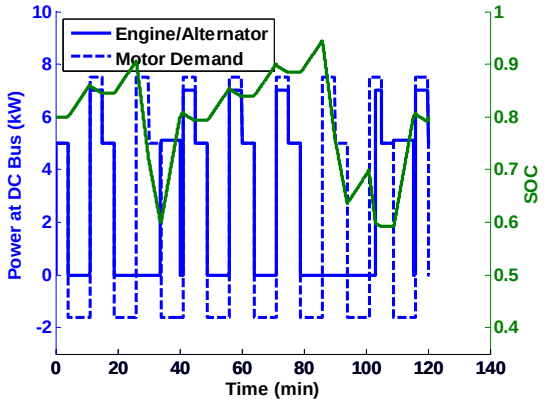


**Figure 7.58: Line control for high power load profile**

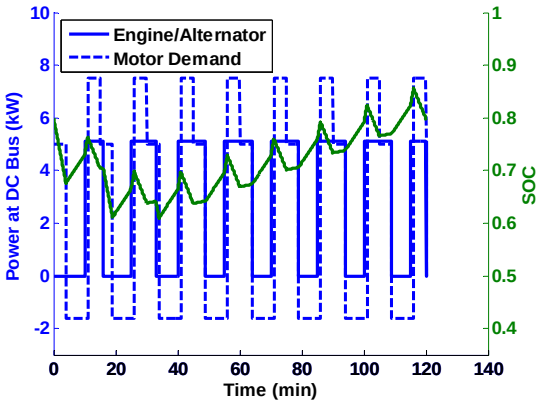


**Figure 7.59: Optimal control for high power load profile**

**Table 7.10: Comparison of EMS for high power load profile**

| High Power | Strategy | Energy Input (MJ) | %Improvement Over Thermostat |
|---|---|---|---|
| On-Off Methods | Thermostat | 79.66 | 0.00% |
| | Point Control | 73.38 | 7.88% |
| | Optimal On-Off | 69.06 | 13.31% |
| Line Methods | Line Control (Engine Optimized) | 72.76 | 8.67% |
| | Line Control (System Optimized) | 72.76 | 8.67% |

A control engineer must also pay attention to the effects of energy management strategies on hard limits of the system. These limits include battery current limits, battery voltage limits, and regenerative torque limits. These nonlinear constraints effect the performance of energy management strategies, and they cannot be ignored. It is interesting how for certain load profiles, extra effort on complicated energy management systems may not pay off. For instance, the single test case used in the majority of this document shows very little difference between energy management strategy performance. Therefore, for a vehicle that commonly has this type of load profile, there is little reason to invest in extensive development of complicated control strategies when thermostat control will achieve nearly the same efficiency. However, for the high power load profile shown above, point and line control methods show a significant improvement over thermostat control. The control designer must be very aware of potential vehicle applications for this reason.

### 7.3 Effect of Quantization on Dynamic Programming Results

An important consideration when examining the "optimal" results of the dynamic programming algorithm is the effect of quantization error. As discussed earlier, dynamic programming finds a path through a state matrix that minimizes cost for a given load profile. This state matrix is a combination of battery state-of-charge and engine states (on or off). The states-of-charge must be quantized into bins to form an even grid. By rounding the transition map to fit the grid, certain possible state paths from start to finish are eliminated. Dynamic programming successfully finds the path that minimizes cost out of the local set of possible state paths, but other lower-cost paths may exist in the global set of actual (non-quantized) possible state paths. The resolution used for SOC (and other states) determines the local set of possible paths that are available. If a very course mesh is used, error accumulation will be very large as seen in Figure 7.60 where the SOC was quantized into 5% bins. The path predicted by the dynamic programming algorithm and the actual path seen in simulation deviate a great deal due to the quantization error.

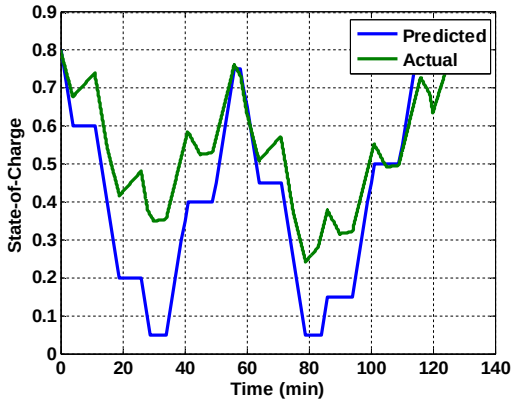


**Figure 7.60: SOC error accumulation due to large SOC quantization bins of 5%**

When a finer mesh is used, error accumulation is reduced, but local possible path sets change as different resolutions are used. Solution sets were derived using 0.5%, 1%, and 2% SOC resolution (in addition to the 5% resolution shown above), as shown in

Figure 7.61 through Figure 7.63. All of these paths were derived using the high power load profile discussed in the previous section.

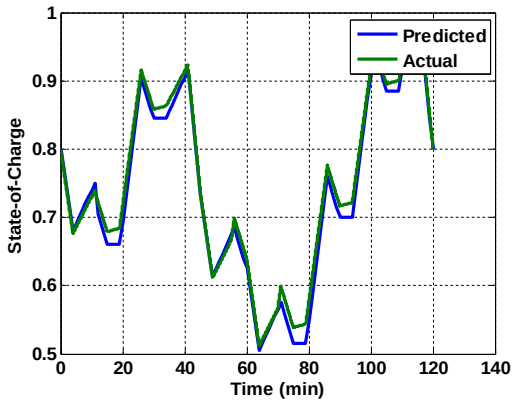


**Figure 7.61: Predicted and actual SOC path for 0.5% SOC resolution**

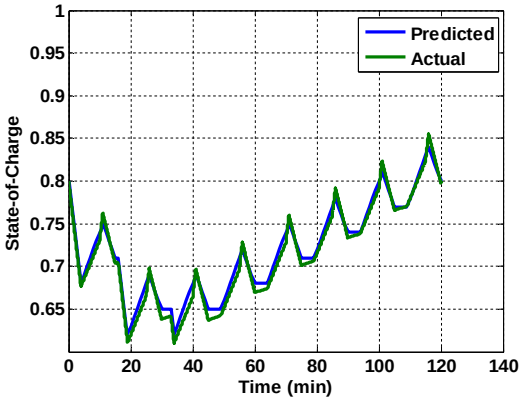


**Figure 7.62: Predicted and actual SOC path for 1% SOC resolution**

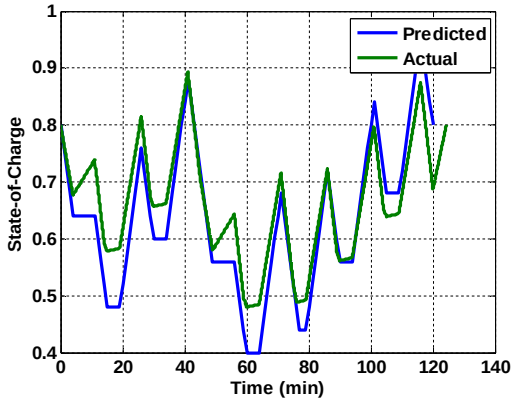


**Figure 7.63: Predicted and actual SOC path for 2% SOC resolution**

The energy consumption results are shown in Table 7.11. Since each quantization level makes different local possible paths available, some sets will contain better paths than others. In this case the 1% path set contains the path that achieves the least energy consumption for the given high power load profile. When searching for the optimal path for a given load profile, it is critical for the control engineer to be aware of the effect of quantization as it can greatly affect the results of the dynamic programming algorithm.

**Table 7.11: Energy consumption comparison for various SOC quantization sizes**

| SOC Quantization for Dynamic Programming Solution | Energy Input (MJ) |
|---|---|
| 0.50% | 73.09 |
| 1% | 69.06 |
| 2% | 78.99 |
| 5% | 76.54 |

# CHAPTER 8

# CONCLUSION AND SUGGESTED FUTURE WORK

A method for modeling and simulating a series hybrid powertrain has been presented. This model was used for testing multiple energy management strategies. If knowledge of the system is limited, it is apparent that simple logic can improve over the common thermostat control method by using point control. If efficiency data for the engine and alternator are known, line control methods can be used to further increase efficiency.

The dynamic programming method was presented and adapted for fuel efficiency optimization. For this application, the engine state was limited to "on" or "off". While the optimal on-off solution achieved better efficiency than other on-off control methods as expected, non-optimal line control methods achieved even higher efficiency results.

The effect of load profiles was shown to affect the choice of energy management strategy for a particular application. If a vehicle is used for a common application, the energy management strategy that best suits that load profile can be used. The effect of quantization on the solution derived via dynamic programming was also presented. The choice of quantization level changes the possible path sets available to choose from, which in turn affects the value of the dynamic programming solution.

Given the advantage of line control methods over point control methods, it would be worthwhile to improve the dynamic programming algorithm to include multiple engine operation states. The problem formulation and computing requirements become more complicated for additional engine states, but the resulting optimal line control may be of academic and commercial interest. This expansion of the dynamic programming problem is left for future work.

It is also suggested that further analysis be performed on the effect of quantization on dynamic programming results. While this project has shown that the effect exists and has commented on how it affects results, the effect of quantization could be characterized in more depth.

A given vehicle has typical uses, and these various uses can be characterized by representative load profiles. For the John Deere Gator, these load profiles could represent common vehicle tasks such as on-road transport, trail riding, towing, blading, and stump pulling. Using dynamic programming, an optimal control law could be developed for each characteristic load profile. A learning algorithm could be implemented to learn how the driver is actively using the vehicle. With an estimate of what type of task the driver is executing, the controller could operate using the optimal control law for that task. For a vehicle where simpler energy management strategies do not achieve high enough efficiencies, this learning method could achieve near-optimal results.

# APPENDIX A

# SIMULINK MODELS

## A.1 Top Level Model

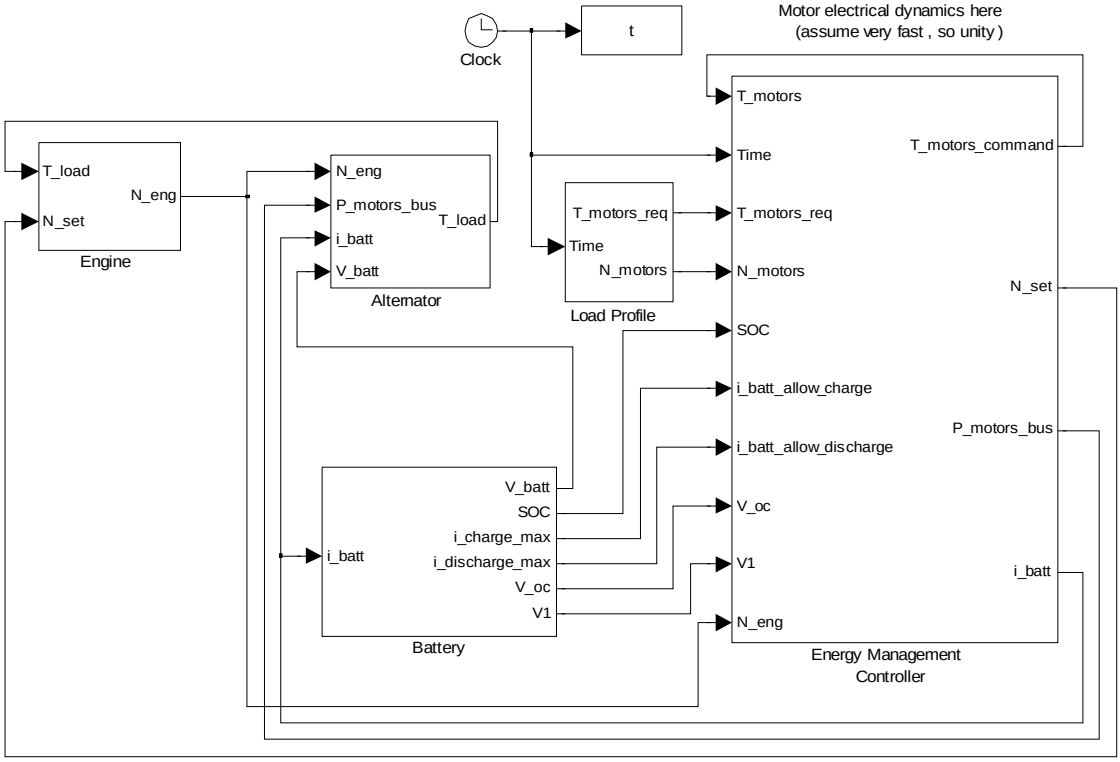


**Figure A.1: Top level model**

## A.2 Engine Model

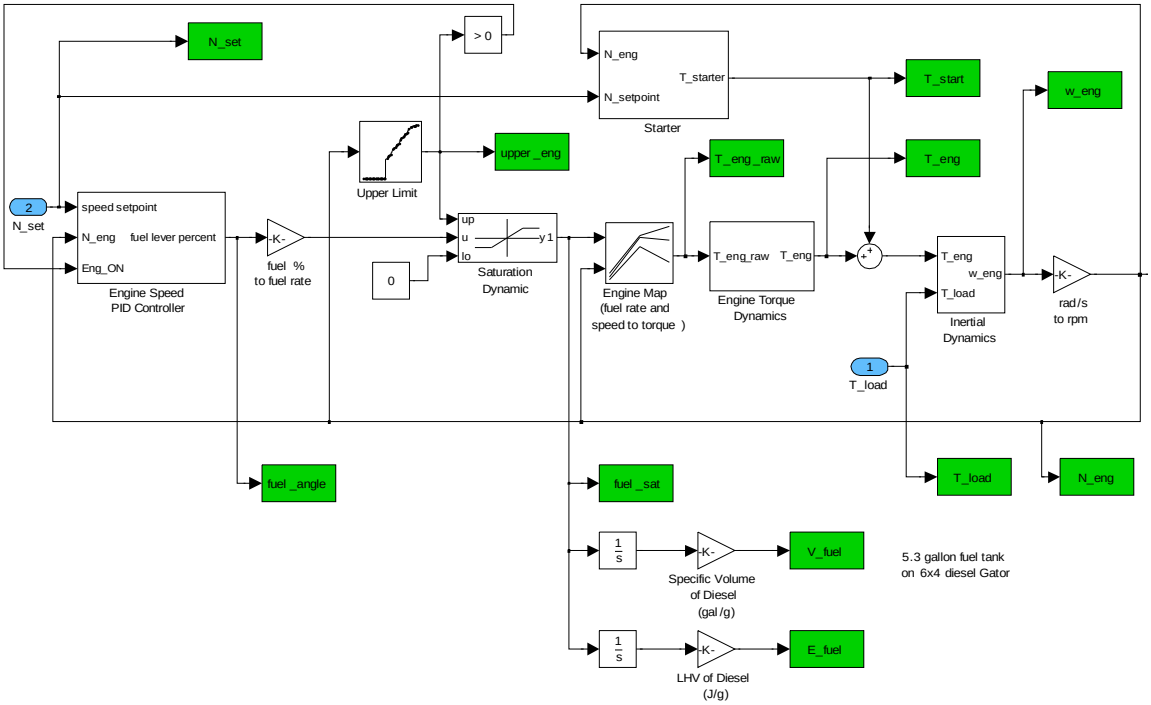


**Figure A.2: Engine subsystem model**

## A.3 Alternator Model

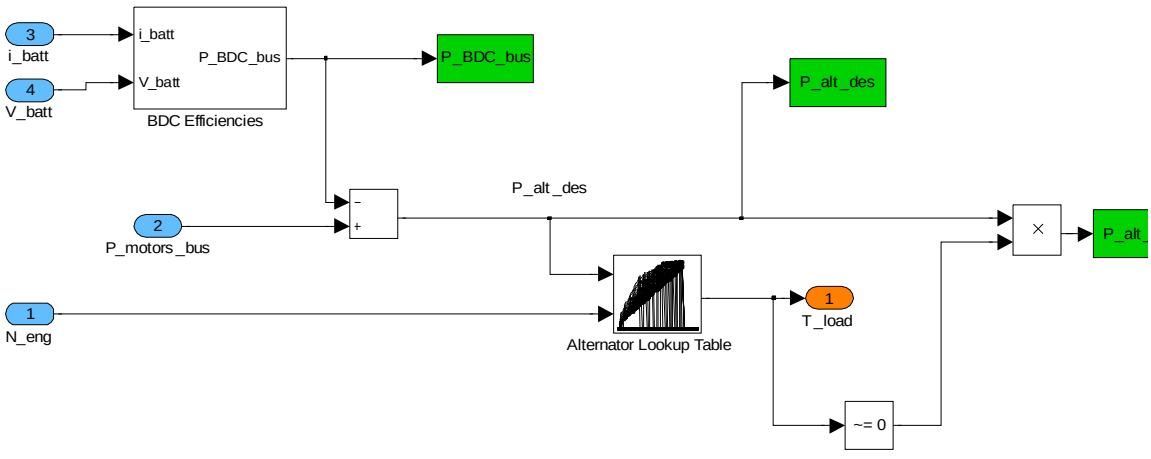


**Figure A.3: Alternator subsystem model**

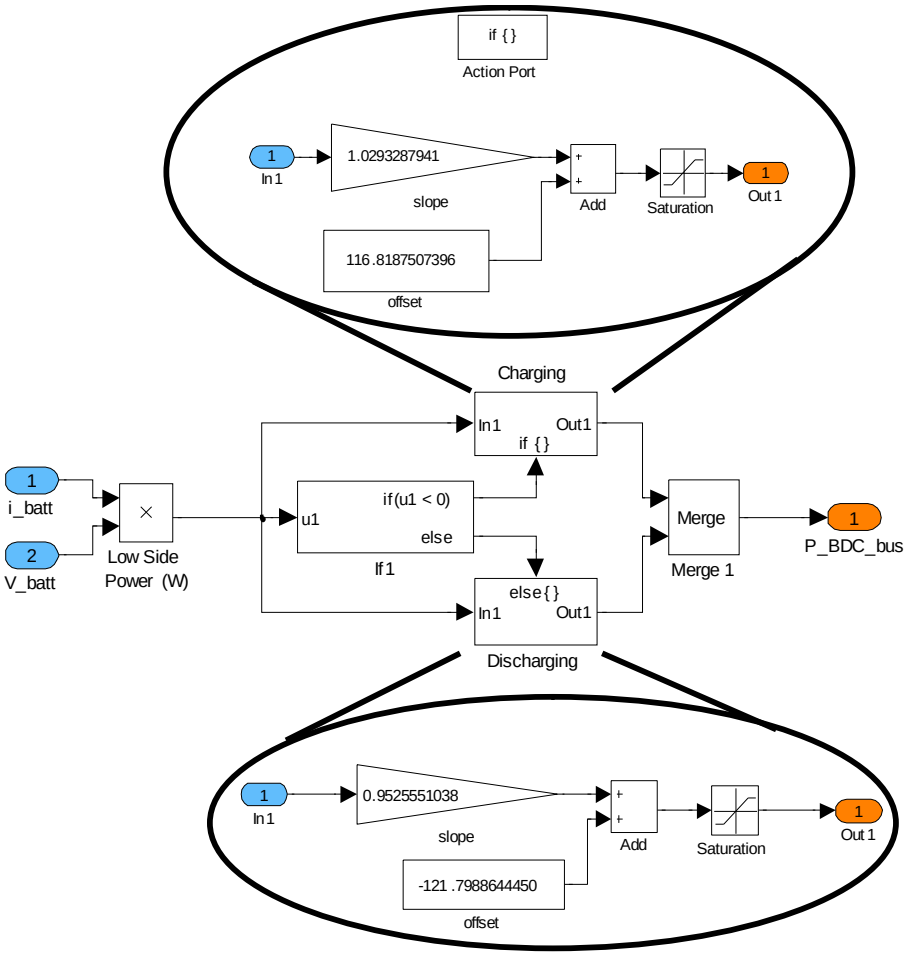


**Figure A.4: BDC (bi-directional converter) efficiency subsystem model**

## A.4 Battery Model

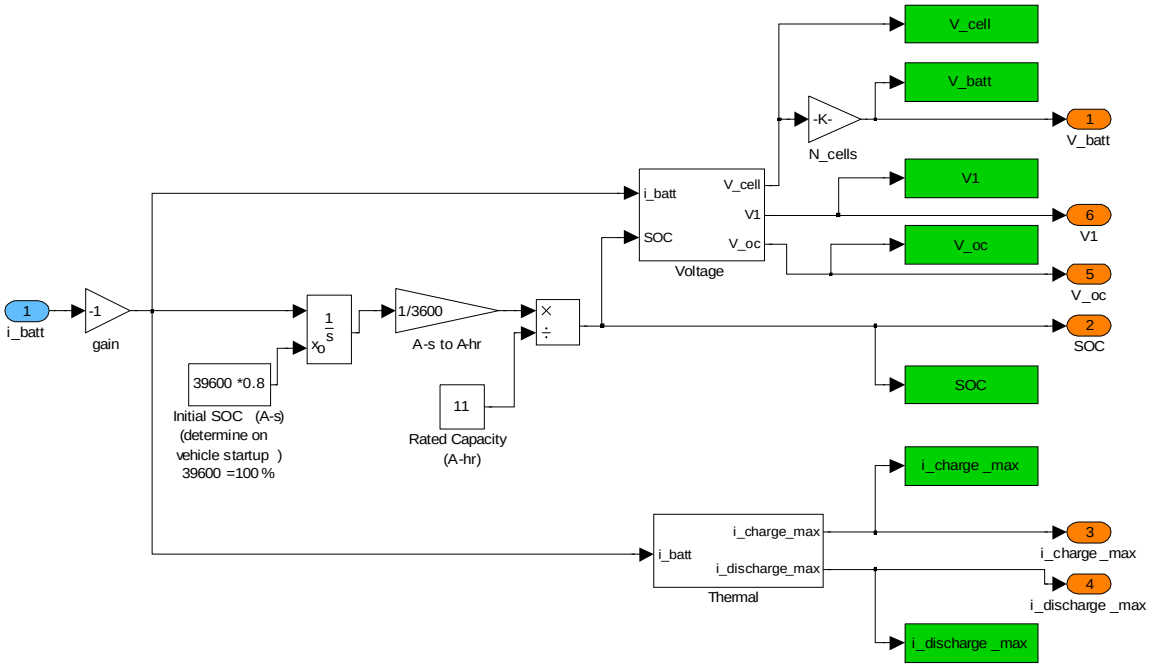


**Figure A.5: Battery subsystem model**

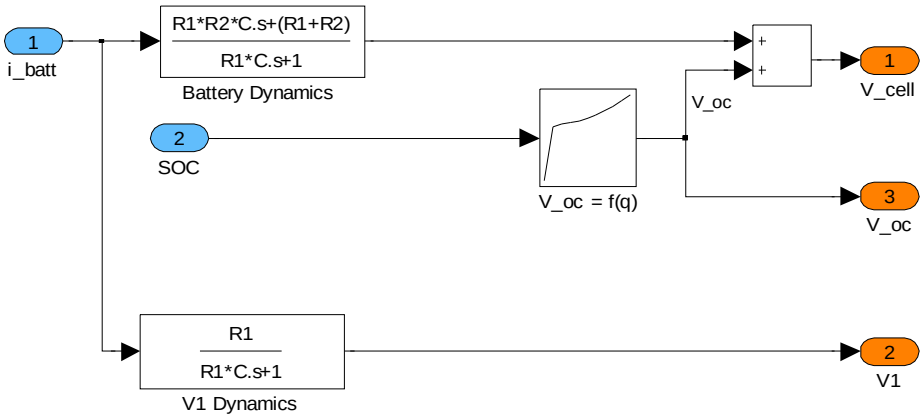


**Figure A.6: Battery voltage subsystem model**

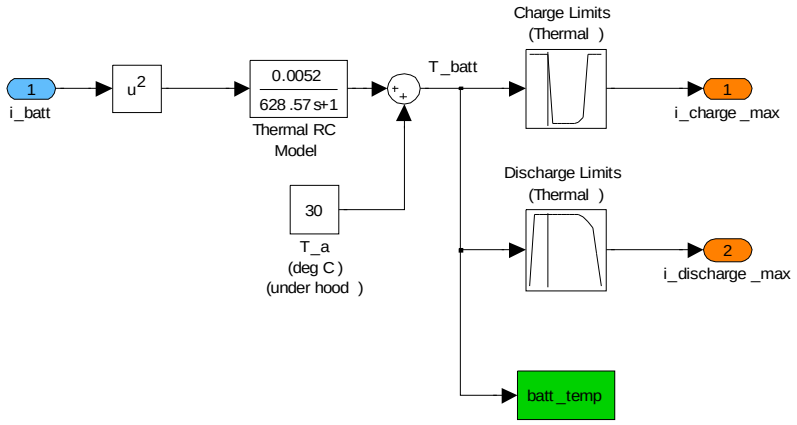


**Figure A.7: Battery thermal subsystem model**

## A.5 Load Profile Model

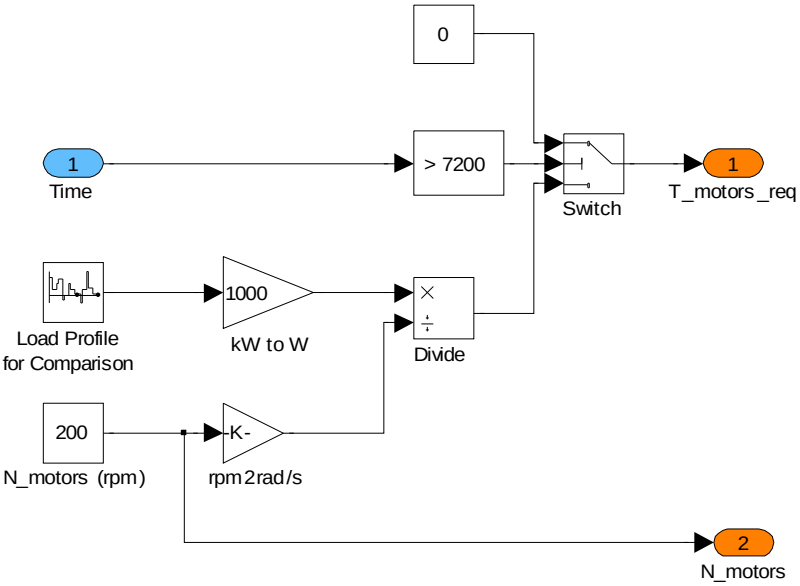


**Figure A.8: Load profile subsystem model**

## A.6 Energy Management System Models

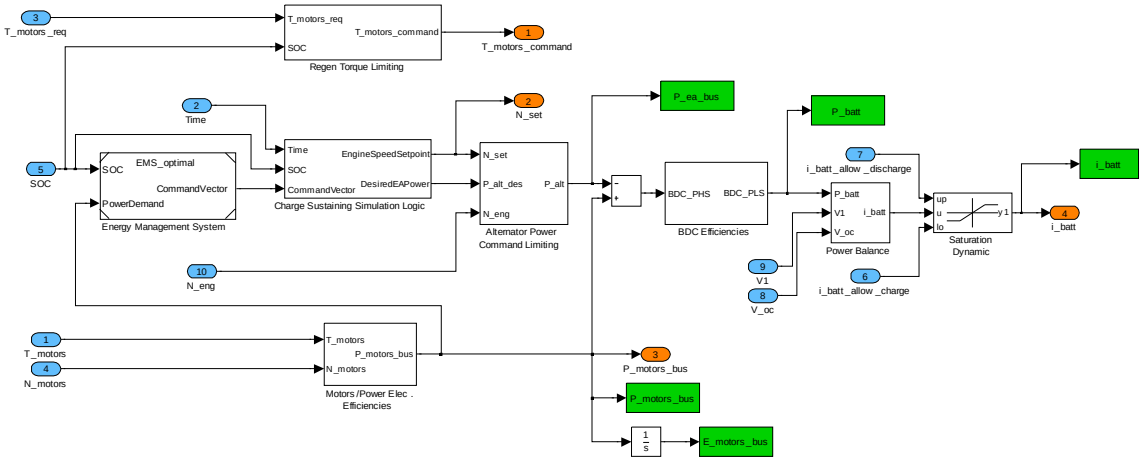


**Figure A.9: Top level EMS model**

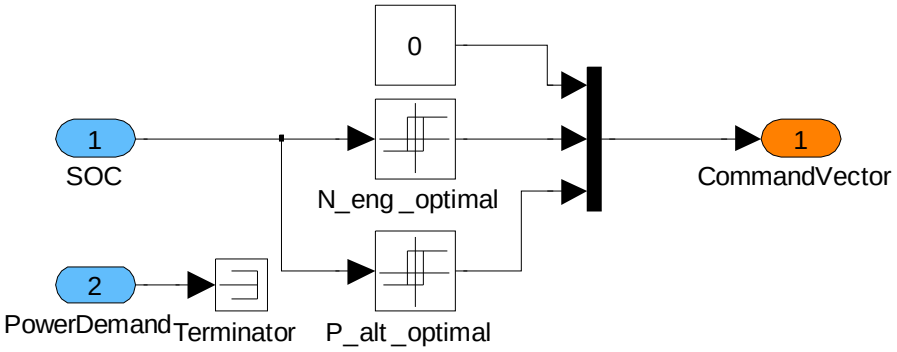


**Figure A.10: Thermostat EMS model**

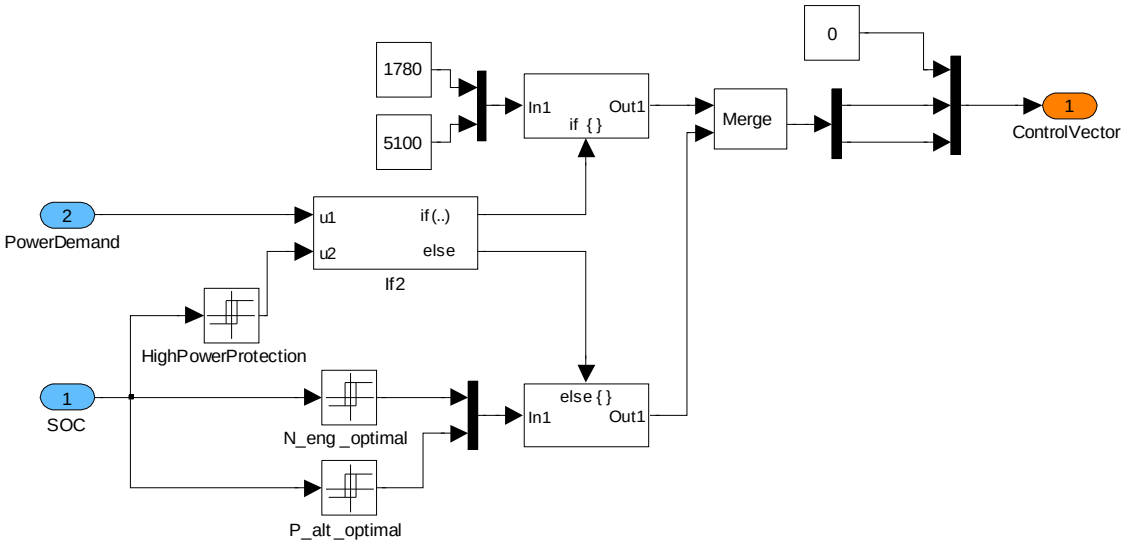


**Figure A.11: Point control EMS model**

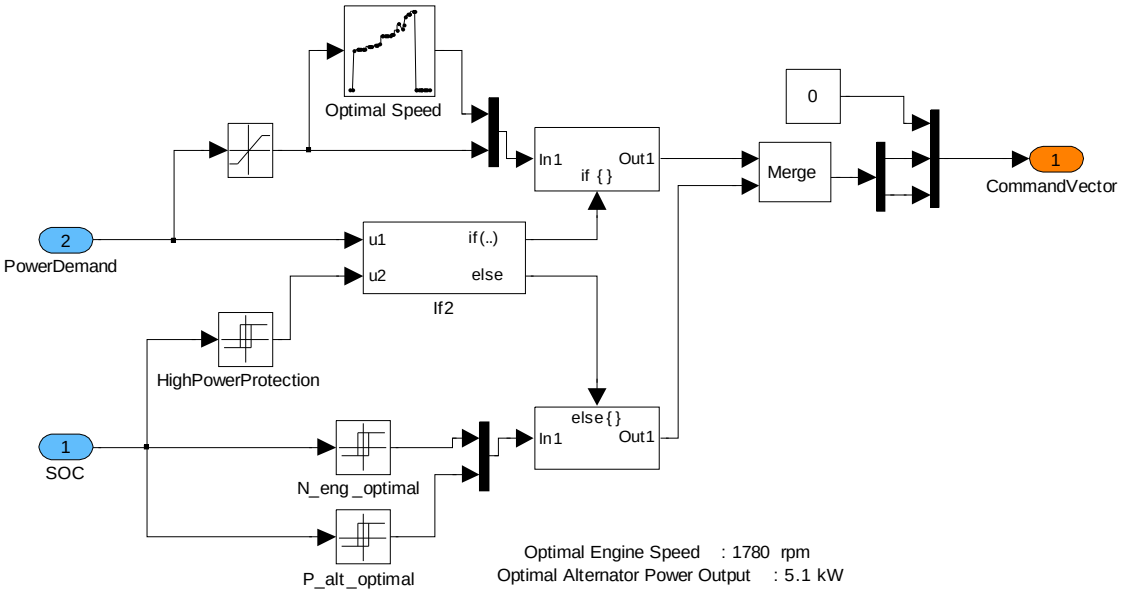


**Figure A.12: Engine optimized line control EMS model**

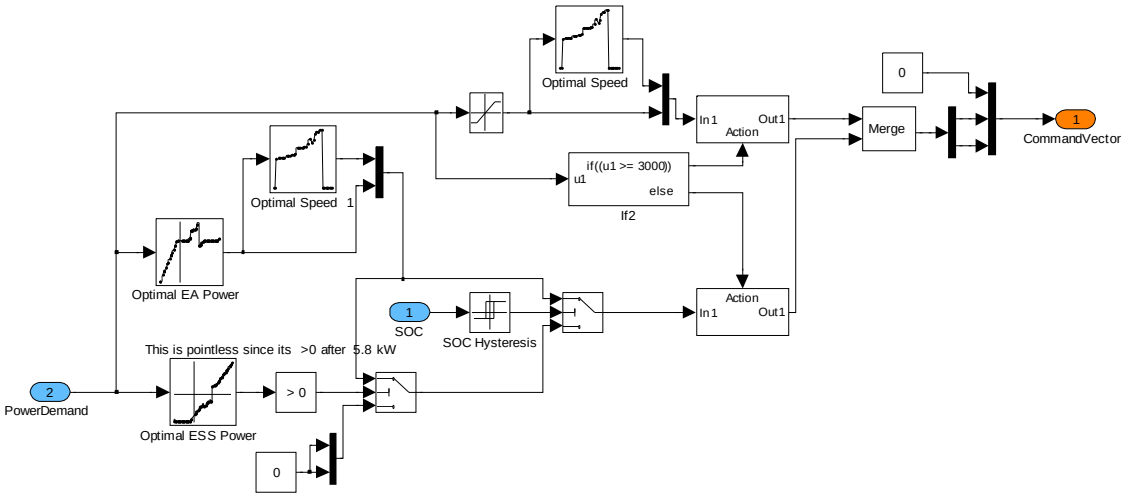


**Figure A.13: System optimized line control EMS model**

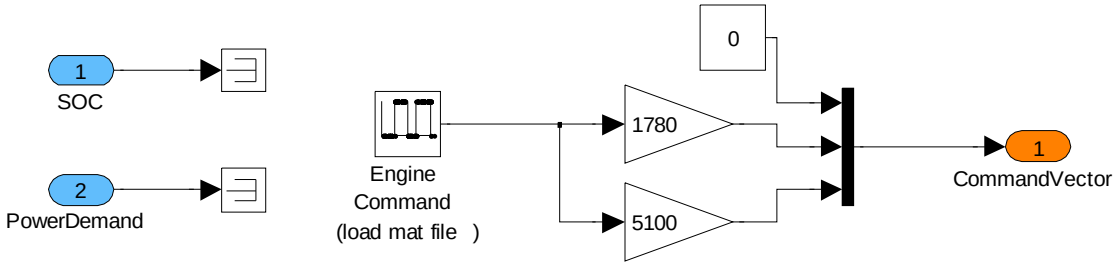


**Figure A.14: Optimal control EMS model**

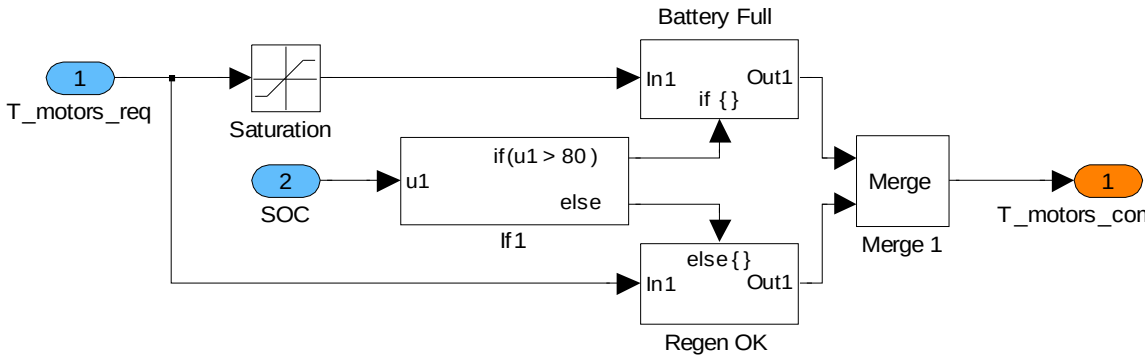


**Figure A.15: Regenerative torque limiting model**

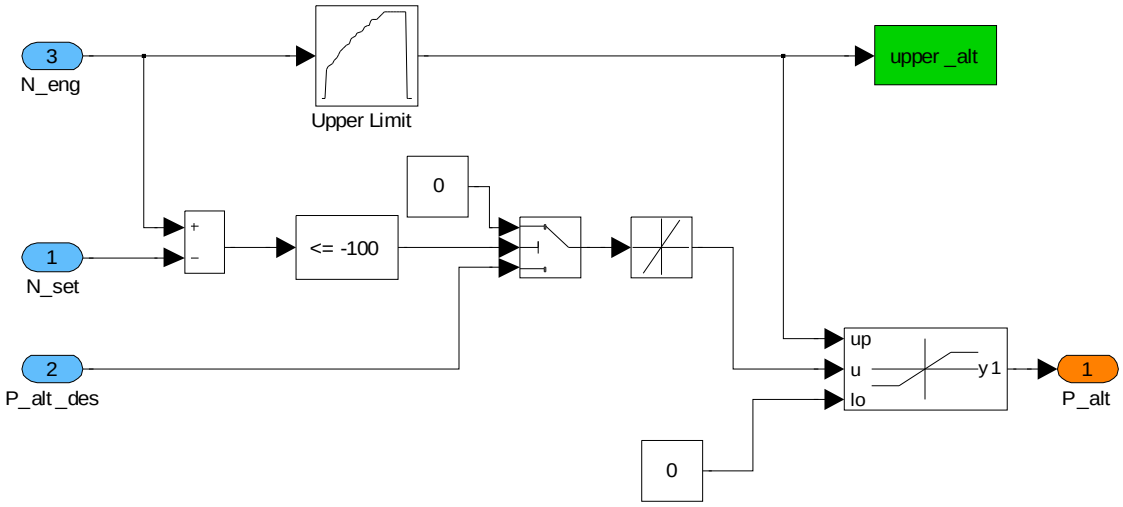


**Figure A.16: Alternator power command limiting model**

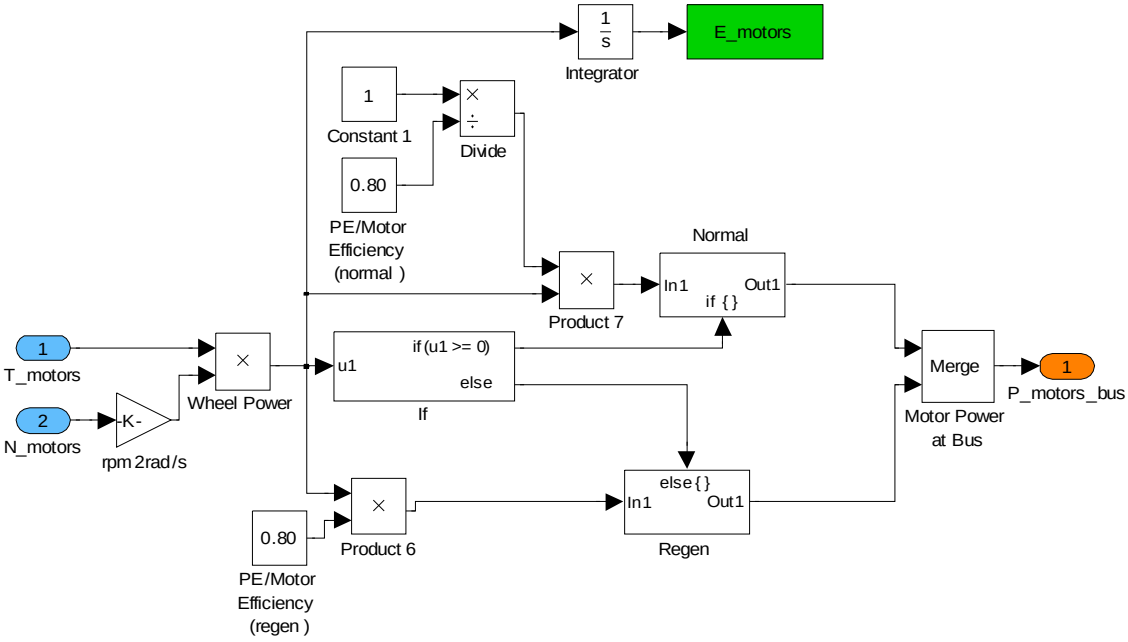


**Figure A.17: Power electronics efficiency model**

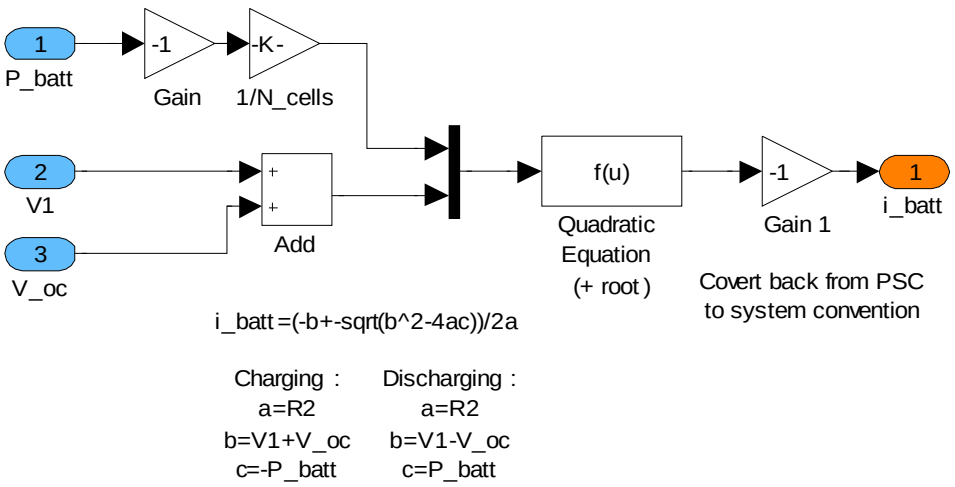


**Figure A.18: Power balance model**

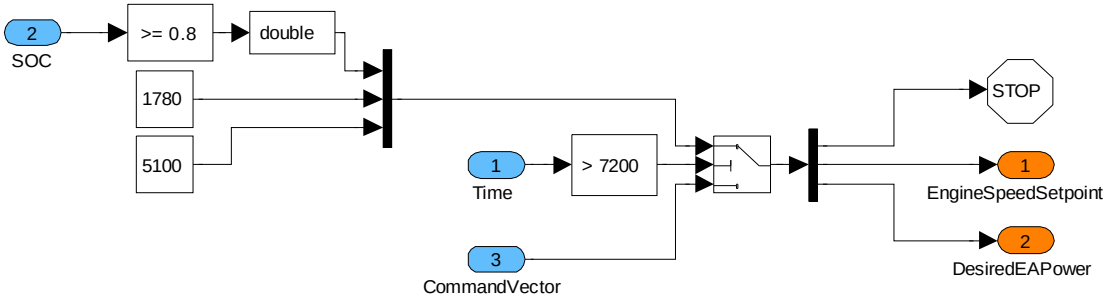


**Figure A.19: Charge sustaining logic model**

# APPENDIX B

# MATLAB CODE

## B.1 Model Initialization Code

```
%% Clean Up
% clear all
% close all
clc

%% Battery Parameters for Battery Transfer Function
R=11e-3; %total internal resistance, Ohms
R1=R/2;
R2=R/2;
C=1000; %effective capacitance, F (for transient response of charge
double
%layers in the porous electrodes)
Cb=39600/(5/.9); %approximate capacitance of the battery (linear
version)
N_cells=72; %number of cells in battery pack

%% Create Battery Voltage Lookup Table (x: SOC, y: Current, z: Voltage)
Batt_data
BDC_data

x=[V_0C(:,1);V_1C(:,1);V_3C(:,1);V_5C(:,1);V_8C(:,1);V_10C(:,1)];
x=abs(x-11)/11; %transform from discharge capacity (11 to 0 A-hr) to
SOC (0 to 1)
y=[zeros(length(V_0C),1);11*ones(length(V_1C),1);33*ones(length(V_3C),1
);55*ones(length(V_5C),1);88*ones(length(V_8C),1);110*ones(length(V_10C
),1)];
z=[V_0C(:,2);V_1C(:,2);V_3C(:,2);V_5C(:,2);V_8C(:,2);V_10C(:,2)];
x_rng=0.01:0.01:0.99;
y_rng=0:1:110;
[xi,yi]=meshgrid(x_rng,y_rng);
zi = griddata(x,y,z,xi,yi);
batt_rows=y_rng;
batt_cols=x_rng;
%Replace NaNs with 2.7 (min voltage)
for i = 1:size(xi,1)
    for j=1:size(yi,2)
        if strncmp(num2str(zi(i,j)),'NaN',3)'
            batt_table(i,j)=2.7;
        else
            batt_table(i,j)=zi(i,j);
        end
    end
end
```

```
%% Engine Efficiency lookup table (Inputs: N_eng, P_eng; Output:
eta_eng)
eng_data2

x=eng_spd;
y=eng_pwr/1000; %(kW)
z=eng_eff;
x_rng=1100:50:3000;
y_rng=0:0.5:11;
[xi,yi]=meshgrid(x_rng,y_rng);
zi = griddata(x,y,z,xi,yi,'linear',{'Pp'}); %Pp clears a warning

eng_rows=y_rng;
eng_cols=x_rng;
%Replace NaNs with zeros (so a zero efficiency means OOR)
for i = 1:size(xi,1)
    for j=1:size(yi,2)
        if strncmp(num2str(zi(i,j)),'NaN',3)'
            eng_table(i,j)=0;
        else
            eng_table(i,j)=zi(i,j);
        end
    end
end


%% Alternator Efficiency lookup table (Inputs: N_alt, P_alt; Output:
eta_alt)
alt_data

x=alt_spd;
y=alt_pwr;
z=alt_eff;
x_rng=2200:100:6000;
y_rng=0:100:10000;
[xi,yi]=meshgrid(x_rng,y_rng);
zi = griddata(x,y,z,xi,yi);

alt_eta_rows=y_rng;
alt_eta_cols=x_rng;
%Replace NaNs with zeros (so a zero efficiency means OOR)
for j = 1:length(x_rng)
    for i=1:length(y_rng)
        if strncmp(num2str(zi(i,j)),'NaN',3)
            alt_eta_table(i,j)=0;
        else
            alt_eta_table(i,j)=zi(i,j);
        end
    end
end

%% Alternator Load Torque lookup table (Inputs: N_eng, P_alt; Output:
T_load=T_eng)
alt_data
eng_data %eng_data, not eng_data2 intentionally
```

```
x=eng_spd;
y=alt_pwr;
z=eng_trq;
x_rng=1100:50:3000;
y_rng=0:100:10000;
[xi,yi]=meshgrid(x_rng,y_rng);
zi = griddata(x,y,z,xi,yi);

alt_trq_rows=y_rng;
alt_trq_cols=x_rng;
%Replace NaNs with zeros (so a zero efficiency means OOR)
upper_bnd_alt=zeros(1,length(x_rng));
lower_bnd_alt=zeros(1,length(x_rng));
for j = 1:length(x_rng)
    lower_set=0;
    upper_set=0;
    for i=1:length(y_rng)
        if strncmp(num2str(zi(i,j)),'NaN',3)
            alt_trq_table(i,j)=0;
            if (lower_set == 1) & (upper_set == 0)
                upper_bnd_alt(j) = y_rng(i-1);
                upper_set=1;
            end
        else
            alt_trq_table(i,j)=zi(i,j);
            if lower_set == 0
                lower_bnd_alt(j) = y_rng(i);
                lower_set=1;
            end
        end
    end
end

%% Bring the upper bound away from edges to prevent bad interpolations
upper_bnd_alt(2)=0;
for i=1:length(upper_bnd_alt)
    if upper_bnd_alt(i) ~= 0
        upper_bnd_alt(i)=upper_bnd_alt(i)-100;
    end
end



%% Create Combined Eng/Alt lookup table grid for simulink
%(combined eff. vs. eng spd and alt output pwr)
eng_alt_data

x=eng_spd;
y=alt_pwr/1000; %(kW)
z=eng_alt_eff;
x_rng=1100:50:3000;
y_rng=0:0.5:10;
[xi,yi]=meshgrid(x_rng,y_rng);
zi = griddata(x,y,z,xi,yi);
```

```
eng_alt_rows=y_rng;
eng_alt_cols=x_rng;
%Replace NaNs with zeros (so a zero efficiency means OOR)
for i = 1:size(xi,1)
    for j=1:size(yi,2)
        if strncmp(num2str(zi(i,j)),'NaN',3)'
            eng_alt_table(i,j)=0;
        else
            eng_alt_table(i,j)=zi(i,j);
        end
    end
end



%% Create Engine lookup table grid for simulink (Torque vs. Speed and
Fuel Consumption Rate)
eng_data2

x=eng_spd;
y=eng_FC;
z=eng_trq;
x_rng=0:25:2850;
y_rng=0:0.02:1;
[xi,yi]=meshgrid(x_rng,y_rng);
zi = griddata(x,y,z,xi,yi);

% x=eng_spd;
% y=eng_trq;
% z=eng_FC;
% x_rng=0:25:2850;
% y_rng=-40:1:40;
% [xi,yi]=meshgrid(x_rng,y_rng);
% zi = griddata(x,y,z,xi,yi);
% figure
% [Cs,hs]=contour(xi,yi,zi,[0:0.1:0.8]);
% clabel(Cs,hs,[0:0.1:0.8]);
% zero_FC=find(eng_FC==0);
% hold on
% plot([0;x(zero_FC)],[0;y(zero_FC)])
% xlabel('Engine Speed (RPM)')
% ylabel('Engine Output Torque')

eng_trq_rows=y_rng;
eng_trq_cols=x_rng;
%Replace NaNs (NaN means engine off, so damping torque)
upper_bnd_eng=zeros(1,length(x_rng));
lower_bnd_eng=zeros(1,length(x_rng));
for j = 1:length(x_rng)
    lower_set=0;
    upper_set=0;
    for i=1:length(y_rng)
        if strncmp(num2str(zi(i,j)),'NaN',3)
            eng_trq_table(i,j)=-0.0132285714285714*x_rng(j);
```

```
            if (lower_set == 1) & (upper_set == 0)
                upper_bnd_eng(j) = y_rng(i-1);
                upper_set=1;
            end
        else
            eng_trq_table(i,j)=zi(i,j);
            if lower_set == 0
                lower_bnd_eng(j) = y_rng(i);
                lower_set=1;
            end
        end
    end
end
upper_bnd_eng(47)=0;
for i=1:length(upper_bnd_eng)
    if upper_bnd_eng(i) ~= 0
        upper_bnd_eng(i)=upper_bnd_eng(i)-0.02;
    end
end

%% Find optimal speed for a given alternator power output (for use in
% line-based control

eng_alt_data

%Combined Efficiency vs. Engine Speed and Alternator Output Power
x=alt_pwr/1000; %(kW)
y=eng_spd;
z=eng_alt_eff;
x_rng=0:0.1:10;
y_rng=1100:10:3000;
[xi,yi]=meshgrid(x_rng,y_rng);
zi = griddata(x,y,z,xi,yi,'cubic');

%Find optimal engine speed for each given alternator power output
for i = 1:length(x_rng)
    eta_opt(i)=max(zi(:,i));
    opt_ind = find(zi(:,i) == eta_opt(i));
    if isempty(opt_ind)
        N_opt(i)=0;
    else
        N_opt(i) = y_rng(opt_ind);
    end
end

P_ea=x_rng;

%% Load System Efficiency Lookup Table Data (for line-based control)
%P_m is input (motor power in W)
%P_EA_opt is the output (engine power to achieve optimal system
efficiency)
load system_efficiency_data

%% Load Engine commands for optimal control
load engine_command
```

## B.2 Dynamic Programming Backwards Recursion Code

```
%% DESCRIPTION
% This uses dynamic programming to determine the optimal power split
for
% a series hybrid given a load profile.

%% Clean up
%clear all
%clc

%% Load relevant data (lookup tables and quantization levels)
trans_map_fitting

%% Choose final (and starting) SOC
SOC_initial=0.8;
SOC_final=0.8;
cost_secondary=1;

%% DEFINE LOAD PROFILE
% T=60; %seconds, sampling time
%sample the load profile at the given sampling rate
% load_profile=[6 2 4 5 -1 0 4 2 1 1 -2 2 8 3 1 ...
%               6 2 4 5 -1 0 4 2 1 1 -2 2 7 3 1 ...
%               6 2 4 5 -1 0 4 2 1 1 -2 2 8 3 1 ...
%               6 2 4 5 -1 0 4 2 1 1 -2 2 8 3 1 ...
%               6 2 4 5 -1 0 4 2 1 1 -2 2 8 3 1 ...
%               6 2 4 5 -1 0 4 2 1 1 -2 2 8 3 1 ...
%               6 2 4 5 -1 0 4 2 1 1 -2 2 8 3 1 ...
%               6 2 4 5 -1 0 4 2 1 1 -2 2 8 3 1 1 0];

% load_profile=[4 0 2 3 -3 -2 2 0 -1 -1 -4 0 6 1 -1 ...
%               4 0 2 3 -3 -2 2 0 -1 -1 -4 0 6 1 -1 ...
%               4 0 2 3 -3 -2 2 0 -1 -1 -4 0 6 1 -1 ...
%               4 0 2 3 -3 -2 2 0 -1 -1 -4 0 6 1 -1 ...
%               4 0 2 3 -3 -2 2 0 -1 -1 -4 0 6 1 -1 ...
%               4 0 2 3 -3 -2 2 0 -1 -1 -4 0 6 1 -1 ...
%               4 0 2 3 -3 -2 2 0 -1 -1 -4 0 6 1 -1 ...
%               4 0 2 3 -3 -2 2 0 -1 -1 -4 0 6 1 -1 0];

load_profile=[4 4 4 4 -2 -2 -2 -2 -2 -2 -2 6 6 6 6 ...
              4 4 4 4 -2 -2 -2 -2 -2 -2 -2 6 6 6 6 ...
              4 4 4 4 -2 -2 -2 -2 -2 -2 -2 6 6 6 6 ...
              4 4 4 4 -2 -2 -2 -2 -2 -2 -2 6 6 6 6 ...
              4 4 4 4 -2 -2 -2 -2 -2 -2 -2 6 6 6 6 ...
              4 4 4 4 -2 -2 -2 -2 -2 -2 -2 6 6 6 6 ...
              4 4 4 4 -2 -2 -2 -2 -2 -2 -2 6 6 6 6 ...
              4 4 4 4 -2 -2 -2 -2 -2 -2 -2 6 6 6 6 0];

              %example profile for development, 2 hours, in kW
% load_profile=[1*ones(1,400), 0];
```

```
%%  INITIALIZE ALL VARIABLES AND MATRICES
engine_state=[0 1]; %possible engine states (off or on, transitions are
accounted for in the cost matrix)
SOC_state=0:SOC_q:1; %state-of-charge quantization
PD_input=-6:PD_q:12; %not part of state vector, but a necessary input

%determine relevant matrix dimensions:
m=length(SOC_state)*length(engine_state);
n=length(load_profile);
o=2; %length of state vector (SOC, engine)

%create state matrix (m-by-n-by-o)
state=zeros(m,n,o);
state_col=[SOC_state' ones(size(SOC_state'));SOC_state'
zeros(size(SOC_state'))];
for i=1:n
    state(:,i,:)=state_col;
end

%PAST
%Determine starting and ending indecies
ind_o=find(abs(state(:,1,1)-SOC_initial)<1e-6 & state(:,1,2)==0);

%Initialize cost matrix (m-by-n-by-m)
% first two indices represent the originating node
% the third index represents the row value of the desination node
c=(zeros(m,n,m));
for i=1:m
    if state(i,n,1) >= 0.80;%SOC_final %PAST
        c(i,n,1)=0;
    else
        c(i,n,:)=NaN;
    end
end

possible_nodes_back=zeros(m,n);
for i=1:m
    if state(i,n,1) >= 0.80;%SOC_final %PAST
        possible_nodes_back(i,n)=1;
    else
        possible_nodes_back(i,n)=NaN;
    end
end

%% COST MATRIX CREATION
% Now determine nodes that exist at each stage (engine/SOC combinations
that satisfy
% the nodes in the following stage) and determine associated cost
for j=n:-1:2
    for i=1:m
%         if ~isnan(possible_nodes_back(i,j))

            socnew=find(abs(SOC_state-state(i,j,1))<1e-6);
            pd=find(abs(PD_input-load_profile(j-1))<1e-6);
            socold_OFFOFF=SOC_old_OFFOFF_output_newPD(pd,socnew);
```

```
            socold_OFFON=SOC_old_OFFON_output_newPD(pd,socnew);
            socold_ONON=SOC_old_ONON_output_newPD(pd,socnew);
            socold_ONOFF=SOC_old_ONOFF_output_newPD(pd,socnew);
            cost_OFFOFF=E_total_OFFOFF_output_newPD(pd,socnew);
            cost_OFFON=E_total_OFFON_output_newPD(pd,socnew);
            cost_ONON=E_total_ONON_output_newPD(pd,socnew);
            cost_ONOFF=E_total_ONOFF_output_newPD(pd,socnew);

            %Round lookup table outputs to state matrix grid
            for g=1:m
                if state(i,j,2) == 0
                    if (abs(state(g,j-1,1)-socold_OFFOFF) <= SOC_q/2)
&& (state(g,j-1,2) == 0)
                        possible_nodes_back(g,j-1)=1;
                        c(g,j-1,i)=0;
                    elseif (abs(state(g,j-1,1)-socold_ONOFF) <=
SOC_q/2) && (state(g,j-1,2) == 1) %ONOFF case
                        possible_nodes_back(g,j-1)=1;
                        c(g,j-1,i)=cost_ONOFF+cost_secondary; %PAST
                    else
                        if possible_nodes_back(g,j-1)==0
                            possible_nodes_back(g,j-1)=NaN; %don't
write over previous entries
                        end
                        c(g,j-1,i)=NaN;
                    end
                elseif state(i,j,2) == 1
                    if (abs(state(g,j-1,1)-socold_OFFON) <= SOC_q/2) &&
(state(g,j-1,2) == 0)
                        possible_nodes_back(g,j-1)=1;
                        c(g,j-1,i)=cost_ONON+cost_secondary; %PAST
                    elseif (abs(state(g,j-1,1)-socold_ONON) <= SOC_q/2)
&& (state(g,j-1,2) == 1)
                        possible_nodes_back(g,j-1)=1;
                        c(g,j-1,i)=cost_ONON;
                    else
                        if possible_nodes_back(g,j-1)==0
                            possible_nodes_back(g,j-1)=NaN; %don't
write over previous entries
                        end
                        c(g,j-1,i)=NaN;
                    end
                end
%                 %Add cost penalty if final SOC is not SOC_final
%                 if j == n
%                     c(g,j-1,i) = c(g,j-1,i) + (SOC_final -
state(i,j,1))*4.1e+007; %adjust final cost depending on deviation from
SOC_final
%                 end
            end
%         else
%             c(:,j-1,i)=NaN; %if node is unreachable, all costs to
that destination are NaN
%         end
    end
end
```

```
%% DYNAMIC PROGRAMMING RECURSIVE ALGORITHM
%Initialize nodal path cost matrix
f=zeros(m,n);

%Define last path cost (no movement, so no cost)
for i=1:m
    f(i,n)=c(i,n,1);
end

%Dynamic Programming recursive algorithm
%Yields a matrix showing the cost from the given node to the final node
path_next=NaN*ones(m,n);
for j=n-1:-1:1
    for i=1:m
        min_vector=zeros(1,m);
        for k=1:m
            min_vector(k)=c(i,j,k)+f(k,j+1);
        end
        f(i,j)=min(min_vector);
        if ~isempty(find(min_vector==min(min_vector), 1))
            %Minimize engine state changes
            path = find(min_vector==min(min_vector));
            for g=1:length(path)
                if (abs(state(path(g),j+1,2) - state(i,j,2)) < 1e-6)
%if engine state remains constant
                    path_next(i,j)=path(g);
                end
            end
            if isnan(path_next(i,j))
                path_next(i,j)=path(1);
            end
        else
            path_next(i,j)=NaN;
        end
    end
end


%% DETERMINE OPTIMAL PATH
%Path vector: yields row index of nodes to take for optimal path
optimal_path=ones(1,n);

%PAST
optimal_path(1)=ind_o;

for j=1:n-1
    optimal_path(j+1)=path_next(optimal_path(j),j);
end

optimal_path_plot=optimal_path;
engine_command=ones(size(optimal_path));
for i=1:length(optimal_path_plot)
    if optimal_path_plot(i) > m/2
        optimal_path_plot(i)=optimal_path_plot(i)-(m/2);
```

```
        engine_command(i)=0;
    end
end

%shift engine command to FUTURE sense for implementation in simulink
engine_command=[engine_command(2:length(engine_command)) 0];

%% PLOT RESULTS
optimal_cost=f(optimal_path(1),1);
disp(['Optimal Cost = ' num2str(optimal_cost) ' kJ'])

figure
plot(0:n-1,engine_command,0:n-1,(optimal_path_plot-1)*SOC_q)
```

## B.3 Dynamic Programming Transition Map Creation Code

```
%% Clean Up and Setup
clear all
clc

SOC_q=0.005; %SOC quantization level
PD_q=1; %power demand quantization level

%%%%%%%%%%%%%%%%%%%%%%%
%% For OFFON map Only
load SOC_new_OFFON

%First form all 5 column vectors (plot using 'plot3')
SOC_old_OFFON=SOC_OFFON_neg6(:,1);
SOC_new_OFFON=SOC_OFFON_neg6(:,2);
E_total_OFFON=SOC_OFFON_neg6(:,3)+SOC_OFFON_neg6(:,4); %fuel+starter
power_demand_OFFON=-6*ones(length(SOC_OFFON_neg6(:,1)),1);

for i=5:-1:1

SOC_old_OFFON=[SOC_old_OFFON;eval(['SOC_OFFON_neg',num2str(i),'(:,1)'])
];

SOC_new_OFFON=[SOC_new_OFFON;eval(['SOC_OFFON_neg',num2str(i),'(:,2)'])
];

E_total_OFFON=[E_total_OFFON;eval(['SOC_OFFON_neg',num2str(i),'(:,3)'])
+eval(['SOC_OFFON_neg',num2str(i),'(:,4)'])];
    power_demand_OFFON=[power_demand_OFFON;-
i*ones(length(SOC_OFFON_neg6(:,1)),1)];
end

for i=0:12

SOC_old_OFFON=[SOC_old_OFFON;eval(['SOC_OFFON_',num2str(i),'(:,1)'])];
```

```
SOC_new_OFFON=[SOC_new_OFFON;eval(['SOC_OFFON_',num2str(i),'(:,2)'])];

E_total_OFFON=[E_total_OFFON;eval(['SOC_OFFON_',num2str(i),'(:,3)'])+ev
al(['SOC_OFFON_',num2str(i),'(:,4)'])];

power_demand_OFFON=[power_demand_OFFON;i*ones(length(SOC_OFFON_neg6(:,1
)),1)];
end

SOC_delta_OFFON=SOC_new_OFFON-SOC_old_OFFON;

%Create lookup table parameters (plot using 'surf' command)
%Inputs
SOC_old_OFFON_input=[0:SOC_q:1];
power_demand_OFFON_input=[-6:PD_q:12];
SOC_new_OFFON_input=[0:SOC_q:1];

[xi,yi]=meshgrid(SOC_old_OFFON_input,power_demand_OFFON_input);
SOC_new_OFFON_output_oldPD =
griddata(SOC_old_OFFON,power_demand_OFFON,SOC_new_OFFON,xi,yi,'cubic');

[xi,yi]=meshgrid(SOC_new_OFFON_input,power_demand_OFFON_input);
SOC_old_OFFON_output_newPD =
griddata(SOC_new_OFFON,power_demand_OFFON,SOC_old_OFFON,xi,yi,'cubic');

[xi,yi]=meshgrid(SOC_old_OFFON_input,power_demand_OFFON_input);
SOC_delta_OFFON_output_oldPD =
griddata(SOC_old_OFFON,power_demand_OFFON,SOC_delta_OFFON,xi,yi,'cubic'
);

[xi,yi]=meshgrid(SOC_new_OFFON_input,power_demand_OFFON_input);
E_total_OFFON_output_newPD =
griddata(SOC_new_OFFON,power_demand_OFFON,E_total_OFFON,xi,yi,'cubic');

[xi,yi]=meshgrid(SOC_old_OFFON_input,power_demand_OFFON_input);
E_total_OFFON_output_oldPD =
griddata(SOC_old_OFFON,power_demand_OFFON,E_total_OFFON,xi,yi,'cubic');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% For ONOFF map Only
load SOC_new_ONOFF

%First form all 5 column vectors (plot using 'plot3')
SOC_old_ONOFF=SOC_ONOFF_neg6(:,1);
SOC_new_ONOFF=SOC_ONOFF_neg6(:,2);
E_total_ONOFF=SOC_ONOFF_neg6(:,3)+SOC_ONOFF_neg6(:,4); %fuel+starter
power_demand_ONOFF=-6*ones(length(SOC_ONOFF_neg6(:,1)),1);

for i=5:-1:1

SOC_old_ONOFF=[SOC_old_ONOFF;eval(['SOC_ONOFF_neg',num2str(i),'(:,1)'])
];
```

```
SOC_new_ONOFF=[SOC_new_ONOFF;eval(['SOC_ONOFF_neg',num2str(i),'(:,2)'])
];

E_total_ONOFF=[E_total_ONOFF;eval(['SOC_ONOFF_neg',num2str(i),'(:,3)'])
+eval(['SOC_ONOFF_neg',num2str(i),'(:,4)'])];
    power_demand_ONOFF=[power_demand_ONOFF;-
i*ones(length(SOC_ONOFF_neg6(:,1)),1)];
end

for i=0:12

SOC_old_ONOFF=[SOC_old_ONOFF;eval(['SOC_ONOFF_',num2str(i),'(:,1)'])];

SOC_new_ONOFF=[SOC_new_ONOFF;eval(['SOC_ONOFF_',num2str(i),'(:,2)'])];

E_total_ONOFF=[E_total_ONOFF;eval(['SOC_ONOFF_',num2str(i),'(:,3)'])+ev
al(['SOC_ONOFF_',num2str(i),'(:,4)'])];

power_demand_ONOFF=[power_demand_ONOFF;i*ones(length(SOC_ONOFF_neg6(:,1
)),1)];
end

SOC_delta_ONOFF=SOC_new_ONOFF-SOC_old_ONOFF;

%Create lookup table parameters (plot using 'surf' command)
%Inputs
SOC_old_ONOFF_input=[0:SOC_q:1];
power_demand_ONOFF_input=[-6:PD_q:12];
SOC_new_ONOFF_input=[0:SOC_q:1];

[xi,yi]=meshgrid(SOC_old_ONOFF_input,power_demand_ONOFF_input);
SOC_new_ONOFF_output_oldPD =
griddata(SOC_old_ONOFF,power_demand_ONOFF,SOC_new_ONOFF,xi,yi,'cubic');

[xi,yi]=meshgrid(SOC_new_ONOFF_input,power_demand_ONOFF_input);
SOC_old_ONOFF_output_newPD =
griddata(SOC_new_ONOFF,power_demand_ONOFF,SOC_old_ONOFF,xi,yi,'cubic');

[xi,yi]=meshgrid(SOC_old_ONOFF_input,power_demand_ONOFF_input);
SOC_delta_ONOFF_output_oldPD =
griddata(SOC_old_ONOFF,power_demand_ONOFF,SOC_delta_ONOFF,xi,yi,'cubic'
);

[xi,yi]=meshgrid(SOC_new_ONOFF_input,power_demand_ONOFF_input);
E_total_ONOFF_output_newPD =
griddata(SOC_new_ONOFF,power_demand_ONOFF,E_total_ONOFF,xi,yi,'cubic');

[xi,yi]=meshgrid(SOC_old_ONOFF_input,power_demand_ONOFF_input);
E_total_ONOFF_output_oldPD =
griddata(SOC_old_ONOFF,power_demand_ONOFF,E_total_ONOFF,xi,yi,'cubic');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% For OFF-OFF map ONLY
```

```
load SOC_new_OFFOFF

%First form all 5 column vectors (plot using 'plot3')
SOC_old_OFFOFF=SOC_OFFOFF_neg6(:,1);
SOC_new_OFFOFF=SOC_OFFOFF_neg6(:,2);
E_total_OFFOFF=SOC_OFFOFF_neg6(:,3)+SOC_OFFOFF_neg6(:,4); %fuel+starter
power_demand_OFFOFF=-6*ones(length(SOC_OFFOFF_neg6(:,1)),1);

for i=5:-1:1

SOC_old_OFFOFF=[SOC_old_OFFOFF;eval(['SOC_OFFOFF_neg',num2str(i),'(:,1)
'])];

SOC_new_OFFOFF=[SOC_new_OFFOFF;eval(['SOC_OFFOFF_neg',num2str(i),'(:,2)
'])];

E_total_OFFOFF=[E_total_OFFOFF;eval(['SOC_OFFOFF_neg',num2str(i),'(:,3)
'])+eval(['SOC_OFFOFF_neg',num2str(i),'(:,4)'])];
    power_demand_OFFOFF=[power_demand_OFFOFF;-
i*ones(size(eval(strcat('SOC_OFFOFF_neg',num2str(i),'(:,1)'))))];

end

for i=0:12

SOC_old_OFFOFF=[SOC_old_OFFOFF;eval(['SOC_OFFOFF_',num2str(i),'(:,1)'])
];

SOC_new_OFFOFF=[SOC_new_OFFOFF;eval(['SOC_OFFOFF_',num2str(i),'(:,2)'])
];

E_total_OFFOFF=[E_total_OFFOFF;eval(['SOC_OFFOFF_',num2str(i),'(:,3)'])
+eval(['SOC_OFFOFF_',num2str(i),'(:,4)'])];

power_demand_OFFOFF=[power_demand_OFFOFF;i*ones(size(eval(strcat('SOC_O
FFOFF_',num2str(i),'(:,1)'))))];
end

SOC_delta_OFFOFF=SOC_new_OFFOFF-SOC_old_OFFOFF;

%Create lookup table parameters (plot using 'surf' command)
%Inputs
SOC_old_OFFOFF_input=[0:SOC_q:1];
power_demand_OFFOFF_input=[-6:PD_q:12];
SOC_new_OFFOFF_input=[0:SOC_q:1];

[xi,yi]=meshgrid(SOC_old_OFFOFF_input,power_demand_OFFOFF_input);
SOC_new_OFFOFF_output_oldPD =
griddata(SOC_old_OFFOFF,power_demand_OFFOFF,SOC_new_OFFOFF,xi,yi,'cubic
');

[xi,yi]=meshgrid(SOC_new_OFFOFF_input,power_demand_OFFOFF_input);
SOC_old_OFFOFF_output_newPD =
griddata(SOC_new_OFFOFF,power_demand_OFFOFF,SOC_old_OFFOFF,xi,yi,'cubic
');
```

```
[xi,yi]=meshgrid(SOC_old_OFFOFF_input,power_demand_OFFOFF_input);
SOC_delta_OFFOFF_output_oldPD =
griddata(SOC_old_OFFOFF,power_demand_OFFOFF,SOC_delta_OFFOFF,xi,yi,'cub
ic');

[xi,yi]=meshgrid(SOC_new_OFFOFF_input,power_demand_OFFOFF_input);
E_total_OFFOFF_output_newPD =
griddata(SOC_new_OFFOFF,power_demand_OFFOFF,E_total_OFFOFF,xi,yi,'cubic
');

[xi,yi]=meshgrid(SOC_old_OFFOFF_input,power_demand_OFFOFF_input);
E_total_OFFOFF_output_oldPD =
griddata(SOC_old_OFFOFF,power_demand_OFFOFF,E_total_OFFOFF,xi,yi,'cubic
');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% For ON-ON map ONLY
load SOC_new_ONON

%First form all 5 column vectors (plot using 'plot3')
SOC_old_ONON=SOC_ONON_neg6(:,1);
SOC_new_ONON=SOC_ONON_neg6(:,2);
E_total_ONON=SOC_ONON_neg6(:,3)+SOC_ONON_neg6(:,4); %fuel+starter
power_demand_ONON=-6*ones(length(SOC_ONON_neg6(:,1)),1);

for i=5:-1:1

SOC_old_ONON=[SOC_old_ONON;eval(['SOC_ONON_neg',num2str(i),'(:,1)'])];

SOC_new_ONON=[SOC_new_ONON;eval(['SOC_ONON_neg',num2str(i),'(:,2)'])];

E_total_ONON=[E_total_ONON;eval(['SOC_ONON_neg',num2str(i),'(:,3)'])+ev
al(['SOC_ONON_neg',num2str(i),'(:,4)'])];
    power_demand_ONON=[power_demand_ONON;-
i*ones(size(eval(strcat('SOC_ONON_neg',num2str(i),'(:,1)'))))];
end

for i=0:12
    SOC_old_ONON=[SOC_old_ONON;eval(['SOC_ONON_',num2str(i),'(:,1)'])];
    SOC_new_ONON=[SOC_new_ONON;eval(['SOC_ONON_',num2str(i),'(:,2)'])];

E_total_ONON=[E_total_ONON;eval(['SOC_ONON_',num2str(i),'(:,3)'])+eval(
['SOC_ONON_',num2str(i),'(:,4)'])];

power_demand_ONON=[power_demand_ONON;i*ones(size(eval(strcat('SOC_ONON_
',num2str(i),'(:,1)'))))];
end

SOC_delta_ONON=SOC_new_ONON-SOC_old_ONON;

%Create lookup table parameters (plot using 'surf' command)
%Inputs
```

```
SOC_old_ONON_input=[0:SOC_q:1];
power_demand_ONON_input=[-6:PD_q:12];
SOC_new_ONON_input=[0:SOC_q:1];

[xi,yi]=meshgrid(SOC_old_ONON_input,power_demand_ONON_input);
SOC_new_ONON_output_oldPD =
griddata(SOC_old_ONON,power_demand_ONON,SOC_new_ONON,xi,yi,'cubic');

[xi,yi]=meshgrid(SOC_new_ONON_input,power_demand_ONON_input);
SOC_old_ONON_output_newPD =
griddata(SOC_new_ONON,power_demand_ONON,SOC_old_ONON,xi,yi,'cubic');

[xi,yi]=meshgrid(SOC_old_ONON_input,power_demand_ONON_input);
SOC_delta_ONON_output_oldPD =
griddata(SOC_old_ONON,power_demand_ONON,SOC_delta_ONON,xi,yi,'cubic');

[xi,yi]=meshgrid(SOC_new_ONON_input,power_demand_ONON_input);
E_total_ONON_output_newPD =
griddata(SOC_new_ONON,power_demand_ONON,E_total_ONON,xi,yi,'cubic');

[xi,yi]=meshgrid(SOC_old_ONON_input,power_demand_ONON_input);
E_total_ONON_output_oldPD =
griddata(SOC_old_ONON,power_demand_ONON,E_total_ONON,xi,yi,'cubic');
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% %Plot results for comparison
% %SOC_delta
% figure
% hold on
% plot3(SOC_old_OFFOFF,power_demand_OFFOFF,SOC_delta_OFFOFF,'o')
% plot3(SOC_old_OFFON,power_demand_OFFON,SOC_delta_OFFON,'or')
% plot3(SOC_old_ONON,power_demand_ONON,SOC_delta_ONON,'og')
% xlabel('Old SOC')
% ylabel('Power Demand (kW)')
% zlabel('Change in SOC')
% legend('Engine Off-Off','Engine Off-On','Engine On-On')
% title('Comparison of Change in SOC for different Engine States')
%
% a=find(abs(SOC_delta_OFFOFF) <= SOC_q)
% figure
% hold on
% plot3(SOC_old_OFFOFF,power_demand_OFFOFF,SOC_delta_OFFOFF,'ro')
%
plot3(SOC_old_OFFOFF(a),power_demand_OFFOFF(a),SOC_delta_OFFOFF(a),'bo'
)
%
% b=find(abs(SOC_delta_OFFON) <= SOC_q)
% figure
% hold on
% plot3(SOC_old_OFFON,power_demand_OFFON,SOC_delta_OFFON,'ro')
% plot3(SOC_old_OFFON(b),power_demand_OFFON(b),SOC_delta_OFFON(b),'bo')
%
% c=find(abs(SOC_delta_ONON) <= SOC_q)
% figure
% hold on
% plot3(SOC_old_ONON,power_demand_ONON,SOC_delta_ONON,'ro')
```

```
% plot3(SOC_old_ONON(c),power_demand_ONON(c),SOC_delta_ONON(c),'bo')
%
%
% %E_total
% figure
% hold on
% plot3(SOC_old_OFFOFF,power_demand_OFFOFF,E_total_OFFOFF,'o')
% plot3(SOC_old_OFFON,power_demand_OFFON,E_total_OFFON,'or')
% plot3(SOC_old_ONON,power_demand_ONON,E_total_ONON,'og')
% xlabel('Old SOC')
% ylabel('Power Demand (kW)')
% zlabel('Energy Cost (kJ)')
% legend('Engine Off-Off','Engine Off-On','Engine On-On')
% title('Comparison of Total Energy Cost for different Engine States')
%
% %SOC_new Table (Off-On)
% figure
%
surf(SOC_old_OFFON_input,power_demand_OFFON_input,SOC_new_OFFON_output_
oldPD)
% xlabel('Old SOC')
% ylabel('Power Demand (kW)')
% zlabel('New SOC')
% title('New SOC for Engine Off-On')
%
% %SOC_old Table (Off-On)
% figure
%
surf(SOC_new_OFFON_input,power_demand_OFFON_input,SOC_old_OFFON_output_
newPD)
% xlabel('New SOC')
% ylabel('Power Demand (kW)')
% zlabel('Old SOC')
% title('Old SOC for Engine Off-On')
%
% %SOC_delta Table (Off-On)
% figure
%
plot(SOC_old_OFFON_input,SOC_delta_OFFON_output_oldPD(9,:),SOC_old_ONON
_input,SOC_delta_ONON_output_oldPD(9,:),SOC_old_OFFOFF_input,SOC_delta_
OFFOFF_output_oldPD(9,:))
% xlabel('Old SOC')
% ylabel('Change in SOC for Engine Off-On')
% legend('Off-On','On-On','Off-Off','On-On & Off-Off Avg.')
%
% figure
%
plot(SOC_old_OFFOFF_input,SOC_delta_OFFOFF_output_oldPD(9,:)*0.63+SOC_d
elta_ONON_output_oldPD(9,:)*0.37,SOC_old_OFFOFF_input,SOC_delta_OFFOFF_
output_oldPD(9,:)*0.5+SOC_delta_ONON_output_oldPD(9,:)*0.5)
%
% figure
%
surf(SOC_old_OFFON_input,power_demand_OFFON_input,SOC_delta_OFFON_outpu
t_oldPD)
% xlabel('Old SOC')
% ylabel('Power Demand (kW)')
```

```
% zlabel('Change in SOC')
% title('Change in SOC for Engine Off-On')
%
% %SOC_new Table (Off-Off)
% figure
%
surf(SOC_old_OFFOFF_input,power_demand_OFFOFF_input,SOC_new_OFFOFF_outp
ut_oldPD)
% xlabel('Old SOC')
% ylabel('Power Demand (kW)')
% zlabel('New SOC')
% title('New SOC for Engine Off-Off')
%
% %SOC_old Table (Off-Off)
% figure
%
surf(SOC_new_OFFOFF_input,power_demand_OFFOFF_input,SOC_old_OFFOFF_outp
ut_newPD)
% xlabel('New SOC')
% ylabel('Power Demand (kW)')
% zlabel('Old SOC')
% title('Old SOC for Engine Off-Off')
%
% %SOC_delta Table (Off-Off)
% figure
%
surf(SOC_old_OFFOFF_input,power_demand_OFFOFF_input,SOC_delta_OFFOFF_ou
tput_oldPD)
% xlabel('Old SOC')
% ylabel('Power Demand (kW)')
% zlabel('Change in SOC')
% title('Change in SOC for Engine Off-Off')
%
% %SOC_new Table (On-On)
% figure
%
surf(SOC_old_ONON_input,power_demand_ONON_input,SOC_new_ONON_output_old
PD)
% xlabel('Old SOC')
% ylabel('Power Demand (kW)')
% zlabel('New SOC')
% title('New SOC for Engine On-On')
%
% %SOC_old Table (On-On)
% figure
%
surf(SOC_new_ONON_input,power_demand_ONON_input,SOC_old_ONON_output_new
PD)
% xlabel('New SOC')
% ylabel('Power Demand (kW)')
% zlabel('Old SOC')
% title('Old SOC for Engine On-On')
%
% %SOC_delta Table (On-On)
% figure
```

```
%
surf(SOC_old_ONON_input,power_demand_ONON_input,SOC_delta_ONON_output_o
ldPD)
% xlabel('Old SOC')
% ylabel('Power Demand (kW)')
% zlabel('Change in SOC')
% title('Change in SOC for Engine On-On')
%


% % Use this (rstool) to export beta values for polynomial model
% X=[x',y'];
% Y=z;
% model='quadratic';
% alpha=0.01;
% XN={'Power Demand (kW)';'Original SOC'};
% YN={'New SOC'};
% rstool(X,Y,model,alpha,XN,YN)
%
% %Then export data before executing code below
% break
%
%
trans_map_beta=beta(1)+beta(2)*x+beta(3)*y+beta(4).*x.*y+beta(5).*x.*x+
beta(6).*y.*y;
% surf([0:12],[0:0.1:1],trans_map_OFFON')
% hold on
% plot3(x,y,trans_map_beta,'o')
```

### B.4 Transition Map Data Collection Code Sample (Off-Off case)

```
T=60; %time step in seconds
Q_SOC=5; %SOC quantization level (%)
set_param('hybrid_transition_testing','StopTime','1e6')
set_param('hybrid_transition_testing/Energy Management Controller/Speed
Setpoint','Value','0')
set_param('hybrid_transition_testing/Energy Management Controller/Power
Setpoint','Value','0')
for i1=[-6:12]
    i1
    set_param('hybrid_transition_testing/Load Profile/P_motors
(kW)','Value',num2str(i1))
    if i1<0
        set_param('hybrid_transition_testing/Battery/Initial SOC (A-
s)','Value','39600*0')
        set_param('hybrid_transition_testing/Compare','relop','>=')
        set_param('hybrid_transition_testing/Compare','const','1')
        SOC_levels=[0:Q_SOC:100];
    elseif i1>0
        set_param('hybrid_transition_testing/Battery/Initial SOC (A-
s)','Value','39600*1')
        set_param('hybrid_transition_testing/Compare','relop','<=')
        set_param('hybrid_transition_testing/Compare','const','0')
        SOC_levels=[100:-Q_SOC:0];
    else
```

```
        SOC_levels=[100:-Q_SOC:0]/100;
        SOC_summary=[SOC_levels' SOC_levels' zeros(size(SOC_levels'))
zeros(size(SOC_levels'))];
        eval(strcat('SOC_OFFOFF_',num2str(abs(i1)),'=SOC_summary;'))
        continue %skip to next for loop iteration
    end
    sim('hybrid_transition_testing')

    clear k_closest k_find err SOC_k SOC_new ...
        t_start t_end k_find_end err_end k_closest_end k_min_err

    final_index=1;
    for j1=1:length(SOC_levels)
        clear err err_end
        k_find=find(round(SOC*100)==SOC_levels(j1));
        for k1=1:length(k_find)
            err(k1)=abs(SOC(k_find(k1))-SOC_levels(j1)/100);
        end
        k_min_err=find(err==min(err));
        if length(k_min_err) > 1
            k_min_err=k_min_err(length(k_min_err));
        end
        k_closest(j1)=k_find(k_min_err);
        t_start(j1)=t(k_closest(j1));
        t_end(j1)=t_start(j1)+T;
        if t_end(j1)>t(length(t))
            k_closest_end(j1)=NaN;
            if final_index==1
                final_index=j1-1;
            end
        else
            k_find_end=find(round(t)==round(t_end(j1)));
            for k2=1:length(k_find_end)
                err_end(k2)=abs(t(k_find_end(k2))-t_end(j1));
            end
            k_closest_end(j1)=k_find_end(find(err_end==min(err_end)));
        end
    end

    if final_index==1
        final_index=j1;
    end

    SOC_k=SOC(k_closest(1:final_index));
    SOC_new=SOC(k_closest_end(1:final_index));
    Ef_k=(E_fuel(k_closest_end(1:final_index))-
E_fuel(k_closest(1:final_index)))/1000; %kJ
    Es_k=(E_starter(k_closest_end(1:final_index))-
E_starter(k_closest(1:final_index)))/1000; %kJ
    % SOC_summary=[SOC_k SOC_new Ef_k Es_k]
    SOC_summary=[SOC_k SOC_new Ef_k zeros(length(Ef_k),1)];
    if i1<0
        eval(strcat('SOC_OFFOFF_neg',num2str(abs(i1)),'=SOC_summary;'))
    else
        eval(strcat('SOC_OFFOFF_',num2str(abs(i1)),'=SOC_summary;'))
    end
```

```
end
```

## B.5 Battery Efficiency Programmatic Simulation Code

```
%% Create battery efficiency table data by programatically running
multiple
% Simulink simulations, altering parameters and saving output data for
% every run.

%% Input Power Levels
Pb=[-6000:2000:-2000,-1000,-500,500,1000,2000:2000:12000];

%% Programatically run the simulation
% for i=Pb
%     set_param('batt_efficiency_test/Step','After',num2str(i))
%     if i < 0
%         set_param('batt_efficiency_test/Compare To
Constant','relop','>=')
%         set_param('batt_efficiency_test/Compare To
Constant','const','1')
%         set_param('batt_efficiency_test/Battery/Initial SOC (A-
s)','Value','39600*0')
%         sim('batt_efficiency_test');
%         eval(strcat('SOC_neg',num2str(abs(i)),'=SOC;'))
%
eval(strcat('eta_batt_inst_neg',num2str(abs(i)),'=eta_batt_inst;'))
%         eval(strcat('t_neg',num2str(abs(i)),'=t;'))
%     elseif i > 0
%         set_param('batt_efficiency_test/Compare To
Constant','relop','<=')
%         set_param('batt_efficiency_test/Compare To
Constant','const','0')
%         set_param('batt_efficiency_test/Battery/Initial SOC (A-
s)','Value','39600*1')
%         sim('batt_efficiency_test');
%         eval(strcat('SOC_',num2str(i),'=SOC;'))
%
eval(strcat('eta_batt_inst_',num2str(i),'=1./eta_batt_inst;'))
%         eval(strcat('t_',num2str(i),'=t;'))
%     else
%         continue
%     end
% end

%% Load Data that has already been created by the code above
load battery_efficiency_data

%% BDC EFFICIENCIES
% BOOST (DISCHARGE, positive power by system convention)
% eta_d=PHS/PLS;
PHS_d_fit=[500:100:12000];
PHS_d=[903 2710 3628 4524 5437 6322 7207 8108 8990 9922 10800 11684
12142];
```

```
PLS_d=[1072 2984 3943 4850 5820 6768 7715 8656 9574 10535 11481 12374
12870];
eta_d=PHS_d./PLS_d;
%Linear fit for PLS=m*PHS+b
md=1.0497912213;
bd=127.9848908436;
%Conversion from linear PLS=f(PHS) relationship to nonlinear
eta_d=f(PHS)
eta_d_fit=PHS_d_fit./(md.*PHS_d_fit+bd);

% BUCK (CHARGE, negative power by system convention
% eta_c=PLS/PHS;
PHS_c_fit=[500:100:6000];
PHS_c=[1270 2424 2423 3181 3571 3750 4361 4672 5115 5518 6106 6576];
PLS_c=[1129 2238 2246 2975 3344 3527 4127 4421 4848 5248 5829 6277];

PHS_c_fit=-fliplr(PHS_c_fit); %convert to system sign convention
PHS_c=-fliplr(PHS_c); %convert to system sign convention
PLS_c=-fliplr(PLS_c); %convert to system sign convention

eta_c=PLS_c./PHS_c;
%Linear fit for PLS=m*PHS+b
mc=0.9714914947;
bc=113.4274611639; %if positive power values are used, this should be
NEGATIVE
%Conversion from linear PLS=f(PHS) relationship to nonlinear
eta_d=f(PHS)
eta_c_fit=mc+bc./PHS_c_fit;

%% Compile data for plotting
SOC=[0:0.01:1];
eta_batt_surf=zeros(length(SOC),length(Pb));
indx=1;
for i=Pb
    if i < 0

eval(strcat('eta_batt_surf(:,',num2str(indx),')=interp1(SOC_neg',num2st
r(abs(i)),',eta_batt_inst_neg',num2str(abs(i)),',SOC);'));
    elseif i > 0

eval(strcat('eta_batt_surf(:,',num2str(indx),')=interp1(SOC_',num2str(i
),',eta_batt_inst_',num2str(i),',SOC);'));
    else
        continue
    end
    indx=indx+1;
end

PHS_fit=[PHS_c_fit PHS_d_fit];
%convert x-axis from LS power to HS power so that they can be
multiplied over a single x-axis vector
Pb_neg=[-6000:2000:-2000,-1000,-500];
Pb_pos=[500,1000,2000:2000:12000];
Pb_negHS=(Pb_neg-bc)./mc;
Pb_posHS=(Pb_pos-bd)./md;
Pb_HS=[Pb_negHS Pb_posHS];
```

```
eta_batt_surf_fit=zeros(length(SOC),length(PHS_fit));
for i=1:length(SOC)

eta_batt_surf_fit(i,:)=interp1(Pb_HS,eta_batt_surf(i,:),PHS_fit);
end

eta_BDC_fit=[eta_c_fit eta_d_fit];

for i=1:length(SOC)
        eta_ESS_surf(i,:)=eta_BDC_fit.*eta_batt_surf_fit(i,:);
end

%% Determine Logic for Engine Based Line Algorithm
%Operates on the premise that any current going into the battery will
have
%to come out again.  Therefore, we compare the efficiency for using
engine
%only to the efficiency of using engine+battery charge+battery
discharge.

%It also makes sense to charge the batteries at a slightly higher rate
than
%may occur naturally from low-level regen since the ESS is not very
%efficient at low power levels.

warning('off','MATLAB:interp1:NaNinY');
load ea_efficiency_data
P_EA=P_ea_out*1000;
P_m=PHS_fit;
eta_ESS_70=eta_ESS_surf(71,:);

eta_ESS_discharge=interp1(P_m,eta_ESS_70,P_EA);
eta_ESS_charge=interp1(fliplr(-P_m),fliplr(eta_ESS_70),P_EA);
eta_charge_combined=eta_ESS_charge.*eta_ea;
eta_max_index=find(eta_charge_combined==max(eta_charge_combined));
figure
plot(P_EA/1000,eta_ESS_charge,P_EA/1000,eta_ea,P_EA/1000,eta_charge_com
bined,P_EA(eta_max_index)/1000,eta_charge_combined(eta_max_index),'*k',
'MarkerSize',14,'LineWidth',2)
set(gca,'FontSize',14,'FontWeight','bold')
clear temp
legend('\eta_{ESS,charge}','\eta_{EA}','\eta_{ESS,charge}*\eta_{EA}','m
ax(\eta_{ESS,charge}*\eta_{EA})')
xlabel('Bus Power (kW)','FontSize',14,'FontWeight','bold')
ylabel('Efficiency','FontSize',14,'FontWeight','bold')
%This plot shows that maximum charging efficiency still occurs at 5.1kW
%(straight charging from the engine to the battery...no motor power
demand)

eta_system_charge=eta_charge_combined(eta_max_index)*eta_ESS_discharge;
% eta_EA_max=max(eta_ea);
% eta_ESS_max=max(max(eta_ESS_surf)); %THIS OCCCURS AT 100% SOC!
% eta_system_charge_max=eta_EA_max*eta_ESS_max*eta_ESS_discharge;
figure
plot(P_EA/1000,eta_ea,P_EA/1000,eta_system_charge,'LineWidth',2)
```

```
set(gca,'FontSize',14,'FontWeight','bold')
legend('\eta_{EA}','\eta_{EA}(5.1)*\eta_{ESS}(-5.1)*\eta_{ESS}')
xlabel('Bus Power (kW)','FontSize',14,'FontWeight','bold')
ylabel('Efficiency','FontSize',14,'FontWeight','bold')
%% System Based Line Algorithm
%Say we are running a standard thermostat algorithm.  On the charge
cycle,
%instead of running the engine at 5.1 kW, you could run it at the
engine
%power that maximizes eta_ea*eta_ESS_charge for a given Pm.  This
should
%require less fuel to charge the battery.

%Determine overall system efficiency (EA & ESS)
for i=1:length(P_EA)
    for j=1:length(P_m)
        eta_ea_temp(i,j)=eta_ea(i);
        eta_ESS_temp(i,j)=interp1(PHS_fit,eta_ESS_70,P_m(j)-P_EA(i));
        eta_sys(i,j)=eta_ea_temp(i,j)*eta_ESS_temp(i,j);
    end
end

for i = 1:length(P_m)
    eta_opt(i)=max(eta_sys(:,i));
    opt_ind = find(eta_sys(:,i) == eta_opt(i));
    if isempty(opt_ind)
        P_EA_opt(i)=0;
    else
        P_EA_opt(i) = P_EA(opt_ind);
    end
end

P_ESS_opt=P_m-P_EA_opt;

figure
surf(P_m/1000,P_EA/1000,eta_sys)
xlabel('Motor Bus Power (kW)')
ylabel('Engine/Alternator Bus Power (kW)')
zlabel('System Efficiency')

figure
surf(P_m/1000,P_EA/1000,eta_ea_temp)
xlabel('Motor Bus Power (kW)')
ylabel('Engine/Alternator Bus Power (kW)')
zlabel('EA Efficiency')

figure
surf(P_m/1000,P_EA/1000,eta_ESS_temp)
xlabel('Motor Bus Power (kW)')
ylabel('Engine/Alternator Bus Power (kW)')
zlabel('ESS Efficiency')

figure
% v=[0.08:0.02:0.26];
v=[0.22:0.005:0.26];
```

```
% v=[0.08:0.02:0.24 0.241:0.01:0.258];
% v=[0.08:0.02:0.24 0.241:0.005:0.258];
[Cs,hs]=contour(P_m/1000,P_EA/1000,eta_sys,v);
clabel(Cs,hs,[0.22:0.01:0.24 0.245:0.005:0.26]);
set(gca,'FontSize',14,'FontWeight','bold')
xlabel('Motor Bus Power (kW)','FontSize',14,'FontWeight','bold')
ylabel('Engine/Alternator Bus Power
(kW)','FontSize',14,'FontWeight','bold')
hold on
plot(P_m/1000,P_EA_opt/1000,'LineWidth',2)
hold off

figure
plot(P_m/1000,P_ESS_opt/1000)
xlabel('Motor Bus Power (kW)')
ylabel('ESS Bus Power (kW)')

%% Create Plots
% figure
% surf(PHS_fit,SOC,eta_batt_surf_fit)
% plot(PHS_fit,eta_BDC_fit,'o')

%Battery Only Plots
figure
surf(Pb,SOC,eta_batt_surf);
xlabel('Battery Input Power (kW)')
ylabel('SOC')
zlabel('Battery Instantaneous Efficiency')

figure
[Cs,hs]=contour(Pb,SOC,eta_batt_surf,[0.8:0.01:1]);
clabel(Cs,hs,[0.8:0.02:0.9,0.91:0.01:1]);
xlabel('Battery Input Power (kW)')
ylabel('SOC')

figure
plot(SOC,eta_batt_surf(:,1),...
SOC,eta_batt_surf(:,2),...
SOC,eta_batt_surf(:,3),...
SOC,eta_batt_surf(:,4),...
SOC,eta_batt_surf(:,5),...
SOC,eta_batt_surf(:,6),...
SOC,eta_batt_surf(:,7),...
SOC,eta_batt_surf(:,8),...
SOC,eta_batt_surf(:,9),...
SOC,eta_batt_surf(:,10),...
SOC,eta_batt_surf(:,11),...
SOC,eta_batt_surf(:,12),...
SOC,eta_batt_surf(:,13))
legend('-6 kW','-4 kW','-2 kW','-1 kW','-0.5 kW','0.5 kW','1 kW','2
kW','4 kW','6 kW','8 kW','10 kW','12 kW')
xlabel('SOC')
ylabel('Battery Instantaneous Efficiency')

%BDC Plots
figure
```

```
plot(PHS_d/1000,eta_d,'bo',PHS_d_fit/1000,eta_d_fit,'b-
',PHS_c/1000,eta_c,'ro',PHS_c_fit/1000,eta_c_fit,'r-')
legend('Actual Discharge Efficiency','Fit Discharge Efficiency','Actual
Charge Efficiency','Fit Charge Efficiency')
xlabel('High Side (Bus) Power (kW)')
ylabel('Bi-Directional Converter Efficiency')

figure
plot(PHS_d/1000,PLS_d/1000,'bo',PHS_d/1000,(md.*PHS_d+bd)/1000,'k-')
legend('Data','Linear Fit')
xlabel('High Side (Bus) Power (kW)')
ylabel('Low Side (Battery Input) Power (kW)')

figure
plot(PHS_c/1000,PLS_c/1000,'bo',PHS_c/1000,(mc.*PHS_c+bc)/1000,'k-')
legend('Data','Linear Fit')
xlabel('High Side (Bus) Power (kW)')
ylabel('Low Side (Battery Input) Power (kW)')

%Combined Energy Storage System (ESS) Plots
figure
[Cs,hs]=contour(PHS_fit/1000,SOC,eta_ESS_surf,[0.80:0.01:0.91]);
clabel(Cs,hs,[0.80 0.82 0.84:0.01:0.91],'FontSize',14)
xlabel('High Side (Bus) Power (kW)')
ylabel('SOC')

figure, surf(PHS_fit/1000,SOC,eta_ESS_surf)
```

# REFERENCES


[1] WANG, W., and SADEGH, N., "Optimal Control for Energy Management of a Series Hybrid Vehicle."

[2] BARSALI, S., MASSIMO, C., POSSENTI, A., "Techniques to Control the Electricity Generation in a Series Hybrid Electrical Vehicle," IEEE Transactions on Energy Conversion, Vol. 17, No. 2, June 2002.

[3] BARSALI S., MIULLI, C., POSSENTI, A., "A Control Strategy to Minimize Fuel Consumption of Series Hybrid Electric Vehicles," IEEE Transactions on Energy Conversion, Vol. 19, No. 1, March 2004.

[4] JALIL, N., KHEIR, N. A., SALMAN, M., "A Rule-Based Energy Management Strategy for a Series Hybrid Vehicle," Proceedings of the American Control Conference, June 1997.

[5] GOKASAN, M., BOGOSYAN, S., GOERING, D. J., "Sliding Mode Based Powertrain Control for Efficiency Improvement in Series Hybrid-Electric Vehicles," IEEE Transactions on Power Electronics, Vol. 21, No. 3, May 2006.

[6] BRAHMA, A., GUEZENNEC, Y., RIZZONI, G., "Optimal Energy Management in Series Hybrid Electric Vehicles," Proceedings of the American Control Conference, June 2000.

[7] SCIARRETTA, A., GUZZELLA, L., "Control of Hybrid Electric Vehicles," IEEE Control Systems Magazine, April 2007.

[8] JOHANNESSON, L, ASBOGARD, M., EGARDT, B., "Assessing the Potential of Predictive Control for Hybrid Vehicle Powertrains Using Stochastic Dynamic Programming," IEEE Transactions on Intelligent Transportation Systems, Vol. 8, No. 1, March 2007.

[9] KLEIMAIER, A., SCHRODER, D., "An Approach for the Online Optimized Control of a Hybrid Powertrain," IEEE Advanced Motion Control Conference, 2002.

[10] PISU, P., RIZZONI, G., “A Supervisory Control Strategy for Series Hybrid Electric Vehicles with Two Energy Storage Systems,” 2005 IEEE Vehicle Power and Propulsion Conference, 2005.

[11] PISU, P., RIZZONI, G., “A Comparative Study of Supervisory Control Strategies for Hybrid Electric Vehicles,” IEEE Transactions on Control Systems Technology, Vol. 15, No. 3, May 2007.

[12] RIZZONI, G., GUZZELA, L., BAUMANN, B., “Unified Modeling of Hybrid Electric Vehicle Drivetrains,” IEEE/ASME Transactions on Mechatronics, Vol. 4, No. 3, 1999.

[13] KUMARESAN, K., SIKHA, G., WHITE, R. E., “Thermal Model for a Li-Ion Cell,” Journal of the Electrochemical Society, Vol. 155, No. 2, 2008.

[14] VERBRUGGE, M., FRISCH, D., KOCH, B., “Adaptive Energy Management of Electric and Hybrid Electric Vehicles,” Journal of the Electrochemical Society, Vol. 152, No. 2, 2005.

[15] ABRAHAM, D.P., KAWAUCHI, S., DEES, D.W., “Modeling the Impedance Versus Voltage Characteristics of $LiNi_{0.8}Co_{0.15}Al_{0.05}O_2$,” Electrochimica Acta 53, 2008.

[16] GAO, L., LIU, S., DOUGAL, R., “Dynamic Lithium-Ion Battery Model for System Simulation,” IEEE Transactions on Components and Packaging Technologies, Vol. 25, No. 3, September 2002.

[17] CHO, D., HEDRICK, J. K., “Automotive Powertrain Modeling for Control,” Transactions of the ASME, Vol. 111, December 1989.

[18] Anon., “John Deere M-Gator A1 Features”, http://www.deere.com/en_US/... contractsales/fedmilitarysales/cce/m_gator/m_gator.html?link=GC_sa_gator_series, (Accessed December 13, 2007).