Electric Vehicle Drive Simulation with MATLAB/Simulink

David McDonald

LSSU Sault Ste Marie, MI 49783

dmcdonald@lssu.edu

Abstract

The paper presents the simulation of a basic electric vehicle motor-drive system that is used to investigate power flow during both motoring and regeneration. The simulation assumes a DC permanent magnet motor, an ideal motor controller combined with a proportional-integral controller, and the electric vehicle battery. The model can be used to evaluate the electric drive's energy flow and efficiency for specific speed and torque load conditions. Some of the key system parameters were specified and others were modeled as ideal. A stable MATLAB/Simulink model was developed and validated. It was then used to determine the system performance and energy flow over a given set of motoring and regeneration speed/torque conditions. The model could be used to augment instruction in energy conversion or vehicle systems courses.

Introduction

This past year electric vehicles were mass produced for the first time in history, and there is a need to include more learning experiences that are related to that topic. "The 2010 – 2020 time period has been described as the upcoming 'tipping point' …the transition from the Internal Combustion Engine (ICE) as the prime mover of vehicles to electric propulsion systems."[1] "Education is really the important foundation for where the industry is headed in this field.[2] Currently there are no ABET accredited Automotive Engineering or Technology degree programs that contain electric vehicle courses[3]. A literature search for electric vehicle educational revealed a few single-offering or special topics courses,[4-9] The Department of Energy has awarded funds under the Advanced Electric Drive Vehicle Education Program to support the development of new courses for graduate, undergraduate, secondary students, teachers, technicians, emergency responders, and the general public.[10, 11] However, industry is largely training engineers 'in-house', and educational experiences in this technology are needed now to prepare a well trained and educated workforce to support the development of Smart Grid and Electric Vehicle applications.

In addition to the growth in new technology, the design process in industry has also experienced significant change in recent years. Model-Based Design is now commonly used in automotive, aeronautical, and other industries for complex embedded systems.[12-15] Traditional design workflow follows a sequential path that involves: a) Requirements, b) Design, c)

Implementation, and d) Test and validation. Problems with traditional design can develop when: 1) specifications must be read and understood by different engineers on different teams, 2) application engineers have to rewrite design engineers' algorithms, or 3) the problem is not found until the testing phase.

Model-Based Design uses models early in the process to create executable specifications that allow engineers to immediately validate and verify specifications against the requirements. Engineers then share models that can demonstrate the performance of the subsystems and components, and also use the automatic code generation capability of Simulink/Real Time and Embedded Coder to facilitate Hardware In the Loop (HIL) testing.

Simulation is a key tool that facilitates design while reducing the cost of product development. As the design process evolves engineers can perform Model-In-The-Loop (MIL), Software-In-The-Loop (SIL), and Hardware-In-The-Loop (HIL) development modeling model is the design. By integrating simulation within the design process engineers can decrease both design costs and design time thus enabling companies to complete and test designed items.

Drive Cycle

To assist in the design process, vehicle driving tests and vehicle driving simulations are completed to help support the design process to determine if the design is appropriate for the desired application.

A driving cycle is a set of second-by-second set of vehicle velocity values that the simulated vehicle is to attain during the simulation. The need of a drive cycle is to reduce the quantity of expensive on-road tests, and also reduce both the time of test and fatigue of the test engineer. The drive cycle process brings the road to the dynamometer or to the computer simulation.

Drive cycles are used in vehicle simulations to model the drive system and predict the performance of the drive system. There are many standard driving cycles used for testing road vehicles for fuel economy and other purposes. Some driving cycles are developed theoretically, and others are direct measurements of a representative driving pattern. A driving cycle can include frequent speed changes or extended periods at constant speed. An example of vehicle simulator is ADVISOR produced by AVL Engineering[16] and other on-line road load and fuel economy simulations.[17]

Speed and Torque Values

The simulation that is presented assumes known speed and torque values. If speed values are assumed then the torque values can be calculated if the wheel dimensions are available and the road load values encountered by the vehicle values are known. The total road load is the sum of

the rolling resistance, air resistance, and gradient resistances are known or can be calculated. Information on these calculations is available in literature.[18-20]

Electric Vehicle Drive Train Operation

In a typical gasoline powered vehicle the gas tank is not a part of the design model. Gasoline is consumed by the engine, but the engine does not put gasoline back into the gas tank. A paradigm shift from internal combustion engines to electric vehicles is that in an electric vehicle the battery is part of the drive train as shown below in Figure 1: Electric Vehicle Drive Train. The drive train consumes energy from the battery during motoring. The drive train can also add charge to the battery if the motor is operated as a generator during regeneration. This can occur during braking or if the vehicle is being powered by an Internal Combustion Engine (ICE). In the diagram, the battery is frequently constructed of Lithium Ion cells, and supplies 300+ volts and high current to the power electronics. A battery controller monitors key battery parameters and controls the battery pack.

The power electronics unit inverts the DC battery voltage into three-phase AC voltage at the proper frequency and voltage for the motor to meet the requested speed and torque. The AC motor is typically a high efficiency AC Induction Motor (IM) or Permanent Magnet Synchronous Motor (PMSM). These motors can supply either acceleration torque or braking torque for both directions of rotation. When the vehicle's brakes are applied the motor operates in regeneration mode thus reversing both the current direction and torque direction. The reversed torque direction provides vehicle braking torque while helping to recharge the battery.

The Vehicle Interface communicates with the Battery Controller and Motor Controller, and provides an interface with the vehicle-level controls and sensors. Communication between the separate units involves the use of a Controller Area Network (CAN) communications system.

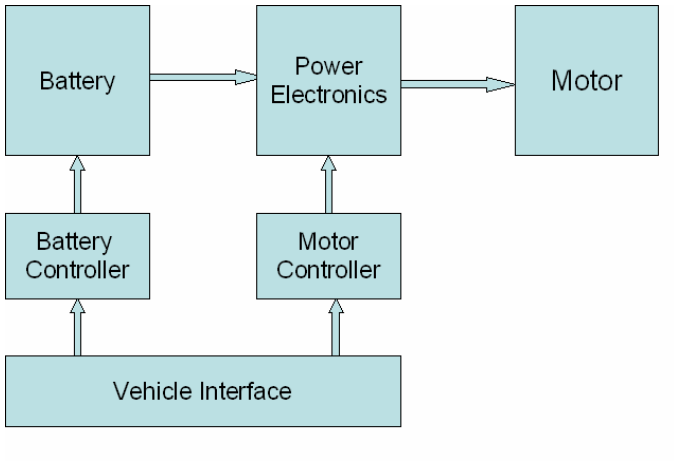

Figure 1: Electric Vehicle Drive Train

Model Development Process:

The model development process consists of 1) determining how the model will be used, 2) identifying the key equations, parameters, and assumptions, 3) building and refining the model, and then 4) the actual model application and evaluation.

The model can be used to evaluate the energy flow of a DC motor drive train, and to determine the ability of the system to meet specific drive cycle speed and torque requirements. The major components of the model are input road torque, input road speed, motor model, motor controller model, battery model, and PI controller.

A block diagram of the model is presented below in Figure 2: DC Drive Simulation Model. In the model the required Road Speed and Road Torque are inputs, and the major model blocks are the Motor Model, Controller Model, Battery Model, PI Controller Model, and feedback from the PI Controller to the main power controller. The feedback includes a one-sample delay with an initial condition to prevent an algebraic loop in the Simulink model.

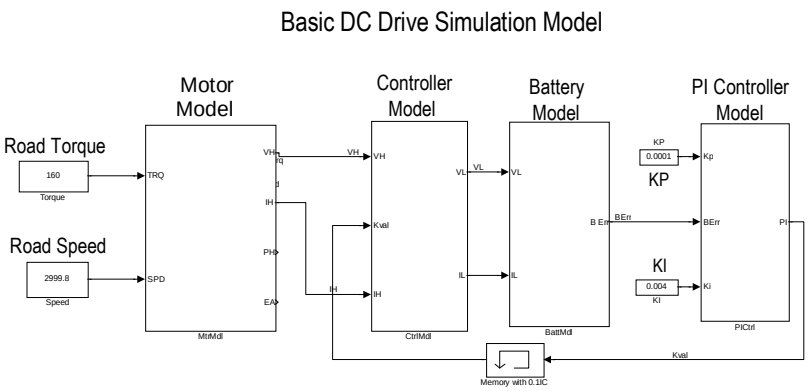

Figure 2: DC Drive Simulation Model

Key Equations

Determining the key equations and their corresponding variables and parameters is a necessary first step in model development. Each block in this simplified model represents one or more major equations as listed below.

DC Motor:

As noted earlier, Battery Electric Vehicles (BEV) and Hybrid Electric Vehicles (HEV) frequently use special, high efficiency Permanent Magnet Synchronous Motors (PMSM). This type of motor may be referred to as a brushless DC motor because it runs from DC voltage but does not have brushes. PMSM motors actually use AC voltage that is supplied by the Motor Controller. The motor controller inverts the DC voltage to produce an AC voltage at the proper voltage and frequency. The motor voltage is frequently a 10-20 KHz Pulse Width Modulated AC voltage where the voltage and frequency are adjusted to provide the proper motor speed and magnetic field values.

A DC permanent magnet motor was used in the simulation model presented below. This type of motor is not appropriate for BEV or HEV applications due to weight and efficiency considerations. This motor was used in the simulation because it frequently covered undergraduate engineering education.

The motor model includes some terms and parameters for power loss and time lag while other terms were omitted from the model. The model accounts for power loss in the winding resistance and time lag due to the energy storage in the magnetic field of the winding inductance. There is no field power loss because it is a permanent magnet field.

The model does not include power loss due to friction and other rotational losses of hysteresis, eddy current, and windage. The model also does not include the time lag due to energy storage in the rotor inertia. The motor model is based on the following equations.

Developed Torque is proportional to armature current:

Equation 1: $T_d(Nm) = K_m * I_A(Amp)$ Developed motor torque

Developed Voltage is proportional to armature speed:

Equation 2: $V_D(Volt) = W_D\ (rad/sec)/K_m$ Developed motor voltage

Motor armature input or terminal voltage is equal to the sum of developed voltage plus resistance and inductance voltage drops. In addition, the motor High Side voltage and current are directly connected to, and therefore identical to, the motor controller High Side voltage and current.

Equation 3: $V_H(Volt) = I_H(Amp) * R_A(Ohm) + L_H(Henry) * di(t)/dt(A/s) + V_D(V)$ Motor Voltage

Shaft output torque is equal to developed torque minus friction loss (Bw) and inertial loss (J*dw(t)/dt). Friction and inertial were not specified in the model and are assumed equal to zero. Therefore developed torque and output torque are equal in this model. However, the model could be easily modified to include these parameters in the future.

The motor physical constant, K_m , is a physical parameter that depends upon the construction of the motor. In the SI system K_m has units of (Amp/Nm) or (Volt/(rad/sec)). At the electrical – mechanical interface inside the motor the developed electrical power ($P = I_A * V_D * K_m$) is equal to the developed mechanical power ($P = K_m * T_d * W_D$).

As noted earlier, in the motor model the mechanical friction and inertia as well as the magnetic power losses have been set to zero. Therefore, the power loss will only occur in the armature resistance, and the time lag will only occur in the armature inductance.

Motor Controller:

The motor controller is assumed to be an ideal controller with no power loss and no time lag. The controller simply raises the battery voltage to meet the higher voltage needs of the motor. The dimensionless constant gain or K ratio of the input and output voltages is determined in order to meet the motor's needs. The same K ratio is used to adjust the current so that input and output power values are equal.

High side voltage is equal to K times the low side voltage:

Equation 4: $V_H = K*V_L$ Controller High Side Current

High side current is equal to 1/K times the low side voltage:

Equation 5: $I_H = (1/K)*V_L$ Controller High Side Voltage

Battery:

The battery is modeled as a voltage source with an internal resistance. The model accounts for internal power loss in the resistance of the battery. There is no time lag component in the model. The battery is assumed to have a constant internal voltage, E_B . The battery terminal voltage, V_B , is equal to the sum of the internal voltage and resistance voltage drop. The battery voltage and battery current are equal to the controller low side voltage and current.

Equation 6: V_B (Volt) = I_A(Amp)*R_A(Ohm) + E_B(Volt). Battery model calculation

V_L (Volt) = I_L(Amp)*R_A(Ohm) + E_B(Volt). Assuming: $V_B = V_L$ and $I_A = I_L$

The battery model uses the current and voltage information from the Motor Controller to calculate the required battery's internal voltage. This voltage is compared with the actual E_B value to create a battery voltage error, B_{EER}, and that error is used by the PI controller model to adjust the loop gain.

Equation 7: $B_{ERR} = E_B$ (actual) - E_B (calculated) Error Voltage Calculation

Proportional Integral (PI) Controller:

The PI controller accepts the B_{ERR} signal from the Battery Model and uses proportional (Kp) and integral (Ki) to calculate the gain K value that is used by the Motor Controller.

Equation 8: $K = (K_p + s*K_I)*B_{ERR}$ PI Calculation

The simulation includes eight equations and eight variables.

Simulation Model Blocks

Motor Model

The simulation block for the motor includes Equations 1 – 3 for the motor. The block is shown below in Figure 3: Motor Model Block

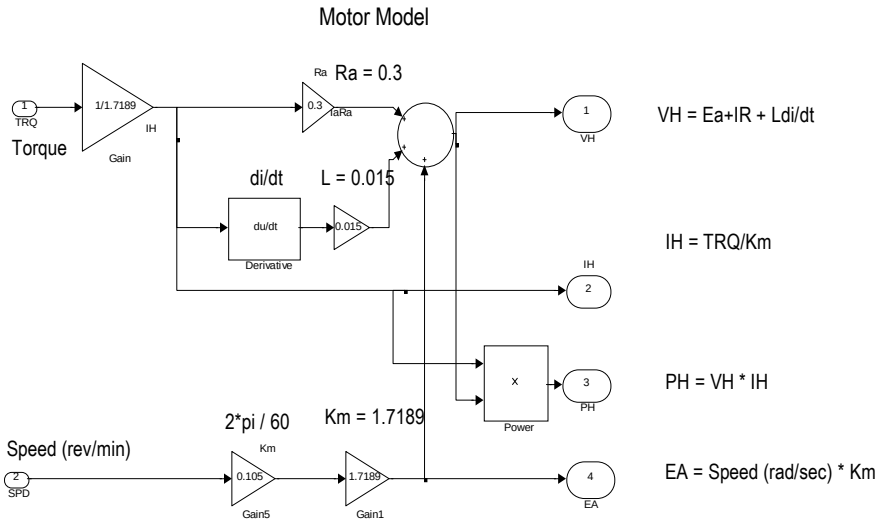

Figure 3: Motor Model Block

Motor Controller Model

The simulation block for the Motor Controller includes Equations 4 and 5 for the motor controller. The block is shown below in Figure 4: Motor Controller Model Block

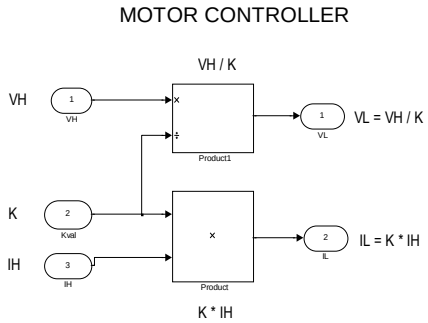

Figure 4: Motor Controller Model Block

Battery Model

The simulation block for the battery model includes Equations 6 & 7 for the battery. The block is shown below in Figure 5: Battery Model Block

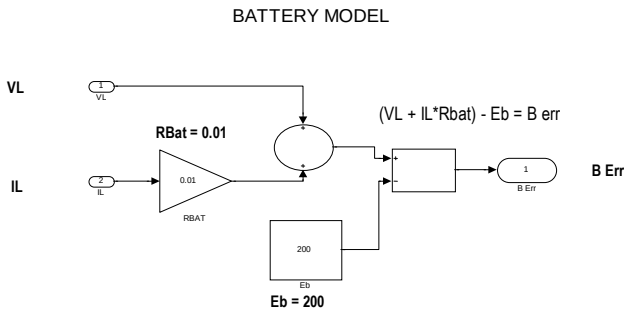

Figure 5: Battery Model Block

PI Controller Model: The block model includes Equation 8 for the controller.

The Gain (K) of the Motor Controller is determined by the output of the PI Controller model. The gain has an initial starting value of 0.1. This value was preset within the controller's integration block to minimize the possibility of a Simulink simulation error due to an algebraic loop. An algebraic loop is basically a divide by zero operation when the simulation is trying to solve the set of linear equations.

The PI Controller checks to see that the output is not zero. If the output is zero then the controller outputs a small value (0.001). This is done to prevent model analysis failure due to dividing by zero when solving the linear equations. The controller also includes a gain limiting block to prevent excess feedback signals.

The block is shown below in Figure 6: PI Controller Model Block

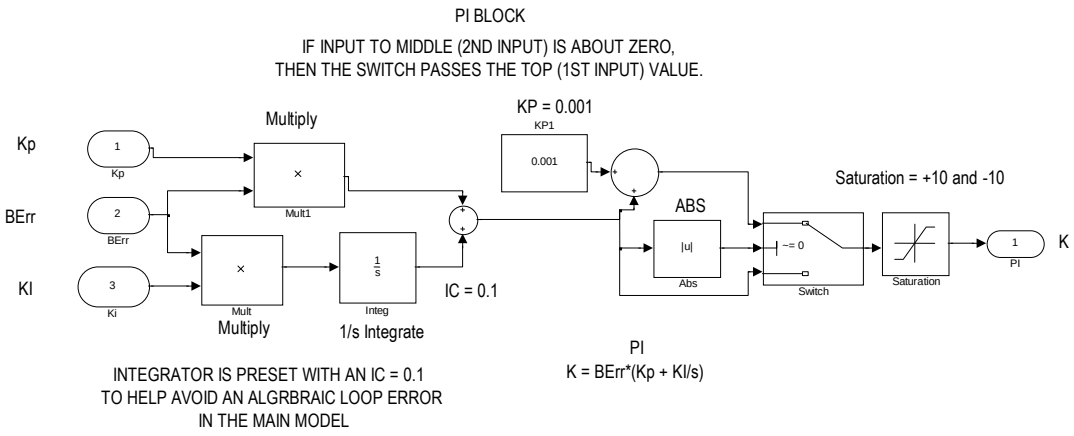

Figure 6: PI Controller Model Block

Drive System Model

The Speed and Torque values were written to the MATLAB Workspace, and the values were then read into the model speed and torque look-up tables. The Clock input to the look-up tables used the following time base values that were setup in the model parameters table: Tmin = 0, Tstep = 0.01, Tstop = 100 seconds.

The displayed Scope values were also written to the MATLAB Workspace as Structures with Time. A MATLAB script was used to pre-load the speed and torque data in the Workspace, Run the Simulation, obtain the key data from the Scope Structures, and plot the data. The complete Motor Drive Model is shown below in Figure 7: Motor Drive Model.

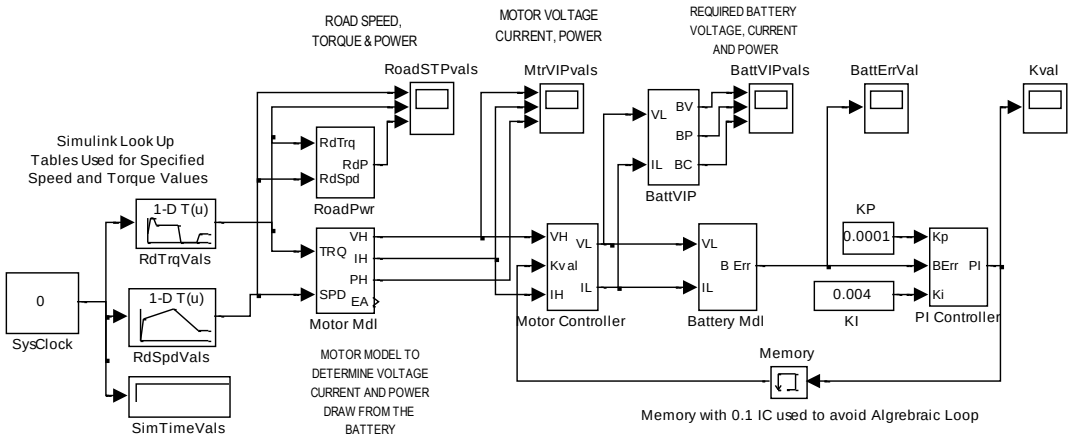

Figure 7: Motor Drive Model

Simulation Road Torque and Speed

The first application of the model focused on processing a given set of Speed and Torque data to determine evaluate the drive system's performance and efficiency.

The key Speed and Torque data was entered into the MATLAB Workspace using the section of MATLAB code shown below in Figure 8: Road Speed and Torque Data.

The data represents Speed-Time and Torque-Time values which correspond to transition times of their corresponding Speed and Torque curves. This data was then processed by the Simulink Speed Look-Up Table and Torque Look-Up Table.

```
% The key Speed and Torque values are loaded into the MATLAB Workspace
% for use by the Simulink Model.
%
% Load Speed vals and times into the Workspace
Svals = [ 0    2000   3000   1000   1000 ];
Stime = [ 0       5     50     85    100 ] ;

% Load Torque vals and times into the Workspace
Tvals = [ 0     330    330    160    160   -220   -220    0     0 ];
Ttime = [ 0       5     10     15     50     55     80   85   100 ];
```

Figure 8: Road Speed and Torque Data

Road Speed, Torque and Power

The speed and torque data were used to calculate the Road Torque Data, and then all three data sets were plotted as shown in Figure 9: Road Speed, Torque and Power. When both torque and speed are positive values the DC Motor is providing torque in the direction of rotation. This is normal motoring operation. However, when the motor torque is in the opposite direction to the speed, then the motor is being pushed and acting as a generator.

A conventional Speed-Torque 4-Quadrant map shows +/-Speed on the x-axis and +/-Torque on the y-axis. When the speed and torque have the same polarity then power is being transferred from the motor to the load, and the motor is in the motoring mode or 1st Quadrant operation. However, when the speed is positive and the torque is negative, then the motor is being pushed by the external mechanical source. This results in energy being transferred back to the battery. In this case the motor is operating in the 4th Quadrant of the Speed-Torque map.

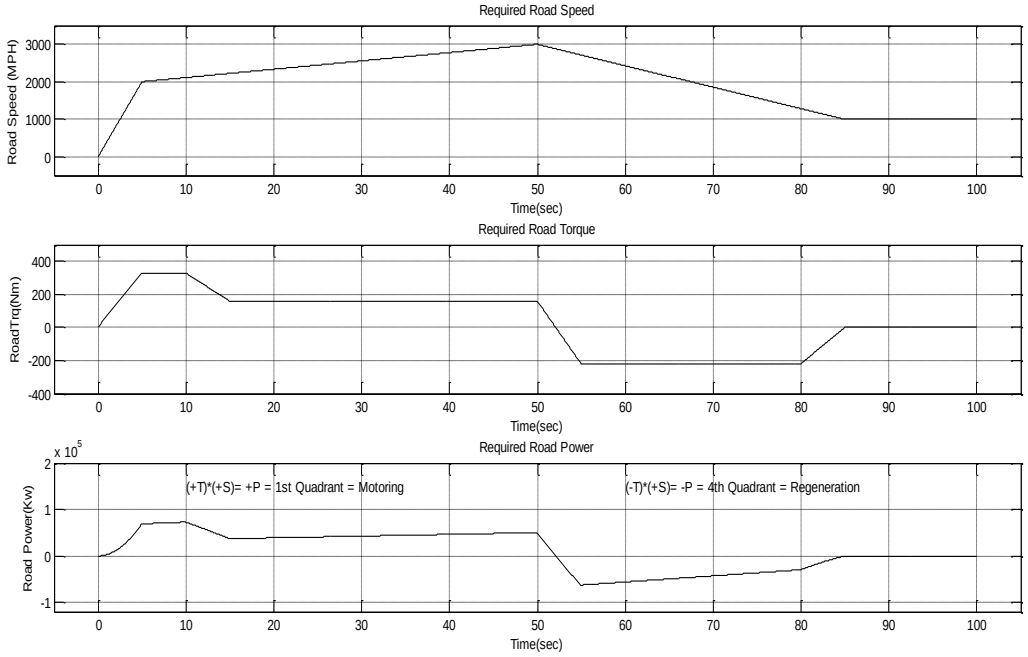

Figure 9: Road Speed, Torque and Power Curves Showing Motoring and Regeneration

Motor Voltage, Current and Power

The motor draws power from the battery as shown below in Figure 10: Motor Voltage, Current, and Power. As can be seen by comparing Figure 9 and Figure 10, the voltage and speed curves generally follow each other, and the torque and current curves also generally follow each other. This general relationship reflects the voltage and torque equations that were discussed earlier.

Motoring and Regeneration

The Motor Power plot in Figure 10 shows both Motoring and Regeneration. When both current and voltage are positive values then the DC Motor is providing torque in the direction of rotation and power is being transferred to the load. This is normal motoring operation. However, when the motor current is in the opposite polarity of the voltage, then the motor is being pushed and acting as a generator with current flow back into the battery.

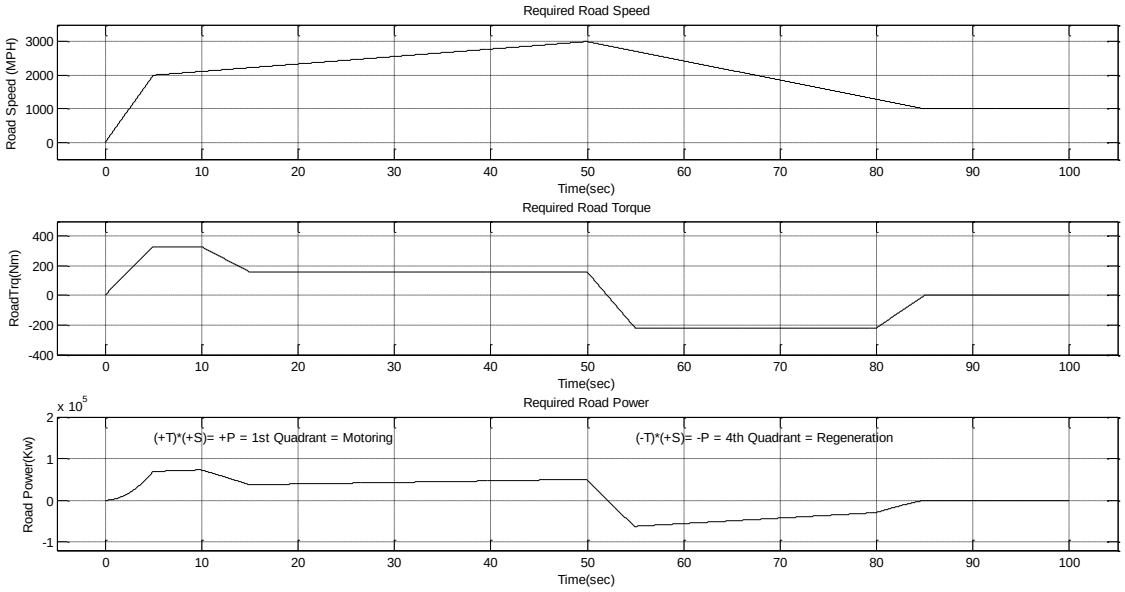

Figure 10: Motor Voltage, Current and Power Curves Showing Motoring and Regeneration

Battery Voltage, Current and Power

The motor draws power from the battery as shown below in Figure 11: Battery Voltage, Current, and Power. As can be seen by comparing Figure 9, Figure 10, and Figure 11, the motor torque, motor current, and battery current curves generally follow each other because torque is

proportional to current. Thus as the torque requirement increases, then the motor must draw more battery current.

Motoring and Regeneration

The Battery Power plot in Figure 11 shows both Motoring and Regeneration. When both current and voltage are positive values then the DC Motor is providing torque in the direction of rotation and power is being transferred to the load. This is normal motoring operation. However, when the motor current is in the opposite polarity of the voltage, then the motor is being pushed and acting as a generator with current flow back into the battery.

Battery Energy

The energy dissipated in motoring and recaptured in regeneration was determined by performing numerical integration of the power curve. Figure 11: Battery Voltage, Current and Power Power (Watt) is to the change of Energy(Joules) with Time (Seconds). Therefore the integral of the power curve is equal to energy in Watt*Seconds. The numerical integration was performed in MATLAB using the trapezoidal rule function, trapz(power,time). The resulting value was then divided by 3600 to get energy in WattHours.

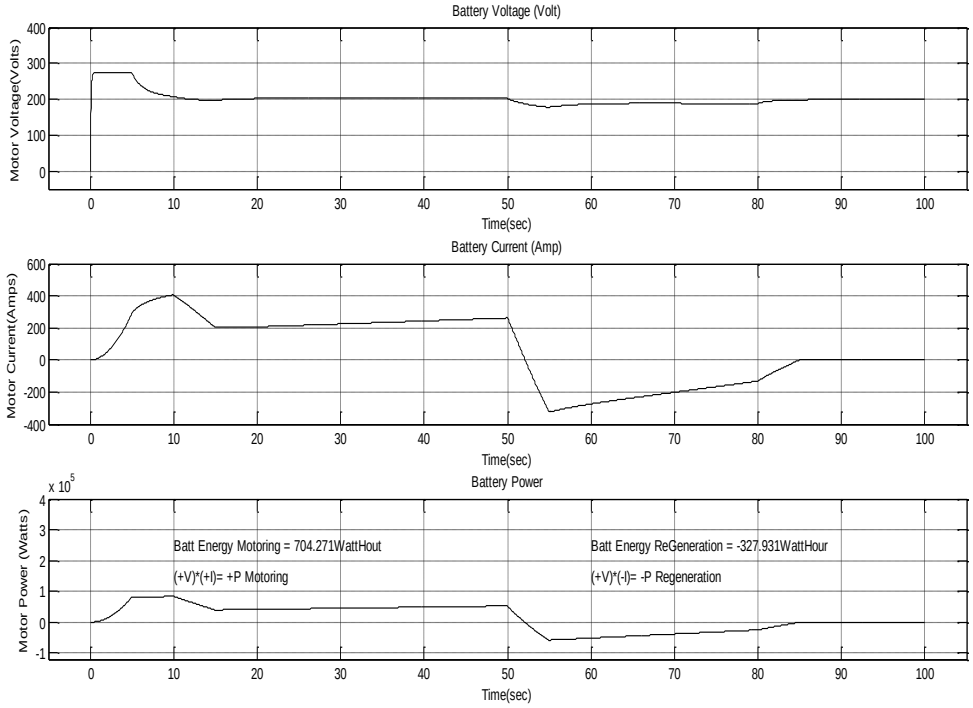

Figure 11: Battery Voltage, Current and Power

Battery Voltage Error (Berr)

The simulation model adjusts the controller gain (K) to meet drive torque and regeneration requirements. The simulation compared the nominal battery internal voltage, VB =200 volts or VBatt(actual), with a calculated battery voltage based on the motor voltage and current values to get VBatt(calculated). The difference, VBerr, was used as an error signal input to the Proportional Integral (PI) Controller. This VBerr signal was plotted over the range of the simulation operation. This plot is shown below in Figure 12: Battery Voltage Error (BErr).

The maximum error of -200 occurs at the very beginning of the simulation. This large error is a natural response to starting the simulation. The simulation quickly recovers and holds an error of about +76 during the initial starting of the motor. It is normal to have a higher error here because the motor developed voltage, $V_D(Volt) = W_D\ (rad/sec)/K_m$, is low during startup, especially when the current is increasing.

The negative error occurs during regeneration. By reviewing the motor voltage drop equation, $V_L\ (Volt) = I_L(Amp)*R_A(Ohm) + E_B(Volt)$, the change in current polarity will cause the reverse polarity of the $I_L(Amp)*R_A(Ohm)$ term. This voltage change will impact the magnitude of the input and output of the PI controller because of the reduced difference between the calculated and actual voltage in the error equation, $B_{ERR} = E_B\ (actual) - E_B\ (calculated)$.

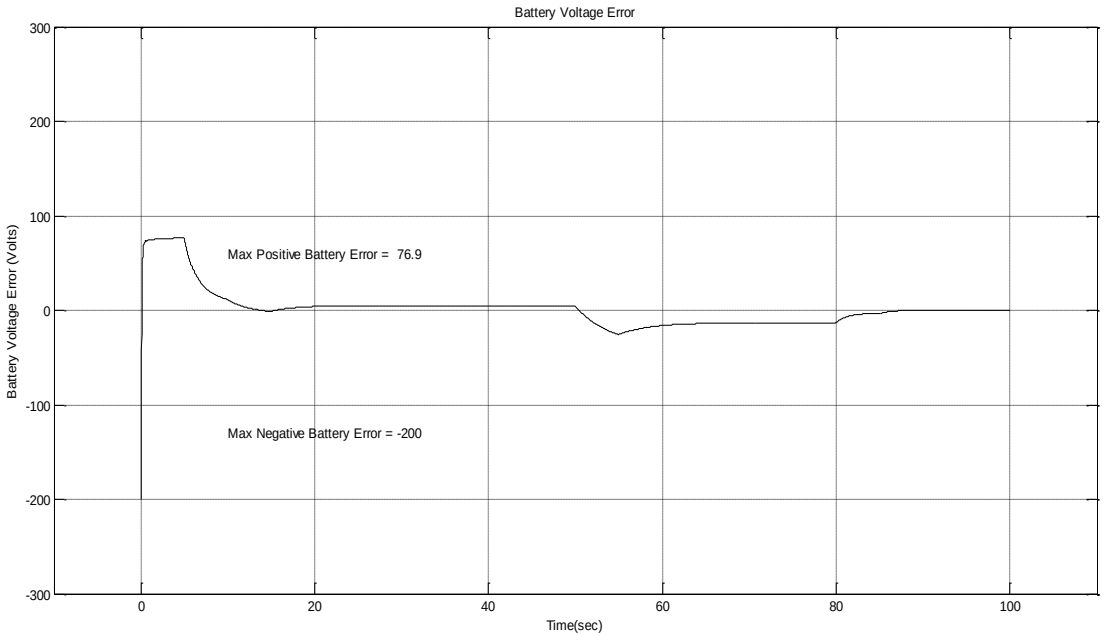

Figure 12: Battery Voltage Error (BErr)

Controller Gain

The Gain (K) of the Motor Controller is determined by the output of the PI Controller model. A plot of the value of the Controller Gain (K) during the simulation is shown below in Figure 13: Controller Gain K Value. The controller gain increases during the time when the motor speed is increasing, and decreases when the motor speed is decreasing.

The gain has an initial starting value of 0.1. This was preset within the controller in the 1/s integration block. This value is set in the simulation by opening up the 1/s block. The addition of the Initial Condition on the integration block helps to minimize the possibility of a Simulink simulation error due to an algebraic loop. An algebraic loop is basically a divide by zero operation when the simulation is trying to solve the set of linear equations.

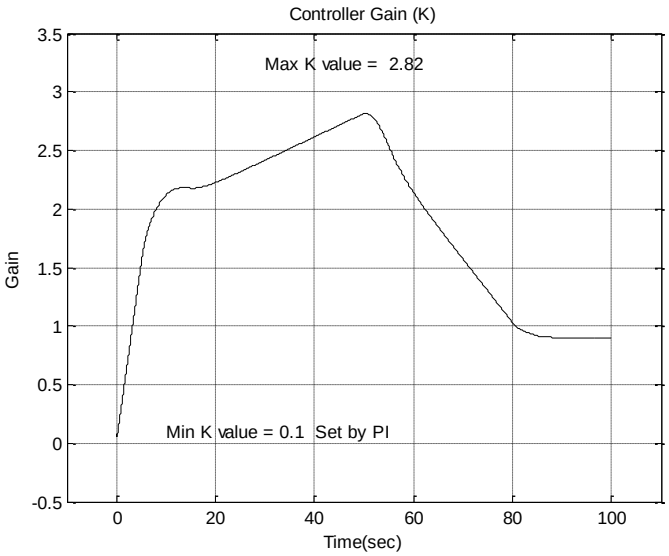

Figure 13: Controller Gain K Value

PI Controller K_p and K_i

The Proportional – Integral Controller (PI) provides an error correction signal that is directly proportional to the system error signal and proportional to the integral of the error signal. The proportional signal helps the controller respond to changes in the system, and the integral signal helps to reduce constant errors by integrating that signal over time.

The K_p and K_i controller constants were determined by trial and error, and the tuning process simply amounted to changing the values while monitoring the magnitude of the Berr signal.

EGEE400 Electric Vehicle Systems Instructional Plan and Assessment

A initial offering of a new course, EGEE400 Electric Vehicle Systems, was offered in Spring 2011 for undergraduate Computer, Electrical, and Mechanical Engineering students. The simulation activity that presented in this paper was not included in that initial course offering, but will be introduced in the next offering.

This initial offering of the course focused significantly on Vector CAN programming. Other topics included the general Battery Electric Vehicle (BEV) power train, an introduction to DC and AC motors, power electronics, and battery systems. The course emphasized software instruction and application. The chassis dyno was not used this time, but should be integrated into the lab in the future.

The textbook Electric and Hybrid Vehicles by Husain was used as a reference for some material. The Vector CAPL programming manual was also required. The appendix from that manual was used for CAN instruction. Handouts from the Vector CAN courses were also used for lecture reference.

Several of the lab assignments used the Vector CAN software, and there was a laboratory exam on CAN. The electric machinery equipment was used for demonstrations and for laboratory exercises on AC power, DC motors, and AC motors.

At the end of the semester the students worked alone or in small groups on technical projects related to the course material. The projects were 1) Use of LabVIEW CAN and interfacing to CANoe, 2) MATLAB Vehicle Network Toolbox, 3) CANoe Simulation Programming, and 4) Investigating the CAN use and extension of CAN on the Vehicle Chassis Dyno.

Some ideas for the next offering include expansion of laboratory exercises such as variable speed control in the machinery laboratory, and possibly touring an BEV assembly plants or colleges that are constructing a BEV. One student suggestion was to convert a Baja vehicle to a BEV.

Specific Course Objectives and Student Self-Assessment

1. Describe electric and hybrid vehicles: 88%
2. Describe basic battery energy storage systems: 88%
3. Describe basic motor systems: 85%
4. Describe power electronics systems in hybrid and electric vehicles: 82%
5. Analyze the operation and simulation of CAN networks: 85%

Summary:

Simulation is a very real and necessary part of electric vehicle development and needs to be integrated into learning experiences within engineering education. Simulation-based testing in the form of Hardware-In-The-Loop testing is also a very necessary part of current engineering development especially in advanced systems such as hybrid and electric vehicle drive systems that rely heavily on complex embedded system subsystems. Student learning experiences that include simulation based design and testing are necessary to include in undergraduate engineering education in order to prepare students for current industry employment.

Acknowledgement:

A portion of the simulation presented here was developed by the author as part of an open ended project in the course AEV 5010 Introduction to Advanced Electric Vehicles. The course is part of the new Advanced Electric Vehicles graduate degree program at the University of Detroit Mercy[22]. The course was taught in Fall 2011 by Dr. M Abul Masrur and used the textbook Hybrid Electric Vehicles that is listed in the references. In addition to teaching at UDMercy, Dr Masrur is a research engineer at the US Army RDECOM-TARDEC where he does research related to hybrid electric vehicles. He was previously with the Scientific Research Labs at Ford Motor Company.

References:

1. Brook, Meeting the technology challenge, AE Int'l, SAE Sept 2009.
2. Ashley, Priming the Green Car Pump, Automotive Engr., Sept. 2009, SAE International.
3. www.abet.org
4. McDonald, Engineering & Technical Education for Electric Vehicle Engr, ASEE AC 2010
5. Rizkalla, Dev Senior Elective for EE and EET Majors in the Design of Electronic Instrumentation for Electric Vehicles, ASEE AC 1998.
6. Rizkalla, Using Senior Research, Design and Development Projects in the Development of a Course in Electric Vehicle Technology, ASEE AC 2000.
7. Rizkalla, Applications of Computer-Based Power Electronics to Electric Vehicle Technology, An Interdisciplinary Senior Course, ASEE AC 2000.
8. Rathod, M., Addressing the Alternative Energy Workforce Needs, ASEE AC 2005.
9. Yet, C., A College-University Partnership Developing a Learning Environment for Hybrid Electric Vehicle Technology" ASEE AC 2007.
10. Staubel, Tesla in article "Meeting the technology challenge" AEInt., SAE Sept 2009.
11. www.doe.gov
12. Waltermann, Hardware-in-the-Loop, The Technology for Testing Electronic Controls in Automotive Engineering. www.dSpace.com
13. Smith, Best Practices for Establishing a Model-Based Design Culture, 2007-01-0777, www.mathworks.com
14. Saurabh, Model-Based Design for Hybrid Electric Vehicle Systems, Paper 2008-01-0085, www.mathworks.com.
15. Herniter, Combining Passion with Fundamentals – Applying Model-Based Design to Education, Paper 2008-01-1292, www.mathworks.com
16. http://www.virtual-car.org/wheels/wheels-road-load-calculation.html
17. http://www1.eere.energy.gov/vehiclesandfuels/pdfs/success/advisor_simulation_tool.pdf
18. Fundamentals of Vehicle Dynamics, Gillespie, SAE, ISBN 1-56091-199-9 pgs 110-115
19. Hybrid Electric Vehicles, Mi, Masrur & Gao, Wiley, ISBN 978-0-470-74773-5 pgs42-45
20. Electric and Hybrid Vehicles Design Fundamentals 2nd Ed. Iqbal Husain, CEC Press, ISBN: 978-1-4398-1175-7

Appendix 1 Table of Model Equations:

Developed Torque is proportional to armature current:

Equation 1: $T_d(Nm) = K_m * I_A(Amp)$ Developed motor torque

Developed Voltage is proportional to armature speed:

Equation 2: $V_D(Volt) = W_D\ (rad/sec)/K_m$ Developed motor voltage

Internal Motor Voltage referred to High Side

Equation 3: $V_H(Volt) = I_H(Amp) * R_A(Ohm) + L_H(Henry) * di(t)/dt(A/s) + V_D(V)$ Motor Voltage

High side voltage is equal to K times the low side voltage:

Equation 4: $V_H = K * V_L$ Controller High Side Current

High side current is equal to 1/K times the low side voltage:

Equation 5: $I_H = (1/K) * V_L$ Controller High Side Voltage

The battery terminal voltage, V_B , is equal to the sum of the internal voltage and resistance voltage drop. The battery voltage and battery current are equal to the controller low side voltage and current.

Equation 6: $V_B\ (Volt) = I_A(Amp) * R_A(Ohm) + E_B(Volt)$. Battery model calculation

$V_L\ (Volt) = I_L(Amp) * R_A(Ohm) + E_B(Volt)$. Assuming: $V_B = V_L$ and $I_A = I_L$

Equation 7: $B_{ERR} = E_B\ (actual) - E_B\ (calculated)$ Error Voltage Calculation

The PI controller accepts the B_{ERR} signal from the Battery Model and uses proportional (Kp) and integral (Ki) to calculate the gain K value that is used by the Motor Controller.

Equation 8: $K = (K_p + s * K_I) * B_{ERR}$ PI Calculation

Appendix 2 MATLAB Script:

A MATLAB Script was used to run the Simulink model for Model Application Road Torque and Speed. The script used the MATLAB Cell mode where the script was divided into functional parts and individual cells could be run alone or sequentially. The first part of the script involved the Clean Up of the Command Window and Workspace to remove residual items.

```
%% AEV5010 Assignment 2
%
% This MATLAB script(AEVAssign2Main.m) performs a simulation on a set of
% Speed, Torque, and Time values. The simulation also provides key output
% Current, Voltage, Power, and Efficiency data.
%
% The script uses the MATLAB Cell Mode to divide the code into the
% following major sections:
% Part 1: Clean Up to remove residual data and commands.
% Part 2: Load Speed and Torque values and Run the Simulink model.
% Part 3: Retrieve signal data from MATLAB Workspace.
% Part 4: Plot the data in MATLAB Figures
%

%% Part 1: Clean Up
% This clean up section removes residual data from the Workspace
% and residual commands from the Command Window.
% Clear the Workspace
clear
% Clear the Command Window
Clc
```

This portion loads the specified Speed, Torque and Time values. It then runs the main Simulink model.

```
%% Part 2: Load Speed and Torque vals and RUN Simulink model

% The key Speed and Torque values are loaded into the MATLAB Workspace
% for use by the Simulink Model.
%
% Load Speed vals and times into the Workspace
Svals = [ 0    2000   3000   1000   1000 ];
Stime = [ 0       5     50     85    100 ] ;

% Load Torque vals and times into the Workspace
Tvals = [ 0     330    330    160    160   -220   -220    0     0 ];
Ttime = [ 0       5     10     15     50     55     80   85   100 ];

% Run Simulink model
% The following sim command will run the simulink model AEVAssign2Main. The
% main model is comprised of several sub-models.
% The models perform key calculations and save the calculated data to the
% Workspace.
% Simulink Scope blocks are connected to the models to view the data.
```

```
% The data from the Scope blocks are stored in structures, and the
% structures are copied to the MATLAB Workspace.
%
% Run the Simulation
sim('AEVAssign2.mdl')
%
```

When the script runs the resulting key data is displayed on Scopes in the simulation and loaded into the MATLAB Workspace as a structure. The individual signals need to be retrieved from the structures so the data can be plotted. This section also uses numerical integration to determine the energy during motoring and regeneration.

```
%% Part 3: Retrieve signal data from MATLAB Workspace.
%
% Data values for each model Scope are packaged as a Structure and saved
% to the MATLAB Workspace. The following commands retrieve each set
% of variable data from the structures in preparation for plotting.
%
% Time data from the Save to Workspace block
Time    = TimeVal.signals.values;
%
% Road Torque, Speed, Power data from the Motor Model Block
RoadTrq = RdTSP.signals(1,1).values;
RoadSpd = RdTSP.signals(1,2).values;
RoadPwr = RdTSP.signals(1,3).values;
%
% Motor Voltage, Current and Power from the Motor Model Block
MtrV    = MtrVIP.signals(1,1).values;
MtrI    = MtrVIP.signals(1,2).values;
MtrP    = MtrVIP.signals(1,3).values;
%
% Battery Voltage, Current and Power from the Battery Model Block
BattV   = BattVIP.signals(1,1).values;
BattP   = BattVIP.signals(1,2).values;
BattI   = BattVIP.signals(1,3).values;
%
% Battery Eb voltage error from the BattErr structure
BattErr = BErr.signals.values;
%
% Motor Controller Gain K value from the Controller structure
K       = Kval.signals.values;

%% Calculate Battery Energy
%
% Used Trapezoidal Rule to determine the integral of battery power from
% zero until the polarity changed to negative. The array index just before
% polarity change was 5211.
%
% Energy Used Motoring
BattE_WHr_Mtr = trapz(Time(1:5211),BattP(1:5211))/3600;
```

```
%
% Repeated to find energy recaptured by regeneration.
BattE_WHr_Gen = trapz(Time(5211:10001),BattP(5211:10001))/3600;
```

The last portion of the MATLAB script plots the data from the Simulation. Because of the length of the code it is divided into three sections. The first section plots the Road and Motor Data.

```
%% Part 4: Plot the data in MATLAB Figures

% Plot Road Torque, Speed, Power

%--------------------------------------------------------------------------
% Plot Road Speed
figure(1)
subplot(3,1,1)
plot(Time,RoadSpd,'k'),xlabel('Time(sec)'),ylabel('Road Speed (MPH)'),grid on
title('Required Road Speed')
axis([ -5 105 -500 3500 ]);
%
% Plot Road Torque
subplot(3,1,2)
plot(Time,RoadTrq,'k'),xlabel('Time(sec)'),ylabel('RoadTrq(Nm)'),grid on
title('Required Road Torque')
axis([ -5 105 -400 500 ]);
%
% Plot Road Power
subplot(3,1,3)
plot(Time,RoadPwr,'k'),xlabel('Time(sec)'),ylabel('Road Power(Kw)'),grid on
title('Required Road Power')
axis([ -5 105 -120000 200000 ]);
text(5,150000,'(+T)*(+S)= +P = 1st Quadrant = Motoring');
text(55,150000,'(-T)*(+S)= -P = 4th Quadrant = Regeneration');

% Plot Motor Voltage, Current, and Power
%--------------------------------------------------------------------------
figure(2)
% Plot Motor Voltage
subplot(3,1,1)
plot(Time,MtrV,'k'),xlabel('Time(sec)'),ylabel('Motor Voltage (Volts)'),grid on
title('Motor Voltage(Volts)')
axis([ -5 105 -400 600 ]);
% Plot Motor Current
subplot(3,1,2)
plot(Time,MtrI,'k'),xlabel('Time(sec)'),ylabel('Motor Current (Amps)'),grid on
title('Motor Current (Amps)')
axis([ -5 105 -500 500 ]);
% Plot Motor Power
subplot(3,1,3)
plot(Time,MtrP,'k'),xlabel('Time(sec)'),ylabel('Motor Power (Watts)'),grid on
title('Motor Power')
```

```
axis([ -5 105 -120000 280000 ]);
text(10,150000,'(+V)*(+I)= +P Motoring');
text(60,150000,'(+V)*(-I)= -P Regeneration');
```

The following portion of the MATLAB Script plots the Battery Voltage, Current and Power as well as the and Battery Error control Berr.

```
% Plot Battery Voltage, Current, and Power
%--------------------------------------------------------------------------
figure(3)
%Plot Battery Voltage
subplot(3,1,1)
plot(Time,BattV,'k'),xlabel('Time(sec)'),ylabel('Motor Voltage(Volts)'),grid on
title('Battery Voltage (Volt)')
axis([ -5 105 -50 400]);
%Plot Battery Current
subplot(3,1,2)
plot(Time,BattI,'k'),xlabel('Time(sec)'),ylabel('Motor Current(Amps)'),grid on
title('Battery Current (Amp)')
axis([ -5 105 -400 600 ]);
%Plot Battery Power
subplot(3,1,3)
plot(Time,BattP,'k'),xlabel('Time(sec)'),ylabel('Motor Power (Watts)'),grid on
title('Battery Power')
axis([ -5 105 -120000 400000 ]);
text(10,150000,'(+V)*(+I)= +P Motoring');
text(60,150000,'(+V)*(-I)= -P Regeneration');
text(10,250000,['Batt Energy Motoring = ' sprintf('%5.0f',BattE_WHr_Mtr) 'WattHour']);
text(60,250000,['Batt Energy ReGeneration = ' sprintf('%5.0f',BattE_WHr_Gen) 'WattHour']);


% Plot Battery Error
%--------------------------------------------------------------------------
figure(4)
% Find maximum value of BattErr in the array
[BEmaxVal BEmaxIndex] = max(BattErr);
% Find minimum value of BattErr in the array
[BEminVal BEminIndex] = min(BattErr);
% Plot
plot(Time,BattErr,'k')
title('Battery Voltage Error')
xlabel('Time(sec)'),ylabel('Battery Voltage Error (Volts)'),grid on
text(10,-130,['Max Negative Battery Error = ' num2str(BEminVal)])
xlabel('Time(sec)'),ylabel('Battery Voltage Error (Volts)'),grid on
text(10,60,['Max Positive Battery Error = '  sprintf('%5.1f',BEmaxVal)]);
axis([ -10 110 -300 300 ]);
```

The final portion of the MATLAB Script plots the K value.

```
% Plot Controller Control Gain (K)
%-------------------------------------------------------------------------
% Find maximum value of K in the array
[Kmax Kindex] = max(K);
% Plot Kval
figure(5)
plot(Time,K,'k')
title('Controller Gain (K)')
xlabel('Time(sec)'),ylabel('Gain'),grid on
text(30,3.25,['Max K value = ' sprintf('%5.2f',Kmax)]);
text(10,0.1,'Min K value = 0.1  Set by PI')
axis([ -10 110 -0.5 3.5 ]);
```