Battery Life Estimation in PHEV Applications: A Monte Carlo Simulation Study

BJ Yurkovich, Oruganti Prashanth Sharma, Yann Guezennec, Simona Onori

Center for Automotive Research - The Ohio State University, Columbus, OH 43212 USA, e-mail: {yurkovich.7, sharma.264, guezennec.1, onori.1}@osu.edu

Abstract: With the onset of the electrification of vehicles, battery life estimation has become a very important issue in the design and commercialization of vehicle electrification technologies. Designing and implementing algorithms to adequately predict battery life is essential for the wide-scale adoption of hybrid vehicles, i.e. both charge-sustaining electric vehicles (HEVs) and plug-in electric vehicles (PHEVs) and pure electric vehicles (EVs). In order to effectively design methods for predicting battery life, accurate and efficient modeling and simulation tools are required to support in the design, identification, and modeling of such methodologies and algorithms. In this paper, we will present a modeling and simulation tool that is intended to be used for battery life estimation in PHEV applications. The dynamic PHEV model is based on many years of development at The Ohio State University's Center for Automotive Research in Columbus OH. The focus of the paper is not so much on each individual vehicle component of the model, but on the design and implementation of the software model. In addition, we will present the design and implementation of parallelized Monte Carlo simulations completed on The Ohio Supercomputer Center using the software model. At the end of the paper, we will present some example results and metrics obtained from the simulation infrastructure that can be used for battery life estimation.

Keywords: hybrid and electric vehicles, powertrain simulation, batteries,driveline simulation, modeling, simulation, monte carlo, supercomputing

1. INTRODUCTION

Since the adoption of electric transportation has become more prevalent in the technology market, issues of battery life have become a major focus of research. In order to design and implement effective battery aging methodologies and techniques, detailed models and simulation infrastructures need to be developed to support the study of battery life estimation. There exists some models and simulators that describe battery and vehicle models for Hybrid and Electric Vehicles (i.e., HEVs/PHEVs/EVs) and the associated components such as the engine, electric machine, and control strategy , such as in Yurkovich and Guezennec (2009), Tulpule et al. (2010), Plett (2004) and Kroeze and Krein (2008), Moura et al. (2011).

There has been a significant amount of work on the the design and implementation of hybrid vehicle models and simulators (see Guzzella L. (2007) and Miller (2003) and references therein). Most of these models are implemented in the industry standard Mathworks Simulink software program. In addition, rigid software design principles, such as Object Oriented design principles, are rarely enforced in the design and implementation of such models. Therefore, a refactoring of the existing models that are found in the literature is required in order to support the research in the area of Battery life estimation.

In this paper, we focus on the development, design, and implementation of such a model and simulator that can be used to complete massive distributed simulations in order to study the effects that driving of a PHEV has on battery life. We have chosen a model that was developed at the Center for Automotive Research at The Ohio State University. In addition to describing the tool, the simulation infrastructure that is required to perform large scale will also be presented.

At the end of the paper, we will show some example battery aging results that were gathered from running the simulation on a supercomputer. Finally, using battery aging methodologies such weighted amp-hour counting defined in Onori et al. (2012), metrics will be defined to showcase how driving effects battery life in a PHEV.

2. PHEV MODEL AND SIMULATOR

2.1 Original Model

We have chosen to use a specific PHEV model and simulator implemented Simulink model as a basis that was

⋆ Sponsor and financial support acknowledgment goes here. Paper titles should be written in uppercase and lowercase letters, not all uppercase.

developed using years of research at the Center for Automotive Research at The Ohio State University Tulpule et al. (2009). Although the model physics are well designed, the software design does not lend itself well to reusability and scalability. Therefore, a more rigid software design approach needs to be taken in order to allow for the model to be extended to support the research of battery life estimation.

For small scale simulation to determine how a single PHEV driving pattern effects the battery pack, the described Simulink Model works well. However, in order to adequately determine the true impact that driving has on a PHEV battery pack, large numbers of simulations must be completed varying battery pack size, engine size, location, temperature, and driving profile distance and velocity. Therefore, using a single Simulink model to simulate all possible cases would take a number of years and significant user interaction.

Therefore, to make the model more accessible, it was decided that a refactorization from the implementation of the Simulink model to a more modular MATLAB implementation was necessary.

2.2 New Model Design and Advantages

Using Object Oriented Software Engineering concepts, a more robust, scalable, and reusable PHEV model was designed and implemented. Because MATLAB's programming language is more suited to large scale simulation (as opposed to Simulink's graphical style), client implementors of the base model will have an easier time extending the model to include the necessary components for their research.

The power of Object Oriented Programming (OOP) becomes extremely apparent when trying to architect an OOP PHEV model. OOP allows for each component in the vehicle, such as the Engine, to be a separate object upon instantiation. These objects can be strung together to create the overall vehicle system. In addition, each of these objects can build on one another to create bigger and more robust components while still retaining the basic properties and actions (called "methods") that the base object had. In this spirit, we create a **VehicleComponent** class which can be inherited by all other components in the vehicle, such as the Engine. The **VehicleComponent** class contains properties such as simulation step size (defaulted to 10Hz) and debugging settings, and a parameter loading method that is generic to all possible vehicle components. Other classes include

- Battery Cell
- Battery Pack
- Battery Pack Controller (or Battery Management System)
- Battery Severity Factor Calculator
- Battery System
- Control Strategy
- Driver
- Electric Machine
- Electric Machine Gearbox
- Engine (Diesel)
- Engine Gear Box
- Fuel Tank
- Gear Selector
- Torque Converter
- Vehicle Dynamics
- Wheels

It should be noted that the entire battery system is broken out into individual objects. This allows for more control over how the battery pack is configured. For the simulations in this paper, we are using the cell model that comes from Yurkovich and Guezennec (2009). In addition, the MATLAB implementation of the model uses 1^{st} and 2^{nd} order backward-looking Euler methods.

As with all Object Oriented architectures, the model is intended to be extended by others who may wish to study specific parts of a vehicle in more detail such as different control strategy implementations or specific engine/electric motor configurations. The model architecture gives a flexible and robust framework for vehicle simulations.

When it comes to loading input parameters in a Simulink model, there are only a certain number of ways to accomplish the task: (1) loading variables in from the MATLAB workspace, or (2) embedding the parameters into the actual model. Neither method allows for much organization and causes many headaches in debugging and extending the model. By porting the implementation to MATLAB, an eXtensible Markup Language (XML) Document Object Model (DOM) can be utilized to assist in the organization and specification of input simulation parameters. By specifying a XML DOM class that is responsible for loading in simulation parameters, all parameters are able to be stored in a human readable textual file that is organized and easily scaled to include future extensions to the model.

In addition to providing an organized method for specifying input parameters, the refactoring from a Simulink implementation to a pure MATLAB implementation provides for a framework that is easily scalable. It is often difficult to scale a Simulink model to support large scale simulations. Refactoring the Simulink model to MATLAB model allows for a much more streamlined process in scaling the simulator to be run in a Monte Carlo-like fashion or to be used in complex large scale simulations that utilizes dynamic programming.

2.3 PHEV Model Example Input

Before we define an entire Monte Carlo simulation infrastructure that uses the discussed model implementation, we must first verify that the PHEV model gives reasonable output. We have chosen a PHEV model that roughly simulates a PHEV Sports Utility Vehicle (SUV). Table 1 describes sample model input parameters. It should be noted the "charge sustaining mode" executes when a 30% State of Charge (SoC) level of the battery pack is reached. In addition, a 1^{st} order battery model described as

$$V(t) = E_0 - IR_0 - V_c \tag{1}$$

where E_0 is the open circuit voltage of the battery, i is the input current to the battery, R_0 is the internal resistance of the battery scheduled on temperature, SoC, and current,

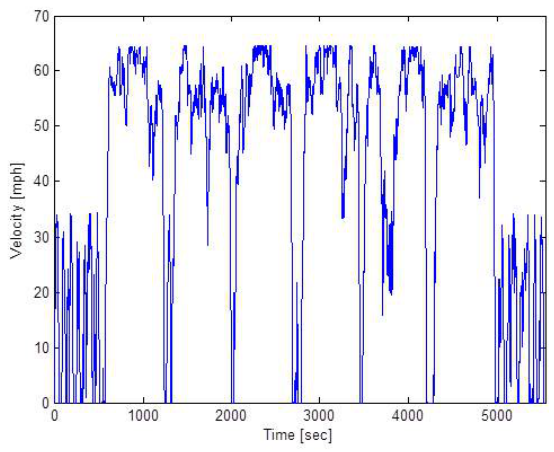

Fig. 1. Example Velocity Profile

and V_c is an RC circuit also with parameters scheduled on SoC, temperature and current is used to model the battery cells in the pack (see Yurkovich and Guezennec (2009).

In order to show the complete operation of a single vehicle, we have chosen an example velocity profile of about 60 miles simulating a combination of both urban and highway driving. By simulating a profile that is around 60 miles long, the charge sustaining mode will execute after the battery pack SoC reaches a certain level. The velocity profile is shown in Figure 1.

2.4 Example Model Results

The best way to first verify the validity of the simulation results from the PHEV Model is by comparing the expected and actual velocities. From simulating the model, we obtain that the simulated velocity tracks the expected velocity well with an RMS error of less than 0.41%.

As one can see in Figure 2, the control strategy of the vehicle model begins in a pure-EV mode, and as soon as the SoC of the battery reaches an SoC of 30%, the mode of the vehicle changes to charge sustaining and the engine starts, resulting in a battery SoC that hovers around 30% by $\pm 3\%$. This can also be seen in Figure 3 which shows the torque of the engine as zero before the ECMS control strategy switches from a pure EV mode to a charge sustaining mode.

The vehicle, since it is modeled after an SUV, obtains an average mile per gallon rating of 32.4 mpg.

Keeping with a C-rate profile typical of a standard PHEV-20 or PHEV-30, we see in Figure 4 that the average C-Rate (defined as the rate at which a battery is discharged

Table 1. Sample Vehicle Parameters

Parameter	Value
Max Engine Torque	325 Nm
Max Electric Motor Torque	325 Nm
Battery Pack State of Charge (SoC) Range	25% to 90%
Batteries in a String	90
Battery Strings in Parallel	25
Battery Pack Size	16.5 kWh
Battery Pack Nominal Voltage	288V
Vehicle Control Strategy	ECMS

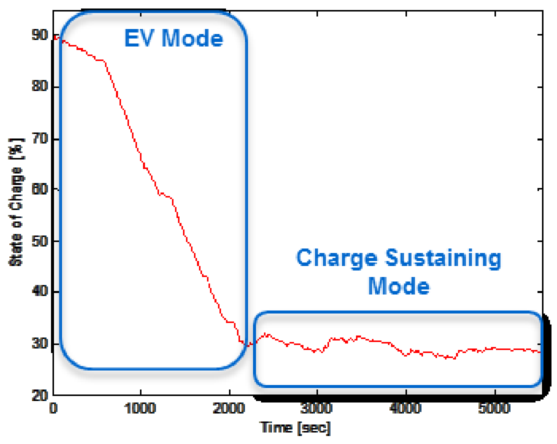

Fig. 2. Example State of Charge

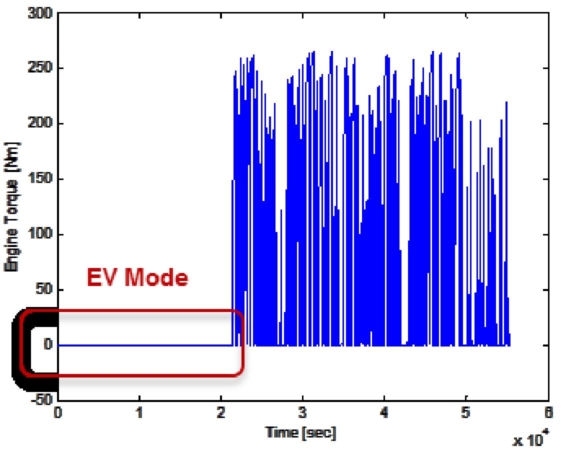

Fig. 3. Example Engine Torque

relative to its capacity) over the entire operation of the Vehicle for the prescribed velocity profile is 0.39C. From the histogram, we can see that most of the output current is clustered around 0C, and rarely rises above 2.5C in the case of a discharge and -2C in the case of a charge. The reason that the C-Rate skew favors the discharge current should be obvious; the vehicle is in a pure EV-mode for the first part of the trip, and therefore experiences a number of situations that require a large amount of acceleration which in turn result in a large current request, and therefore discharge, from the battery pack. In addition, the battery pack is really only charged on regenerative breaking and therefore does not experience a large amount of negative current over a typical drive cycle.

3. LARGE SCALE SIMULATION SOFTWARE INFRASTRUCTURE

One of the main reasons for the development of such a vehicle model was to provide a robust tool for large scale simulation since large scale simulation cannot be easily done using Simulink or other similar implementation tools. By defining an example large scale simulation infrastructure, we hope to show the usage of such a tool that will allow for the simulation of multiple different vehicles, environments, and drive cycles. With results from these

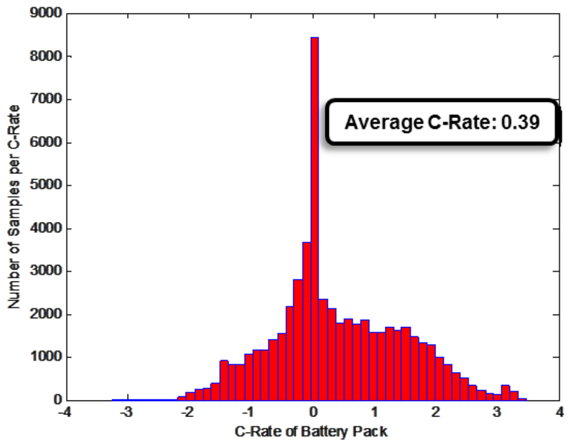

Fig. 4. Example Current C-Rate Histogram

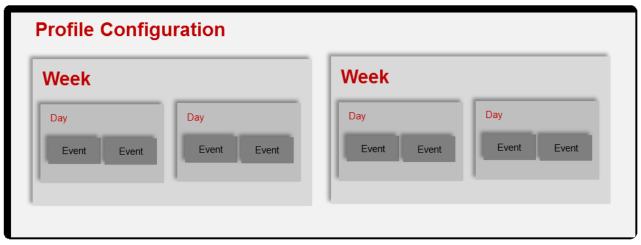

Fig. 5. Simulation Architecture Diagram

multiple different types of simulations, we can perform an analysis of battery aging, study typical PHEV usage, and achieve a better understanding of long-term PHEV usage on a large scale.

3.1 Large Scale Simulation Architecture

In much the same way as the Vehicle model was designed, the large scale simulation modeling uses Object Oriented principles. By doing so, the simulation architecture can be compartmentalized into discrete objects, thus allowing for easier experiment design and organization.

Software System Architecture The system is designed using Object Oriented concepts, thus allowing for straightforward implementation in MATLAB. In order to design an experiment that will encompass a typical driving year for an individual, we must define a **User** object. A **User** is an individual who can own a **Vehicle**.

This **User** has certain driving habits, commutes to and from work, as well as charging habits. In order to complete a number of random different years, a **Profile Configuration** (which can also be thought of as a year) class is defined, which contains a number of **Week** objects. The **Week** object, in turn, is made up of a number (usually 7) **Day** objects. A typical day for a user is made up of a number of trips called an **Event** . An **Event** object may represent a drive to work or home from work, an errand, or a trip on the weekend. In addition, an **Event** may also be a charging task. A graphical representation of this description can be found in Figure 5.

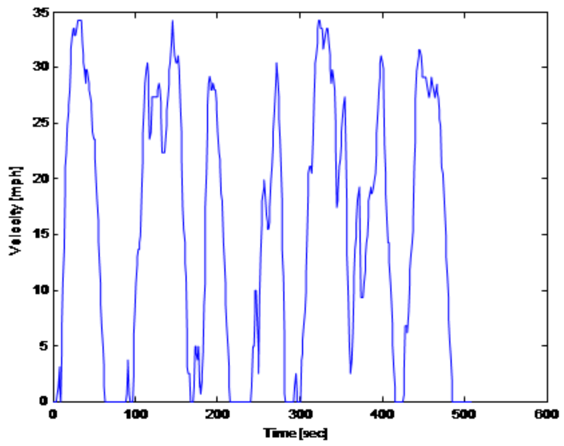

Fig. 6. Example Urban No Traffic Drive Cycle

Each **Event** is made up of a number of driving cycles, each of which is called a **Driving Segment** , that have been generated using a Markov Chain approach.

3.2 Using a Markov Chain to Generate Drive Cycles

In order to meaningfully perform simulations on a large scale, we must first generate a library of multiple different driving cycles that a **User** could potentially perform during a typical year (or **Profile Configuration**). There are 6 different types of **Driving Segment** profiles that are typical of the different **Driving Segment** profiles that the Markov Chain generates:

(1) Urban Traffic
(2) Urban No Traffic
(3) Freeway Traffic
(4) Freeway No Traffic
(5) Highway Traffic
(6) Highway No Traffic

Each **Driving Segment** cycle has different characteristics that reflect the type of drive expected from an Urban, Freeway, or Highway drive cycle. The Markov Chain model for each of the typical driving profiles is used to generate 1000 different velocity profiles for each of the 6 different **Driving Segment** profiles. Figures 6, 8, and 7 show examples of a Highway No Traffic, Urban No Traffic, and a Freeway No Traffic drive cycle. The Driving Cycles were generated using a training set based on experimental vehicle velocity traces.

When running large scale simulations, a complete **Event** can be constructed by choosing a predetermined sequence of **Driving Segment** profiles at random and stringing them together. For example, to make an **Event** that exemplifies a typical user's 14 mile drive to work during rush hour, one Urban Traffic **Driving Segment** cycle can be combined with 2 Highway Traffic **Driving Segment** cycles chosen at random out of their respective pools. For more information on the Markov Model, see Gong et al. (2010).

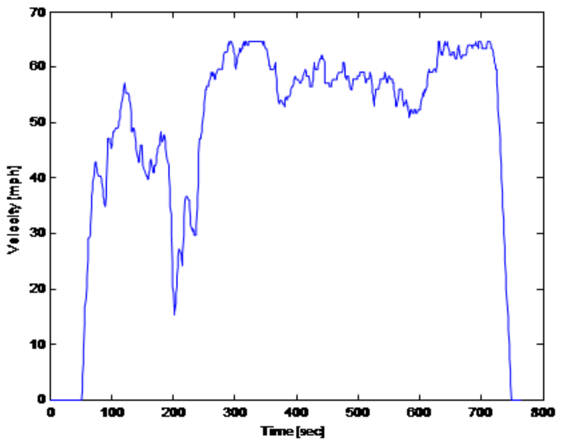

Fig. 7. Example Freeway No Traffic Drive Cycle

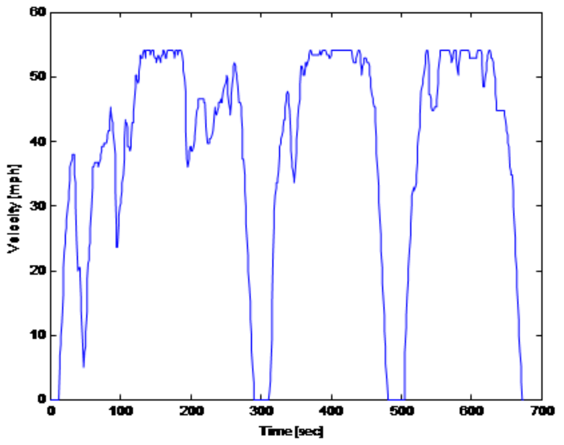

Fig. 8. Example Highway No Traffic Drive Cycle

3.3 Example Monte Carlo Simulation Infrastructure

In order to show how this simulation infrastructure can be used, we will create an example **User** to be simulated in a Monte Carlo fashion. For our example purposes we will only define a single **User** . It should be noted, however, that the simulation architecture has the ability (and is intended) to support a large number of users in order to explore a number of different kinds of driving styles and vehicles.

In true Monte Carlo fashion, the **User** that is defined will have 100 **Profile Configuration** sets representing 100 different possible years that a user could drive that will be simulated. Each of the **Profile Configuration** sets will contain 4 **Week** sets, representing 4 season in each year: Winter, Spring, Summer, and Fall. Each **Week** contains 7 days - 5 weekdays and 2 weekend days.

Each of the 5 weekdays contains a commute to and from work, totaling a total of 29 miles both ways. Each commute (both to and from work) is assumed to be made in traffic and consists of an Urban Traffic **Driving Segment** , a Highway Traffic **Driving Segment** , and another Urban Traffic **Driving Segment** . Once at work, there is an assumed mandatory Level 2 charge (220V at 32A, 7kW). After work, there is a possibility of an errand composed of 0 to 3 Urban No Traffic **Driving Segment** cycles. The total distance of the errands have the possibility of being between 0 miles and 6 miles. At night, it is assumed that the **User** will always charge the Vehicle at Level 1 (110V, 12A, 1.3kW).

On the first day of the weekend, it is assumed that the **User** will make a some sort of errand that can range from 2 miles to 8 miles (1 to 4 Urban No Traffic **Driving Segment** cycles). At night, there is again another assumed Level 1 charge. On the second day of the weekend, there will be a 2 mile commute (one Urban No Traffic) errand and one long trip that will total 65 miles. The long trip is composed of 2 Urban No Traffic, 6 Freeway No Traffic, and another two Urban No Traffic cycles.

For every event, the location basis is assumed to be Columbus Ohio. In order to get temperatures, a historical weather archive was consulted to obtain average temperatures for each of the 4 seasons. In this example, only 4 temperatures are considered. In a larger simulation, more temperature variation may want to be considered. The infrastructure is capable of handling multiple temperature variations over the course of a **Day** . The season temperatures that were decided upon are annotated in Table 2.

Table 2. Season Temperatures

Season	Temperature
Winter	5^{o}
Spring	23^{o}
Fall	19^{o}
Summer	29^{o}

More input parameters for the Monte Carlo simulation are shown in Table 3. It should be noted that there is only 1 ”long trip” assumed per season. In addition, the distance traveled per year is calculated by multiplying the distance of 4 seasons by 13, which assumes 52 weeks in a year. As it can be seen in Table 3, it is clear that 12,739 miles is a reasonable mileage for a typical user that has a fairly long commute to work.

Table 3. Simulation Example Parameters

Parameter	Value
Average # of driving hours per weekday	1.21 hours
Average # of driving hours per weekend	1.36 hours
Average # Errands per Week	4
Average errand distance	4 Miles (max: 8mi)
Est. Distance Traveled per Year	12,736 miles
# of charging events per year	624 Events
Total # Charging Hours	420
Vehicle Control Strategy	ECMS

3.4 Running the Simulation

Since each **Event** of the Simulation Architecture requires an insignificant number of seconds (20-30 sec) to simulate, completing thousands of simulations linearly is not a practical option. Therefore, the simulations were dispatched on a supercomputer.

The Ohio Supercomputer Center (OSC) in Columbus Ohio was used to accomplish the parallelization task. OSC boasts over 3000 Opteron processors and the ability to perform 75 trillion flops per second. In addition, OSC also has a software service that is called Remote MATLAB Service (RMS), based on the pMALTAB implementation from MIT. RMS allows for the OSC users to utilized the OSC job manager to submit MATLAB applications and scripts to the scheduler to be run. In this way, the PHEV model was submitted to OSC to be run.

Although OSC's RMS tool is very usable, it does not give adequate feedback as to the progress of the simulations as they are run on the supercomputer. In the case of the PHEV model, the simulation takes a number of hours (or possibly days) to complete, and without proper feedback from the simulation, very little is known as to whether or not the simulation was actually successful in completing its task. Therefore, as an extension to the Monte Carlo Simulation Infrastructure, a Representational State Transfer (REST) web service with a complimentary MATLAB/Java and Python API to report in real time that status of PHEV Model (such as velocity, battery pack SoC, etc) was designed and implemented. A front-end web and mobile application was also developed to allow for easy analysis the status of the simulations.

Using the REST web service and front-end reporting website in conjunction with the PHEV model and Monte Carlo Simulation Infrastructure allows for seamless completion and analysis of the simulations.

4. DEVELOPMENT OF METRICS FOR ANALYZING THE EFFECT OF DRIVING ON BATTERY PACK

In order to quantify the effect that PHEV driving has on the battery pack in terms of battery life, we must first define a metric that describes the severity that certain types of driving and environment has on the battery pack. We will call this the Severity Factor.

4.1 Severity Factor

The Severity Factor is defined as a metric that measures the effect that the acts charging and discharging has on a battery based on Battery SoC, Battery Depth of Discharge, and Battery Temperature. As one can see in Figure 9, the severity factor is more severe in points of extreme temperature (both hot and cold) and high depth of discharge. The trick for maintaining optimum battery life, then, is to operate the battery in the "sweet spot" where there is mild temperatures and low depth of discharge. For more information on how the severity factor map is derived, see Onori et al. (2012).

4.2 Weighted Amp Hour vs. Amp Hour

The severity factor map and the SoC from a vehicle simulation can be used to obtain the "weighted amp hour." The weighted amp hour is a measurement of the amp hours used over a simulated drive cycle with the previously defined severity factor applied.

Using the concept of a weighted amp hour, a metric can be defined to show the difference between a normal amp hour usage over a drive cycle compared to a weighted amp hour usage. The equation

$$\lambda = \frac{A_W - A_N}{A_N} \tag{2}$$

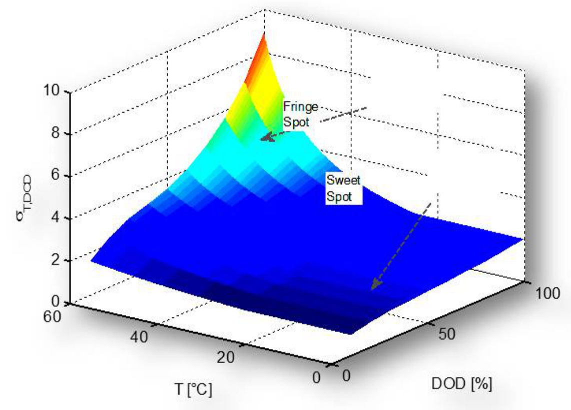

Fig. 9. Severity Factor Map based on Temperature and DOD

where A_W is the Weighted Amp Hour of the battery, and A_N is the Normal Ampere Hour of the battery defines a way of normalizing the difference between a normal amp hour measurement and an amp hour measurement where a severity factor is applied.

Using the comparison described in Equation 2, we can look at how different environments, vehicle configurations, and PHEV users effect the aging of a battery pack over a number of years of PHEV operation.

5. RESULTS

Although the specific results of Monte Carlo simulations is not the main focus of this paper, it is important to show some results to motivate how such a tool can assist in the analysis of PHEV operation. Even from the smaller number of **Profile Configuration** sets and only one **User** , we are able to draw some meaningful conclusions about how the driving of the PHEV effects battery life.

Firstly, we can analyze the SoC Band distribution over the different seasons. In Figure 10, it can be seen that the majority of SoC range over all operation of the vehicle is around 80% to 90% SoC. This is expected since there are so many opportunities for the PHEV **User** to charge the battery pack, and the pack is recharged after every PHEV usage. In addition, we see that there is a significant amount of SoC clustering in the 60% band. This is to expected since it is around the 60% SoC range that the battery pack reaches on a drive to work and a drive from work to home. Along that same reasoning, there is an SoC clustering around 30% which is to expected, since the ECMS puts the mode into charge sustaining and therefore keeps the battery pack charge around 30% SoC.

In Figure 11, we see the majority of the C-Rates congregate around zero, which also is expected, since the charging C-Rates for Level 1 and Level 2 are 0.2C and 0.5C respectively. In addition, we see that there is a larger skew towards the positive C-Rates (discharging), and can easily be explained by they fact that the only charging C-Rates that the pack would experience would be from regenerative breaking. Therefore, we would expect a significant skew in

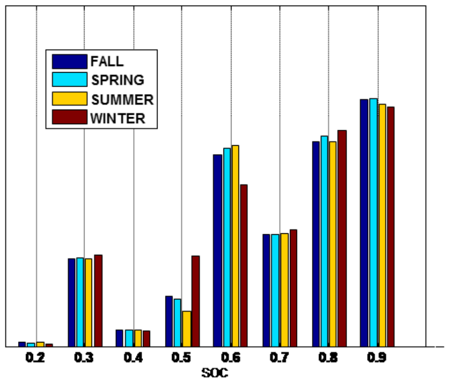

Fig. 10. Battery Pack SoC Band Distribution

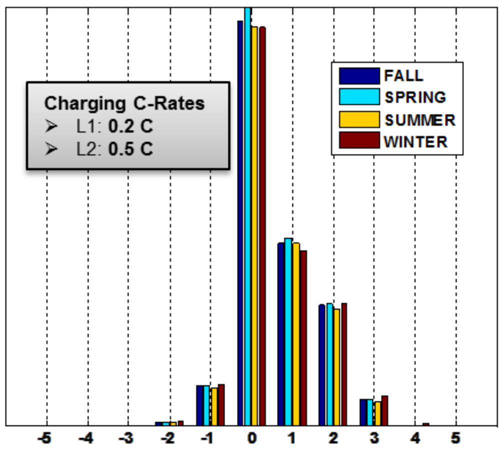

Fig. 11. C-Rate Histogram for All Events

the discharging direction on the histogram resulting from PHEV battery pack usage.

Figure 12 shows some interesting trends in the how the different seasons, and therefore temperatures, can effect the battery pack of a PHEV. Recalling from Table 2, we know that the Summer **Week** in our simulation has the highest temperature. Therefore, knowing what we know about the severity factor map as shown in Figure 9, higher temperatures have the greatest severity impact on a battery pack. Thus, as Figure 12 shows, the most significant severity factor can be found in the Summer **Week** of the simulations (a factor of almost 1.8). As described before, the higher the severity on the battery pack, the worse the aging effects are.

Figure 13 shows the normal amp hour vs. the weighted amp hour measurement over the entire 100 **Profile Configuration** sets resulting from applying the severity factor to the normal amp hour. Noting that the normal amp hour is linear, we can again see that the effects of seasonal temperature effect battery operation in a large way. As we

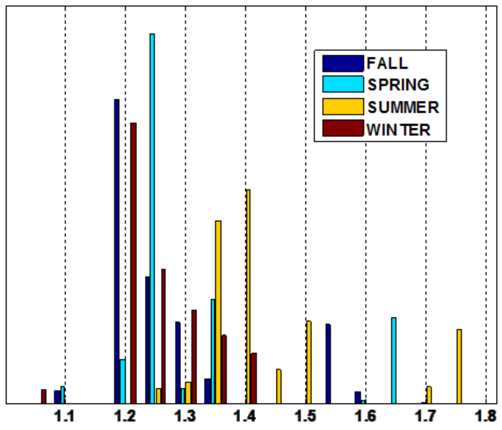

Fig. 12. Severity Factor Distribution for All Events

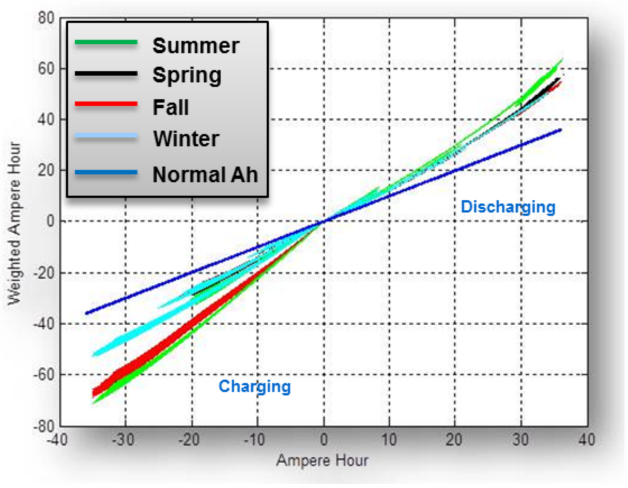

Fig. 13. Severity Factor Distribution for All Events

would expect, the most moderate temperatures (in Spring) are closest to the normal amp hour. In addition, from our analysis of the severity factor in Figure 12, we would expect that the **Event** set that occurred in the summer would show the greatest effect on battery aging due to the higher temperatures. This analysis shows that in using this model tool, we can actually quantify the specific severity that the battery pack experiences in normal operation. This analysis could be extended in future simulations to include temperature variation over a number of days based on historical temperature data, instead of just a few constant temperatures.

From the data shown in Figure 13, we can apply the normalization metric defined in Equation 2 to see the trends in the effect that driving has on the battery pack. By normalizing the data in Figure 13, we find that the most significant λ value of 0.5 belongs to the summer months as seen in Figure 14. In fact, the effect that of the temperature in the summer months is magnified in the summer months in a large way.

With larger simulations sets using the defined tool such as more **User** definitions, more **Profile Configuration** sets per PHEV owner, and different vehicle configurations,

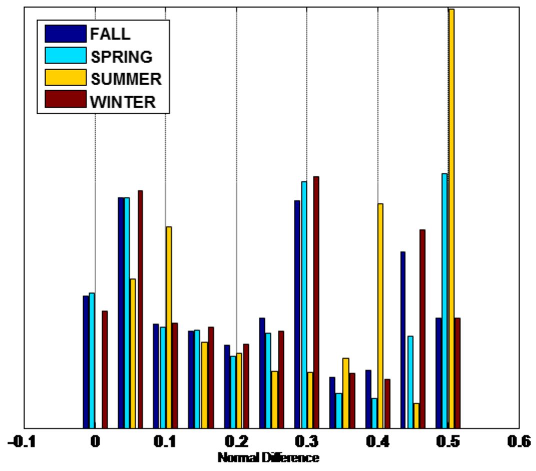

Fig. 14. Distribution of Weighted vs. Normal Ah - λ

even more information can be gathered that can be used to quantify how varied parameters effect battery aging.

6. CONCLUSION

In this paper, a Plug In Hybrid Electric Vehicle modeling and simulation infrastructure and model implementation was introduced that is capable of providing and extendable and resulable tool as well as simulating parallelized PHEV operation on a massive scale. By performing large scale simulations using the tool, analysis of how PHEV operation and the environment can effect the life of the on-board battery pack can be completed.

In addition to presenting the tool, an example Monte Carlo simulation infrastructure was defined and simulated in order to motivate the tool and provide some initial insights into the effects of PHEV operation on the battery pack.

Future work will include the simulation of more PHEV users to gain a better understanding and quantify the effects that different locations, temperatures, vehicle configurations, velocity profiles, and driving habits impact the life of the PHEV battery pack.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the Ohio Supercomputer Center and the The Center for Automotive Research at The Ohio State University.

REFERENCES

Gong, Q., Midlam-Mohler, S., Marano, V., and Rizzoni, G. (2010). An iterative markov chain approach for generating vehicle drive cycles. *SAE International*.

Guzzella L., S.A. (ed.) (2007). *Vehicle Propulsion Systems - Introduction to Modeling and Optimization*. Springer Verlag, 2nd. Ed.

Kroeze, R. and Krein (2008). P. electrical battery model for use in dynamic electric vehicle simulations. *J. Am. Chem. Soc.*, 1336–1342.

Miller, J. (ed.) (2003). *Propulsion systems for hybrid vehicles*. IEE Power and Energy Series.

Moura, S.J., Fathy, H.K., Callaway, D.C., and Stein, J.L. (2011). A stochastic optimal control approach for power management in plug-in hybrid electric vehicles. *IEEE Transactions on Control Systems Technology*, 19(3), 545–555.

Onori, S., Spagnol, P., Marano, V., Guezennec, Y., and Rizzoni, G. (2012). A new life estimation method for lithium-ion batteries in plug-in hybrid electric vehicles applications. *Int. J. Power Electronics*, 4(3).

Plett, G. (2004). High performance battery pack power estimation using a dynamic cell model. *IEEE Transactions on Vehicular Technology*, 1336–1342.

Tulpule, P., Marano, V., and Rizzoni, G. (2010). Energy management for plug-in hybrid electric vehicles using equivalent consumption minimization strategy. *Int. J. Electric and Hybrid Vehicles*, 2(4).

Tulpule, P., Stockar, S., Marano, V., and Rizzoni, G. (2009). Optimality assessment of equivalent consumption minimization strategy for phev applications. *Proceedings of Dynamic Systems and Control Conference*.

Yurkovich, B. and Guezennec, Y. (2009). Lithium ion battery pack modeling and simulation for automotive applications. *Proceedings of Dynamic Systems and Control Conference*.