The Assessment of Battery-Ultracapacitor Hybrid Energy Storage Systems

by

Yiou He

B.E., Tsinghua University, Beijing, China (2012)

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2014

Author: Signature redacted

Department of Electrical Engineering and Computer Science
May 21, 2014

Certified by: Signature redacted

John G. Kassakian
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by: Signature redacted

Professor Leslie A. Kolodziejski
Chair, Department Committee on Graduate Theses

Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

The Assessment of Battery-Ultracapacitor Hybrid Energy Storage Systems

by

Yiou He

Submitted to the Department of Electrical Engineering and Computer Science
on May 21, 2014, in partial fulfillment of the requirements for the degree of
Master of Science

ABSTRACT

Battery-ultracapacitors hybrid energy storage systems (ESS) could combine the high power density and high life cycle of ultracapacitors with the high energy density of batteries, which forms a promising energy storage system.

In this thesis, an assessment of the benefits of the hybrid ESS relative to its battery-only counterpart in pulse-load applications is investigated for both Nickel-Metal Hydride (NiMH) batteries and Lithium-ion (Li-ion) batteries, and under different load profiles. Specifically, the hybrid ESS in this assessment is of the simplest type – paralleling the ultracapacitors across the batteries without any power electronics interface between them.

To quantify this assessment, Discharge Capacity(Ø) is defined as the amount of energy one can draw out of an ESS per unit charge supplied by this ESS. The metric for quantifying the benefits is energy efficiency gain, defined as the percentage increase in the discharge capability of the hybrid ESS over its battery-only counterpart.

The investigation proves that the hybrid system is more beneficial over the battery-only system in terms of how much energy it can output at a specific state-of-charge level. Among the test cases covered by this thesis, the increase in the output energy of Li-ion battery systems by incorporating ultracapacitors can reach to 17% and that of Ni-MH battery systems can reach to 33%.

This thesis also shows that the benefits of paralleling ultracapactors across batteries depended upon the discharge profile of the load, the battery type and the capacitance. The benefits increase quadratically with the pulse amplitude, decreases linearly with the duty cycle and inverse with the pulse period. Moreover, capacitors with higher capacitance and lower ESR yield to larger benefits. And for batteries with a higher ESR, the ultracapacitors will show more benefits than for batteries with low ESR.

Thesis Supervisor: John G. Kassakian
Title: Professor of Electrical Engineering and Computer Science

Acknowledgement

In retrospect, the past two years at MIT has been a great fortune and an invaluable experience for me. There are many people have helped me, encouraged me and contributed to this thesis, and I could hardly find words to express my gratefulness to them.

First of all, my deepest gratitude is to my advisor, Professor John G. Kassakian, for giving me this opportunity to explore and enjoy the energy and power electronics world. His patience, his immense amount of knowledge, and the way of thinking have given me a precious guidance not only in research, but also in life, and in how to become a better person. I couldn't feel more fortunate and grateful to be his student.

I must extend my thanks to Mr. Otten and Richard Zhang, for giving me so many valuable instructions on circuit design and numerical simulation respectively. They have shared their vast amount of knowledge with me so generously, and have "unstuck" me from problems again and again. This thesis could not be finished without your help.

I would also like to sincerely thank my lab mates in LEES. Thanks to Minjie, Nelson, Juan and Jorge, for listening to my grumble as well as happiness in countless days, and for giving me valuable advice and suggestions. Thanks to Lisa, for accompanying me so many times for lunch and going through the darkness together. Thanks to Wei, Kendall, Seungbum, Alex, John and little John, Youngjin, Mark, Anas, Matt, Wardah, Jimmy Peng and everybody else in 10-061 and 10-050, for bringing sunshine and memorable moments to the basement of the MIT dome. Special thanks to Sam Gunter. Without her super computer, I would have spent weeks to run the simulation and process my data.

In addition, I would like to thank all my dear friends in or outside of MIT. A particular thank to Jiankang, Mie, Sumaiya, Xiaoxue, Nahan and Yihui, for caring me so much, calming me down and teaching me to enjoy research and enjoy life. I greatly value their friendship and deeply appreciate their understanding and patience to me. Thanks to my friends in MIT Energy Club and Ashdown House, for making my two years at MIT a rich and rewarding experience. Thanks to all my friends and mentors back in China who have been so supportive and have my back.

Most importantly, I want to express my deep gratefulness to my parents, my grandparents and my cousins, for their forever support and love. My study in the US would have been impossible without the love, concern and patience from you. May this thesis serve as a dedication to the values that I learned from you, my families.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Motivation

Pulse current or power releases the stored energy in a concentrated short-period pulse or pulses. Numerous processes and applications in industrial production and daily life involve or could be seen as pulse current loads. For example, wireless sensor signals or communication signals of electronics devices often consist of pulses of milliamp and milisecond periods [1]; power tools such as metal cutters, power drills, which usually cut materials or drill a screw in several seconds and rests in between for minutes [2]; an automotive engine starter, which requires an initial high power pulse of hundreds of amperes [3]; fluctuations and faults in electronics systems could also be seen as pulsed loads.

Batteries are known for their high energy density and have been widely used as the main energy source in many pulsed load applications. However unlike running at constant load, when high current pulses are impressed on batteries, the relatively large internal resistance of the batteries will create a voltage drop, which may cause the output voltage unable to hold as required. Moreover, it will generate heat inside the batteries, reduce the available state-of-charge (SoC) window and decrease the life time.

Thus, batteries are usually not capable of running the same time length under pulsed loads, comparing to under the constant load which have the same average current, as shown the run-time comparison in Fig. 1-1 [4]. Considering the high usage rate of batteries in pulse load applications, how to improve their run-time and discharge efficiency has become a critical issue.

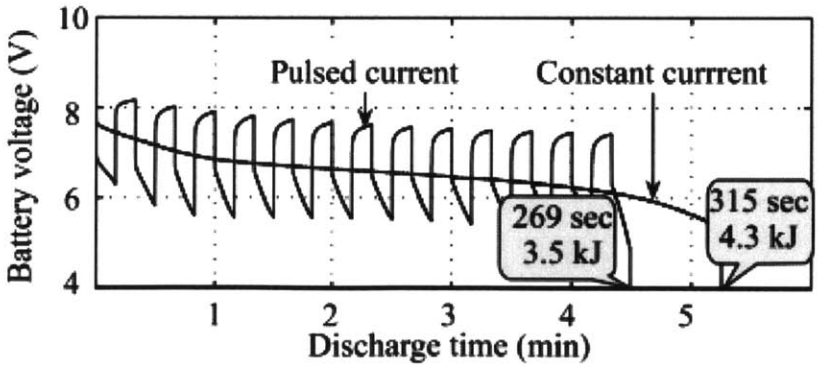


Fig. 1- 1 Pulse and Constant Current Comparison (Discharging at a 6C constant current and 12C pulsed current of a 20 s period and a 50% duty cycle) [4]

The ultracapacitor, or more descriptively the electrochemical double-layer capacitor (EDLC), is a capacitor having an extremely high capacitance and a very low equivalent series resistance (ESR) comparing with an electrolytic capacitor of similar physical size. It is not a new technology but has advanced rapidly in recent several years, and has been widely exploited to mitigate pulse load applications using batteries [4], [5].

Many investigations of battery-ultracapacitors hybrid systems have focused on combining EDLCs with batteries through a power electronics interface, such as a dc-to-dc converter [6]. The most popular application of ultracapacitors is in the field of hybrid/electric vehicles (HEV/EV) [5], [7]. Other research is being conducted on integrating battery-ultracapacitors hybrid systems into renewable generation systems, mostly solar arrays [8]. With the use of power electronics as the interface, it has been shown that adding an ultracapacitor to batteries (or other energy storage systems) can relieve the stress on the battery by sharing the load current.[6], [8].

1.2 Literature Review and Previous Work

Little research has been done on quantifying the benefits of the simple battery-ultra-capacitor paralleled systems [9], [10], [11].

Reference [9] describes a similar pulse-load application but only through simulation and lacks hardware experiments. Besides, the models of the battery and ultracapacitor were the ideal 'ESR-only' models which can only show a general approximation of actual results.

Reference [10] focused on transient load suppression - the voltage drop due to a transient current load (single pulse). Experiments were run on a 10.8 V Li-Ion laptop battery, which was discharged by low current pulses of 2 A. The voltage drop during discharge pulses decreased from 0.7 V to 0.08 V after adding a 7 F ultracapacitor. The unique point in [10] is that they also conducted the same experiments with tantalum electrolytic capacitors, which shows that the ultracapacitor has great advantages over normal capacitors in terms of suppressing voltage drop, as well as size and volume. The results in [10] are very helpful to explain the theory of how paralleled ultracapacitors benefit the battery, but it lacks a direct explanation of energy efficiency saving and neglects the importance of long-time experiment. In addition, there is no further analysis to correlate benefits with the load characteristics.

Researchers in [11] focused on improving battery lifetime with a hybrid energy storage system, and shared similar goals with this thesis. Both experiments and simulations were conducted for a 14.4 V Li-ion battery with a pulse-current load of 2.3 A. The experimental results show that the increase in the battery run-time when adding an equivalent 3.34 F ultracapacitor could reach 4.3%; and when adding an equivalent 6.68 F ultracapacitor, the

increase could reach 12%. However, the simulation results in their investigation, which are respectively 3.6% and 6%, differ substantially from their experimental results,. This is due to the models of batteries and ultracapacitors in their simulation being Thevenin ESR-only circuit models which provide simplicity but sacrifies accuracy. These models do not take the transient behaviors of batteries and ultracapacitors into consideration. The battery's ESR-only model used in [11] also cannot describe the characteristics of battery's open circuit voltage changing with its state-of-charge in a long-term run.

This thesis will continue investigating based on the previous work done by Ian Smith from 2010 to 2012 [12]. His experiments also consisted of applying a pulsed current load to both hybrid ESS and battery-only ESS and measuring the energy delivered to the load as well as the charge drawn from the batteries. His charging and discharging circuit and experimental board are shown in Fig. 1-2. The experiments were done by paralleling three 25 F ultracapacitors (equivalent 8.33 F) with five 1.5 V Ni-MH batteries in series (equivalent 7.5V), under four different discharging current profiles. Results are listed in Table 1-1. It shows that the hybrid ESS can provide significant efficiency gains over the battery-only ESS, which could be as high as 35% (when the load current is 16 A, 10% duty cycle, 400 ms period).

His experimental results were also used to verify a circuit model for the hybrid ESS, allowing for future testing and design based upon simulation.

Table 1- 1 Energy efficiency gain of hybrid ESS over battery-only ESS (period: 400ms) [12]

Amplitude / Duty Cycle	8 A	16 A
10%	13%	35%
25%	11%	27%

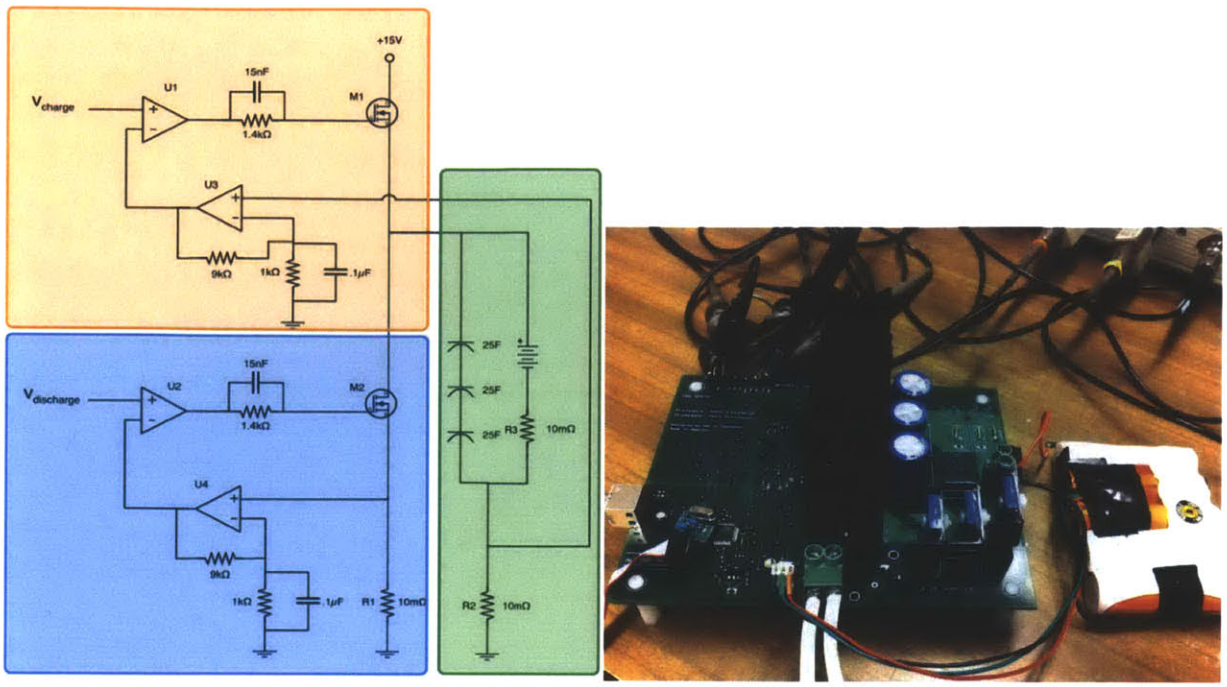


Fig. 1- 2 Charging and discharging circuit (left) and its experiment PCB board (right) [12]

1.3 Statement of the Thesis

This thesis assesses benefits of battery-ultracapacitor parallel energy storage systems (ESSs) in pulsed load applications. For an electric vehicle, people care most about how long their battery can run from one charge; for a power drill, people care about how many more screws it could drill before recharging. These all depend on how much energy one could get out of the batteries before depletion.

Thus the main benefit this thesis investigates is dischargeable energy, i.e., how much more energy the hybrid ESSs can draw out of the same capacity batteries comparing to the battery-only ESSs. This could also be seen as a type of "energy density". By simply paralleling the EDLCs across batteries, they absorb high frequency and large peak current, and thus reduce the current as well as the energy loss inside the batteries. In addition, there will also be benefits in terms of cost, volume, temperature and complexity.

This thesis will focus on investigating the benefits in high-current low-frequency pulsed load applications, with pulse amplitudes from 3A to 20A, pulse periods from 50 ms to 400 ms and duty cycles from 5% to 50%. These load profiles can be seen in power tools and power equipment, and if the power and voltage level were extended further, the application would be in the automotive industry and public transportation system.

The assessment of the benefits covers two different types of rechargeable batteries: the nickel-metal hydride (Ni-MH) battery and the lithium ion (Li-ion) battery (particularly lithium iron phosphate battery, LiFeP04). The former one is of relative high energy density, safe and environment friendly, widely used in telecommunication, wireless communication and automotive industry [13], [14]. The latter one is one type of Li-ion batteries, with longer life time, higher power density and more safety than commonly used LiCoO2 batteries [15].

The investigation also covers several other degrees of freedom, including various capacitance values of ultracapacitors, as well as the period, the duty cycle and the amplitude of the pulse loads, as illustrated in Fig. 1-3. Both experiments and simulations were conducted in these several degrees of freedoms.

Results of this thesis reveal how much more energy one could get out of batteries by paralleling ultracapacitors across them, which relates to how much longer the batteries could run. The results also show the trend of the benefits and give a guideline of how to select proper ultracapacitors for a certain load profile to get higher efficiency.

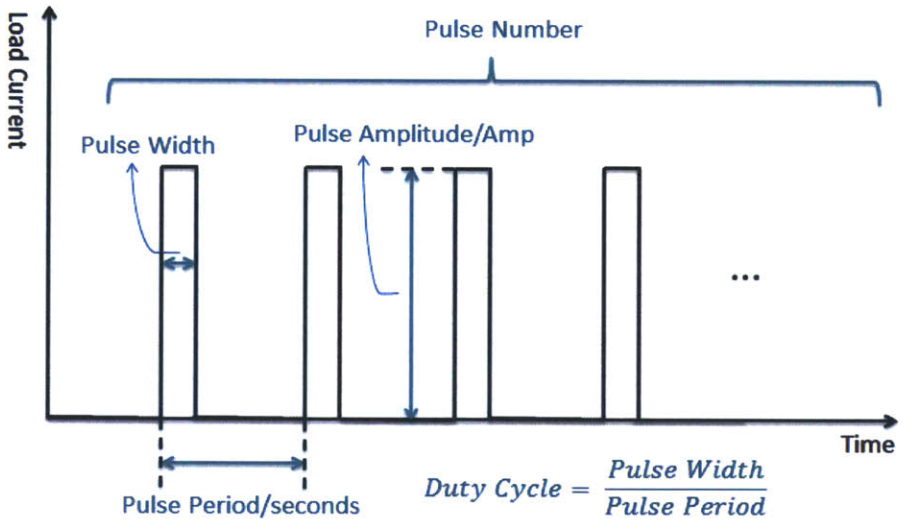


Fig. 1- 3 Information of pulse load profile

This thesis consists of five chapters. Chapter 2 covers background information and knowledge, including the chemical characteristics of EDLCs, the modeling of Li-ion batteries and ultracapacitors. Chapter 3 presents the design of experiments, including how the metrics of this assessment were defined and how the investigation process was planned. A simple theoretical analysis is also presented to analyze what variables will influence the metrics and how the metrics will change with these variables. Chapter 4 describes the design of the hardware apparatus and hardware experiments. A test rig was designed in order to draw the desired current profiles and measure the critical variables of

the energy storage systems. In Chapter 5, a circuit model for the hybrid ESSs was built and verified by the hardware experiment, and it was further used to simulate more sets of experiments and helped further study the trend of the benefits over different degrees of freedom. Chapter 5 compares and interprets both experimental and simulation results, as well as addresses future work.

Chapter 2 Background

2.1 Characteristics and Applications of Ultracapacitors

The capacitors generally can be classified into three different types: electrostatic capacitors, electrolytic capacitors and electrochemical capacitors. The electrostatic capacitors are simply formed by two metal electrodes with non-conductive dielectric between them. In the electrolytic capacitors, the dielectric is replaced by a much thinner oxide layer coated on one of the metal electrodes, and a conductive soaked paper inserted between the oxide layer and another metal electrode, serving as the electrolyte [16]. At a macroscopic level, both these capacitors can be evaluated using the equation

$$C \propto \frac{A}{d} \tag{1}$$

Where C is the capacitance, A is the surface area of one plate of either positive or negative charges and d is the distance between positive and negative plates.

The electrochemical double-layer capacitor (EDLC) is a type of electrochemical capacitors built based on the theory of "electrical double layer effect". The "electrical double layer effect" actually appears at every boundary between two dissimilar materials, which means that there always exists an array of charged particles and dipoles at the interface of two different materials [17] [18]. Specifically if the two materials are electrode and electrolyte, when impressing an electric field across their boundary, the charge accumulates and forms a strong internal electric field, illustrated in Fig. 2-1. This charge and the generated electric field form an equivalent capacitor. As with traditional capacitors, this capacitance is given by (1).

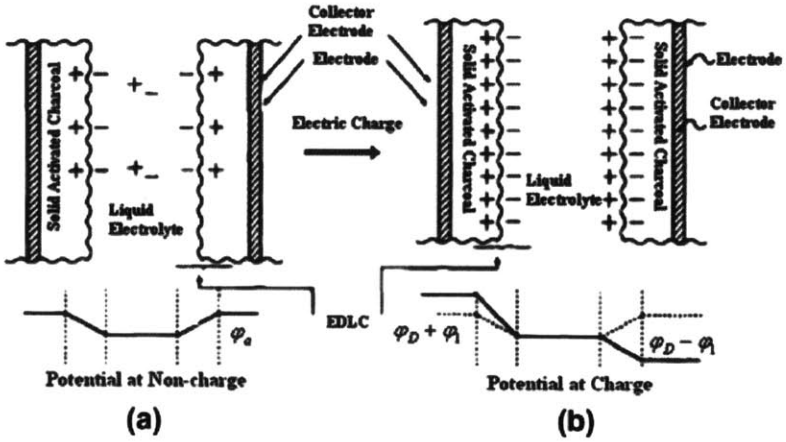


Fig. 2 - 1 Theory of electrical double layer [16]

The extremely high capacitance of EDLCs is a result of using porous (active) electrode material such as carbon and metal-oxides. The surface area of these materials is on order of 10^5 times that of smooth electrodes, which brings the surface area A to the extreme. And the very low ESR comes from the use of a high conductivity electrolyte, such as organic and aqueous [16] [19] [20].

Because the energy stored in a capacitor increases linearly with its capacitance, and the low ESR allows high currents to go through which means it can also provide higher peak power, the EDLC combines both high power density and a relatively high energy density. The Ragone plot shown in Fig. 2-2 gives a qualitative comparison of the energy and power densities of batteries, traditional capacitors and EDLCs. The EDLCs can be seen as transitional between traditional capacitors and batteries.

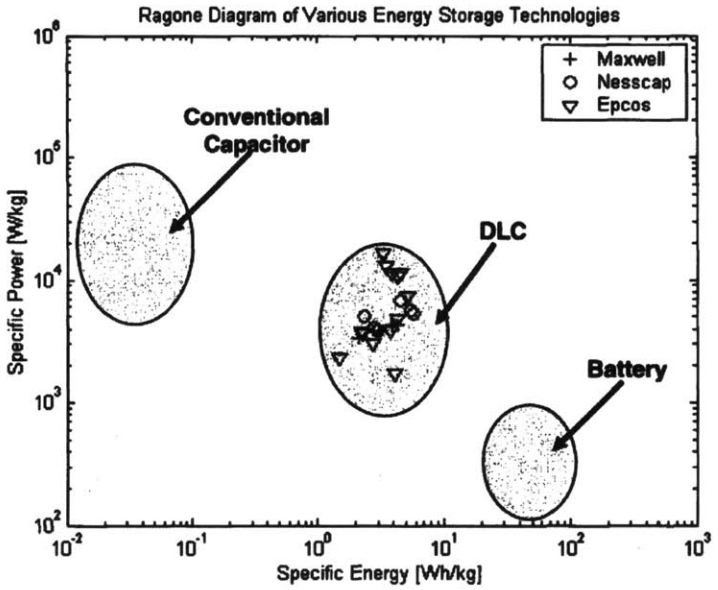


Fig. 2 - 2 Energy density and power density comparison using Ragone plot (conventional capacitors, EDLCs and batteries) [19]

Other promising advantages of ultracapacitors include their long lifetime as well as strong reversible storing and releasing charge capabilities. They are capable of withstanding a large number of charge/discharge cycles without sacrificing the available total state of charge [16]. A comparison table of the specifications of ultracapacitors, Ni-MH batteries and Li-ion batteries is shown in Table 2-1, from which one can get a general overview of how the energy density, the life cycle and the power density of these three energy storage devices compare.

Table 2- 1 Comparison of Specs of different energy storage device [21] [22]

	Ni-MH battery (typical)	Li-ion battery (typical)	Ultracapacitors (typical)
Energy density (Wh/kg)	60 – 120	120 – 200	5 – 8
Power density (kW/kg)	0.25 – 1	0.3	8 – 15
Rated voltage (V)	1.2	3.6/3.2	2.7
Longevity (cycles)	$10^2 - 10^3$	$10^2 - 10^3$ $10^3 - 10^4$	$10^5 - 10^6$
Temperature dependency	High	High	Minimal

The history of ultracapacitors is summarized in reference [16]. They were first designed for military uses such as power sources for electronic fuses and arming devices, missile guidance systems and laser weapons [17]. Then, the EDLCs were used as the standby power for integrated circuit memories, power systems and electronic devices [23], [24]. However, the energy density of EDLCs is still relatively low compared with that of commonly used batteries, thus rather than being used as a main source of energy, the EDLCs are most beneficial when used as a complement power source for other energy sources, typically batteries, fuel cells, etc. A good example of using the EDLCs to complement batteries is regenerative braking in the automotive industry, such as electric vehicles [25] [26] and subway transportation systems [27]. The EDLCs are used to capture the energy at very high power normally lost during braking and reuse it. In this way, ultracapacitors help to both save total energy and to drive the engine during peak power demand, thus improving the power density of the whole energy storage system.

2.2 Modeling of Ultracapacitors

A number of electric circuit models for EDLCs have been developed for simulation purposes and to better understanding their behavior. Characteristics and trade-offs of different types of models are compared in [16], [23], [28]. Depending on their application and desired accuracy, the models for EDLCs can vary from very simple to very complex.

Because the charge diffuses at different speeds based on the different physical sizes of the pores in the active electrode, RC pairs with different time constants are typically used to model the charge balancing inside the EDLCs [29], [30], [31]. Different models chose various combinations of RC pairs (in series, in parallel, or in transmission line form), however, different RC pairs usually represent distinct time constants with an order of magnitude separation from each other.

The EDLCs have been observed to have voltage-dependent capacitance due to the interfacial tension in the double layer [32], which is usually modeled as a nonlinear

capacitor. In addition, the ESR and the capacitance of EDLCs are also changing with frequency and temperature, the effect of which has also been modeled [33], [34].

Based on the goal of this particular investigation, several requirements were considered when selecting the model, as listed below:

- The model needs to accurately simulate the performance of EDLCs in the pulse period range of 1 ms to 10 s. Because the pulse period of interest was designed to be from 50 ms to 400 ms, the upper and lower boundaries were selected to leave some space to guarantee accuracy. The accuracy outside of this pulse period range could be compromised.

- The model should not be too complicated or contain too many components. On the other hand, it should not be too simple to fail to represent crucial characteristics of the EDLCs, for example, the nonlinear capacitance behavior.

- Frequency and thermal effects on the EDLCs will not be considered. A fan was designed to cool down the EDLCs, Li-ion batteries and the switches and keep the temperature almost constant (details in Chapter 4).

The model proposed by Zubieta [30], shown in Fig. 2-3, contains three branches of RC pairs, a leakage resistor and a nonlinear capacitor. Because it provides a simple circuit configuration and relatively accurate representation of the EDLC's performance, this investigation will model the EDLCs based on this equivalent circuit.

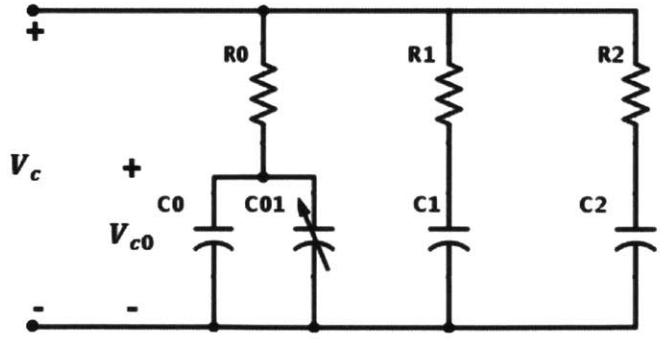


Fig. 2 - 3 Equivalent circuit model for EDLC proposed in [30]

Parameter Identification Methods for EDLC models:

The parameter identification methods of this model have been studied in details in [19] and [30]. Usually a pulse charge experiment, shown in Fig. 2-4, needs to be done: a fully discharged EDLC (terminal voltage is zero) is charged by a current pulse with a given value until its voltage reaches the rated voltage, then charging is stopped and the voltage of the

EDLC is monitored during a long duration. The values of all R and C will be calculated by analyzing the voltage waveform. The parameter extraction experiments were also conducted on the same hardware board, and described in detail in Chapter 5.

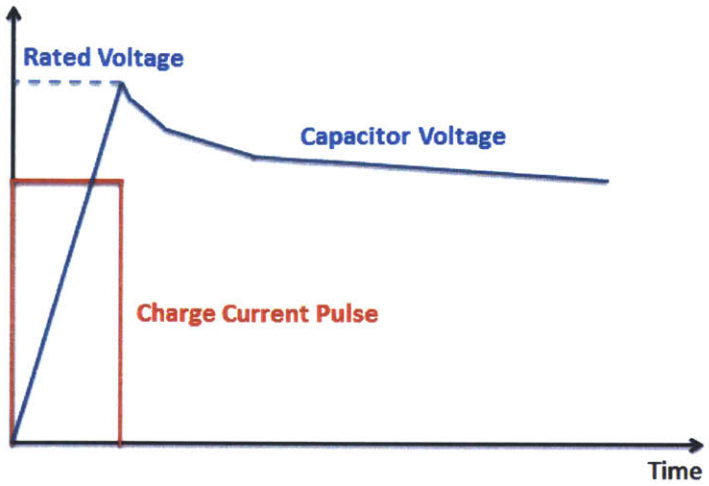


Fig. 2 - 4 Single pulse charge experiment for EDLC parameter identification

2.3 Modeling of Batteries

There are three popular types of existing battery circuit models which are compared in [35] [36]. Each of them features short-term time domain, frequency domain and long-term time domain characteristics of a single-cell battery.

A battery can be simply modeled as a Thevenin equivalent with an open circuit voltage source and series resistor or impedance. In a long time period, the open circuit voltage (OCV) V_{oc} changes according to the change of the state-of-charge (SOC) of the battery, thus V_{oc} could be modeled as a charge-dependent (or current-dependent) voltage source.

Since the goal of this thesis cares about both long-term and short-term performance of the batteries and how much overall benefits the ultracapacitors could bring to the batteries from fully charged to fully discharged, the widely adopted runtime model proposed by [35] was used, as shown in Fig. 2-5. The left part of the model represents long-term performance of the battery, which consists of the self-discharge resistance $R_{self-discharge}$ and the capacitor $C_{capacity}$ indicating total capacity of the battery. The right part of the model consists of the equivalent series resistance (ESR) R_s, and RC pairs modeliing transient performance at different time constant. Two or three pairs of RC are usually used in most models.

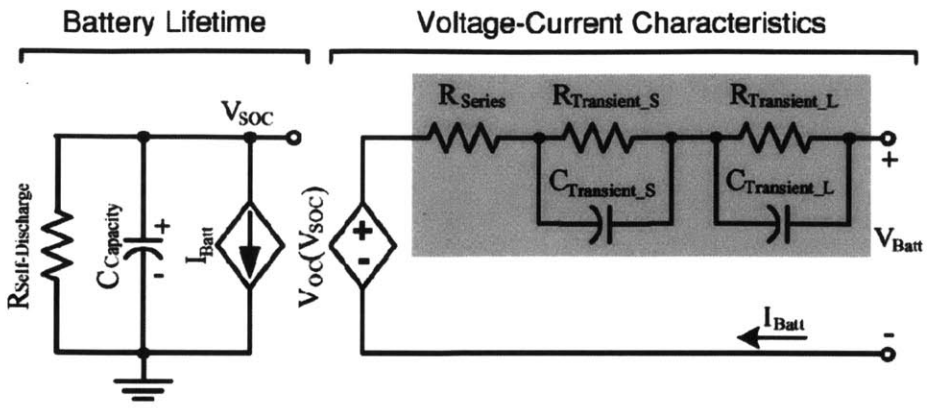


Fig. 2 - 5 Circuit Model for batteries proposed by [35]

The selected battery model has been widely revised and tested for commonly used 4.2 V Li-ion batteries, Lithium Polymer batteries and Nickel-metal Hydride battery (Ni-MH) and Lead-acid batteries [35], [37], [38]. Relatively less research has been conducted to verify whether the model would fit for LiFePO4 batteries [39], [40]. Even though some results indicate that there are errors in the model when LiFePO4 batteries are discharged to low SOC [39], for this investigation, because the 90% SOC to 20% SOC part of the performances influences most to our results, the accuracy is acceptable and the model proposed in [35] is used, but with different experimental sets to extract parameters.

Parameter Identification Methods for Battery models:

Based on Fig. 2-5, there are several parameters whose values need to be determined through parameter identification experiments: $R_{self-discharge}$, $C_{capacity}$, SOC – OCV relationship, series resistance R_s, short-time transient RC pair R_{ts}/C_{ts}, long-time transient RC pair R_{tl}/C_{tl}, and other possible time constant RC pairs. By assuming the self-discharge resistance will not influence the performance over a times on the order of an hour, and ignoring capacity decreases, we will set $R_{self-discharge}$ infinite and $C_{capacity}$ equal to 1.

In general, all parameters could be extracted by proper pulse-charging experiments from deep discharge to fully charge or vice versa. To get a sense of how to better design the experiments, several different processes described in existing papers have been compared in Table 2-2 [35], [39], [41]. Because this investigation cares more about discharge behavior of batteries, the test steps were designed based on pulse discharge experiments only, and the discharge current amplitude, duty cycle and period were designed to accommodate our experiments. Details of the parameter extraction process are described in Chapter 5.

Table 2- 2 Comparison of different parameter extraction methods of batteries

	[35]	[39]	[41]
Conducted Tests	Pulse discharge tests.	Continuous discharge; pulse discharge	Pulse charge tests; pulse discharge tests
Discharge/Charge Current Level	0.1C, 0.3C, 0.7C,1C	0.5C, 0.75C, 1C	0.33C, 1C
Pulse Width (for pulse tests)	Each pulse discharges 0.1Ah	30 min	First pulse discharges 1Ah, then every pulse discharges 10Ah.
Rest Width	10 min	2 hours	1min
How to define SOC-OCV curve?	Regression based on discharge curve	Regression based on discharge curve	Average of both charge curve and discharge curve
How to get RC parameters?	Calculation based on the average of different discharge curve	Calculation based on different discharge curves	Calculation based on the first pulse

Chapter 3 Design of Experiments

One way of evaluating the effectiveness of a battery is to consider the amount of energy that can be drawn from it, i.e., its energy capacity. As previously discussed, the energy capacity can be enhanced, in specific scenarios, by paralleling an ultracapacitor across the battery. Accordingly, the objective of this thesis is to quantify the improvement in energy capacity, between a battery-ultracapacitor hybrid system versus the battery used on its own.

Ideally, the energy capacity could be measured directly. For example, one may discharge the battery, or the hybrid system, from its fully-charged state to depletion, and record the total output energy. The issue with such an approach is the amount of time required. Each instance of the experiment could take on the order of hours. Time would become a bottleneck, preventing us from conducting experiments at a wide variety of load conditions.

Instead, in this thesis, we propose a proxy metric for energy, named the *discharge capacity*, denoted as $\emptyset$. As discussed in Section 3.1, the *discharge capacity* indicates the effectiveness of an energy storage system, in a similar manner to the energy capacity described above. However, and most importantly, the discharge capacity can be measured in a shorter time, thus allowing for more data to be gathered in a given time period.

To compare the effectiveness of two different energy storage systems, a second metric *normalized energy gain* is defined in Section 3.2, denoted as $\delta\emptyset$. It can be used to describe the difference between the effectiveness of two energy storage systems, or more intuitively, the "gain" in energy capacity of one energy storage system over another.

3.1 Discharge Capacity

The *Discharge Capacity* of an energy storage system, with units of Joule/Coulomb, is defined as the amount of energy delivered to the load per unit charge drawn out of the ESS during the discharge, as shown in (2),

$$\emptyset = \frac{\Delta E}{\Delta Q} = \frac{\int_{t1}^{t2} v(t)i(t)dt}{\int_{t1}^{t2} i(t)dt} \tag{2}$$

where E and Q are respectively, the total energy and total charge delivered to the load, from time t_1 to t_2. The former is computed by integrating the load power, i.e., the

product of load voltage, $v(t)$, and load current, $i(t)$, over time. The latter is computed by integrating the load current over time.

The derivation of *discharge capacity* is justified by the conservation of charge and energy between a battery (or a hybrid system) and its load. Ignoring aging effect, it can be shown that any given battery contains a fixed total amount of charge. Eventually, all of this charge will be transferred from the source to the load, and the energy transfer that results is by definition,

$$E = \int v dq. \tag{3}$$

It implies a phase plane of v and q, and energy is the area enclosed by any $v(q)$ curves and the q axis. For a pulse load application, an example of forming a trajectory on the $v-q$ plane is shown in Fig. 3-1. The load voltage $v(t)$ and the load current $i(t)$ are measured, and the charge is computed as the integral of the current. One can see that on the $v-q$ trajectory, the voltage waveform has separate impulses along on the smooth curve.

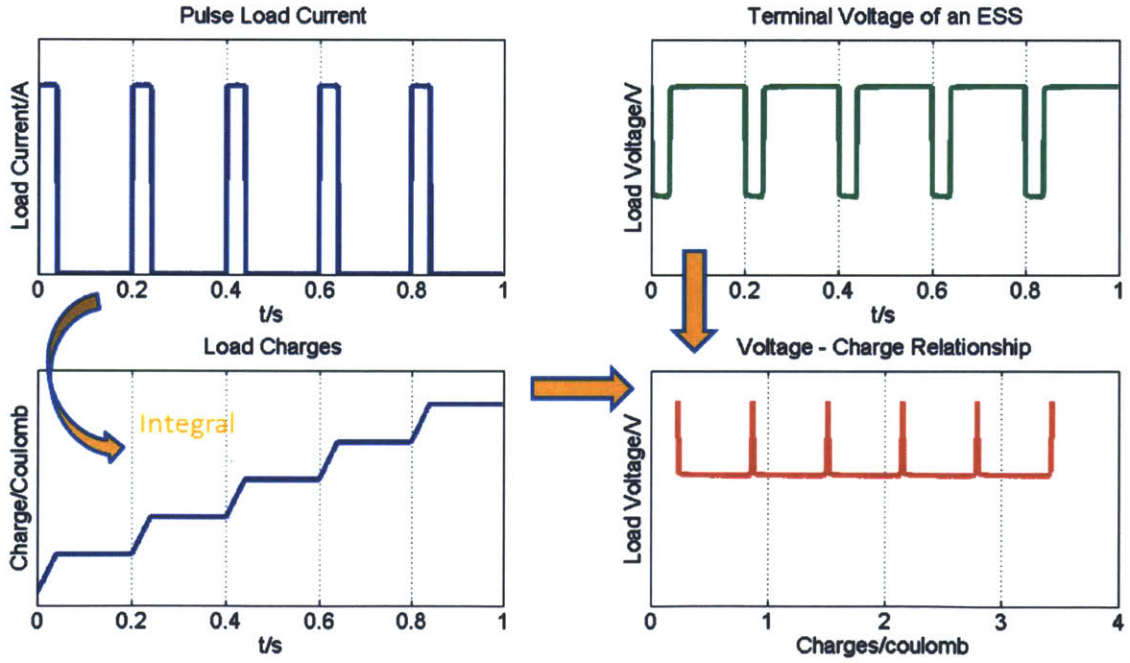


Fig. 3- 1 Trajectory on the $v-q$ plane for a pulse load application

The load charge is equivalent to the SoC of the energy storage system, thus the trajectory of the terminal voltage of the ESS versus its SoC on the $v-q$ plane can be shown as Fig. 3-2.

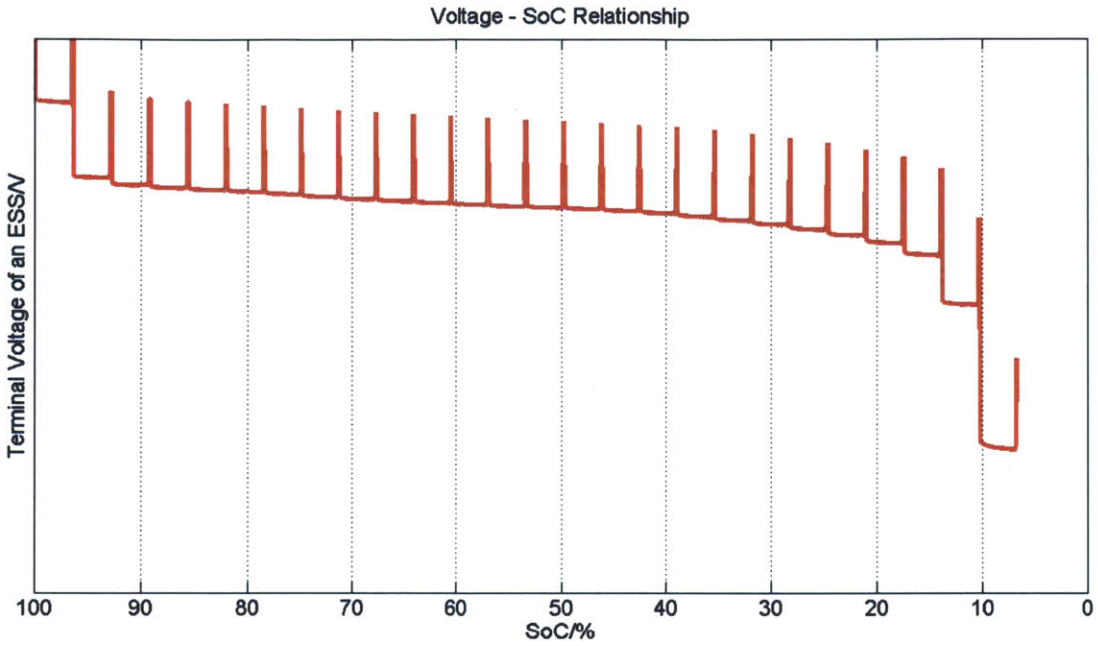


Fig. 3- 2 Trajectory of the termial voltage of an ESS vs SoC of the ESS

The spikes on the curve have zero width and a height equal to the voltage between two current pulses. There is no area in these spikes thus no change in energy. We pick the *discharge capacity* $\emptyset$ such that at these spikes, the derivative of $\emptyset$ over q is finite and at other places, $\emptyset$ is equal to the terminal voltage v. Thus $\emptyset$ is a smooth function of q, and

$$E = \int \emptyset(q)dq = \int vdq \tag{4}$$

Since $\emptyset(q)$ is smooth, using the mean value theorem,

$$\Delta E = \int_{q_1}^{q_2} \emptyset(q)dq \approx \emptyset(q')(q_2 - q_1), \qquad q' \in [q_1, q_2] \tag{5}$$

Thus there comes our definition of discharge capacity,

$$\emptyset(q) = \frac{\Delta E}{\Delta Q}\,|_q \tag{6}$$

An example curve of $\emptyset$ versus SoC is shown in Fig. 3-3. For a certain state-of-charge, the higher $\emptyset$ is, the greater the area is beneath the $\emptyset - q$ curve, meaning more energy can be supplied to the load by the ESS.

The discharge capacity is a metric fully relating energy and charge, and itself could be a function of temperature, pulse profiles, battery type, etc.

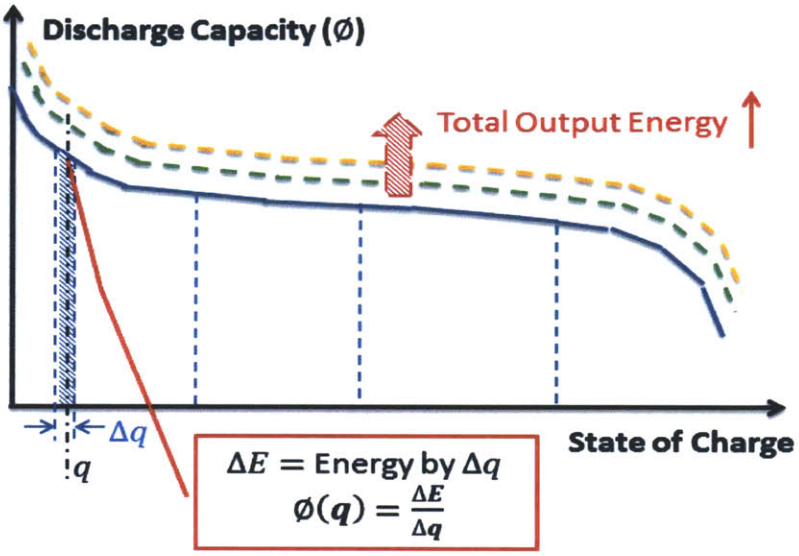


Fig. 3- 3 Discharge Capacity $\emptyset$ – SoC Trajectory

3.2 Energy Gain and Normalized Energy Gain

The *energy gain* is defined as the absolute difference between discharge capacities of a hybrid energy storage system, Φ_H, and its battery-only counterpart, Φ_B, under the same SoC, denoted as $\Delta\emptyset$ and shown in (7). It indicates the absolute value of how much more energy one could draw out from the hybrid ESS than the battery-only ESS.

$$\Delta\emptyset|_q = \emptyset_H|_q - \emptyset_B|_q \tag{7}$$

To have a standard for comparing the *energy gain* under different conditions (such as different pulse loads, battery types, etc.), a percentage number rather than the absolute value is used. The *normalized energy gain* is denoted as $\delta\emptyset$ and defined in (8). It can also be translated to an increase in percentage of the ability to output more energy with the same amount of charge.

$$\delta\emptyset|_q = \frac{\Delta\emptyset}{\emptyset_B}|_q = \frac{\emptyset_H|_q - \emptyset_B|_q}{\emptyset_B|_q} \tag{8}$$

A positive $\delta\emptyset$ indicates an improved performance by incorporating EDLCs in the system. The larger the $\delta\emptyset$ is, the more improvements the hybrid energy storage system has.

3.3 Theoretical Analysis

There are many variables that potentially influence the two metrics defined above. A few examples are the pulse load profile, the battery's state-of-charge, the temperature and the discharge rate.

A theoretical analysis was conducted to shed light on what these possible variables are and their influence on the trend of the metrics. The analysis also gives instructions to the design of the experiments.

The simplest Thevenin models for both batteries and ultracapacitors were used: a voltage source or a capacitor with its series resistance, as shown in Fig. 3-4. The load current i is the periodically pulsed load, with a duty cycle D and period T:

$$i(t) = \begin{cases} I, & 0 \le t \le DT \\ 0, & t \ge DT \end{cases} \tag{9}$$

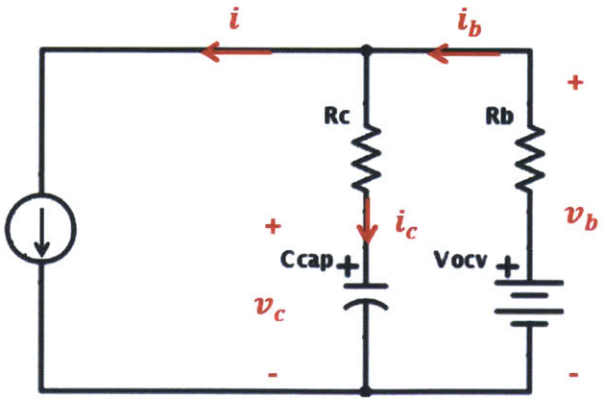


Fig. 3- 4 A simple model of battery-ultracapacitor parallel system

Considering the circuit entering in its periodic steady state, the *discharge capacity* of the hybrid energy storage system and the battery-only system can be calculated as in (10) and (11) respectively. The derivation process is described in detail in Appendix 1.

$$\emptyset_{Battery-Only} = \frac{\int (v_b i)dt}{\int idt} = V_{ocv} - R_b I \tag{10}$$

$$\emptyset_{Hybrid} = \frac{\int (v_b i)dt}{\int idt} = V_{ocv} - R_b \frac{\int_0^{DT} i_b dt}{DT} \approx V_{ocv} - R_b[I - \frac{R_b I}{DT}(\frac{e^{\frac{-(1-D)T}{\tau_{eq}}} - 1}{e^{\frac{-T}{\tau_{eq}}} - 1})(1 - e^{\frac{-DT}{\tau_{eq}}})] \tag{11}$$

Where $\tau_{eq} = C(R_b + R_c)$. And the *normalized energy gain* is

$$\delta\emptyset = \frac{\emptyset_{Hybrid}}{\emptyset_{Battery-Only}} - 1 \approx \frac{{R_b}^2 I}{V_{ocv} - R_b I} \frac{1}{DT}(\frac{e^{\frac{-(1-D)T}{\tau_{eq}}} - 1}{e^{\frac{-T}{\tau_{eq}}} - 1})(1 - e^{\frac{-DT}{\tau_{eq}}}) \tag{12}$$

From (12), the *normalized energy gain* depends on the pulse information (duty cycle D, period T, pulse amplitude I), and battery parameters (ESR R_b, open circuit voltage V_{ocv}) and ultracapacitor parameters (ESR R_c and capacitance C). As previously mentioned, V_{ocv} and R_b are changing with the battery' SoC, temperature, frequency, etc. The parameters of the ultracapacitor are also changing with its SoC, temperature and frequency. The trends of the $\delta\emptyset$ when changing these seven variables are shown in Fig. 3-5.

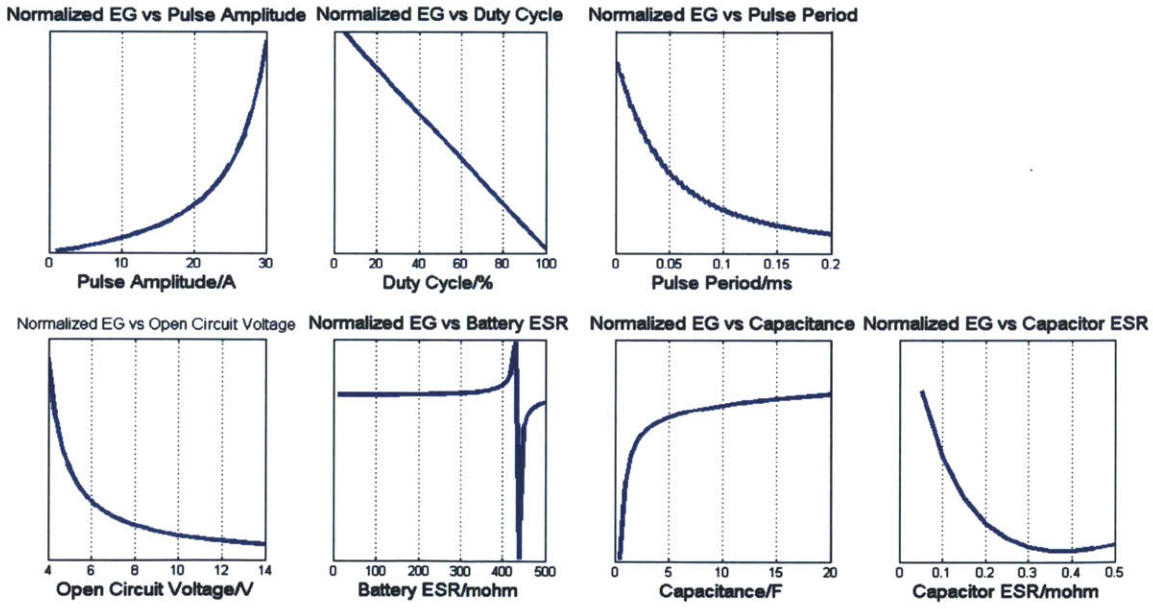


Fig. 3- 5 Possible changes of $\delta\emptyset$ versus different variables

This trend is not very accurate because in reality the changes of open circuit voltage and battery ESR, as well as the changes of capacitance and its ESR could not be separated. For example, typically for ultracapacitors, larger capacitance value corresponds to a smaller capacitor ESR. And the sharp change in the plot "normalized EG vs Battery ESR" is a mathematical artifact.

But still one can find important trends such as: the *normalized energy gain* will increase as pulse amplitude increases, and will decrease as duty cycle and period increases. It will increase as the open circuit voltage decrease and decrease when the capacitance increases.

3.4 Design of Experiments

The theoretical analysis gives some direction as to what to choose as the variables for this investigation.

For pulse information, pulse amplitude, duty cycle and period were chosen. For ultracapacitors, since its capacitance and ESR could not be separated, various values of capacitors were chosen.

For batteries, two types of batteries (Ni-MH and Li-ion) were chosen to provide different ESR values. And to match the open circuit voltages for these two types of batteries, different numbers of cells were put in series.

In addition, because of the impossibility of controlling RC parameters and the open circuit voltage of the batteries, a specific final SoC level was chosen. The assumption of this choice is that the open circuit voltages and the battery RC parameters will stay relatively unchanged at this SoC.

Therefore, there are six degrees of freedom in the experiments: pulse amplitude, pulse frequency, pulse duty cycle, capacitance, battery type and SoC. Hardware experiments as well as simulations based on a more accurate model were designed to reveal the exact trend of $\emptyset$ and $\delta\emptyset$.

3.4.1. Devices for the Experiments

The devices for the experiments are one type of Nickel-metal Hydride battery (Ni-MH), one type of Lithium Ferro Phosphate battery (LiFePO4) and three values of ultracapacitors. The Ni-MH battery pack consists of five 1.2 V Panasonic HHR370AH cells in series connection, with a total voltage of 7 V. The Lithium-ion battery pack consists of two 3.3 V A123 ANR26650 cells in series connection, to get a comparable total voltage of 6.6 V. The ultracapacitor banks are three 2.7 V Maxwell ultracapacitors in series connection, with a total rated voltage of 8.1 V. Three different values of ultracapacitors were used to construct three capacitor banks with capacitances of 1.67 F, 8.33 F and 16.67F. The pictures of the

batteries and the ultracapacitors are shown in Fig. 3-6 and their specifications from datasheets are listed in Table 3 - 1.

Fig. 3- 6 Devices used in the investigation
Ni-MH battery pack, Lithium-ion battery pack and
ultracapacitor cell (left to right: 5F, 25F, 50F)

Table 3 - 1 Specifications of devices used in the investigation

	Ni-MH [42]	LiFePO4 [43]	Ultracapacitors [44]		
Norminal Voltage/cell	1.2 V	3.3 V	2.7 V	2.7 V	2.7 V
Capacity/cell	3.5 Ah	2.3 Ah	-		
Capacitance/cell	-		5 F	25 F	50 F
Estimated ESR/cell	20 mΩ	10 mΩ	170 mΩ	42 mΩ	20 mΩ
Diameter/cell	18.2 mm	26 mm	10 mm	16 mm	18 mm
Length/cell	67 mm	66.5 mm	20 mm	26 mm	40 mm
Mass/cell	60 g	70 g	2.3 g	7.5 g	13 g
Manufacturer	Panasonic	A123	Maxwell		
Part number	HHR370AH	ANR26650-M1	BCAP0005	BCAP0025	BCAP0050
Cells in Series	5	2	3	3	3

3.4.2. Design of Pulse Loads

As mentioned in section 1.1, this investigation is interested in pulse amplitudes from 3 A to 20 A, and pulse periods from 50 ms to 400 ms and duty cycles mainly from 5% to 50%.

For the hardware experiments, discrete numbers in these ranges and different combinations of these numbers were selected. As discussed in Section 3.1, the SoC level to which the two ESSs will discharge needs to be specified. Considering the time limitation of

experiments, the SoC level of 99% was chosen. Depending on the SoC level, pulse numbers for different pulse load profiles were decided.

The specified pulse load profiles are listed in Table 3-2. Pulse numbers are calculated based on the 99% SoC level. However to simply the hardware experiments, each of the pulse load profile takes 2000 or 4000 pulses. And in the latter data processing part, a desired number of pulses were separated and the two metrics were calculated based on these pulses. For example, for a pulse load with the pulse amplitude of 8 A, duty cycle of 20% and a pulse period of 100 ms (as in No. 8 in the Table 3-2), the desired pulse number is 500. We first took 2000 pulses data, then calculate $\emptyset$ and $\delta\emptyset$ based on the first 500 pulses.

Because the *discharge capacity ($\emptyset$)* is the derivative of energy with respect to charge at a given charge level, $\emptyset$ and $\delta\emptyset$ were both calculated based on the last 10 pulses of the total desired pulses. An example is shown in Fig. 3-7.

Table 3 - 2 Pulse load profiles in the hardware experiments

No.	Amplitude /A	Period /ms	Duty Cycle /%	Pulse number	Capacitance /F
1	4	100	10%	2000	1.667/8.333/16.667
2	4	200	10%	1000	
3	8	50	5%	4000	1.667/8.333/16.667
4	8	50	10%	2000	
5	8	50	20%	1000	
6	8	100	5%	2000	
7	8	100	10%	1000	
8	8	100	20%	500	
9	8	200	2.5%	2000	
10	8	200	5%	1000	
11	8	200	10%	500	
12	8	200	20%	250	
13	8	200	50%	100	
14	8	200	75%	67	
15	8	400	10%	250	
16	12	100	10%	667	1.667/8.333/16.667
17	12	200	10%	333	
18	16	50	5%	2000	1.667/8.333/16.667
19	16	50	10%	1000	
20	16	50	20%	500	
21	16	100	5%	1000	
22	16	100	10%	500	

23	16	100	20%	250	
24	16	200	2.5%	1000	
25	16	200	5%	500	
26	16	200	10%	250	
27	16	200	20%	125	
28	16	200	50%	50	
29	16	200	75%	34	
30	16	400	10%	125	

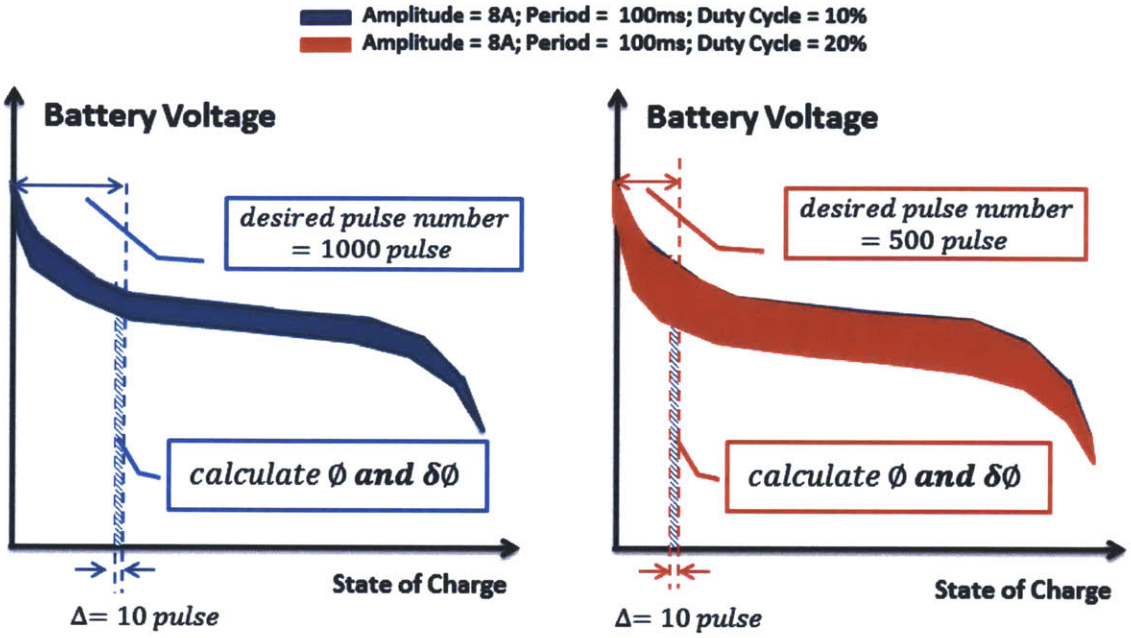


Fig. 3- 7 Examples of using the last 10 pulses to calculate $\emptyset$ and $\delta\emptyset$

Chapter 4 Experimental Apparatus

The experimental apparatus was designed and improved based on the system in [12]. To freely control different degrees of freedom, a discharging circuit and a measurement circuit were designed; to guarantee each discharging experiment starts at the same fully charge point, and give us freedom to charge different types and combinations of batteries, a charging circuit was also designed and implemented.

The core of the test rig is a microcontroller, which controls the charging and discharging circuit as well as takes measurements. There are several requirements that the rig must meet:

- To be able to source/sink current in a range of 0 to 20 A, and provide good accuracy;
- To precisely sample data as well as output pulses with various duty cycles and periods. This is critical for conducting any sort of experiments sweeping along pulse amplitude, frequency and duty cycle.
- Experiments on batteries often run for hours, thus the apparatus needs to record a large amount of data, which requires either separate SRAMs on the printed circuit board or communication with a personal computer.

Several improvements need to be considered for the new apparatus compared to the previous test rig in [12]:

- The microcontroller based measurement system needs to be improved in terms of clock accuracy, sampling frequency, analog-to-digital converter (ADC) accuracy as well as the speed of data transfer. The previous test rig was designed with an 8 MHz internal clock, a 10-bit internal ADC and a Universal Asynchronous Receive/ Transmit (UART) rate of 115200 BAUD, which were all proved to be not sufficient.

- The flow map of the program needs to be redesigned to guarantee a more accurate timing and achieve a more efficient data sampling process. In the previous work, the period and width of the pulsed load was substantially differenent from what we expected and programed in the microcontroller.

- The floating ground problem needs to be solved. Since there was no isolation between the digital communication circuits and the high-current-side circuit, there was an analog ground shift caused by closing the loop from the high side power supply ground, through the low voltage analog and digital circuitry, to the data-collecting computer

through the USB port. This floating ground increases the noise on the ADC data collection, and shifts the signals up by over 10 mV.

The discharge part of experiments was also conducted later on a borrowed Agilent Advanced Power System (APS) N7951A to verify the accuracy of the experimental results.

Fig. 4-1 shows the block diagram of the new test rig. The Atmel microcontroller ATxmega192A3U was chosen and put in the center of the diagram. Its internal 12-bit Digital-to-Analog Converter (DAC) was used to generate proper analog signals to control the current level of charging or discharging. The microcontroller also controls a separate 8-channel 16-bit ADC (Analog Device Inc., AD7606) to conduct simultaneous sampling. Furthermore, it communicates and transfers data to the personal computer through a UART-to-USB chip FTR232R.

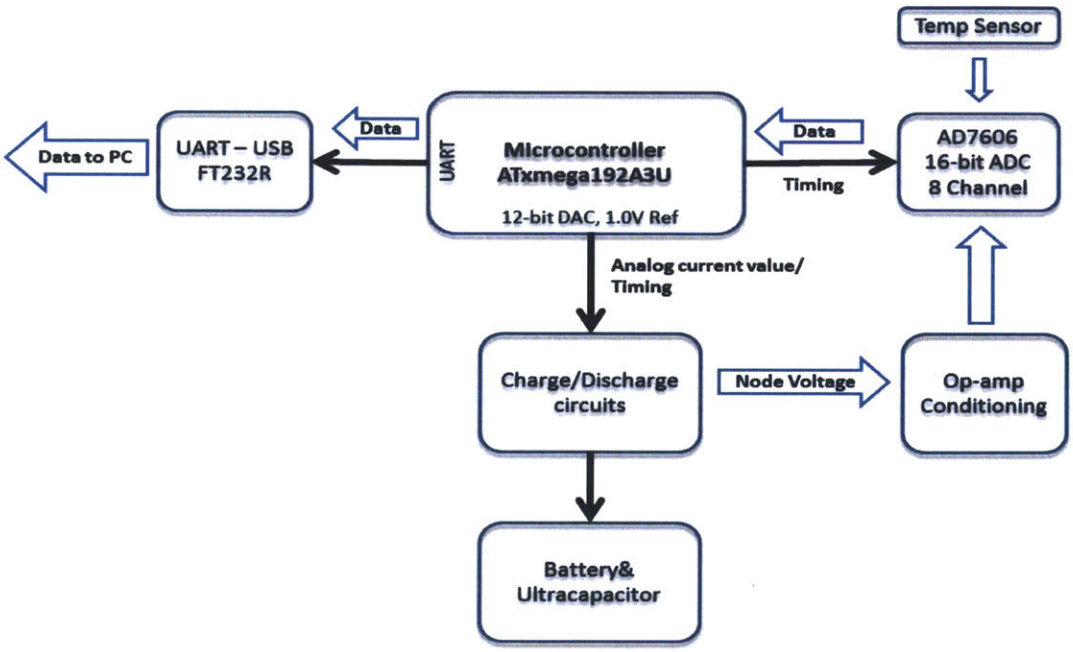


Fig. 4 - 1 Block diagram of the experimental apparatus

4.1 Design and Improvement of Charging and Discharging Circuit

General Circuit:

The charging circuit and the discharging circuit were both designed based on the same theory to function as constant current sources, as shown in Fig. 4-2 [12], [45]. The operational amplifier (Op Amp) U_c serves as a controller, compares the desired current value with the measured current value to set the gate signal to the power switch Q_s, which

operates in its saturation region and can be considered as a voltage-controlled current source. U_{IA} is an instrumentation amplifier, serving as both a signal conditioner and a filter of ground floating noise.

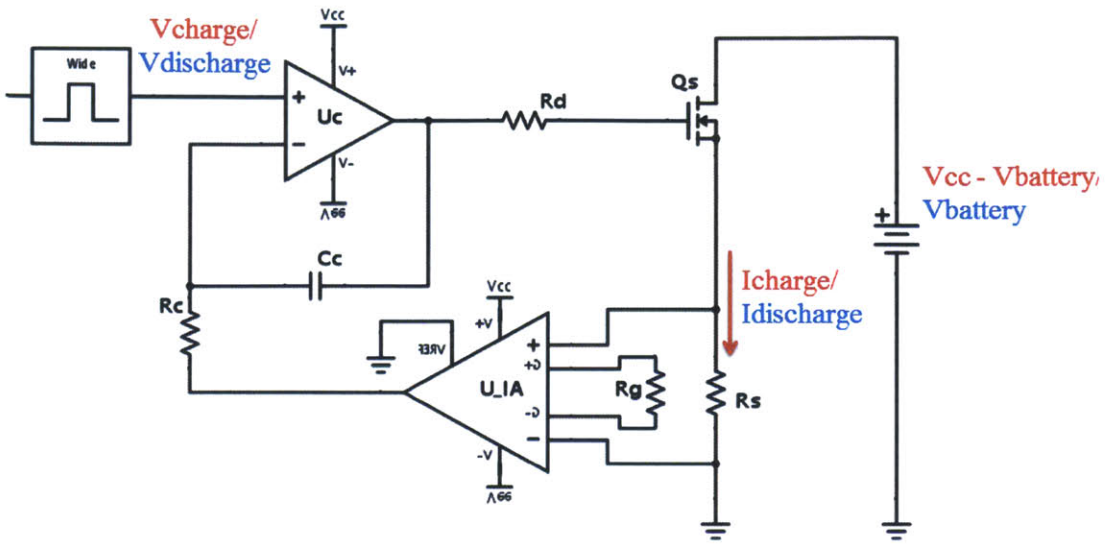


Fig. 4 - 2 The constant current source used in the charging/discharge circuit

The difference between the charging and discharging circuits is the static operating point voltage between source and drain of the power switch: for the former one it is Vcc minus the battery's terminal voltage and for the latter one it is the battery's terminal voltage.

Comprehensively considering the pulse amplitude we would like to achieve, and the upper limit of the reference current signal (set by the DAC of the microcontroller to 1.0 V), the shunt resistor was chosen as 10 mΩ and the gain of the instrumentation amplifier U2 was set to 4.984. This number was determined by the gain equation of the instrumentation amplifier when choosing a precise gain resistor of 12.4 kΩ, as shown in (13).

$$\text{GAIN} = \frac{49.4\ k\Omega}{R_G} + 1 \tag{13}$$

Then the relationship between input analog signal V_{charge} or $V_{discharge}$ and the output current I_{charge} or $I_{discharge}$ is

$$I_{dis/charge} = \frac{V_{dis/charge}}{0.01 * 4.984} = 19.985 V_{dis/charge} \quad (14)$$

By staging two switches into a half bridge, the upper MOSFET functions as a charging circuit and the lower MOSFET functions as a discharge current source, as shown in Fig. 4-3. All devices with index 1 are for the discharging circuit and those with index 2 are for the charging circuit. The only exceptions are that the shunt resistor R1 is for the discharge circuit, R2 is for the current measurement in the capacitor branch, and R3 is for the charge circuit.

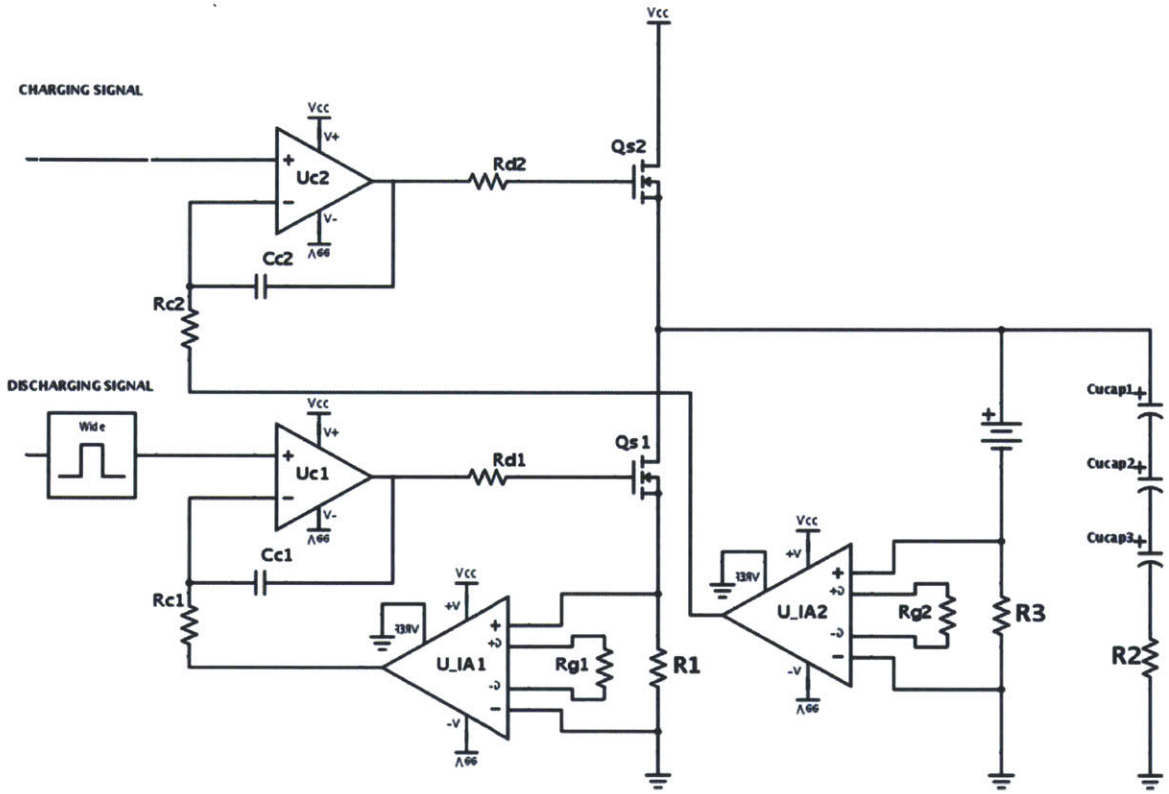


Fig. 4 - 3 Circuit diagram of the charging and discharging circuit

Both U_c and U_{IA} are selected as dual-supply devices, so that they can either control and measure currents during both the charging and the discharging processes. In addition, the input range and the output range of U_c and U_{IA} can reach exactly zero without a close-to-rail problem of many single supply amplifiers, which improves the accuracy of controlling and measuring small current signals.

The disadvantage of the dual-supply devices is that during discharging, the current in the R3 branch (i.e., battery branch) is opposite to the charging direction, thus the output of

U_{IA2} will be negative and accordingly the inverting input of U_{c2} is also driven negative. However the non-inverting input of U_{c2} is constrained by the microcontroller to have a lowest value of 0 V (neglecting any offset), thus the gate signal of Q_{s2} will be positive and Q_{s2} will be on instead of off as it should be. An external circuit was added to manually short the output of U_{c2} to ground during discharging to guarantee that Q_{s2} turns off.

Compensation:

The power MOSFET has capacitance C_{gs} between gate and source, which can be seen as a capacitive load to the Op Amp U_c. When driving capacitive loads, an Op Amp tends to have peaking and oscillation problems and may cause reduced bandwidth and additional phase shift [46].

By analyzing the mathematical model and the open-loop transfer function of the constant current source circuit, the frequency response is plotted in Fig. 4-4. It can be seen that without any compensation, the phase margin of the open-loop transfer function is only 18°. A feedback compensation between U_c's output and its inverting input needs to be designed to add more phase margin at the crossing frequency [47], [48].

Three different feedback compensations have been simulated: resistive feedback, capacitive feedback and integrator feedback. It shows that the integrator feedback can increase the phase margin to 67° without a large influence on the dc performance. Thus the compensation resistor-capacitor pairs R_{c1}/C_{c1} and R_{c2}/C_{c2} in Fig. 4-3 were added.

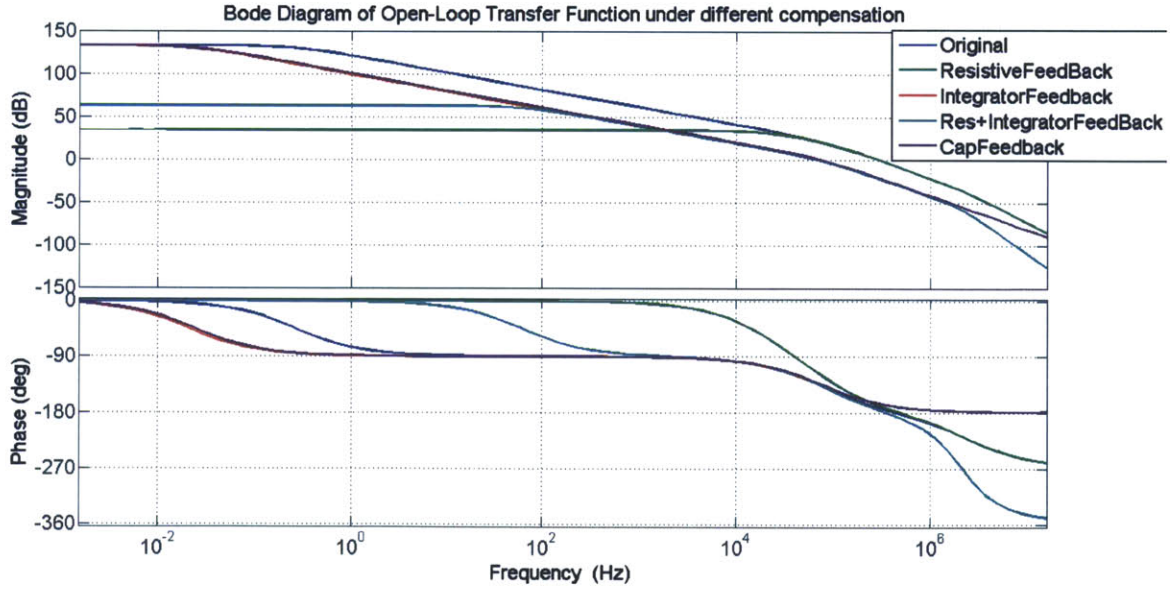


Fig. 4 - 4 Frequency responses of the constant current source with different compensation methods

Offset adjustment:

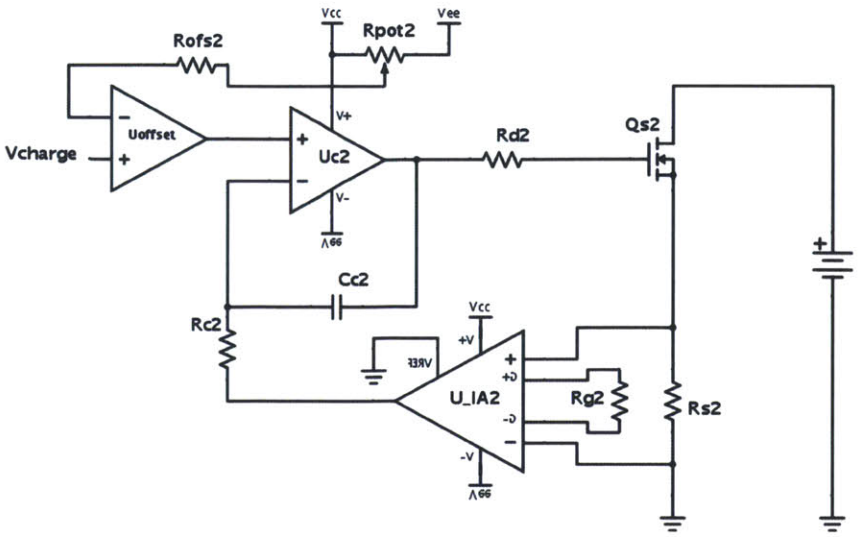
The charging circuit needs to output current from 50 mA to at least 3 A for reasons discussed in section 4.1.3, however the output Op Amp of the built-in DAC of the microcontroller is not able to reach down to its rail [49]. The lowest voltage it can output is 16 mV, which according to (14) still gives the battery branch a 0.3 A charging current. Thus a subtraction circuit was added to the non-inverting input of U_{c2} to guarantee the charging current can go below 50 mA, as shown in Fig. 4-5(a).

The discharging circuit needs to output currents up to 20 A during pulses and also shut down completely in between the pulses. However, the input offset voltages of the Op Amp and the instrumentation amplifier, even though only typically 80 μV [50], prevent the MOSFET from completely turning off when we program the analog signal $V_{discharge}$ to zero. An offset of minus 2 mV was added through a potentiometer to the inverting input of U_{c1} to guarantee the circuit can output zero current, as shown in Fig. 4-5(b).

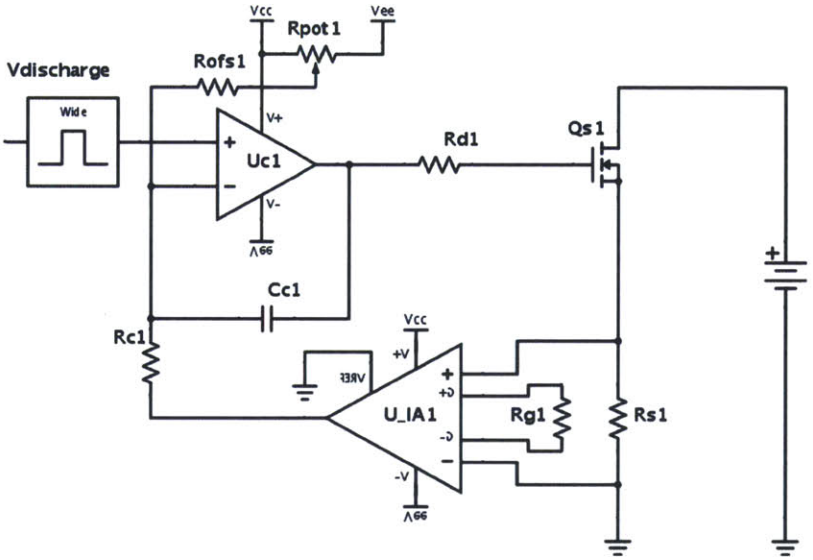
The specifications of devices used in the circuit Fig. 4-3 and Fig. 4-5 are listed in Table 4-1. Because of the symmetry, most devices are the same in the charging and discharging circuits.

Table 4- 1 Device value and part number

Device	Function	Part number/value
Ucontrol	Low-offset precise operational amplifier	AD8671
U_InstruAmp	Instrumentation amplifier	AD8221
Qswitch	Power MOSFET	ST43MN60N
Rshunt	Precise shunt resistor (FPR 2 – T218)	10 mΩ ± 0.5%
Rdamp	Damping resistor	10 Ω
Rcompensation	Compensation resistor	10 kΩ
Ccompensation	Compensation capacitor	270 pF
Roffset1	Offset resistor for discharging circuit	10 MΩ
Roffset2	Offset resistor for charging circuit	100 Ω
Rpot1/Rpot2	Potentiometers for offset adjustment	2 kΩ
Uoffset	Subtraction operational amplifier	AD8672



(a) Offset adjustment for charging circuit



(b) Offset adjustment for discharging Circuit

Fig. 4 - 5 Offset adjustment using potentiometer and Op-amp

4.2 Design of the Measurement and Control Circuit

To improve the accuracy and guarantee simultaneous sampling of data, a separate 16-bit 8 channel analog-to-digital converter (ADC) AD7606 is used. It can sample eight inputs simultaneously at a maximum rate of 4 μs, and outputs 16-bit digital data, which we use parallel data bus to transfer to the microcontroller.

Since the use of the 16-bit parallel data bus, ground design becomes more important on the printed circuit board. A four-layer board was designed, including one layer for ground and one layer for supplies. On the ground plane, analog and digital ground was connected underneath the ADC, and the analog plane was connected through a narrow copper path with the power ground. The ground floating problem mentioned in the beginning of this section was solved by implementing instrumentation amplifiers to condition the signals. In addition, there were only two DAC digital signals traveling across the digital ground to the analog ground, which minimized the influence between the two ground planes. The CAD layout of the print circuit board can be seen in Fig. 4-6.

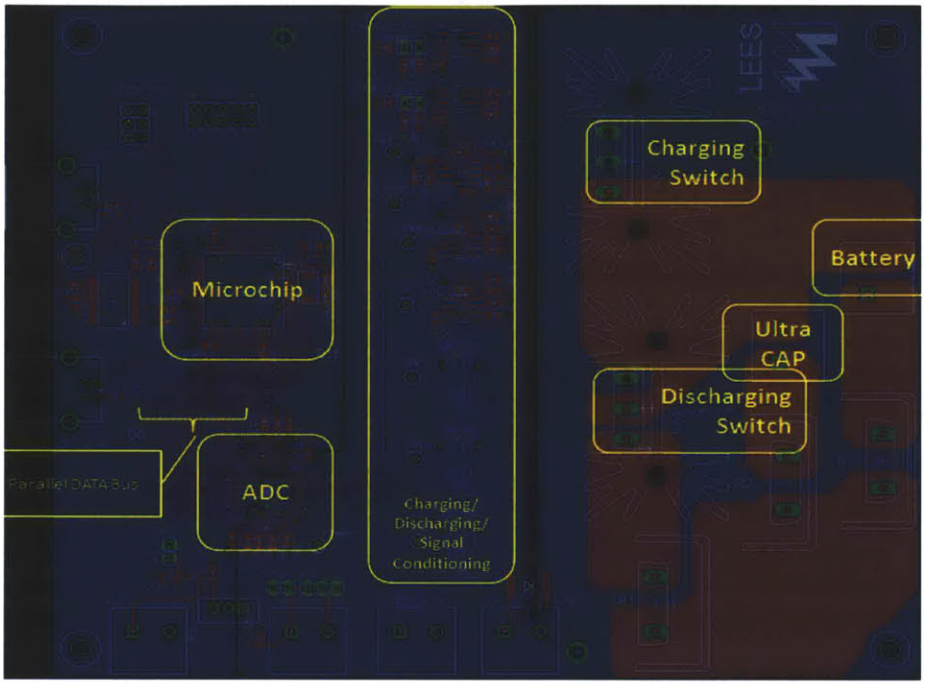


Fig. 4 - 6 CAD layout of the printed circuit board

To both improve the accuracy of the timer as well as speed up the data sampling and transferring in the whole system, an external crystal of 14.7654 MHz and more interrupts were used to guarantee precise timing. The timer of the microcontroller was set to 1ms, during which data from all 8 channels was sampled, sent to the PC through the UART and the next status determined. The flow map of the program is shown in Fig. 4-7, and an example of data sampling is shown in Fig. 4-8.

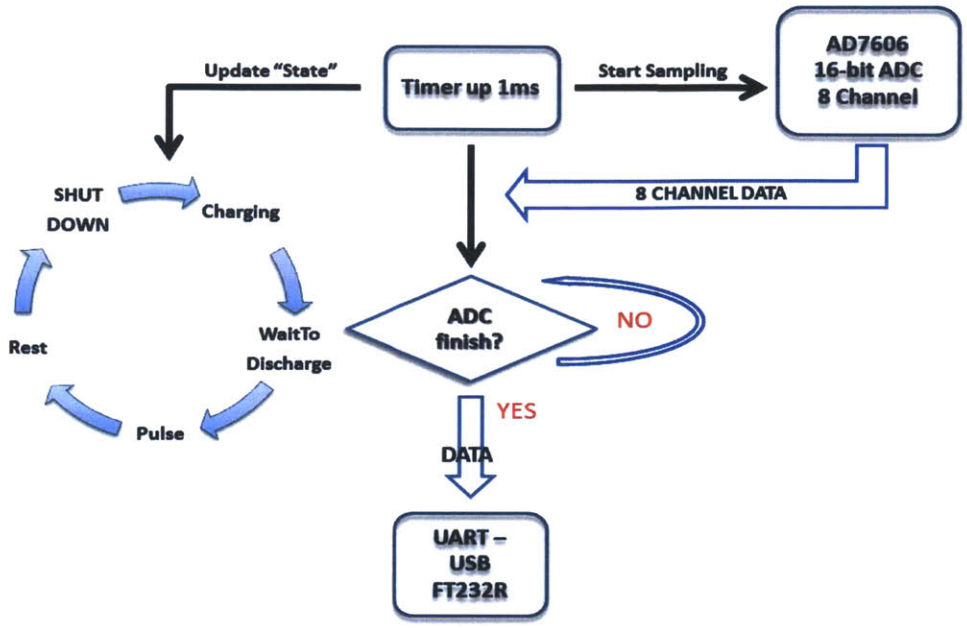


Fig. 4 - 7 Flow map of the program

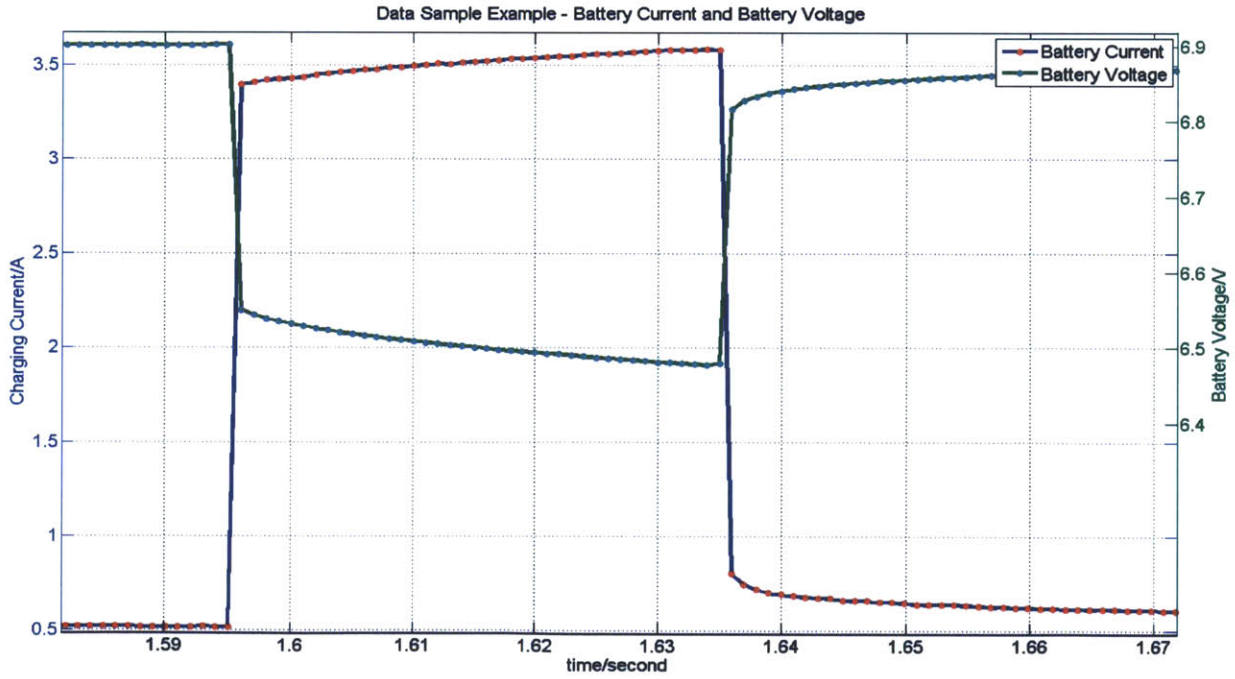


Fig. 4 - 8 Data sampling example – battery current and battery voltage in a hybrid ESS experiment (discharge at a load profile of 8A, pulse period 400ms, duty cycle 10%)

To increase the speed and efficiency of data communication between the UART and the PC, the baud rate was increased from 115200 BAUD to 460800 BAUD. And instead of being coded in ASCII, the data was coded into hex bits with no separation marks to reduce elements which need to be sent. An example is shown in Fig. 4-9. The UART interface was used to receive data on the PC side, the screenshot of which is shown in Fig. 4-10.

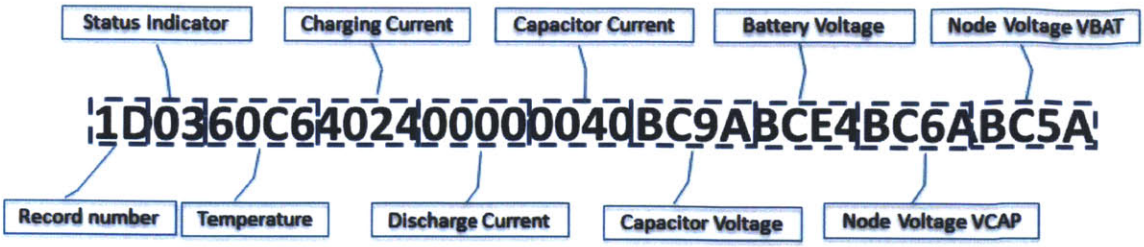


Fig. 4 - 9 HEX coding of the UART data transfer

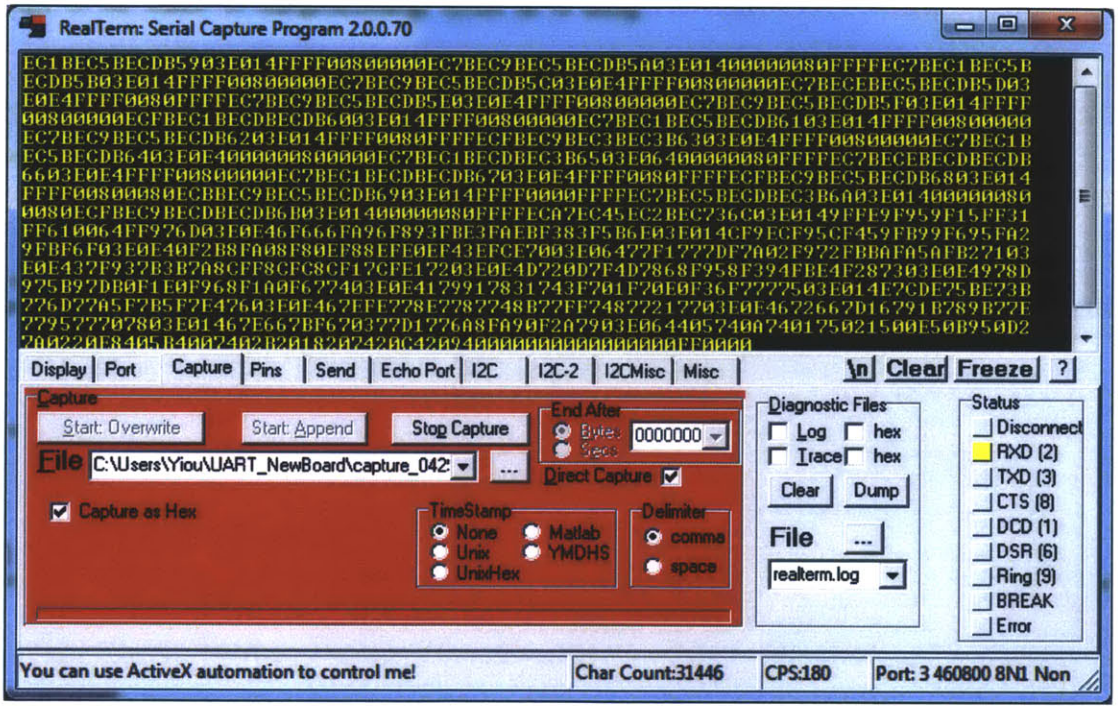


Fig. 4 - 10 UART receiver interface

4.3 Battery Charging Strategy

To shorten the total experimental time, rapid charging methods for both Ni-MH and Lithium-ion batteries were studied and used. The charging specifications of both batteries are listed in Table 4-2. Ni-MH batteries were charged at constant current, and terminated when either the temperature derivative or the voltage derivative reached its limit; Li-ion

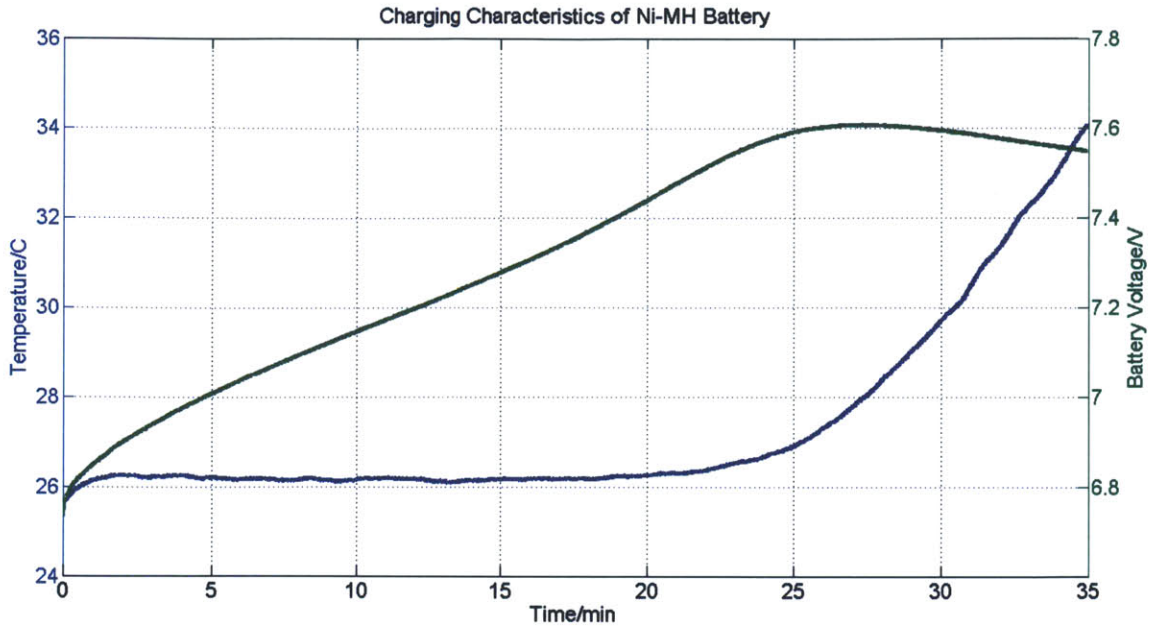
batteries were charged at constant current first, then at constant voltage until either the current limitation or the time limitation was reached [51], [52], [53]. Different termination strategies were used for the batteries to guarantee the proper charge termination and eliminate the danger of over charge.

Table 4- 2 Charging Specifications of Batteries

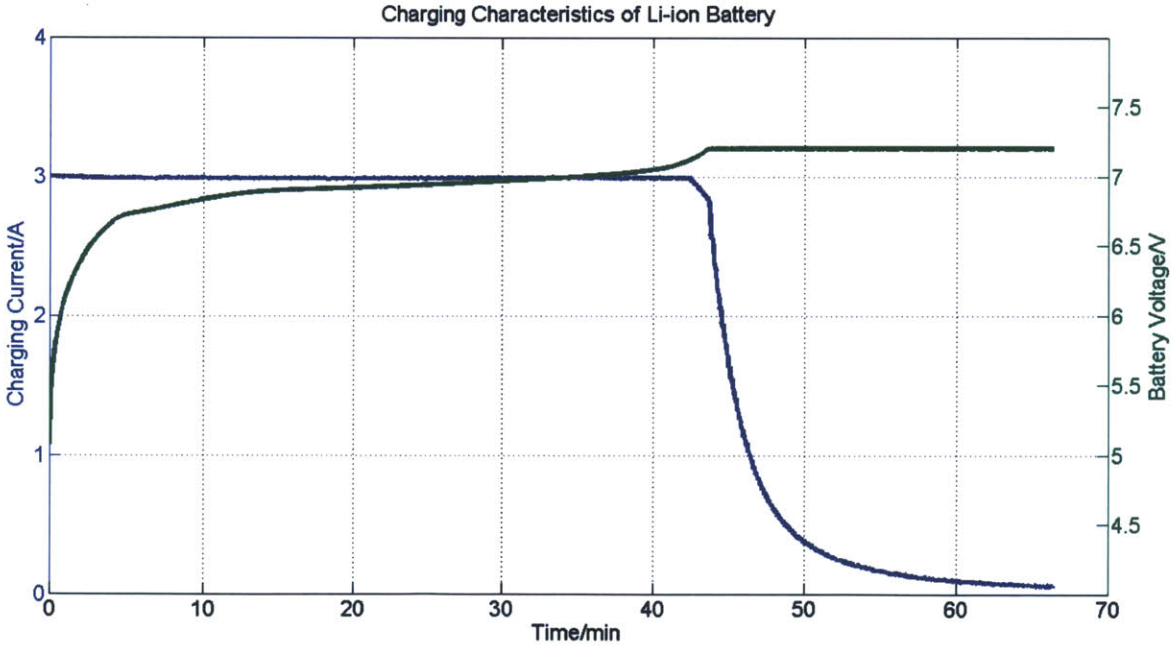
	Ni-MH battery One Cell [42]	**Li-ion battery One Cell [43]**
Recommended charging method	Constant current	Constant current then constant voltage
Constant Current	1.6 A (0.43C)	3 A (1.304C)
Constant Voltage	------	3.6 V
Termination rules	Temperature rises 1°C per minute.	Current goes down to 50mA
	Voltage drops 5 – 10 mV per cell.	Constant voltage charging goes longer than 30 minutes.

The charging characteristic of Ni-MH batteries obtained from experiments is shown in Fig. 4-11 (a) and that of Li-ion batteries is shown in Fig. 4-11 (b).

To guarantee each charging process reaches the same fully charge point and to verify the reliability of the system, multiple charging curves have been compared as shown in Fig 4-12 (a) and (b).

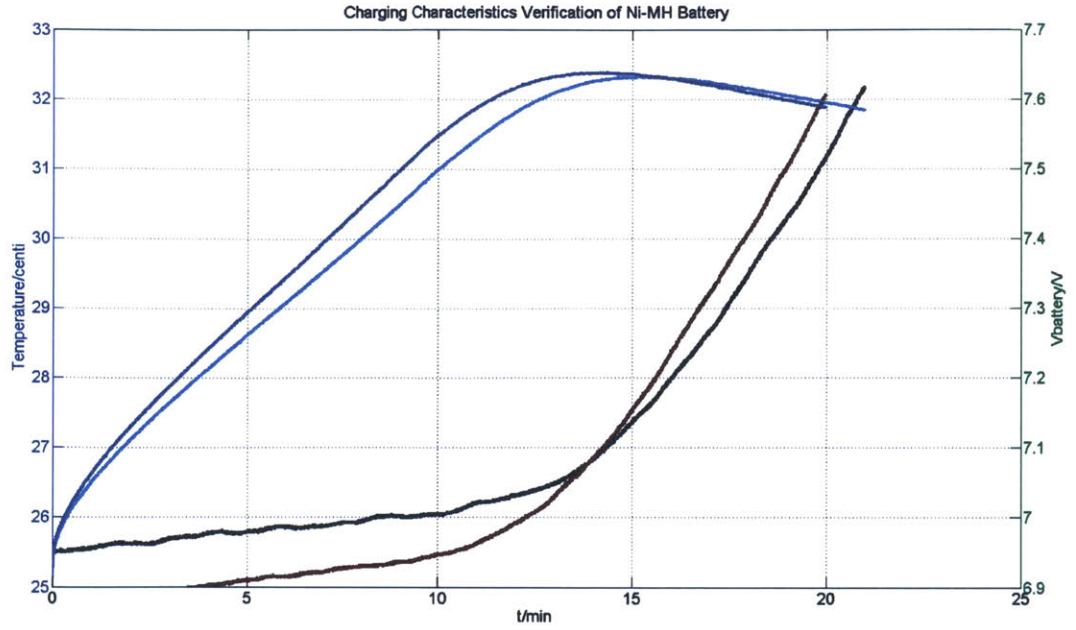


(a) Charging characteristics of five-cell Ni-MH batteries

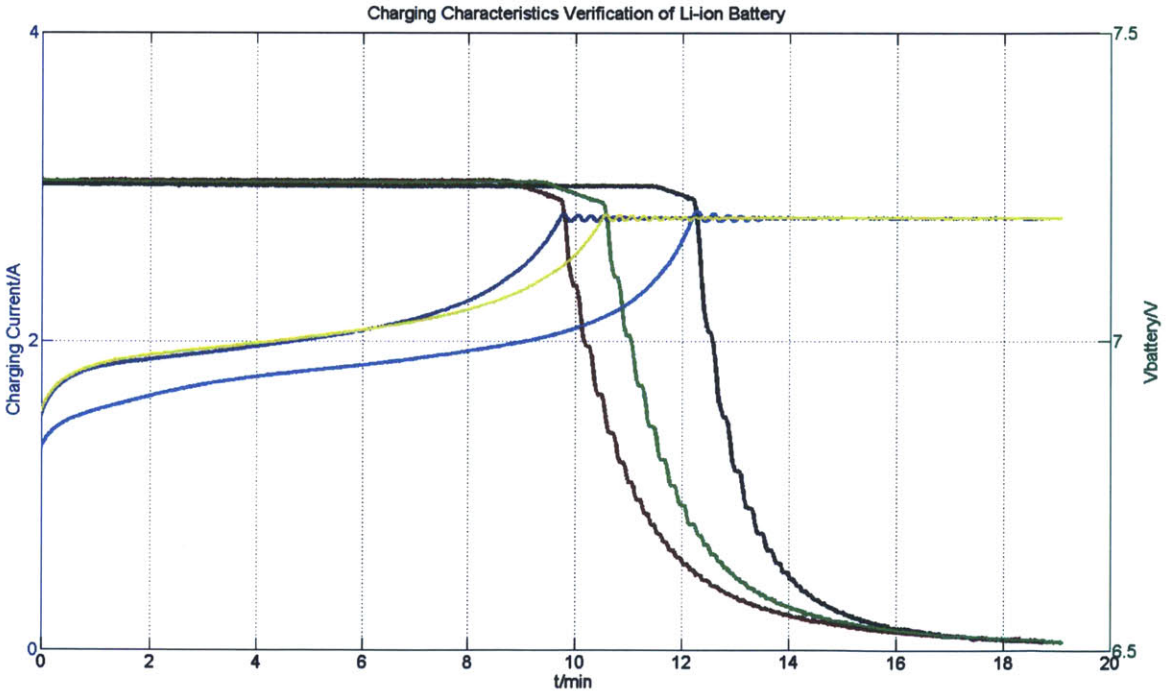


(b) Charging characteristics of two-cell Li-ion batteries

Fig. 4 - 11 Charging characteristics of Ni-MH and Li-ion batteries



(a) Ni-MH Battery two different charging tests



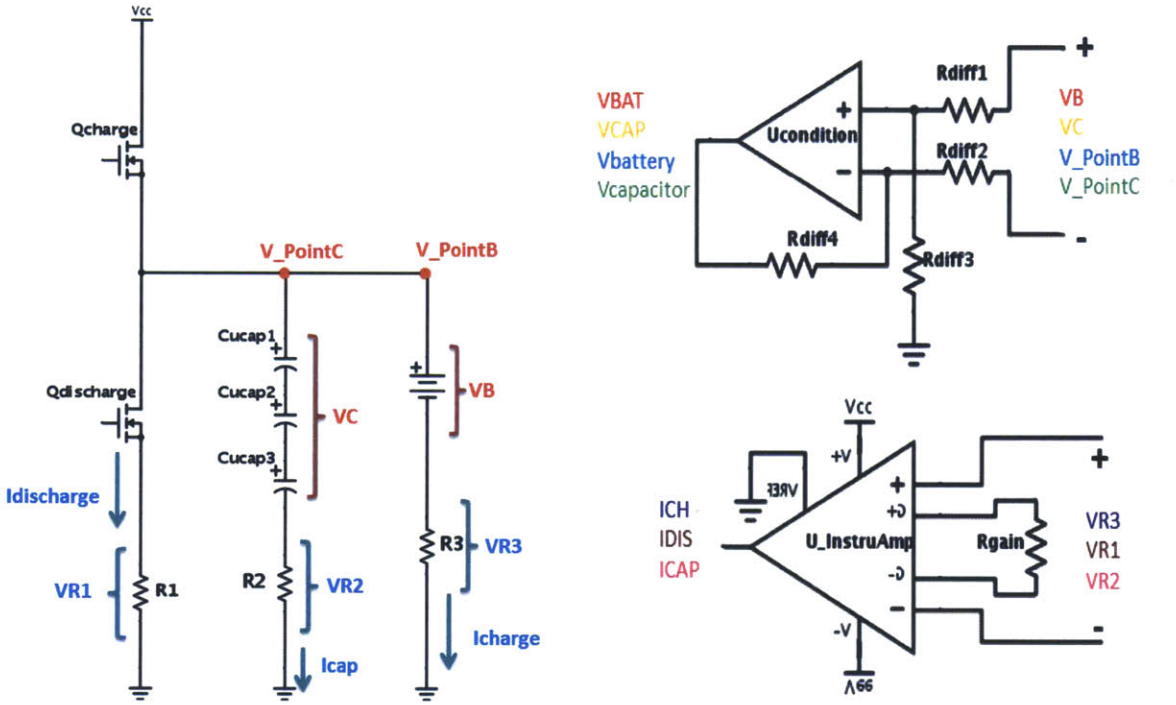
(b) Li-ion Battery three different charging tests

Fig. 4 - 12 Verification of different charging test

4.4 Data processing and Measurement Points

The 8-channel ADC was used to sample data at 8 measurement points. Except for the temperature data coming from the temperature sensor LM35, the seven other measurement points and their conditioning circuits are shown in Fig. 4-13 (a) and (b) respectively.

There are two types of conditioning circuits: one is using differential amplifiers to get a fraction of 1/3 and condition VB, VC, V_PointB and V_PointC; another type is using instrumentation amplifiers to get a gain of 4.984 and condition VR1, VR2 and VR3. The conditioned signals going into different channels of the ADC and their corresponding channel numbers are listed in Table 4-3.



(a) Measurement Points; **(b) Conditioning Circuits**

Fig. 4 - 13 Measurement points and conditioning circuits

Table 4- 3 Measurement Points of 8 Channels into ADC

Channel	Input Signal Name	Direct Measurement	Indirect Measurements
1	Temp	Voltage out of temp sensor	Battery temperature
2	ICH	Voltage across R_{shunt3}, VR3	Current through the battery branch
3	IDIS	Voltage across R_{shunt1}, VR1	Current through the discharge switch branch
4	ICAP	Voltage across R_{shunt2}, VR2	Current through the capacitor branch

5	Vcapacitor	Voltage across capacitors, VC
6	Vbattery	Voltage across batteries, VB
7	VCAP	Voltage at the point above capacitors, V_PointC
8	VBAT	Voltage at the point above batteries, V_PointB

Due to the layout of the parallel data bus from the ADC to the microcontroller, the 16-bit digital data cannot be directly used to represent an actual value, and an interpretation function was used to process the data. Refer to the "Temperature (0x60C6)" in Fig 4-9 as an example:

$$0x60C6 = 0b0110\ 0000\ 1100\ 0110$$

Digit 15th to 8th turns over to digit 8th to 15th; Digits 7th to 0th turns over to 0th to 7th.

$$0b0000\ 0110\ 0110\ 0011 = 0x0663 = 1635$$

$$\text{TEMP} = \frac{1635}{32768} * 5 = 0.2495\ \text{V}$$

$$\text{Temperature} = \text{TEMP} * 100 = 24.95°\text{C}$$

Thus, the data string in Fig. 4-9 (**1D0360C6402400000040BC9ABCE4BC6ABC5A**) can be fully interpreted as below:

Temperature	Charging Current	Discharge Current	Capacitor Current	Capacitor Voltage	Battery Voltage	Nodal VCAP	Nodal VBAT
24.95°C	1.677 A	0 A	0.006 A	7.1892 V	7.1663 V	7.1878 V	7.1896 V

To calculate the *Discharge Capacity* in (2) from the measured points, equation (15) and (16) are used:

$$\Delta E = \int (V_{\_PointC} I_{discharge} + V_{\_PointC} I_{cap} + {I_{charge}}^2 R_3) dt \quad (15)$$

$$\Delta q = \int I_{system} dt = \int I_{discharge} dt \quad (16)$$

Where

$$I_{discharge} = \frac{V_{R_1}}{R_1}, I_{cap} = \frac{V_{R_2}}{R_2}, I_{charge} = \frac{V_{R_3}}{R_3} \quad (17)$$

All variables used in the above equations are referred to Fig. 2-13(a). In addition, not all eight measurement points are independent because redundancy was designed to verify the accuracy of our measurement:

- Charge conservation: the integration of charging current and capacitor current needs to equal the discharging current, as shown in (18);

- Energy conservation: the energy from battery equals the sum of energy delivered to the load and the losses. The latter part should be far less than the former part. Thus the energy delivered to the load needs to be approximately equal to the total energy from the batteries, as shown in (19).

$$\int I_{charge}dt = \int I_{discharge}dt + \int I_{cap}dt \tag{18}$$

$$\Delta E \approx \Delta E_{Battery} = \int V_B I_{charge}dt \tag{19}$$

The final experimental apparatus is shown in Fig. 4-14.

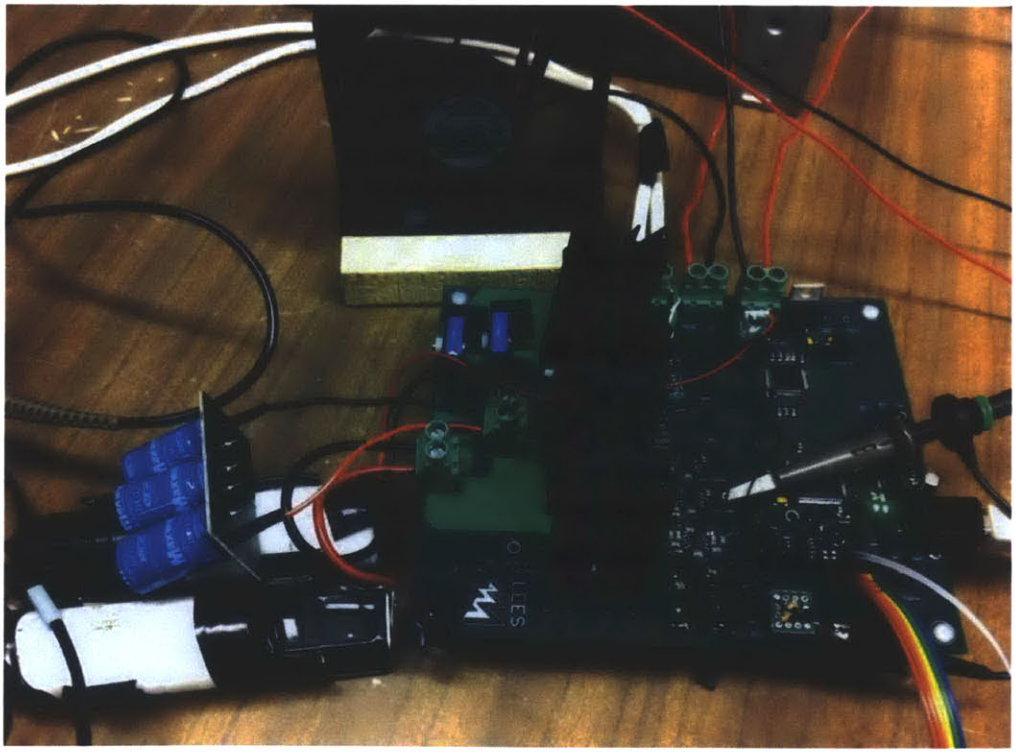


Fig. 4 - 14 Experimental apparatus

4.5 Sources of Errors

4.5.1. Errors in $\emptyset$

Major sources of errors for the hardware experiment are considered as follow:

- ϵ_{shunt}, errors of shunt resistors. The tolerance of the shunt resistors in our experiments are 0.5% [54]. Since we are using three shunt resistors, the total error would be 1.5%.

- ϵ_{ADC}, errors of analog-to-digital converter. According to AD7606 data sheet [55], the LSB of the ADC is 152 μV. Using (14), we calculate the errors brought by ADC as

$$\epsilon_{ADC} \approx \frac{0.152}{I \times 10 \times 4.984} \tag{20}$$

- ϵ_{offset}, the input offset voltage of the instrumentation amplifier is typically 80 μV [50]. It also brought the errors as in

$$\epsilon_{offset} \approx \frac{0.08}{I \times 10} \tag{21}$$

- ϵ_{digit}, small number of data points for low duty cycle experiments. The sample rate of the measurement system was set to 1 ms. For test cases of smaller duty cycle, there will be fewer data points during the pulse width, which results in inaccuracy of the data. For example, for an 8A-200ms-10% experiment, there were 20 data points during pulsing but for an 8A-50ms-10% experiment, there were only 5 data points. We estimate $\delta e_{digit} \approx 0.2\%$ for experiments with data points below 10.

Considering the above mentioned sources of errors, we have a function of a total error in (22),

$$\begin{aligned} \epsilon_{total} &= |\epsilon_{shunt}| + |\epsilon_{ADC}| + |\epsilon_{offset}|(+|\epsilon_{digit}|) \\ &\approx 1.5\% + \frac{0.152}{I \times 10 \times 4.984} + \frac{0.08}{I \times 10} + (0.2\%) \end{aligned} \tag{22}$$

For a 8 A 50ms 10% experiment, $\epsilon_{total} \approx 1.513\%$. For a higher duty cycle, the error would be lower than for a lower duty cycle.

4.5.2. *Errors of* $\delta\emptyset$

$\delta\emptyset$ is deduced from $\emptyset_H$ and $\emptyset_B$, thus its error should consider the errors of both $\emptyset_H$ and $\emptyset_B$. An example of the calculation is explained as follow.

Assume $\emptyset_{LH} \leq \emptyset_H \leq \emptyset_{UH}$ and $\emptyset_{LB} \leq \emptyset_B \leq \emptyset_{UB}$, and $\delta\emptyset$ has its upper bound $\delta\emptyset_{UH}$ and lower bound $\delta\emptyset_{LH}$. Thus according to its definition in (8), there is

$$\delta\emptyset_{UH} = = \frac{\emptyset_{UH} - \emptyset_{LB}}{\emptyset_{LB}} \tag{23}$$

$$\delta\emptyset_{LH} = = \frac{\emptyset_{LH} - \emptyset_{HB}}{\emptyset_{HB}} \tag{24}$$

Usually $\delta\emptyset_{UH}$ and $\delta\emptyset_{LH}$ tend to have very large differences, it is because $\delta\emptyset$ is a relative value and the errors of $\emptyset$ time together to form the error of $\delta\emptyset$. For example, we assume $\emptyset_H \approx 6.5 \pm 1.5\%$ and $\emptyset_B \approx 5.5 \pm 1.5\%$, then according to (23) and (24), we have

$$\delta\emptyset_{UH} = \frac{6.5(1 + 1.5\%) - 5.5(1 - 1.5\%)}{5.5(1 - 1.5\%)} \approx 21.781\% \tag{25}$$

$$\delta\emptyset_{LH} = \frac{6.5(1 - 1.5\%) - 5.5(1 + 1.5\%)}{5.5(1 + 1.5\%)} \approx 14.689\% \tag{26}$$

Chapter 5 Modeling and Simulation

Building models for batteries and ultracapacitors is an important and intuitive method to expand the experiments and estimate the performances of interest in every degree of freedom. In previous works, simple models for Ni-MH batteries and EDLCs have already been built and investigated [12]. However due to aging and usage, the parameters of EDLCs changed and could not predict the behaviors of EDLCs used in this experiments. Thus this thesis focuses on the modeling of Li-ion batteries and remodeling the ultracapacitors.

5.1 Modeling of Ultracapacitor

Three ultracapacitors were connected in series in our experiments. To simplify, they were considered to be identical. Thus the model of a single ultracapacitor was derived, based on Zubieta [30]. The idea is to find a proper number of RC branches and a set of RC parameters which can make simulation waveforms match experimental waveforms.

The experiments contain three parts: a single pulse charging experiment (shown in Fig. 2-6), a 2-pulse discharge and a 100-pulse discharge experiment. In both discharge experiments, the pulse amplitude is 8 A, the period is 200 ms and the duty cycle is 10%. The model is expected to simulate both experiments and give out relatively accurate waveforms of capacitor voltages.

Because of the nonlinear capacitance of the ultracapacitor, it cannot be represented by a transfer function. A set of nonlinear differential equations was deduced to illustrate the ultracapacitor, as shown in Appendix 2. Variables of the equations are the charge on the capacitors in each branch, see Fig. 2-3. The input to the equations is the capacitor's current. MATLAB ordinary differential equation command ODE23tb was used to solve the charge on the capacitors in each branch, and accordingly the voltage of each capacitor was calculated.

Genetic Algorithm (GA), a MATLAB optimization and nonlinear fitting function, was used to find the desired branch number and RC parameters. The working theory of the GA function can be simply explained as follow: the variables of the GA function are RC parameters. It randomly picks an initial set of RC parameters, solves the ultracapacitor's ODE functions to get the waveform. If the simulated waveform substantially different from the experimental one, GA will use its optimization algorithm to find another set of RC parameters and repeats the process again, until the optimization goal is met. Here the optimization goal is to minimize the norm of the difference between the actual voltage waveform and the simulated voltage waveform. Both an infinite norm fitting and a second order norm fitting were conducted. In addition, following the detailed instruction in [19]

and [30], a set of RC parameters were "hand extracted" and were used as a reference to set boundary conditions on the GA function.

Considering the trade-off between how accurate the model is and how many branches the model has, a four branch model was chosen, as shown in Fig. 5-1. The periods of pulse loads we are interested in are from 50 ms to 400 ms, but the total discharge time is on the order of minutes considering the pulse number to be 2000. Thus the RC pairs need to represent ultracapacitors' behaviors in the several millisecond range, several tens of millisecond range, and several second range. The extracted parameters are listed in Table 5-1.

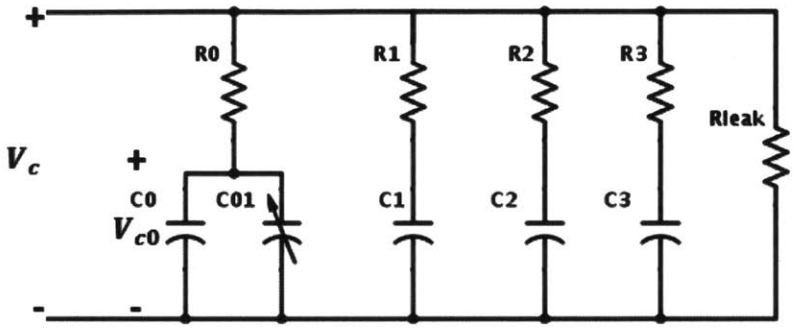


Fig. 5- 1 Four-branch model for the Ultracapacitor

Table 5- 1 Parameters of Ultracapacitor model

R_0	0.0261 Ω	Time constant	0.529 second
C_0	20.28 F		
R_1	0.06 Ω	Time constant	25 ms
C_1	0.417 F		
R_2	0.1313 Ω	Time constant	200 ms
C_2	1.5374 F		
R_3	186.4 Ω	Time constant	195.72 second
C_3	1.05 F		
C_{01}	0.05	-	
R_{leak}	74.737 kΩ	-	

Using this set of parameters, simulation waveforms were compared with experimental waveforms for the single pulse charging experiment, the two-pulse charging experiment and the 100-pulse discharging experiment, as shown in Fig. 5-2 (a), (b) and (c). The waveforms show a good fit for discharge experiments at both short-time period and long-time period, however they have larger differences for the charging experiments.

This indicates the parameters fail to describe the ultracapacitor's behavior well in the second or minute time range. It might be caused by the inaccuracy of the RC pair featuring the longest time constant (195.72 second) as well as the inaccuracy of the nonlinear capacitance. However finding a perfect fit is difficult. The boundary of the GA function significantly influences its result, and slight changes in the boundary may result in a different set of optimized parameters. Thus it requires many times of try-and-fail to find optimized parameters. Time is at last the limitation for us to find a better set of parameters.

5.2 Modeling of Lithium Ferro Phosphates (LiFePO4) battery

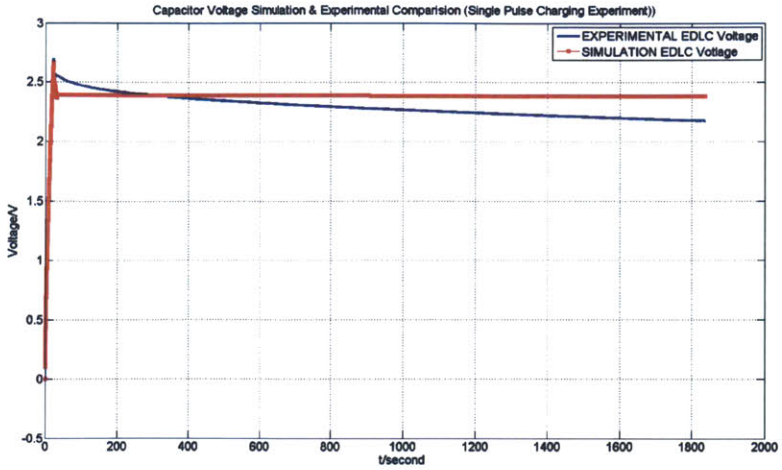
5.2.1. Design of Parameter Extraction Experiments:

In our investigation, we have two Li-ion battery cells in series, and for simplicity, they are modeled as a one-cell battery. Based on the proposed model in [35], there are several parameters needing to be determined through parameter identification experiments: SOC – OCV relationship, series resistance R_s, several transient RC pairs R_{ts}/C_{ts}, R_{tm}/C_{tm}, R_{tl}/C_{tl}, etc. with different time constants.

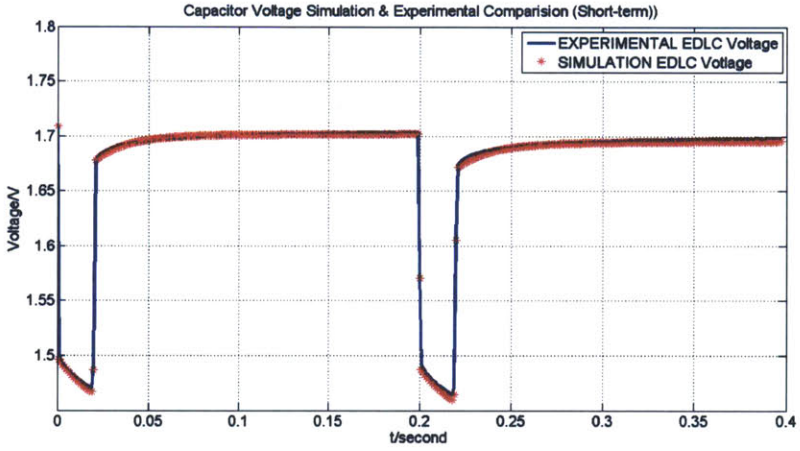
The steps of the parameter identification experiments were designed after comparing the extraction methods listed in Section 2.4 and considering the following requirements of our experiments:

- To conduct pulse discharge experiments from fully charged to depletion to obtain the SOC-OCV curve. Usually the SOC-OCV curves during charging and discharging are different, however this investigation focuses more on the discharge process;
- To obtain reasonable amount of points (at least 10 points) on the SOC-OCV curve. Thus each pulse discharge should take out 0.2 Ah of charge from the battery;
- To reduce the size of data, the sampling time should be no less than 0.5s;
- To get an accurate open circuit voltage of the batteries, a rest time of 30 minutes was set between two discharge pulses;
- To conduct multiple experiments under different discharge rates and take the average to reduce errors; experiments with a discharge rate of over 1C should also be tested, because the pulse amplitudes we are interested in are from 4 A to 20 A, which are all above 1C.

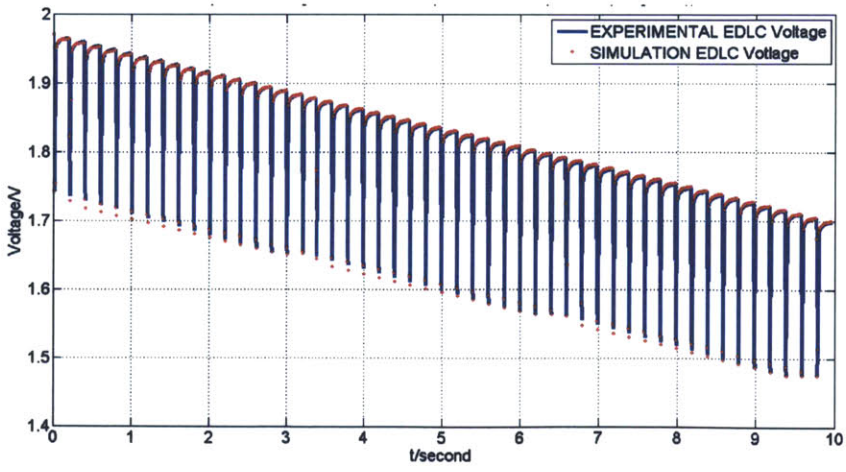
The test steps of parameter extraction experiments are shown in Fig. 5-3. And referring to several important specifications in the data sheet of the A123 ANR26650-M1 Li-ion cell [43], such as the cut-off voltage and the maximum discharge current, three different sets of experiments were designed, as shown in Table 5-2.



(a) The ultracapacitor voltage in the single pulse charging



(b) The ultracapacitor voltage in a 2-pulse discharge



(c) The ultracapacitor voltage in a 100-pulse discharge

Fig. 5- 2 Simulation and experimental comparison for the ultracapacitor model

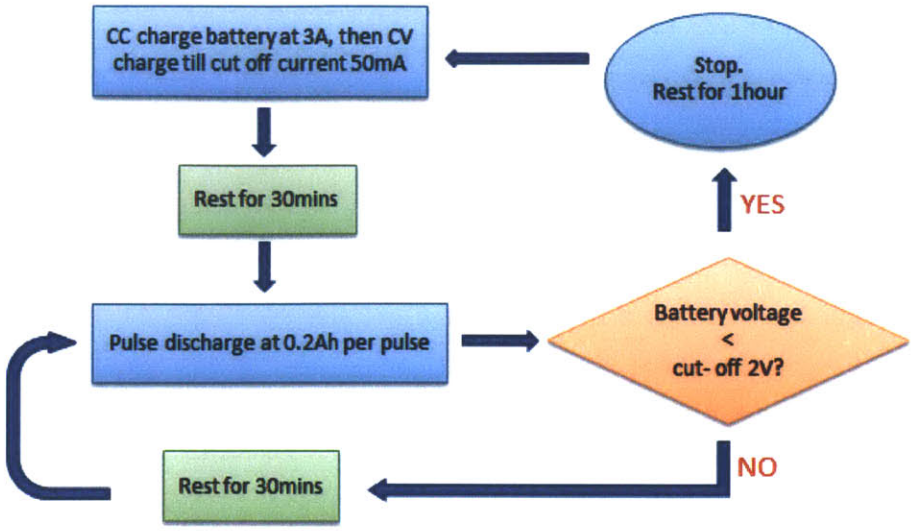


Fig. 5- 3 Test steps of parameter extraction experiments for Li-ion batteries

Table 5- 2 Parameter extraction experiments sets for Li-ion batteries

	Pulse Amplitude	Pulse Width	Rest Width	Rest time between Charge and Discharge	Total time
1	0.4C = 0.92A	13.043 min (782.61s)	30min	Rest over 45 minutes	7.88hr
2	1C = 2.3A	5.2174 min (313.04 s)	30min		6.42hr
3	3.478C = 8A	1.5 min (90s)	30min		5.775hr

5.2.2. Parameter Identification Results:

1) SOC OCV Curve:

As mentioned in Section 3.4.2, the ESSs are discharged over 2000 pulses to 99% SoC. For a 200 ms pulse load, it takes 7 minutes to complete the discharge. Thus it is more reasonable to consider batteries' OCV changing with their SoCs during this process rather than considering them as constant voltage sources.

Connecting the open circuit voltage measured under different state-of-charge, the SOC-OCV curves are shown in Fig. 5-4. The 1C discharge experiment was conducted twice to reduce errors. Four sets of experiments show a very good consistence in the range of 95% to 20% SOC.

Based on the average of these curves, an OCV-SOC look-up table was built by using interpolation, as shown in Table 5-3.

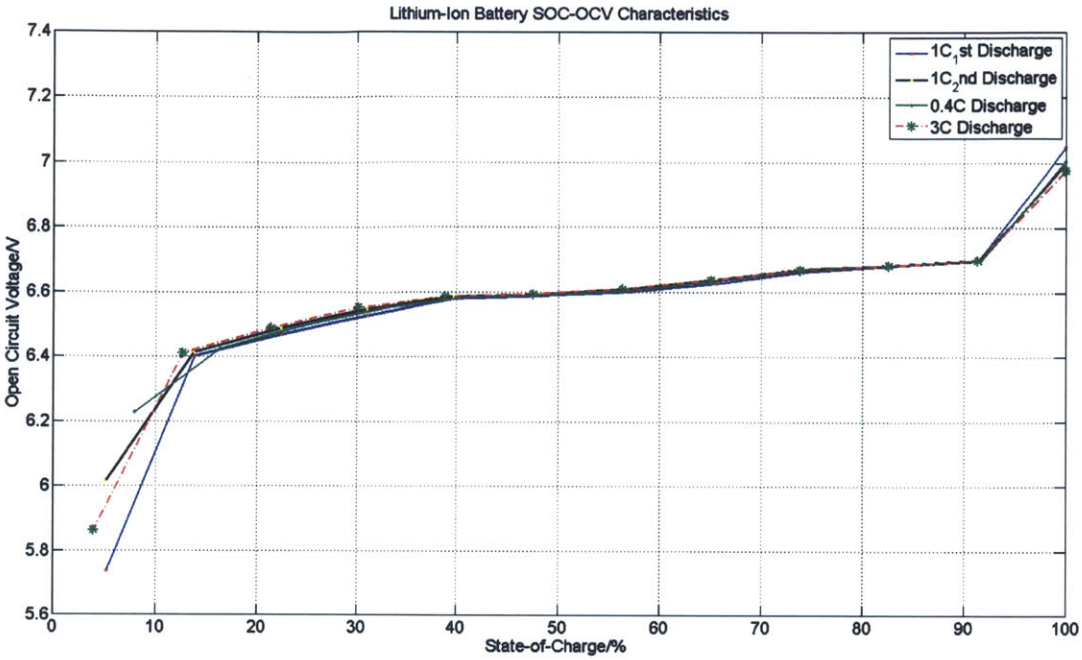


Fig. 5- 4 The tested SOC-OCV curve of Li-ion batteries used in the investigation

Table 5- 3 SoC-OCV of the Li-ion battery look-up table

SOC/ %	OCV/ Volt	SOC/ %	OCV/ Volt	SOC/ %	OCV/ Volt	SOC/ %	OCV/ Volt
1	7.094	0.99	6.945133	0.98	6.886163	0.97	6.840192
0.96	6.800221537	0.95	6.787233	0.94	6.771434	0.94	6.760232
0.93	6.752323484	0.92	6.743341	0.91	6.734213	0.9	6.724575
0.89	6.709681121	0.88	6.690788	0.87	6.688894	0.86	6.687001
0.85	6.685107407	0.84	6.683257	0.83	6.681616	0.82	6.680019
0.81	6.678420764	0.8	6.676823	0.79	6.675225	0.78	6.673627
0.77	6.672029346	0.76	6.670431	0.75	6.668094	0.74	6.664293
0.73	6.660399713	0.72	6.656506	0.71	6.652613	0.7	6.648719
0.69	6.644825624	0.68	6.640932	0.67	6.63709	0.66	6.633545
0.65	6.630292555	0.64	6.62704	0.63	6.623787	0.62	6.620535
0.61	6.617282231	0.6	6.61403	0.59	6.610818	0.58	6.608002
0.57	6.606023439	0.56	6.604357	0.55	6.602691	0.54	6.601025
0.53	6.599359106	0.52	6.597693	0.51	6.596027	0.5	6.594529
0.49	6.593163778	0.48	6.592112	0.47	6.591139	0.46	6.590166
0.45	6.589192608	0.44	6.588219	0.43	6.587246	0.42	6.585981
0.41	6.584245076	0.4	6.581124	0.39	6.576689	0.38	6.572161
0.37	6.567632244	0.36	6.563104	0.35	6.558575	0.34	6.554047

0.33	6.548819526	0.32	6.543575	0.31	6.537443	0.3	6.530514
0.29	6.523584401	0.28	6.516655	0.27	6.509726	0.26	6.502797
0.25	6.495569743	0.24	6.488172	0.23	6.480491	0.22	6.47224
0.21	6.4637834	0.2	6.455327	0.19	6.446871	0.18	6.438414
0.17	6.428891423	0.16	6.416741	0.15	6.40459	0.14	6.370868
0.13	6.322724103	0.12	6.270133	0.11	6.217543	0.1	6.164952
0.09	6.112361414	0.08	6.065097	0.07	6.018353	0.06	5.977858
0.05	5.962110913	0.04	5.961456	0.03	5.7	0.02	5.5
0.01	5						

2) ***RC parameters:***

Obtaining parameters for RC pairs in Fig. 5-5 was a similar process with that of modeling ultracapacitors. Considering the assumption that the RC parameters will stay relatively constant under the same SoC (see section 3.4), the parameters under 99% SoC were extracted.

The idea is also to find a proper number of RC pairs and a set of RC parameters, which can make simulation waveforms match experimental waveforms. This time, to further increase the accuracy, we hope to obtain one set of parameters which can fit multiple pulse load profiles at the same time.

The experiments contain two sets: one set of the pulse load has a pulse amplitude of 16 A, a period of 200 ms and a duty cycle of 10%; another set has an amplitude of 8A, a period of 200 ms and a duty cycle of 50%. Battery voltages under these two different pulse load conditions were simulated.

Because all RC parameters were considered linear, the simulation of battery-only model was based on Laplace transfer function. Again, the GA function in MATLAB was used to optimize the parameters. The optimization goal is to minimize the norm of the difference between the actual voltage waveforms and the simulated voltage waveforms under two types of pulse profiles.

Similarly, following the extraction steps in reference [35] and [41], a set of RC parameters were "hand extracted" and used as a reference to set boundary conditions on the GA function.

Trade-offs have been made when deciding the number of RC pairs in the battery model and it was decided to use a model of four RC pairs, as shown in Fig. 5-5. The optimized parameters under 99% SOC are listed in Table 5-4.

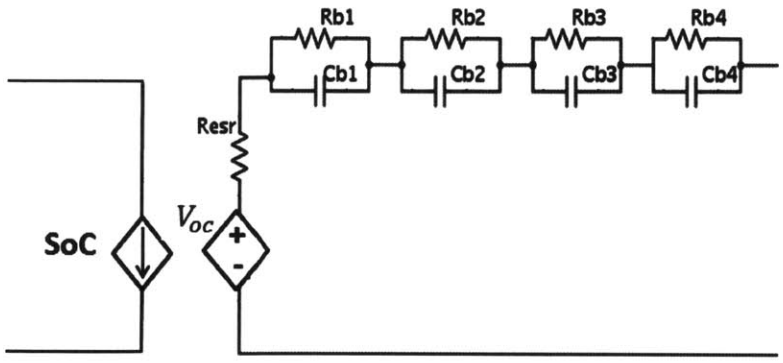


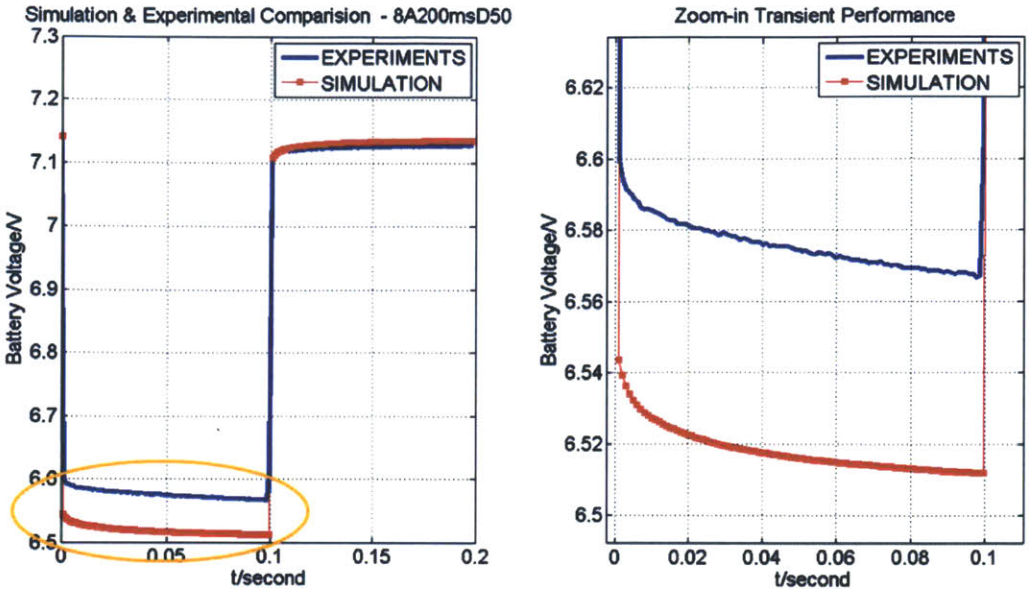
Fig. 5- 5 The Li-ion battery model used in the simulation (Four RC-pair)

Table 5- 4 Parameters of the Li-ion battery model

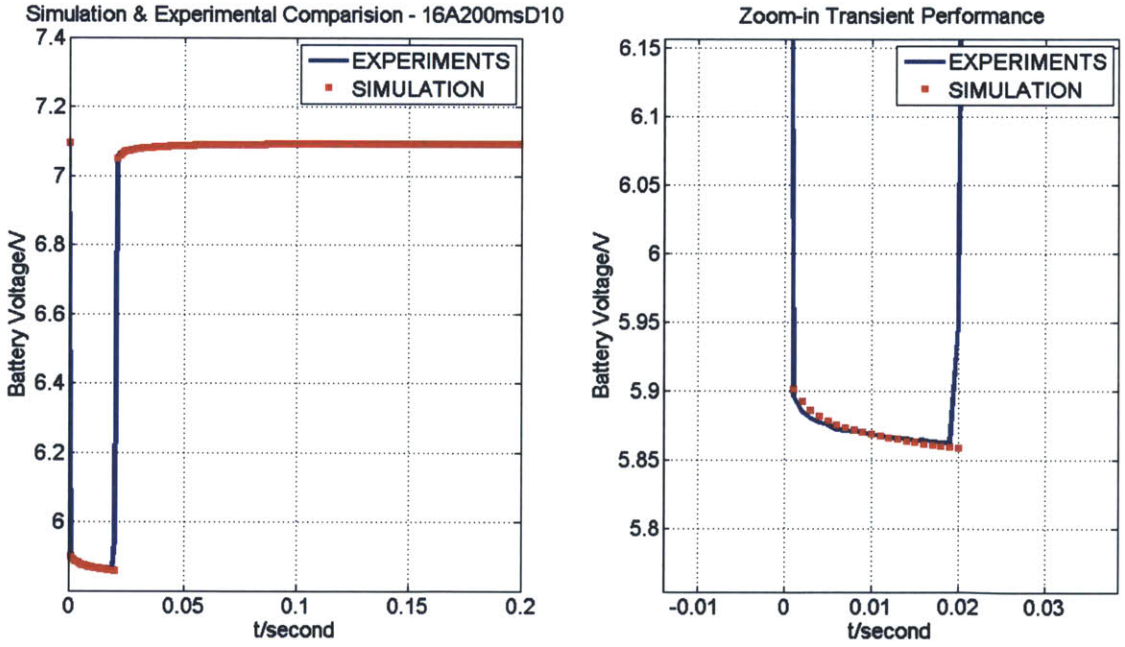
Rb1	1.216 mΩ	Time constant	2.28 ms
Cb1	1.8784 F		
Rb2	1.57 mΩ	Time constant	15.88 ms
Cb2	10.12 F		
Rb3	3.65 mΩ	Time constant	237 ms
Cb3	65.736 F		
Rb4	74.812 mΩ	Time constant	13.95 s
Cb4	186.47 F		
Resr	74.5 mΩ	-	

The waveforms comparisons of two different pulse load conditions are shown in Fig. 5-6 (a) and (b). (a)'s pulse profile is 8 A, 200ms, 50% duty cycle and (b)'s profile is 16 A, 200 ms, 10% duty cycle. One can see the transient performance fits well for both waveforms but there are still some mismatches during pulsing (circled in orange) in (a). The reasons are analyzed in the next subsection "Analysis of Sources of Errors".

To further verify the parameters and the SoC-OCV curve in a long-term run, simulations and experiments were done on a 2000 pulses discharge with an amplitude of 16 A, a period of 200 ms and a duty cycle of 10%, as shown in Fig. 5-7.



(a) **Battery current and voltage under a 8 A, 200 ms, duty cycle 50% discharge**



(b) **Battery current and voltage under a 16 A-200 ms-duty cycle 10% discharge**

Fig. 5- 6 Simulations and experiments comparison using parameters from Table 5-4

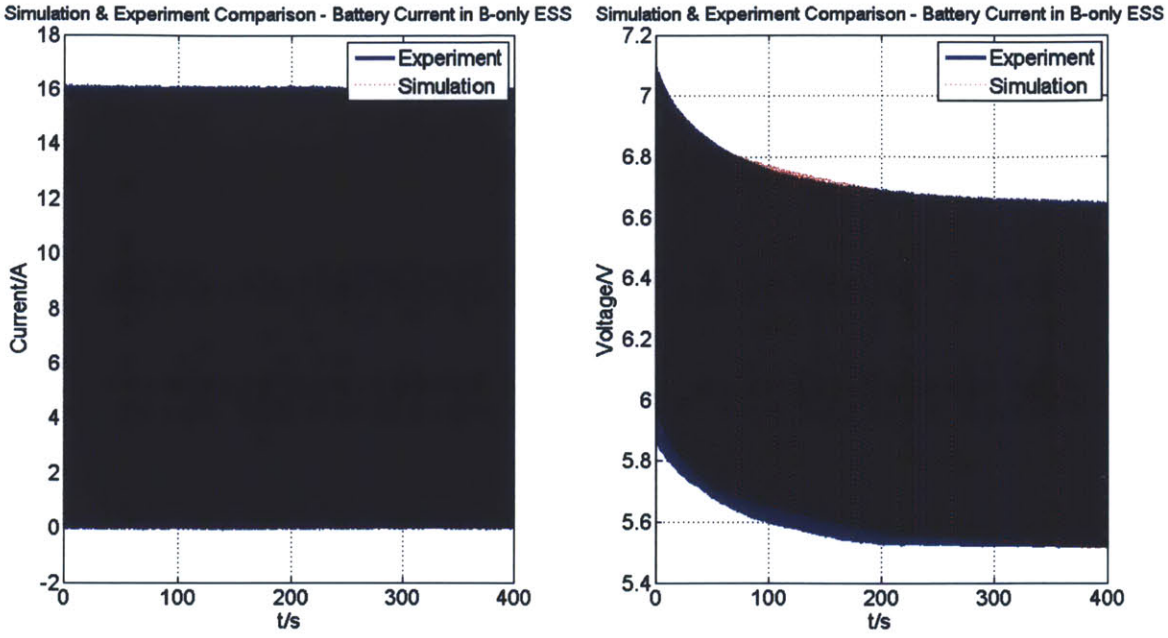


Fig. 5- 7 Simulations and experiments comparison using parameters from Table 5-4 (Battery current and voltage under a 2000 pulse discharge with a 16 A- 200 ms-10% duty cycle pulse load)

5.2.3. Analysis of Sources of Errors:

Figure 5-6 illustrates the short-term performance of the model and Fig. 5-7 illustrates the long-term performance. Both figures show a relatively good fit but still have errors. The sources of errors in the modeling are considered as follow:

- RC parameters change with discharge rate and state-of-charge of the battery. As mentioned in Section 3.4, an assumption is made for the investigation that the SoC-OCV curve as well as the RC parameters will stay relatively constant under the same SoC. However in reality, they change with discharge rates, temperatures, discharge frequency, etc. Thus it is very hard to find a group of fixed parameters to fit discharge curves under different conditions. In Fig. 5-6(a), zooming in the orange circled part on the left graph, one can get the right side graph. The battery voltage of the simulation and the experiment differ by about 0.06 V. Consider the pulse amplitude of 8 A, this difference means a difference of 1 mΩ in the ESR resistance. For different pulse profiles, the differences are always several mΩ for different pulse profiles.

- Linearization of SOC-OCV curve. From Fig. 5-7, one can see that the envelop of the simulation voltage is more linear than that of the experimental voltage. Due to the time limitation, only 10 points on the SoC-OCV curve were measured and interpolation was

used to calculate the points in between. This may cause linearization on the curve which might not be the case in reality.

- SoC-OCV curve change with discharge rate. Same with the RC parameters, the SoC-OCV curve of a battery does not stay constant as well. Here we assume that the SoC-OCV curve will stay relatively constant. The similarity between SoC-OCV curves obtained under different discharge rate, as shown in Fig. 5-4 also supports this assumption. However in reality, the curve changes with different discharge rate of the battery, which should be taken into consideration for further research.

5.3 Simulation of Li-ion Battery-Ultracapacitor Hybrid System

Using both the models of batteries and ultracapacitors, the circuit model of the hybrid ESS was built and shown in Fig. 5-8.

Because of the nonlinearity of the ultracapacitors, this model is also described by a set of nonlinear differential equations, as shown in Appendix 3. Variables of the equations are the charge on each capacitor, including both the batteries' and the ultracapacitors'. The nonlinear differential equation solver in MATLAB "ode23tb" was used again to solve the system.

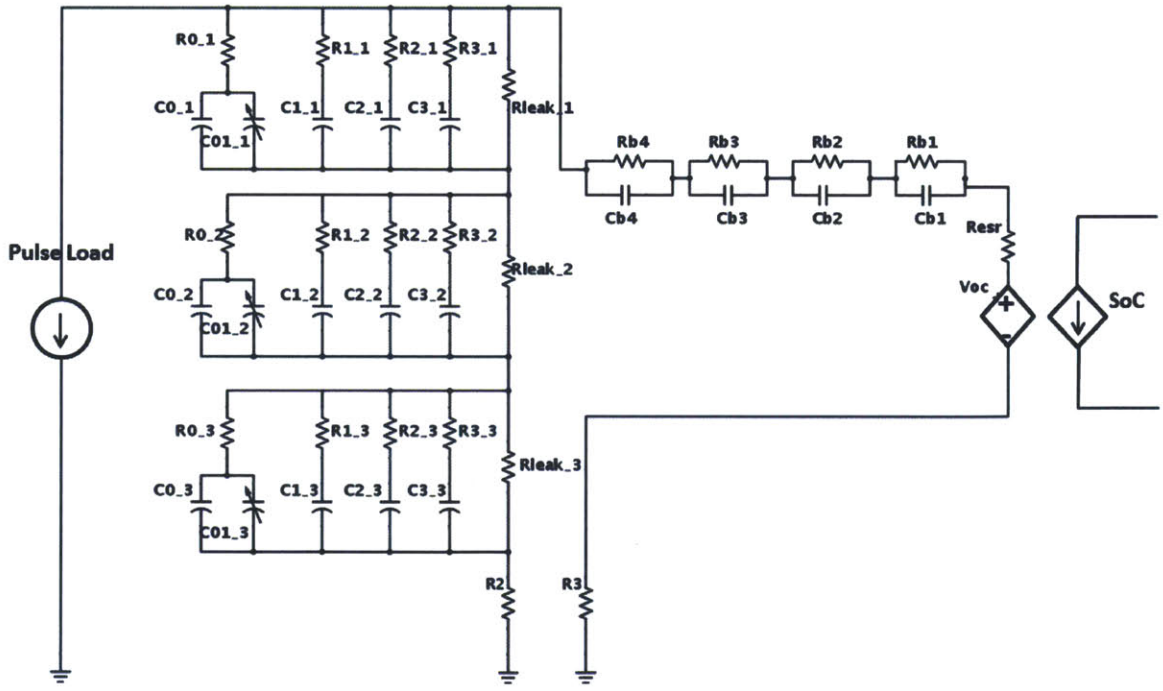
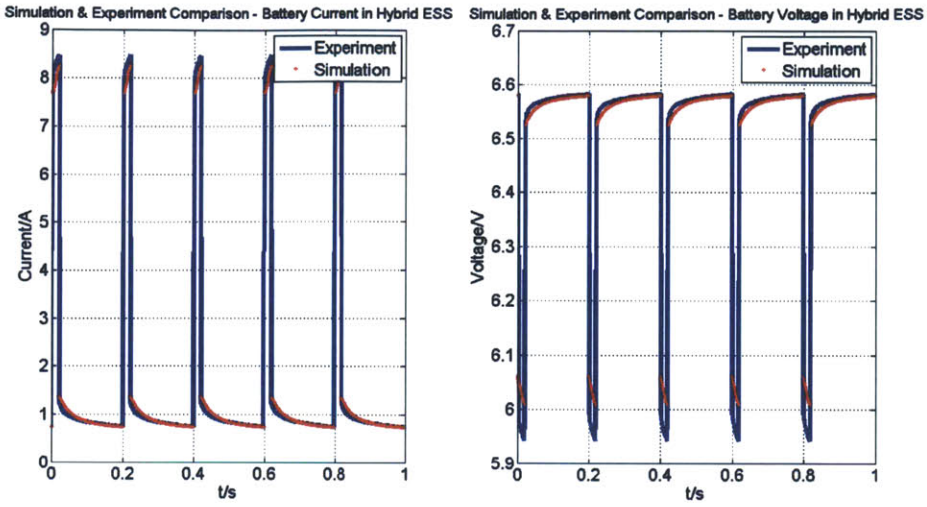
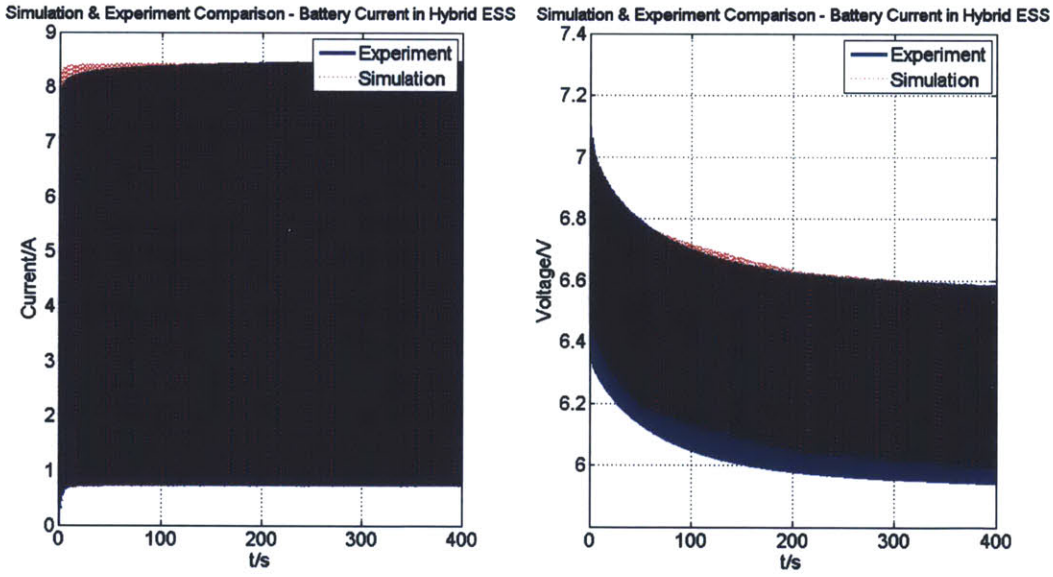


Fig. 5- 8 Model of battery-ultracapacitor hybrid system

Both experiments and simulations were conducted on a 5-pulse discharge and a 2000 -pulse discharge (pulse amplitude of 16 A, period of 200 ms and 10% duty cycle). The waveforms are compared in Fig. 5-9 (a) and (b).



(a) Battery current and voltage under a 5-pulse load (16 A, 200 ms, 10% Duty Cycle)



(b) Battery current and voltage under a 2000-pulse load (16 A, 200 ms, 10% Duty Cycle)

Fig. 5- 9 Simulations and experiments comparison of the hybrid model

One can see that even though the current waveform fits relatively well, the battery voltage during pulsing in the simulation is not as low as that in the experimental results.

A possible reason is as mentioned in Section 5.1, the inaccuracy of the ultracapacitor model. The ultracapacitor model did not predict the ultracapacitors' behavior very well in

the long-term period. Another more important reason is thought to be the current sharing relationship between the battery branch and the ultracapacitor branch. This is usually decided by the total resistances of the battery branch and the ultracapacitor branch. We tried to revise R_0 in the ultracapacitor model and R_{esr} in the battery model, but it is hard to get both current and voltage waveform fit perfectly at the same time because all RC parameters are correlated and influence the current and voltage together. More effort should be made on balancing R_0, R_1, R_2, R_3 as well as R_{esr}, R_{b1}, R_{b2}, R_{b3}, R_{b4}. The main idea of finding a more accurate set of parameters is to first find a proper overall resistance of the battery which can provide the proper voltage drop, and then adjust the overall resistance of the ultracapacitor to provide the right current sharing ratio.

Chapter 6 Results

In Chapter 3, the metric *discharge capacity* $\emptyset$ was defined to indicate the energy capacity of an energy storage system (ESS), i.e., how much energy one can draw out of an ESS; the metric *normalized energy gain* $\delta\emptyset$ was defined to compare the energy capacity of the battery-ultracapacitor hybrid system with that of the battery-only system, i.e., how much more energy one can draw out by paralleling ultracapacitors across batteries.

$\emptyset$ and $\delta\emptyset$ are functions of multi variables. In this thesis, six degrees of freedom have been investigated: pulse amplitude, pulse frequency, pulse duty cycle, capacitance, battery type and state-of-charge of the energy storage system. Hardware experiments were conducted for both Li-ion batteries and Ni-MH batteries under different pulse load profiles. Due to time limitation, simulations were conducted only for Li-ion batteries with 25 F ultracapacitors while sweeping other variables. In all cases, $\emptyset$ and $\delta\emptyset$ were calculated based on (15) and (16). Representative results are shown and analyzed in this chapter.

6.1 Trend of Discharge Capacity ($\emptyset$)

Fig. 6-1, 6-2, 6-3 shows respectively the trends of discharge capacity changing with pulse amplitude, duty cycle and pulse period. There are several obvious trends which can be summarized:

- In all cases, $\emptyset$ of the hybrid system is better than that of the battery-only system, which means the hybrid system performance is better than the battery-only system.
- $\emptyset$ depends strongly upon the pulse amplitude. It decreases when the pulse amplitude increases.
- $\emptyset$ changes mildly when the duty cycle and the pulse period change. However it still shows a decreasing trend when the duty cycle and the pulse period increase.
- $\emptyset$ is larger when the paralleled capacitance is larger.

Several instructions of reading the result graphs are listed as following:

- In all figures, the error bars for experimental data were created based on the calculation in section 4.5.

- In each figure, (a) shows the comparison of simulation results and experimental results for Li-ion batteries. Data labeled with "Simu" in the legend shows simulation results and data labeled with "Expe" in the legend shows experimental results. Except Fig. 6-2, all (b) shows more experimental results for Li-ion batteries and (c) shows experimental results for Ni-MH batteries. For Fig. 6-2, because there are few experimental results for Li-ion batteries other than in (a), (b) shows experimental results for Ni-MH batteries.

- All blue lines/dots show the Ø of the battery-only system, labeled with "- B" in the legend;

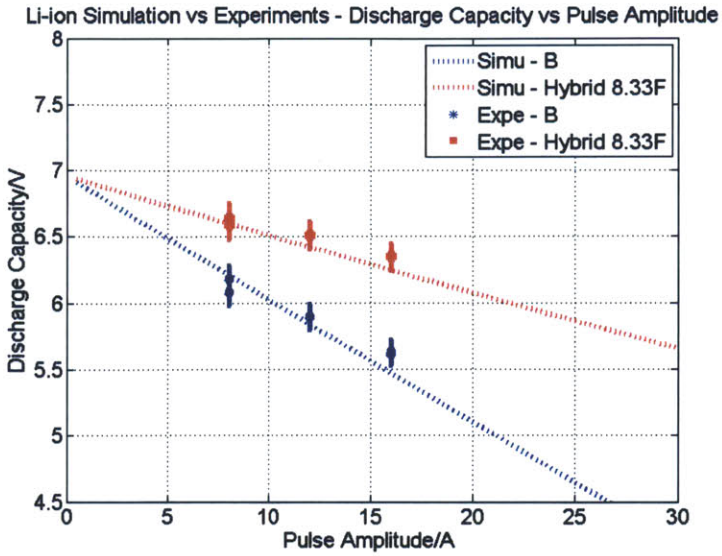
- All light blue dots/lines show those for the hybrid system with 1.667 F ultracapacitors (three 5 F EDLCs in series), labeled with "- Hybrid 1.667 F" in the legend;

- All red lines/dots show results for the 8.333 F hybrid system (three 25 F in series), labeled with "- Hybrid 8.33 F" in the legend;

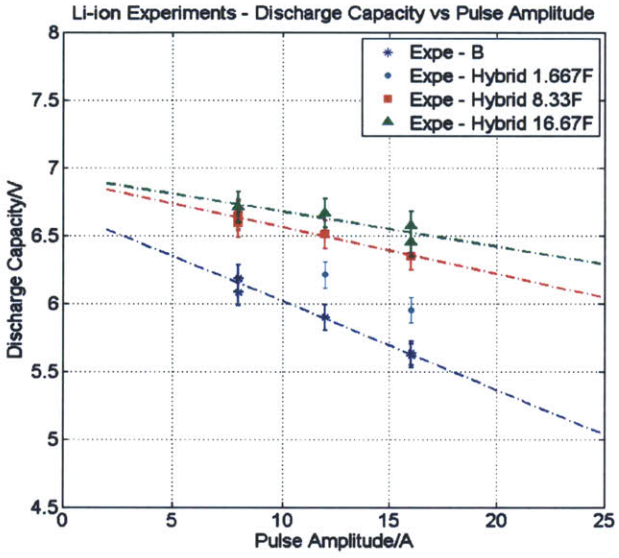
- All green dots/lines show those for the 16.667 F hybrid system (three 50 F in series), labeled with "- Hybrid 16.67F" in the legend.

The detailed analysis is described as following.

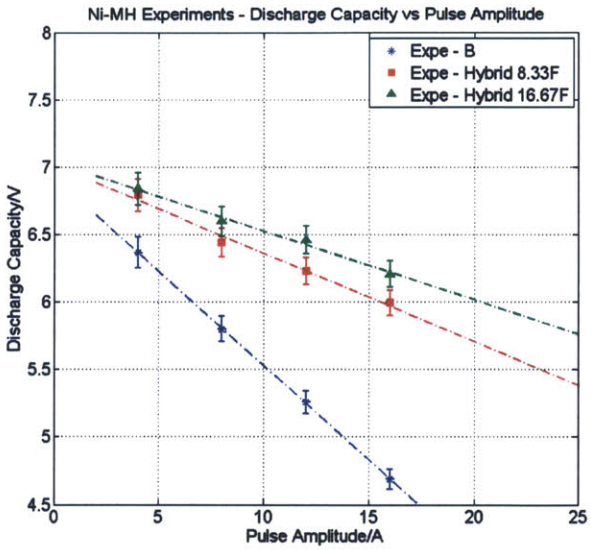
From Fig. 6-1, one can see that when the pulse amplitude increases, Ø goes down linearly. This is because when the pulse amplitude gets higher, the voltage drop on the series impedance of the energy storage system increases linearly. The ultracapacitors help to decrease the equivalent impedance of the ESS, thus the slope of the voltage drop is smaller than it of the battery-only system. In another word, the battery-only system is getting worse faster than the hybrid system.



(a) Comparison of simulations and experimental results (Li-ion & 8.33 F EDLC)



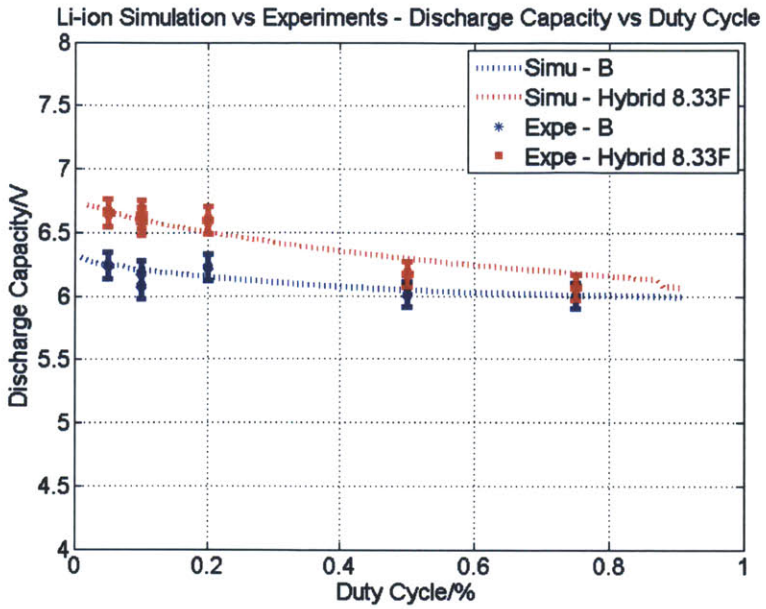
(b)More experimental results of Li-ion



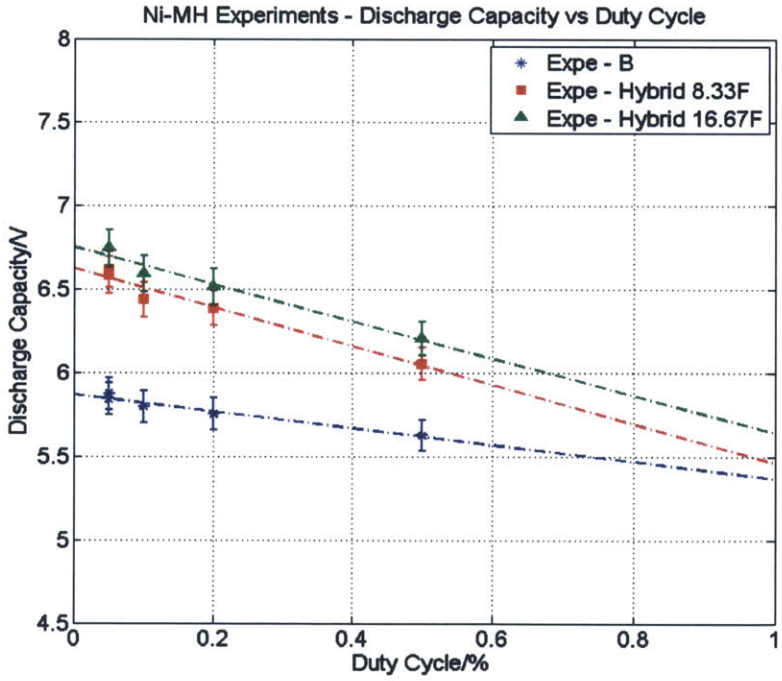
(c) Experimental results of Ni-MH

Fig. 6- 1 Discharge capacity decreases linearly as pulse amplitude increases (Pulse duty cycle 10%, pulse period 200ms)

Fig 6-2 shows Ø changing with the duty cycle when amplitude is fixed at 8 A and pulse period 200 ms. Ø also decreases when the duty cycle increases. When the pulse period is fixed, increasing duty cycle increases pulse width and thus extracts more charge during one pulse. Because the battery voltage will slightly decrease when lowering SoC, as the duty cycle increases the battery voltage will be lower. An extreme example would be that the duty cycle increases to 100% - the load is a constant current load. The ultracapacitors will have no benefits and only serve as a giant RC load to the battery. Thus the discharge capacity of hybrid system might be even worse than that of the battery-only system.



(a) Comparison of simulations and experimental results (Li-ion & 8.33 F EDLC)

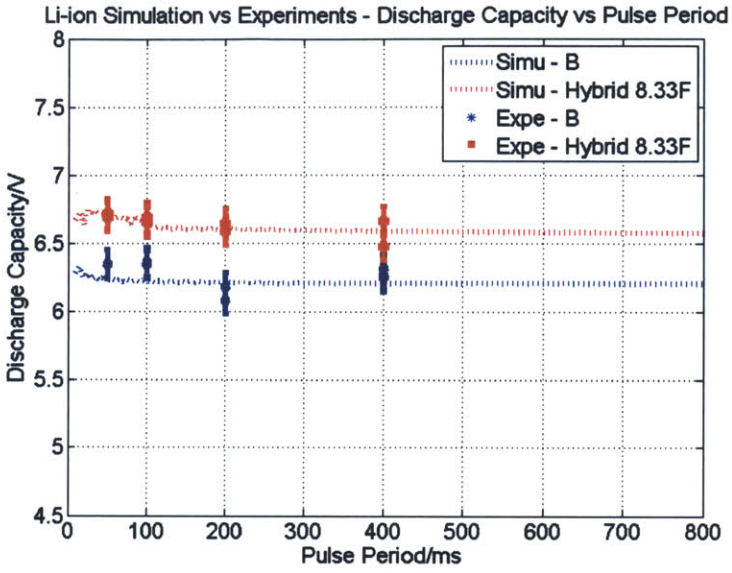


(b) Experimental results of Ni-MH

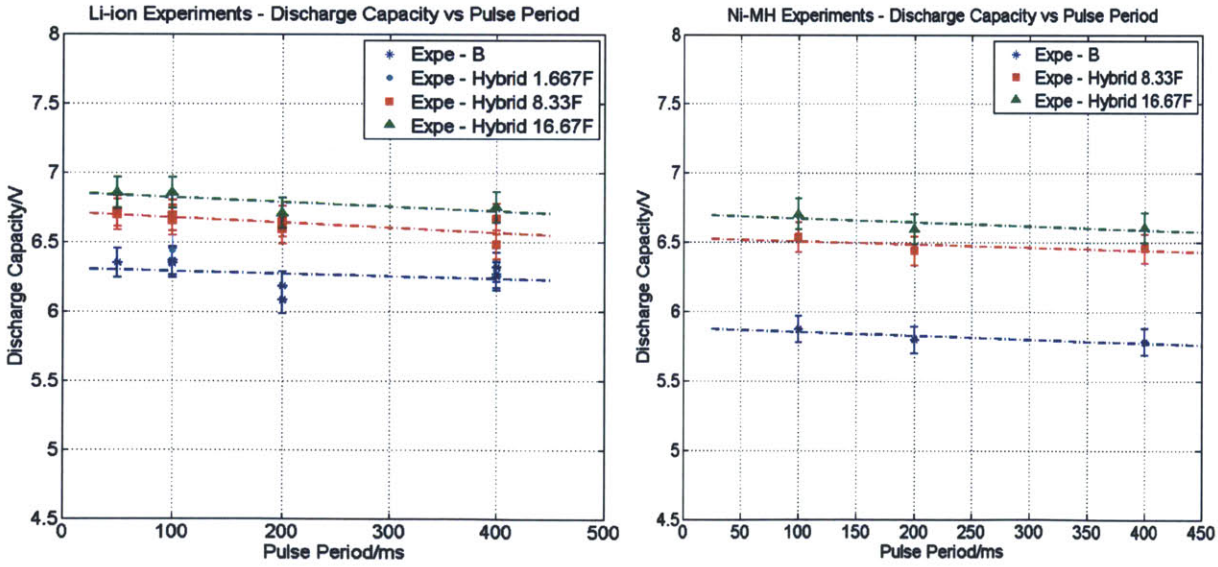
Fig. 6- 2 Discharge capacity decreases as pulse duty cycle increases (Pulse amplitude 8A, pulse period 200ms)

Fig 6-3 shows Ø changing with the pulse period when the amplitude is 8 A and the duty cycle is 10%. When the pulse frequency is low, Ø stays almost constant, and when it gets higher, Ø gets larger. Simply consider the overall impedance of the ultracapacitor as

$\frac{1}{2\pi fC}$, then when f increase, the impedance decreases thus lower the impedance of the hybrid energy system, resulting in a higher terminal voltage. However if the period increases to the extreme, as the same example mentioned in the duty cycle part, $\emptyset$ will eventually go down.



(a)Comparison of simulations and experimental results (Li-ion & 8.33 F EDLC)



(a)More experimental results for Li-ion **(b)Experimental results for Ni-MH**

Fig. 6- 3 Discharge capacity changes not significantly with pulse period (Pulse duty cycle 10%, pulse amplitude 8 A)

6.2 Trend of Normalized Energy Gain ($\delta\emptyset$)

Figures 6-4, Fig. 6-5 and Fig. 6-6 show the *normalized energy gain* $\delta\emptyset$ changing with pulse amplitude, duty cycle and pulse period, respectively. Several obvious trends can be picked out.

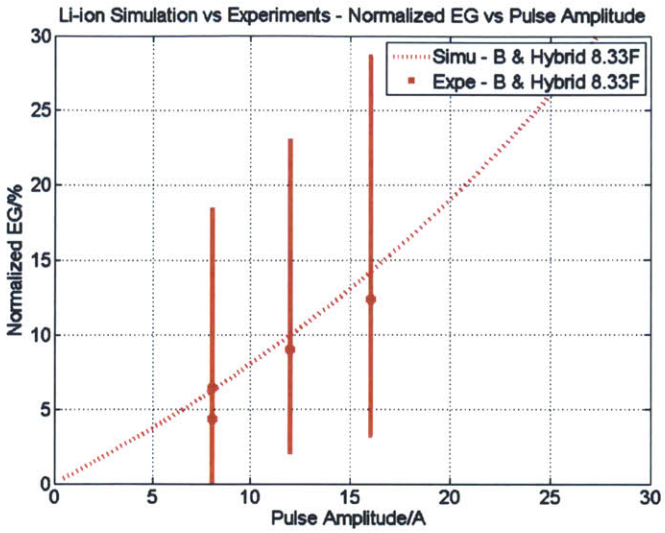
- $\delta\emptyset$ for Ni-MH batteries are larger than that for Li-ion batteries.
- $\delta\emptyset$ increases quadratically with the pulse amplitude, and decreases linearly when the duty cycle increases.
- $\delta\emptyset$ changes mildly when the pulse period changes.
- Larger capacitance yields a larger $\delta\emptyset$.

Several instructions of reading the result graphs are listed as following:

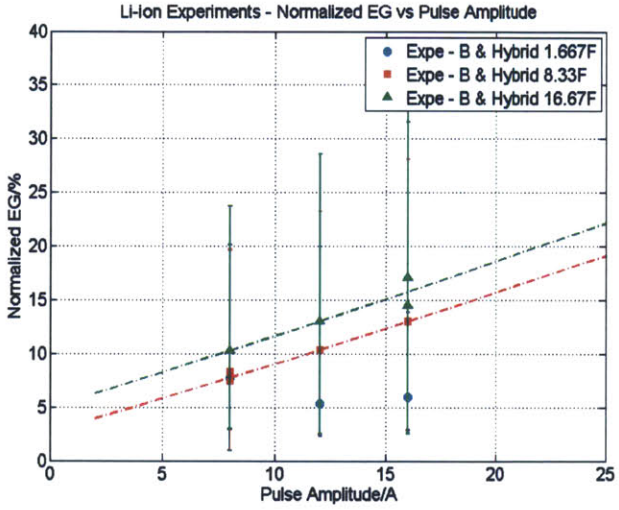
- In all figures, the error bars were created based on the calculation in section 4.5.
- In each figure, (a) shows the comparison of simulation results and experimental results for Li-ion batteries. Data labeled with "Simu" in the legend shows simulation results and data labeled with "Expe" in the legend shows experimental results.
- Except Fig. 6-5, all (b) shows more experimental results for Li-ion batteries and (c) shows experimental results for Ni-MH batteries. For Fig. 6-2, because there are few experimental results for Li-ion batteries other than in (a), (b) shows experimental results for Ni-MH batteries.
- All light blue dots/lines show the improvement percentage by incorporating the 1.667 F ultracapacitors (three 5 F in series), labeled with "- B & Hybrid 1.667 F" in the legend;
- All red lines/dots show the improvement percentage by incorporating the 8.333 F ultracapacitors (three 25 F in series), labeled with "- B & Hybrid 8.33 F" in the legend;
- All green dots/lines show the improvement percentage by incorporating the 16.67 F ultracapacitors (three 50 F in series), labeled with "- B & Hybrid 16.67F" in the legend.

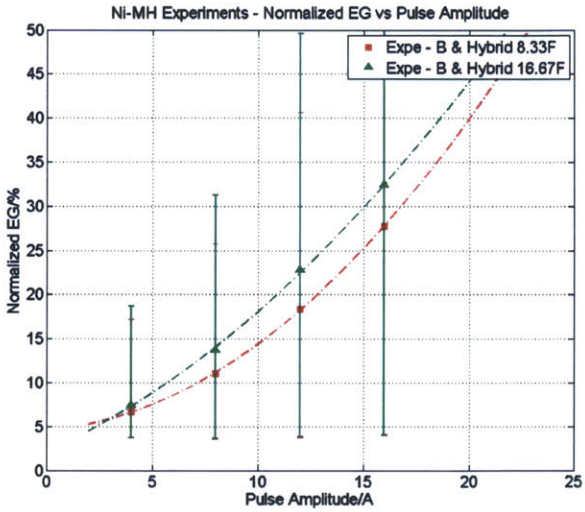
The detailed analysis is as following.

Figure 6-4 shows higher pulse amplitudes yield much higher gains, which is consistent with our theoretical analysis in Section 3.3.



(a)Comparison of simulations and experimental results (Li-ion & 8.33 F EDLC)





(b) More experimental results of Li-ion **(c) Experimental results of Ni-MH**

Fig. 6- 4 Normalized energy gain increases quadratically as the pulse amplitude increases (Pulse duty cycle 10%, pulse period 200ms)

A way to analyze the trends of $\delta\emptyset$ changing with pulse amplitude is by considering the energy dissipation. First, we want to demonstrate the hybrid system has a similar amount of total energy with the battery-only system. A rough calculation is conducted in (27) based on the specifications from datasheets [43], [44]. Note E_B and E_H represent the total energy in the hybrid system and the battery-only system respectively; E_{HB} is the part of E_H coming from the batteries and E_{HUC} is the part of E_H coming from the ultracapacitors.

$$E_{HB} = E_B = 6.6\,V \times 2.3\,Ah = 15.18\,Wh \tag{27}$$

$$E_{HUC} = \frac{1}{2}CV^2 = \frac{1}{2}8.33^2 \times 6.6\,J = 0.06\,Wh$$

$$E_{HUC} = 0.4\%E_{HB}$$

The extra energy by adding ultracapacitors is only 0.4% of that of the batteries. Thus the total energy of the hybrid system and the battery-only system is thought to be the same.

For both systems, energy is conserved between the source and the load. Only part of energy is transferred to the load and the rest of it is wasted as losses. $\delta\emptyset$ indicates how much more energy one can deliver to the load by paralleling ultracapacitors, which is equivalent to how much fewer the losses inside the hybrid system comparing to the battery-only system. E_{lB} and E_{lH} are losses inside battery-only system and hybrid system respectively, thus

$$\delta\emptyset \approx \frac{(E_H - E_{lH}) - (E_B - E_{lB})}{E_B - E_{lB}} \approx \frac{E_{lB} - E_{lH}}{E_B} \tag{28}$$

To simplify, only resistive losses are considered. I_{rms} is the rms current of the load; I_{brms} and I_{crms} are the rms currents of batteries and ultracapacitors respectively. Then,

$$\delta\emptyset \approx \frac{r_b {I_{rms}}^2 - (r_c {I_{crms}}^2 + r_b {I_{brms}}^2)}{E} \approx k{I_{rms}}^2 \tag{29}$$

Because the losses increase with the square of the current amplitude, $\delta\emptyset$ has the same trend – increase quadratically with pulse amplitude.

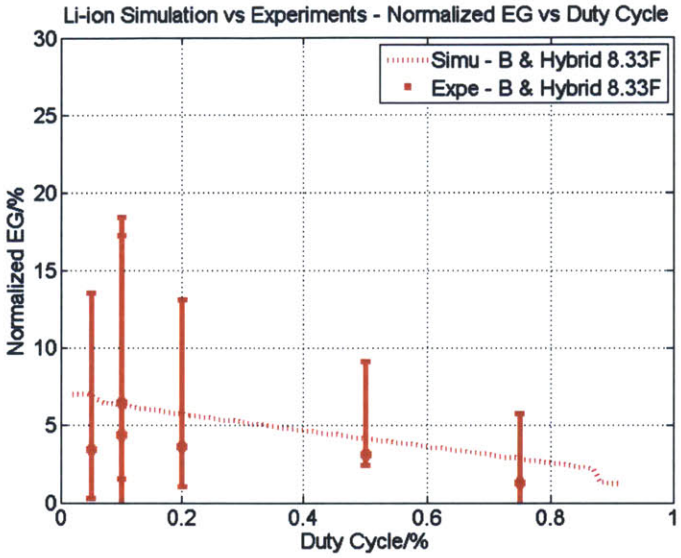
Figure 6-5 shows $\delta\emptyset$ decreasing when the duty cycle increases, because the ratio of rms current increases when the duty cycle decreases.

The rms current of the pulses is a square root of the pulse duty cycle, as shown in (30).

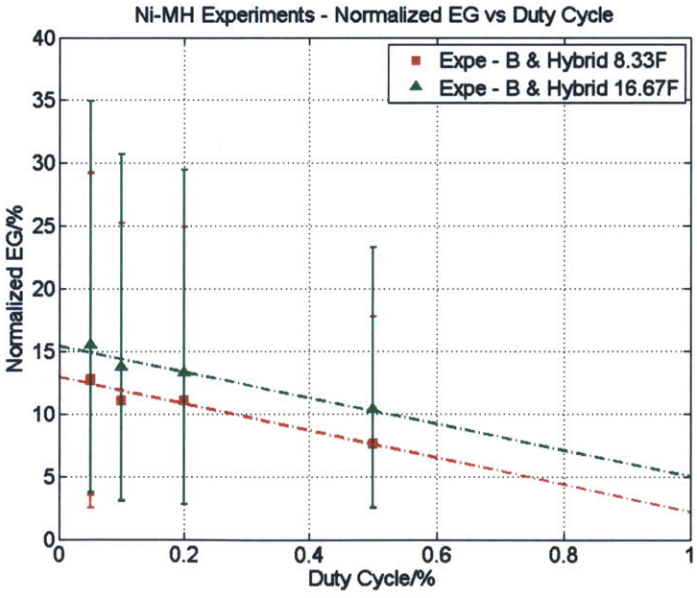
$$I_{rms} = I\sqrt{D} \tag{30}$$

Substituting the rms current in (29), the *normalized energy gain* is directly proportional with respect to duty cycle,

$$\delta\emptyset \approx kI^2D \tag{31}$$



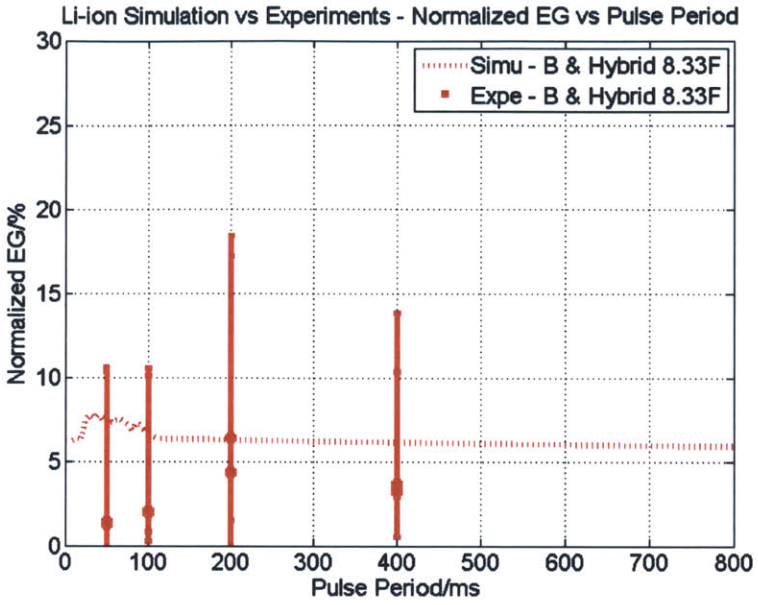
(a)Comparison of simulations and experimental results (Li-ion & 8.33 F EDLC)



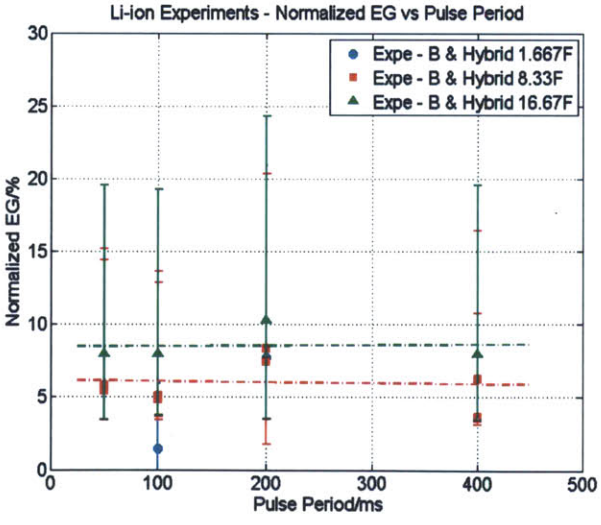
(b) experimental results of Ni-MH

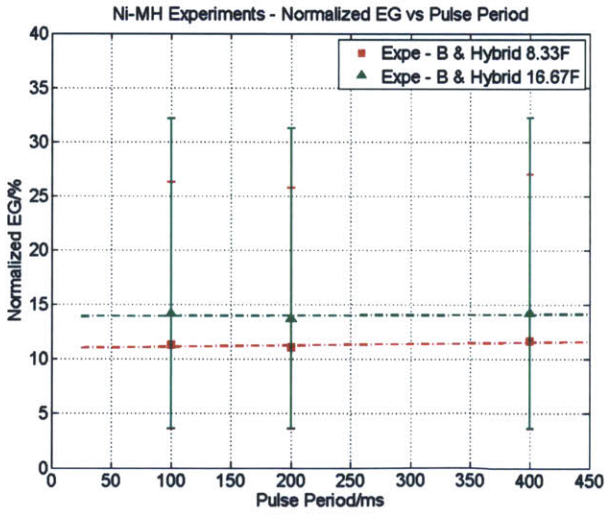
Fig. 6- 5 Normalized energy gain decreases linearly as the duty cycle increases (Pulse amplitude 8A, pulse period 200ms)

Fig. 6-6 displays the relationship between $\delta\emptyset$ and the pulse period. It is not very implicit from the experimental data but the simulation shows as the pulse period increases, $\delta\emptyset$ decreases. It makes sense because considering the extreme case of an infinite pulse period, the energy storage systems work under a constant load, then the ultracapactiors will have no benefits and $\delta\emptyset$ will decrease to zero.



(a)Comparison of simulations and experimental results (Li-ion & 8.33 F EDLC)





(b)More experimental results of Li-ion (c) Experimental results of Ni-MH

Fig. 6- 6 Normalized energy gain changes mildly with pulse period (Pulse duty cycle 10%, pulse amplitude 8 A)

Eventually, the improved performance by incorporating EDLCs in the system is a consequence of ultracapacitors' low serial resistance sharing current with the battery's serial resistance. It results in a lower battery rms current and a lower power dissipation in the battery, and thus a lower battery temperature and increased lifetime.

Hybrid systems with 16.667 F ultracapactors perform better than those with 8.333 F ultracapactors but the improvement is slight. One can see from the specifications (Table 3-1) that the ESR of 25 F ultracapacitor is 42 mΩ and it of 50 F EDLC is 20 mΩ, without that many differences. However the ESR of 5 F ultracapacitor is 170 mΩ. Thus the improvement between 1.667 F and 8.333 F hybrid systems is larger than the improvement between 8.333 F and 16.667 F systems. There were relatively few data taken under 1.667 F hybrid system, but an example could be seen in Fig. 6-1 (b) and Fig. 6-4 (b).

In addition, the improvement over Ni-MH system is larger than it of the Li-ion battery. This is because the Li-ion battery itself has a lower series resistance and even comparable to that of ultracapacitors. Thus the current sharing benefits of ultracapacitors are not as effective as for the Ni-MH batteries.

Chapter 7 Conclusions and Future Works

The desire to prolong batteries' runtime and their discharge efficiency in pulse load applications has stimulated the development of battery-ultracapacitor hybrid energy storage systems (ESS). In the hybrid systems, the high power density of ultracapacitors and the higher energy density of batteries complement each other, forming a promising energy storage system. Many research have been done on studying the hybrid system with a power electronics interface, however few research have thoroughly studied the benefits of battery-ultracapacitor hybrid system.

The purpose of this thesis is to assess the benefits of the battery-ultracapacitor parallel system over the battery-only system in pulse load applications. The benefits that this investigation focused on are energy savings, i.e., how much more energy one can draw out from the hybrid system versus the battery-only system.

7.1 Metrics

To quantify the benefits of the hybrid system, the metric *discharge capability* $\emptyset$ was defined in Chapter 3 as a proxy indicator of energy, i.e., how much energy one can get out from an energy storage system. Another metric *normalized energy gain* $\delta\emptyset$ was defined to compare two discharge capabilities. It is a percentage value and indicates how much more energy one system can provide to the load than another.

$\emptyset$ is calculated as the energy an ESS output to the load per unit charge draw out from the ESS. It has the unit of voltage and is a function of the state-of-charge of the ESS. All batteries' datasheets have their voltage-SoC curves under constant rate discharge, and the $\emptyset - SoC$ curves could be seen as a similar curve under pulse load discharge.

There is a potential to use these metrics to assess any energy storage systems under all kinds of load applications as long as they meet the following conditions: the source-load system is charge conserved and energy conserved.

7.2 Methodology

The investigation has been conducted in both hardware experiments and simulations, along six degrees of freedom: pulse amplitude, duty cycle, pulse period, capacitance, battery type and state-of-charge of the batteries.

An experimental apparatus was designed and described in Chapter 4. A custom PCB was built and fulfills charging and discharging of two kinds of batteries, as well as measuring and collecting data. This test rig was used to run each test case, and the fore-mentioned metrics were calculated based on the measured data.

Circuit models of ultracapacitors and Li-ion batteris were extracted and simulated in Chapter 5. The ultracapacitor model simulates the transient performance and the non-linear capacitance effect of the ultracapacitor; the battery model simulates both transient and long-term performance of the battery. Two models were then combined to create a hybrid ESS model, which was used in simulations of the same test cases from the experiments. The models were then used to sweep along several degrees of freedom to predict the benefits of the hybrid system.

7.3 Results

The results from both experiments and simulations show that the hybrid system could output more energy over the battery-only system at a specific final state-of-charge level. In the limited test cases of this investigation, it shows that the increase of Li-ion battery systems can reach to 17% and the increase of Ni-MH battery systems can reach to 33%.

The benefits of paralleling ultracapactors across batteries depended upon the discharge profile of the load, the battery type and the capacitance. The *normalized energy gain* $\delta\emptyset$ increases quadratically with the pulse amplitude, decreases linearly with the duty cycle and inverse with the pulse period. Thus the batteries benefit more from ultracapacitors when used under higher pulse current, smaller duty cycle and shorter pulse period. Moreover, higher capacitance yields to higher benefits, but more importantly, if the ESR of the ultracapacitors is lower, the benefits are more significant. Similarly, for batteries with a high ESR, the ultracapacitors will significantly increase its dischargeable energy.

There are also benefits in terms of volume, temperature and complexity. According to the volume information in Table 3-1, the volume of one-cell Ni-MH battery equals to that of 3.3 cells 25 F ultracapacitors, and the volume of one-cell Li-ion battery equals to that of 6.8 cell 25 F ultracapacitors. Considering the increase of 33 % for the Ni-MH battery, the energy density per volume of the Ni-MH batteries at the specific SoC is actually increased by paralleling ultracapacitors.

7.4 Future Works

First, the accuracy of the Li-ion-ultracapacitor model needs to be further improved. From Fig. 5-9, one can see that the battery voltage in the hybrid system simulation could not go as low as the experimental battery voltage during pulsing. This also results in errors when calculating $\emptyset$ and $\delta\emptyset$ of the hybrid system. The reasons are analyzed in Section 5.3 and two more accurate set of RC parameters for batteries and for ultrcapacitors needs to be developed. One more branch of RC pair for ultracapacitor featuring second and minute time range could be added. Furthermore, modeling of Ni-MH batteries and 50 F ultracapacitors is recommended to be done to improve the integrity of the research.

Second, as mentioned previously, $\emptyset$ and $\delta\emptyset$ depend on the state-of-charge of the batteries. In this investigation, only a 99% SoC was selected as a section to evaluate $\emptyset$ and $\delta\emptyset$. More experiments under different SoC values are recommended to do. The author would suggest SoC levels of 85%, 70%, 50% and 30% to be selected, as shown in Fig. 7-1. With enough experiments conducted, one can get a sense of the performance of the battery or the hybrid system throughout its state-of-charge. Overall $\emptyset - q$ curves and $\delta\emptyset - q$ curves could be obtained by interpolating $\emptyset$ and $\delta\emptyset$ at different SoCs. Furthermore, because the batteries' parameters and ultracapacitors' parameters are highly dependent on SoC, if simulations need to be conducted under different SoC, the corresponding parameters should be extracted following the same procedure in Chapter 5.

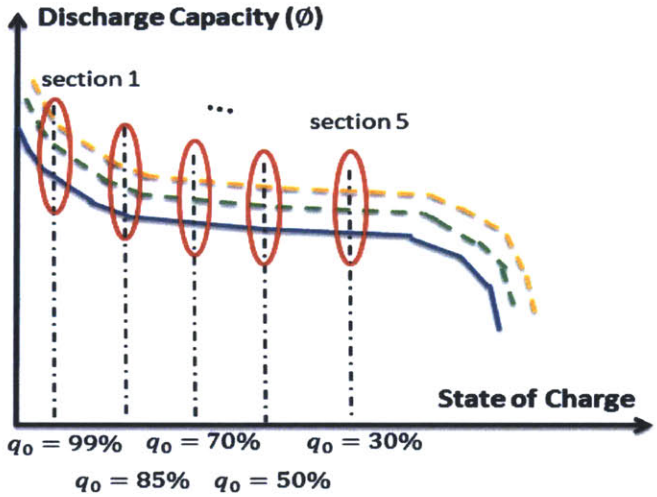


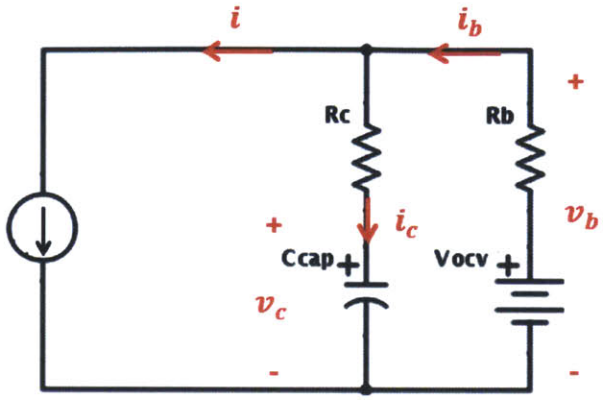
Fig. 7- 1 Extend the investigation to different SoC level

Last but not the least, the assessment could be extended to much higher power level and higher frequency level, such as hybrid vehicles, trucks or pulse power applications. As the results in this thesis indicate, the benefits of the hybrid system will be larger as the current goes higher and the pulse frequency goes higher, which points out a promising direction of incorporating ultracapacitors.

Appendix 1 Theoretical analysis

This appendix gives details of theoretical analysis in Section 3.3.

$$i(t) = \begin{cases} I, & 0 \le t \le DT \\ 0, & t \ge DT \end{cases} \tag{A1-1}$$



Assume the initial capacitor voltage is V_0, in the first period $0 \le t \le T$, the capacitor voltage is

$$v_c(t) = \begin{cases} (V_0 + IR_b - V_{oc})e^{-\frac{t}{\tau_{eq}}} + V_{oc} - IR_b, & 0 \le t \le DT \\ ((V_0 + IR_b - V_{oc})e^{-\frac{DT}{\tau_{eq}}} - IR_b)e^{-\frac{t-DT}{\tau_{eq}}} + V_{oc}, & DT \le t \le T \end{cases} \tag{A1-2}$$

Where $\tau_{eq} = C(R_b + R_c)$. Similarly, write several more periods, one could conclude the following rules, for the period $nT \le t \le (n+1)T$, the capacitor voltage is

$$v_c(t) = \begin{cases} A_n e^{-\frac{t-nT}{\tau_{eq}}} + V_{oc} - IR_b, & nT \le t \le nT + DT \\ B_n e^{-\frac{t-(n+D)T}{\tau_{eq}}} + V_{oc}, & nT + DT \le t \le (n+1)T \end{cases} \tag{A1-3}$$

Where

$$\begin{aligned} A_0 = 1, A_{n+1} &= A_n e^{-\frac{T}{\tau_{eq}}} - e^{-\frac{(1-D)T}{\tau_{eq}}} + 1 \\ B_n &= A_n e^{-\frac{DT}{\tau_{eq}}} - 1, \end{aligned} \tag{A1-4}$$

And if the time is long enough, the system enters steady state, n goes to ifinite, then

$$A_n \to \frac{e^{-\frac{(1-D)T}{\tau_{eq}}} - 1}{e^{-\frac{T}{\tau_{eq}}} - 1} \tag{A1-5}$$

$$B_n = e^{-\frac{DT}{\tau_{eq}}} \frac{e^{-\frac{(1-D)T}{\tau_{eq}}} - 1}{e^{-\frac{T}{\tau_{eq}}} - 1} - 1,$$

We can write the function of battery current as

$$i_b = i + i_c \to I - \frac{IR_b}{\tau_{eq}} \frac{e^{-\frac{(1-D)T}{\tau_{eq}}} - 1}{e^{-\frac{T}{\tau_{eq}}} - 1} e^{-\frac{t-nT}{\tau_{eq}}} \tag{A1-6}$$

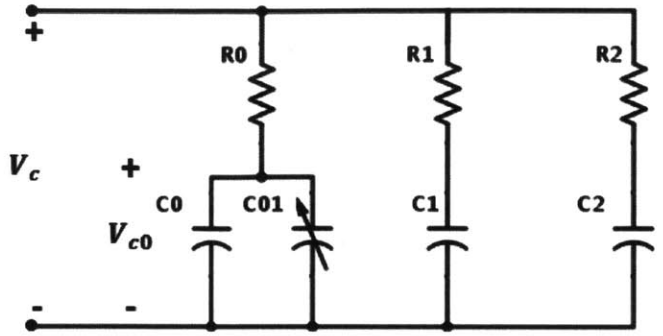
Then we have

$$\int_{nT}^{(n+D)T} i_b dt \to I - \frac{IR_b}{\tau_{eq}} \frac{e^{-\frac{(1-D)T}{\tau_{eq}}} - 1}{e^{-\frac{T}{\tau_{eq}}} - 1} (1 - e^{-\frac{DT}{\tau_{eq}}}) \tag{A1-7}$$

This was used in (11) to calculate $\emptyset$ and $\delta\emptyset$.

Appendix 2 Differential equation of the ultracapacitor model

This appendix gives details of the deduction process of the nonlinear differential equation describing the ultracapacitors.



Define Q_0 is the charge in the capacitors of branch #0 (has the resistor R_0), Q_1 is the charge in the capacitor of branch #1 (has the resistor R_1), etc. Thus

$$\begin{aligned}
&\frac{dQ_0}{dt}+\frac{dQ_1}{dt}+\frac{dQ_2}{dt}+\frac{dQ_3}{dt}=i_c(t) \\
&\frac{1}{R_1}\left(\frac{dQ_0}{dt}R_0+v_0-v_1\right)=\frac{dQ_1}{dt} \\
&\frac{1}{R_2}\left(\frac{dQ_0}{dt}R_0+v_0-v_2\right)=\frac{dQ_2}{dt} \\
&\frac{1}{R_3}\left(\frac{dQ_0}{dt}R_0+v_0-v_3\right)=\frac{dQ_3}{dt}
\end{aligned} \tag{A2-1}$$

Where

$$\begin{aligned}
Q_0 &= (C_0+kv\times v_0)v_0 \\
Q_1 &= C_1v_1 \\
Q_2 &= C_2v_2 \\
Q_3 &= C_3v_3
\end{aligned}$$

Define

$$\gamma=\frac{R_0}{R_0}+\frac{R_0}{R_1}+\frac{R_0}{R_2}+\frac{R_0}{R_3} \tag{A2-2}$$

We have

$$\begin{bmatrix} \frac{dQ_0}{dt} \\ \frac{dQ_1}{dt} \\ \frac{dQ_2}{dt} \\ \frac{dQ_3}{dt} \end{bmatrix} = \begin{bmatrix} \frac{1}{R_0}\left(\frac{R_0}{R_0\gamma}-1\right) & \frac{1}{\gamma R_0}\frac{R_0}{R_1} & \frac{1}{\gamma R_0}\frac{R_0}{R_2} & \frac{1}{\gamma R_0}\frac{R_0}{R_3} \\ \frac{1}{\gamma R_1}\frac{R_0}{R_0} & \frac{1}{R_1}\left(\frac{R_0}{R_1\gamma}-1\right) & \frac{1}{\gamma R_1}\frac{R_0}{R_2} & \frac{1}{\gamma R_1}\frac{R_0}{R_3} \\ \frac{1}{\gamma R_2}\frac{R_0}{R_0} & \frac{1}{\gamma R_2}\frac{R_0}{R_1} & \frac{1}{R_2}\left(\frac{R_0}{R_2\gamma}-1\right) & \frac{1}{\gamma R_2}\frac{R_0}{R_3} \\ \frac{1}{\gamma R_3}\frac{R_0}{R_0} & \frac{1}{\gamma R_3}\frac{R_0}{R_1} & \frac{1}{\gamma R_3}\frac{R_0}{R_2} & \frac{1}{R_3}\left(\frac{R_0}{R_3\gamma}-1\right) \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix} + \begin{bmatrix} \frac{R_0}{R_0\gamma} \\ \frac{R_0}{R_1\gamma} \\ \frac{R_0}{R_2\gamma} \\ \frac{R_0}{R_3\gamma} \end{bmatrix} i_c(t) \tag{A2-3}$$

Define

$$M_{qv} = \begin{bmatrix} \frac{1}{R_0}\left(\frac{R_0}{R_0\gamma}-1\right) & \frac{1}{\gamma R_0}\frac{R_0}{R_1} & \frac{1}{\gamma R_0}\frac{R_0}{R_2} & \frac{1}{\gamma R_0}\frac{R_0}{R_3} \\ \frac{1}{\gamma R_1}\frac{R_0}{R_0} & \frac{1}{R_1}\left(\frac{R_0}{R_1\gamma}-1\right) & \frac{1}{\gamma R_1}\frac{R_0}{R_2} & \frac{1}{\gamma R_1}\frac{R_0}{R_3} \\ \frac{1}{\gamma R_2}\frac{R_0}{R_0} & \frac{1}{\gamma R_2}\frac{R_0}{R_1} & \frac{1}{R_2}\left(\frac{R_0}{R_2\gamma}-1\right) & \frac{1}{\gamma R_2}\frac{R_0}{R_3} \\ \frac{1}{\gamma R_3}\frac{R_0}{R_0} & \frac{1}{\gamma R_3}\frac{R_0}{R_1} & \frac{1}{\gamma R_3}\frac{R_0}{R_2} & \frac{1}{R_3}\left(\frac{R_0}{R_3\gamma}-1\right) \end{bmatrix} \tag{A2-4}$$

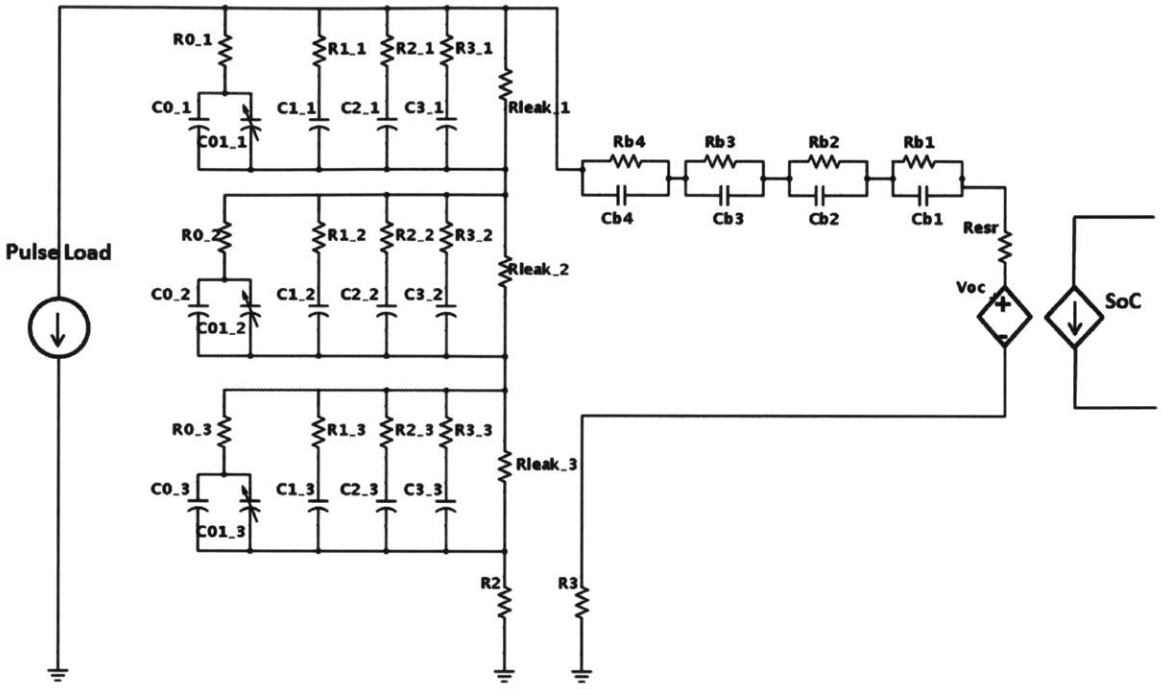
$$M_{qi} = \begin{bmatrix} \frac{R_0}{R_0\gamma} \\ \frac{R_0}{R_1\gamma} \\ \frac{R_0}{R_2\gamma} \\ \frac{R_0}{R_3\gamma} \end{bmatrix}, v_c = \begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

The differential function of ultracapacitor is

$$\dot{Q}_c = M_{qv} v_c + M_{qi} i_c \tag{A2-5}$$

Appendix 3 Differential equation for the hybrid system

This appendix gives details of the deduction process of the nonlinear differential equation describing the hybrid system. The process is similar with deducing the model of ultracapacitors.



Define m is the number of ultracapacitors in series. Assume each ultracapacitor is identical. Thus define Q_0 is the charge in the capacitors of branch #0 (has the resistor R_0), Q_1 is the charge in the capacitor of branch #1 (has the resistor R_1), etc. Same with appendix 3, we have

$$\dot{Q}_c = M_{qv} v_c + M_{qi} i_c \tag{A3-1}$$

For battery, we have

$$\dot{Q}_b = M_{bv} v_b + M_{bi} i_b \tag{A3-2}$$

Where

$$Q_b = \begin{bmatrix} Q_{cb1} \\ Q_{cb2} \\ Q_{cb3} \\ Q_{cb4} \end{bmatrix}, v_b = \begin{bmatrix} v_{b1} \\ v_{b2} \\ v_{b3} \\ v_{b4} \end{bmatrix} \tag{A3-3}$$

$$M_{bv} = diag(\frac{1}{R_{b1}}, \frac{1}{R_{b2}}, \frac{1}{R_{b3}}, \frac{1}{R_{b4}})$$

$$M_{bi} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Then use KCL and KVL, we have

$$\begin{gathered} V_{oc} - i_b(R_{esr} + R_3) - M_{bi}{}' v_b = m[(M_{qv}(1,:)v_c + M_{qi}(1)i_c)R_0 + v_0] + i_c R_2 \\ i_b = i + i_c \end{gathered} \tag{A3-4}$$

Then we have

$$i_b = \frac{[V_{oc} + (mM_{qi}(1) + R_2)i - \{M_{bi}{}', m[(M_{qv}(1,:)R_0 + [1,0,0,0]\}v_x]}{R_{esr} + R_3 + R_2 + mM_{qi}(1)} \tag{A3-5}$$

Where

$$v_x = [v_b; v_c]$$

Define

$$\alpha = \frac{1}{R_{esr} + R_3 + R_2 + mM_{qi}(1)} \tag{A3-6}$$

$$\begin{gathered} \beta = \alpha(mM_{qi}(1) + R_2) \\ A = \alpha\{M_{bi}{}', m[(M_{qv}(1,:)R_0 + [1,0,0,0]\} \end{gathered}$$

We have

$$i_b = \alpha V_{oc} + \beta i - A v_x \tag{A3-7}$$

Combine battery and ultracapacitor together, we have

$$M_x = \begin{bmatrix} M_{bv} & 0 \\ 0 & M_{qv} \end{bmatrix} - \begin{bmatrix} M_{bi} \\ M_{qi} \end{bmatrix} A \tag{A3-8}$$

$$M_v = \alpha \begin{bmatrix} M_{bi} \\ M_{qi} \end{bmatrix}$$

$$M_i = \begin{bmatrix} \beta M_{bi} \\ (\beta - 1) M_{qi} \end{bmatrix}$$

The differential function of the hybrid system is

$$\dot{Q}_x = M_x v_x + M_v V_{oc} + M_i i \tag{A3-9}$$

Appendix 4 Programs on the microcontroller

This appendix contains 3 c code for programming ATxmega192A3U, as listed below:

- Master_LiIonBoard_LiIonB_YHv1.c – the main file of microcontroller to charge, discharge batteries and collect data. One can specify battery type, pulse amplitude, duty cycle, etc in the file.
- ATXmega192A3U_init.h – head file for the initial file below
- ATXmega192A3U_init.c – set initial conditions for all units in ATxmega192A3U, including timer, ADC, UART, interrupt, etc.

Master_LiIonBoard_LiIonB_YHv1.c

```
/*
 * Master_LiIonBoard_LiIonB_YHv1.c
 *
 * Created: 1/15/2014 1:30:42 PM
 *  Author: Yiou
 * For charging and experimenting on LiIon battery, control part is changed.
 */

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdio.h>
#include <math.h>
#include "ATXmega192A3U_init.h"

//----------------------
//Function Definitions
void init();
int setCurrent(double,int);

void pulse_cycles(double, unsigned int);
void rest_cycles(unsigned int);
void chargeNH();
void chargeLI();
void wait_then_discharge();
int tempslope();
int voltdrop();
int interpret(unsigned int); //want it to be signed

//----------------------
//Variable Definitions
int ISRflag = 0; //To help read ADC properly
//ADC Variables
unsigned int ADCDATA[8];

/*--------
state_int = 3 - Charge
          5 - Wait until discharge
          2 - Pulse
```

```
          1 - Rest
--------*/

/*----------------------
  ADC CHANNELS
  1 - TEMP - Temperature of Battery (INCLUDE CONVERSION FORMULA)
  2 - ICH - Charging current, should consider sign (IR3)
  3 - IDIS - Dishcharging current, should consider sign (IR1)
  4 - ICAP - Capacitor current, should consider sign (IR2)
  5 - VCAPACITOR - Voltage across cap.
  6 - VBATTERY- Voltage across battery.
  7 - VBATT - Voltage at VB
  8 - VCAP - Voltage at VCap
----------------------*/

//Start Charging current, in 0.1A
float ChargeCurrent = 3.02;  //1.6A for NH battery, 3A for LI battery, 0.07 is offset
float prechargecurrent = 0.3;
float prevoltage = 4; //4;  //2V*2, 2V is recommended discharge cut off voltage,
assume this is deeply discharge voltage
int BatteryType = 2;  //indicate what kind of battery it is, 1 for Ni-MH, 2 for
Li-ion;

//Charging Termination
//NI-MH: Battery Temperature, for protection and terminal detection
float inttemp;  //sampled from ADC
float prevtemp = 0; //prevtemp records the last one
//NI-MH: Battery Voltage, for protection and terminal detection
double intvol; //sampled from ADC
double maxNHvol = 0;  //For Ni-MH battery, compare the battery voltage drop
//LI-ION: Battery Voltage
const float cutoffvol = 7.3; //For Li-ion battery, norminal cutoff voltage, 3.6*2
= 7.2
const float maxLivol = 7.2; //For Li-ion battery, norminal recommended charge
voltage
const float SlowDownLIvol = 7.12; //Start to slow down charging.3.56*2 =
const float MaxDiscrementVol = 5e-3;  //if voltage is not close to 3.56, than
increase current
const float HighDiscrementVol = 5e-4; //if incease in votage is higher than this,
deltaI doubles
```

```
const float LowDiscrementVol = 5e-6; //if incease in votage is lower than this,
deltaI half
//LI-ION: Battery current, for charging cut-off
float intcur; //sampled from ADC
double deltaI = 2e-4;
const float orideltaI = 2e-4;  //need for turning to CV, back to original current
decrease
const float maxdeltaI = 8e-3;
const float mindeltaI = 5e-6;
const float recoHighCHcur = 3.0; //Standard CC, highest charging current (charging
circuit has offset of around 0.02A) 0.07 is offset
const float recoLowCHcur = 2.0;  //Standard CC, lowest charging current 0.07 is
offset
const float recoMinCHcur = 0.02; //For CV, to prevent current has big changes,
because dac has offset, the best it can do is 0.152A.
const float cutoffcur = 0.05;  //Cut off current, measurement, recommendation is
0.05, make it bigger to see

//Discharging termination
const float discharge_cutoffvol_Li = 4.0; //Lowest voltage for Li-ion battery
const float discharge_cutoffvol_NiMH = 5.0; //For depleting discharge of Ni-MH
battery

//counters, for charging and resting control
long int charge_cnt_loop1 = 0;  //check voltage
long int charge_cnt_loop2 = 0;  //check temp
long int relaxcnt = 0;  //count rest time, cool down;
int relaxcnt2 = 0; //count rest time

//Pulse information
const unsigned int PULSEMAX = 2000; //pulse 100 and then stop, DISCHARGE 10000
FOR COMPARE CHARGING EXPERIMENT
unsigned long pulsecount = 0;  //count pulse number
const unsigned int Period = 50;   //pulse T = 400ms, 50ms
const float DC = 10;       //duty cycle 10%, 50%
int discharge_flag = 1;  //discharge_flag = 1, pulse count; discharge_flag = 2,
depletion discharge to cutoff voltage
int discharge_stopflag = 0; //discharge_stopflag = 1, stop
const float Amp = 8.0;

//every sample is 1 ms, make period the multiplers of 1 ms
```

```
long pulse_length;   //how many points in pulse
long pulse_cnt = 0;      //count how many 1ms in a pulse
long rest_length;    //how many points in rest
long rest_cnt = 0;       //count how many 1ms in a rest

int loop_cycles = 1;                 //Keep track of the number of charge/discharge
cycles we're doing.

//Sample and Packet variables
long record_number = 0;
int sendnum;  //sending data during rest follow sequence, from 1
int record_flag = 1;  //Set this number to 1 to send packets.  0 sends no packets.
always 1, consider get rid of record_flag
int chargesend = 0;   //Set this as flag to send data during charging, 100ms once

//State Variables
char next_state;
int state_int;
int LIcharge_state_int;  //6 is CC charging, 7 is CV charging
char LIcharge_next_state; //start with CC charge

//Main Program
//----------------------

void main(void) __attribute__((noreturn));  // more efficient form of int main()
void main(void)
{

    init();    //Setup IO pins and defaults
    setCurrent(0,3); //change with 'next_state'
    LIcharge_next_state = 'C';  //should be placed here, because there will be delay
when writing LIcharge_next_state if being put after next_state.
    LIcharge_state_int = 6;

    //wait 10s before start charging
    _delay_ms(1000);
    next_state = 'C';   //'P' is discharge, 'C' is charge

    //Start the logging timer, use timer0.
    startlogging();
    while(1)
```

```
    {

    }
}

void init()
{
    sysclk_init();
    io_init();
    tc_init();
    usart_init();
    dac_init();
    ISRflag = 1;

    //interrupt_init
    CPU_SREG |= CPU_I_bm; //Set GIE
    PMIC.CTRL |= (PMIC_HILVLEN_bm | PMIC_MEDLVLEN_bm); //enable low level interrupt

    //Set initial value
    record_number = 0;  //count from 0x00 - 0xFF
    pulse_length = Period * DC / 100 - 1;   //how many points in pulse
    rest_length = Period * (100 - DC) / 100 - 1;    //how many points in rest
}

void startlogging(char next_state)
{//This function will set the count length of Timer0, Let it work, sampling events
every 1ms
    TCC0_PER = 0x3980;
}

//Interrupt Functions:
ISR(TCC0_OVF_vect)
{
    PORTC_OUT |= PIN3_bm; //CONVT = 1, rising edge trigger ADC to work
    PORTC_OUT &= 0xF7;
    //PORTD_OUTTGL = 0x40; //Toggle Pin 6;
    //PORTD_OUTTGL = 0x00; //Toggle back Pin 6;ADD delay time
    //change status
    switch(next_state)
    {
        case 'C':  //This means we now want to charge.
```

```
        record_flag = 1;
        state_int = 3;
        if (BatteryType == 1) chargeNH();
        else if (BatteryType == 2) chargeLI();
        break;
        case 'L':
        record_flag = 0;
        state_int = 5;
        setCurrent(0,state_int);
        prevtemp = 0;
        wait_then_discharge();
        break;
        case 'P':
        record_flag = 1;
        state_int = 2;
        prevtemp = 0;
        pulse_cycles(Amp,pulse_length); //8A;
        break;
        case 'R':
        record_flag = 1;
        state_int = 1;
        prevtemp = 0;
        rest_cycles(rest_length); //rest multipliers number of 1 ms
        break;
        default:
        state_int = 0;
        prevtemp = 0;
        setCurrent(0,state_int);
        next_state = 'O';
        record_number = 0;  //put record_number = 0 here.
        record_flag = 0;
        break;
    }
}

//read ADC
ISR(PORTC_INT0_vect)
{
    //
    //PORTD_OUTTGL = 0x40; //Toggle Pin 6;
    //PORTD_OUTTGL = 0x00; //Toggle back Pin 6;ADD delay time
```

```
    if (record_flag == 1)
    {
        if (ISRflag == 1)
        {
            PORTC_OUT &= 0xFD;  //should read one first, PORTE&F_IN are all 0.
            PORTC_OUT |= PIN1_bm;
            ISRflag = 0;
        }
        else
        {
            //PORTD_OUT |= PIN6_bm;
            if (state_int == 3)
            {
                if (chargesend == 1)   //During charging, send data 100ms once.
                {
                    sendUART(record_number);
                    if (BatteryType == 2 && state_int == 3)
sendUART(LIcharge_state_int);
                    else sendUART(state_int);
                    record_number ++;
                }
            }/*
            else if (state_int == 5)
            {
                if (chargesend == 1)   //During charging, send data 100ms once.
                {
                    sendUART(record_number);
                    sendUART(state_int);
                    record_number ++;
                }
            }*/
            else
            {
                sendUART(record_number);
                sendUART(state_int);
                record_number ++;
            }

            for (int i=0;i<=7;i++)  //read 8 values
            {
```

```
                PORTC_OUT &= 0xFD;
                _delay_us(0.03); //30ns(*8 = 0.24us)
                ADCDATA[i] = PORTE_IN * 256 + PORTF_IN; //together read 16 bits,
still not in right order.
                if (state_int == 3)
                {
                    if (chargesend == 1)   //During charging, send data 100ms once.
                    {
                        sendUART(PORTE_IN);
                        sendUART(PORTF_IN);
                    }
                }/*
                else if (state_int == 5)
                {
                    if (chargesend == 1)   //During charging, send data 100ms once.
                    {
                        sendUART(PORTE_IN);
                        sendUART(PORTF_IN);
                    }
                }*/
                else
                {
                    sendUART(PORTE_IN);
                    sendUART(PORTF_IN);
                }
                PORTC_OUT |= PIN1_bm;
            }
            //PORTD_OUT &= 0b10111111;
            inttemp = (float)(interpret(ADCDATA[0]))/32768*5*1000; //translate to
TEMP * 10
            intvol = (float)interpret(ADCDATA[5])/32768*5*3;  //translate Battery
Voltage, unit: Volt
            intcur = (float)interpret(ADCDATA[1])/32768*5/4.984/0.01;
//translate to battery current, unit: Amp
        }
    }
    else   //only save in ADCDATA but not use UART to send data.
    {
        if (ISRflag == 1)
        {
            PORTC_OUT &= 0xFD;  //should read one first, PORTE&F_IN are all 0.
```

```
            PORTC_OUT |= PIN1_bm;
            ISRflag = 0;
        }
        else
        {
            for (int i=0;i<=7;i++)  //read 8 values
            {
                PORTC_OUT &= 0xFD;
                _delay_us(0.03); //30ns(*8 = 0.24us)
                ADCDATA[i] = PORTE_IN * 256 + PORTF_IN; //together read 16 bits,
still not in right order.
                PORTC_OUT |= PIN1_bm;
            }
            inttemp = (float)(interpret(ADCDATA[0]))/32768*5*1000; //translate
TEMP
            intvol = (float)interpret(ADCDATA[5])/32768*5*3;  //translate Battery
Voltage, because too small, so don't divide, use original value
            intcur = (float)interpret(ADCDATA[1])/32768*5/4.984/0.01;
        }
    }

}

void pulse_cycles(double I, unsigned int length)
{//This function will set the proper state variables for pulsing the current.
  //Sets the current to I amps.
  //Sets the next state to R if pulse period ends

  //Set the current
  setCurrent(I,2);

  if (pulse_cnt >= length)
  {
      pulse_cnt = 0;
      pulsecount++;  //Increment the pulsecount counter
      next_state = 'R';
  }
  else
  {
      pulse_cnt ++;
      next_state = 'P';
```

```
  }

        // judge whether to stop discharge or not
        if (BatteryType == 1) //NiMH
        {
            if (intvol > discharge_cutoffvol_NiMH) discharge_stopflag = 0;
            else
            {
                discharge_stopflag = 1;
                next_state = 'O';
            }
        }
        else if (BatteryType == 2) //Li
        {
            if (intvol > discharge_cutoffvol_Li) discharge_stopflag = 0;
            else
            {
                discharge_stopflag = 1;
                next_state = 'O';
            }
        }
}

void rest_cycles(unsigned int length)   //rest with cycles, how many cycles, every
cyc 1ms
{//This function will rest the battery at 0 current.
//As a state, it will set the following parameters:
  //Check the pulse count. If it's >= the PULSEMAX then next state = "O"
  //If the pulse count < PULSEMAX, then next state = "P"

  //Set the current to 0
  setCurrent(0,1); //Sets the current to 0.

  if (rest_cnt != length)
  {
      rest_cnt ++;
      next_state = 'R';
  }
  else
  {
      rest_cnt = 0;
```

```
    //Set the next state, depending on the pulsecount or depletion
        if (discharge_flag == 1)
        {
            if(pulsecount < PULSEMAX) next_state = 'P';
            else next_state = 'O';
        }
        else if (discharge_flag == 2)
        {
            if (discharge_stopflag == 0) next_state = 'P';
            else next_state = 'O';
        }
  }
}

void chargeNH()
{//This function will charge the Ni-MH battery at given current value, ususally
1.6Amps (< 0.5C = 1.75A).
//It will also check the temperature every minute.  If the temperature rises >1
degree C/minute, then tempslope() returns 0, and we stop charging.

  setCurrent(ChargeCurrent, 3);  //charging current

  //Hard rules: if temp is over 40, turns to "L"
  if (inttemp >= 400)
  {
      next_state = 'L';
      return;
  }


  //check voltage drop, 8mV per cell
  if (!voltdrop())
  {
      //False means we're done charging - Cool Down
      next_state = 'L';
      return;
  }

  //decrease charge_cnt_loop1 every 101 rounds ~ 100 ms
  if (charge_cnt_loop1 == 0)
  {
```

```
      chargesend = 1;
      charge_cnt_loop1 = 100;
      charge_cnt_loop2 ++;
  }
  else
  {
      charge_cnt_loop1 --;
      chargesend = 0;
  }

  //check the tempslope every min
  if (charge_cnt_loop2 == 601) //0.1*(601 - 1) = 1min, accurately.
  {
    //No matter what, make sure we reset the charge_count
    charge_cnt_loop2 = 0;
    //Check the temperature slope after every minute.
    if(!tempslope())
    {
      //False means we're done charging - Cool Down
      next_state = 'L';
      return;
    }
  }
  //True, or not time to check temp, means everything's fine - continue.
  next_state = 'C';
}

int tempslope()
{//This function will check the slope of the battery temperature.
    //Note:  This function must be called only ONCE PER MINUTE!!
    //As a state, it will set the following parameters:
    //If tempslope() returns true when called, then next state = "C"
    //If tempslope() returns false when called, then next state = "O"
    //If tempslope is not called, then wait for charge_count and set next state
= "C"

    //If the temp is raising at least 1C per minute STOP CHARGING
    if ( ((inttemp - prevtemp) >= 10) && (prevtemp != 0) ) return 0;  //sometimes
10 is too insensitive....><

    //Update the prevtemp value with the current value
```

```
    prevtemp = inttemp;
    return 1;
}

int voltdrop()
{
    //check voltage drop, delta v = 8mV * 5 = 40 mV (40mV/3/5*32768 = 87)
    if (intvol >= maxNHvol)
    {
        maxNHvol = intvol;
        return 1;
    }
    else if (((maxNHvol - intvol) >= 87) & (maxNHvol != 0)) return 0;
}

void chargeLI( )
{//This function will charge the Li-Ion battery: CC at 3A - CV when current goes
down to 0.05A.
//It will also check the temperature every minute.

  //Hard rules: if temp is over 40, turns to "L"
  if (inttemp >= 400)
  {
      next_state = 'L';
      return;
  }

  //Hard rules: if voltage is large than cutoffvol, do not charge anymore, directly
discharge
  if (intvol >= cutoffvol)
  {
      next_state = 'L';
      return;
  }

   //change current every 101 rounds ~ 100 ms
   if (charge_cnt_loop1 == 0)
   {
       charge_cnt_loop1 = 100;
       chargesend = 1;
       //in other cases, first CC and then CV
```

```
        switch(LIcharge_next_state)
        {
            case 'B':
            if (intvol >= prevoltage)
            {
                LIcharge_next_state = 'C';
                LIcharge_state_int = 6;
                return;
            }
            setCurrent(prechargecurrent,3);
            next_state = 'C';
            LIcharge_next_state = 'B';
            LIcharge_state_int = 8;
            break;
            case 'C':  //Constant Current Charging Stage
            //goes to CV if voltage reach required
            if (intvol >= maxLivol - HighDiscrementVol)
            {
                deltaI = orideltaI;
                ChargeCurrent = ChargeCurrent - deltaI;
                LIcharge_next_state = 'V';
                LIcharge_state_int = 7;
                charge_cnt_loop2 = 1;
                return;
            }
            //update ChargeCurrent
            if (ChargeCurrent > recoHighCHcur) ChargeCurrent = recoHighCHcur;
//protect, do not exceed recommended charging current;
            if (ChargeCurrent < recoLowCHcur) ChargeCurrent = recoLowCHcur;
//protect, do not exceed recommended charging current;
            if (intvol >= SlowDownLIvol) ChargeCurrent = ChargeCurrent - deltaI;

            setCurrent(ChargeCurrent,3);
            next_state = 'C';
            LIcharge_next_state = 'C';
            LIcharge_state_int = 6;

            //update delta I
            if (intvol != 0)
            {
                if (maxLivol - intvol <= MaxDiscrementVol)
```

```
            {
                deltaI = deltaI + orideltaI * 0.1;
                if (deltaI >= maxdeltaI) deltaI = maxdeltaI;
                else if (deltaI <= mindeltaI) deltaI = mindeltaI;
            }
            /*else if (maxLivol - intvol <= LowDiscrementVol)
            {
                deltaI = deltaI + orideltaI * 0.1;  //not by 0.5, but decrease
deltaI step by step
                if (deltaI >= maxdeltaI) deltaI = maxdeltaI;
            }*/
        }
        break;

        case 'V':  //Constant Voltage Charging Stage
        //Time rules, must < 30min
        if (charge_cnt_loop2 >= 18000)
        {
            charge_cnt_loop2 == 0;
            next_state = 'L';
            return;
        }
        else charge_cnt_loop2 ++;

        //Current rules, reach cutoff current
            if (intcur <= cutoffcur) //charge_cntT >= 3 && intcur <= cutoffcur)
            {
                next_state = 'L';
                return;
            }

            if (maxLivol - intvol >= MaxDiscrementVol)  //if voltage is not as
close enough to recommended charging voltage, then increase current
            {
                ChargeCurrent = ChargeCurrent + deltaI;
            }
            else ChargeCurrent = ChargeCurrent - deltaI;

            if (ChargeCurrent > recoHighCHcur) ChargeCurrent = recoHighCHcur;
//protect, do not exceed recommended charging current;
```

```
            if (ChargeCurrent < recoMinCHcur) ChargeCurrent = recoMinCHcur;
//protect, do not exceed recommended charging current;
            setCurrent(ChargeCurrent,3);
            next_state = 'C';
            LIcharge_next_state = 'V';
            LIcharge_state_int = 7;

            //update on deltaI, only update the decrease type, not for increase
            if ((intvol - maxLivol <= HighDiscrementVol) && (intvol >= maxLivol))
            {
                deltaI = deltaI + orideltaI * 0.1;
                if (deltaI >= maxdeltaI) deltaI = maxdeltaI;
                else if (deltaI <= mindeltaI) deltaI = mindeltaI;
            }
            else if ((intvol - maxLivol >= HighDiscrementVol) && (intvol >=
maxLivol))
            {
                deltaI = deltaI + orideltaI * 0.5;
                if (deltaI >= maxdeltaI) deltaI = maxdeltaI;
                else if (deltaI <= mindeltaI) deltaI = mindeltaI;
            }
            else if ((maxLivol - intvol <= HighDiscrementVol) && (intvol <=
maxLivol))
            {
                deltaI = deltaI - orideltaI * 0.1;
                if (deltaI >= maxdeltaI) deltaI = maxdeltaI;
                else if (deltaI <= mindeltaI) deltaI = mindeltaI;
            }
            else if ((maxLivol - intvol >= HighDiscrementVol) && (intvol <=
maxLivol))
            {
                deltaI = deltaI - orideltaI * 0.5;
                if (deltaI >= maxdeltaI) deltaI = maxdeltaI;
                else if (deltaI <= mindeltaI) deltaI = mindeltaI;
            }

        break;
        default:
        next_state = 'O';
        break;
    }
```

```
   }
   else
   {
       charge_cnt_loop1 --;
       chargesend = 0;
   }
}

void wait_then_discharge()
{//This function will keep the current off, and check the temperature.
//When the temperature is less than 25C, it will move on to the next discharge
cycle.

//This function is meant to be called only after charging.
//The equivalent after discharge is called wait_then_charge.
//Its state is 'L' for cooLdown

  PORTD_OUT |= PIN6_bm;  //light on the red LED
  if (charge_cnt_loop1 == 0)
  {
      charge_cnt_loop1 = 100;
      chargesend = 1;
  }
  else
  {
      charge_cnt_loop1 --;
      chargesend = 0;
  }

  //Set the next state, depending on the temperature and check every 30s
  if (BatteryType == 1)
  {
       if (relaxcnt == 10) //1000 = 1s; set this to ensure rest certain time.
       {
         relaxcnt = 0;
         if(inttemp > 250) next_state = 'L'; //if after 10 min, temp still > 25,
stop; here change to 260,just try
         //else if (intvol >= 7.14) next_state = 'L'; //7.2-0.02*3
         else
         {
              //The temperature has fallen to a reasonable level.
```

```
            //If we still have more cycles to do...
            if(loop_cycles > 0)
            {
                //Get ready to start the next discharge cycle
                next_state = 'P';
                PORTD_OUT &= 0b10111111;
                loop_cycles--;
                pulsecount = 0;
                return;
            }
            else
            //We're done.  Set the next state to Off.
            next_state = 'O';
          }
        }
      else
      {
         relaxcnt++;
         next_state = 'L';
      }
  }
  else
  {
       if (relaxcnt == 29999) //1000 = 1s; now 30s. set this to ensure rest certain
time.
       {
         relaxcnt = 0;
         relaxcnt2 ++;
         if(inttemp > 250) next_state = 'L'; //if after 10 min, temp still > 25,
stop; here change to 260,just try
         else if (intvol >= 7.1) next_state = 'L'; //wait 30 min.
         else
         {
             //The temperature has fallen to a reasonable level.
            //If we still have more cycles to do...
                if(loop_cycles > 0)
                {
                    //Get ready to start the next discharge cycle
                    next_state = 'P';
                    PORTD_OUT &= 0b10111111;
                    loop_cycles--;
```

```
                    pulsecount = 0;
                    return;
                }
                else
                //We're done.  Set the next state to Off.
                next_state = 'O';
            }
          }
        else
        {
           relaxcnt++;
           next_state = 'L';
        }
    }
}

int setCurrent(double I, int state_int) //I is mA level
{//This function will change the DAC value to make the current match the given
I.
  //Returns 1 if successful, 0 otherwise.
  //So make sure this is the value you actually want to set!

  int DACvalue;    //This is the value we will set the DAC to.
  int V0offset = 0.0133;  //CH0, need set extra offset, already calibrated.
  int V1offset = 0;  //CH1, don't need set extra offset, already calibrated.

  if (state_int == 3) //charging, disable discharge.
  {
      DACvalue = (I + 0.28) * 0.01 * 4.984/1.0*4096;  //0.28A, output 13.2mV
      DACB_CTRLB = 0;  //select CH0
      DACB_CTRLA &= 0xF7;
      DACB_CTRLA |= DAC_CH0EN_bm;
      DACB_CH0DATA = DACvalue;
  }
  else if (state_int == 2) //discharging, disable charge. Only during pulse.
  {
      DACvalue = (I * 0.01 * 4.984 - V1offset)/1.0*4096;
      DACB_CTRLB = 0x20; //select CH1
      DACB_CTRLA &= 0xFB;
      DACB_CTRLA |= DAC_CH1EN_bm;
      DACB_CH1DATA = DACvalue;
```

```
  }
  else  //other including rest and wait, shut down all charging/discharging, set
DATA to 0. Doesn't need to worry about not to 0. Because shut down the block already.
  {
      DACB_CTRLA &= 0xF3; //disable charging & dishcharging
      DACB_CTRLB = 0;
      DACB_CH0DATA = 0;
      DACB_CH1DATA = 0;
  }
}

//ADC read opposite direction, should reorder direction. Basically hex fenjie
int interpret(unsigned int hexnum)
{
    unsigned int BITScnt = sizeof(hexnum)*8;
    unsigned int reverse_answer = 0;

    int i;
    for (i = 0; i < BITScnt; i++)
    {
        if((hexnum & (1 << i)))
        {
            if (i < 8) reverse_answer |= 1 << (7 - i);  //bit 7 move to bit 0
            else reverse_answer |= 1 << ((BITScnt + 7) - i);  //bit 15 move to bit
8
        }
    }
    return reverse_answer;
}
```

ATXmega192A3U_init.h

```
/*
  define new header file for init.c, avoid multiple definition
*/
#ifndef _ATXmega192A3U_init_H
#define _ATXmega192A3U_init_H


//Declare Functions-------------------------------
void sysclk_init(void);
void event_init(void);
void tc_init(void);
void io_init(void);
void usart_init(void);
void dac_init(void);
int sendUART(unsigned char data);

#endif //_ATmega168_timers_H
```

ATXmega192A3U_init.c

```
/*=======================
  use for initial the device
  ====================*/

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdio.h>
#include <math.h>

#include "ATXmega192A3U_init.h"

void sysclk_init()
{
    OSC_XOSCCTRL = 0xCB;  //OSC_FRQRANGE_12TO16_gc; 12-16MHz
    OSC_CTRL = OSC_XOSCRDY_bm; //external clock
    do {_delay_ms(1000);}
    while (!(OSC_STATUS & OSC_XOSCRDY_bm)); //wait for stability

    /*OSC.CTRL = OSC_RC32MEN_bm; // 32MHz internal
    while (!(OSC.STATUS & OSC_RC32MRDY_bm));*/
    CCP = 0xD8;
    CLK_CTRL = 0x03; //CLK.CTRL = 0x03
    CCP = 0xD8;
    CLK_PSCTRL = 0x00; //no division

    /*OSC.DFLLCTRL = 0x00;
    DFLLRC32M.CTRL = 0x01;*/
}

void io_init()
{
    //CLKper4 output
    PORTCFG_CLKEVOUT = 0x02; //SET clk output to CLKper

    //PORTD:os1,0; UART;
    PORTD_DIR = 0xFB; //DIR Registers:  0 = input.  1 = output. //PD2 is input
    PORTD_OUT = (PIN3_bm | PIN0_bm); //OS1 = 1; TxD = 1; RxD = 1(pull-up resistor);
```

```
    //Control ADC
    PORTC_DIR = 0xFE; //OS2,CONVT,ResetADC,CS*&RD*,Busy
    PORTC_OUT = (PIN2_bm | PIN1_bm); //CS* = 1; RESETADC = 1;
    PORTC_PIN0CTRL |= PORT_ISC_FALLING_gc;  //Input sensing on PC0-BUSY
    PORTC_INTCTRL = 0x02; //Medium level 0x02;
    PORTC_INT0MASK = 0x01;  //pin 0 is used as interrupt 0.
    PORTC_OUTCLR |= PIN2_bm; //RESETADC = 0;
    PORTC_OUTCLR = 0; //Clear all the OUTCLR bits

    PORTC_OUT |= PIN3_bm; //CONVT = 1, rising edge trigger ADC to work
    PORTC_OUT &= 0xF7;


    PORTE_DIR = 0;
    PORTF_DIR = 0;

    //DA
    PORTB_DIR = 0xFF;
    //PORTB_OUT |= PIN1_bm;
}

void tc_init()
{
    //use for sampling and counting
    TCC0_CTRLA = TC_CLKSEL_DIV1_gc;  //Choose SYS clk
    TCC0_PER = 0; //0x3980 in startlogging();
    TCC0_INTCTRLA = 0x03; //High level
    TCC0_INTFLAGS = 0x01; // clear any initial interrupt flags

    //use for generating pulse to read ADC data. (maybe not used, time of one step
is enough)
    TCC1_CTRLA = TC_CLKSEL_DIV1_gc;
    TCC1_PER = 0x05;  //How long we want the Data to stay on the Parallel bus.
}

void usart_init()
{
    USARTD0_CTRLC = 0x03; //8bit
    USARTD0_BAUDCTRLA = 0x03; //7 - 230400; 3 - 460800;
    USARTD0_CTRLB = (USART_TXEN_bm | USART_CLK2X_bm); //Enable transmission AND
2 SPEED CLOCK.
```

```
    sendUART(0);
}

void dac_init()
{
    DACB_CTRLA = DAC_ENABLE_bm; //not enable any channel, JUST ENABLE DAC
    DACB_CTRLB = 0; //default channel 0
    DACB_CTRLC = 0; //this is 1.0V internal; choose AVcc if set as DAC_REFSEL0_bm;
    DACB_CH0DATA = 0x0;
    DACB_CH0OFFSETCAL = 0x24; //26,0x15
    DACB_CH0GAINCAL = 0x82;   //85,0x32
    DACB_CH1DATA = 0x0;
    DACB_CH1OFFSETCAL = 0x1f; //1F,0X0B
    DACB_CH1GAINCAL = 0x1a;   //1A,0X1A
}

int sendUART(unsigned char data)
{//This function will send 1 byte of data via the UART line.
    while((USARTD0_STATUS & USART_DREIF_bm) == 0) {}
    USARTD0_DATA = data;
    while((USARTD0_STATUS & USART_DREIF_bm) == 0) {}
    return 1;
}
```

Appendix 5 Matlab codes for processing experimental data

This appendix contains 3 MATLAB code files, as listed below:

- reverseturns.m – reverse the order of the UART data, because on the PCB layout, the pins of ADC and the pins of microcontroller are reversely connected.
- interpret.m – translate hex data into decimal
- DataAnalyze_BCAPcompare_Hex_v1.m – process UART data, plot current and voltage waveforms, calculate discharge capacity and normalized energy gain.

reverseturns.m

```
%TO reverse turns of hex data
function y = reverseturns(decdataE,decdataF)
i = 1;
E_bi = zeros(1,8);
F_bi = zeros(1,8);
while (decdataE > 0)
    E_bi(i) = mod(decdataE,2);
    decdataE = floor(decdataE/2);
    i = i+1;
end
%E_bi
i = 1;
while (decdataF > 0)
    F_bi(i) = mod(decdataF,2);
    decdataF = floor(decdataF/2);
    i = i+1;
end
%F_bi
ans_bi = [E_bi,F_bi];
y = 0;
for k = 1:1:16
    y = y + ans_bi(k)*2^(16-k);
end
%y
if (ans_bi(1) == 1)
    y = -(65536 - y);
end
end
```

Interpret.m

```
%interpret the hex file into right order, and into dec
function y = interpret(Edata,Fdata)
E = 16*hex2dec(Edata(:,1)) + hex2dec(Edata(:,2));
F = 16*hex2dec(Fdata(:,1)) + hex2dec(Fdata(:,2));
y = reverseturns(E,F)/32768*5;  %add ADC ratio
end
```

DataAnalyze_BCAPcompare_Hex_v1.m

```
%Read the UART data file, plot out waveforms and calculate out the discharge capacity
and normalized energy gain. efficiency of Bonly and cap experiments, using data
clc,clf,clear all,close all;
format long;

DateBID = '040714';
DateCAPID = '031114';
BatteryType = '_Li_';  %'_Li_' meanse Li-Ion battery;'_' means NIMH
IID = '16A';
PeriodID = '200ms';
DutyID = 'D20';
BID = 'C502000';
CAPID = 'C50';
BatteryID = '23';
CHDISID = 'DIS';

%Differential Amp Coefficiency
R1 = 0.01;R2 = 0.01;R3 = 0.01;
AMP1 = 4.984; AMP2 = 4.984;AMP3 = 4.984; %for data after Jan, 2014

CA_VBAT = 3;
CA_VCAP = 3;
CA_Vbattery = 3;
CA_Vcapacitor = 3;
CA_ICH = 1/(R3*AMP3);
CA_IDIS = 1/(R1*AMP1);
CA_ICAP = 1/(R2*AMP2);

%%
%calculating energy efficiency of battery only system
filename =
strcat('capture_',DateBID,BatteryType,IID,PeriodID,DutyID,BID,'_',BatteryID,'
_',CHDISID,'.txt');
savefilename =
strcat(DateBID,BatteryType,IID,PeriodID,DutyID,BID,'_',BatteryID,'_',CHDISID,
'.mat');

pulse_num = 0;
dt = 1/(14.7654*1e6)*14720;
```

```
EDIS = 0;
ECAP = 0;
ER2 = 0;
ER3 = 0;
Ebatt = 0;
current_integralBatt = 0;
current_integralDis = 0;
current_integralCap = 0;

%read data from file, need manually clean the data, clear the char part.
%Multiple calibration factor (?)
All_Data = textread(filename,'%2c');
[mm,nn] = size(All_Data);

i = 0;
k = 1;
for i = 1:1:mm
    res = mod(i,18);
    switch res
        case 1
            recordnums(k,:) = All_Data(i,:);
            cnt = k;
        case 2
            state(k,:) = All_Data(i,:);
            statenum(cnt) = state(k,2);
            if (cnt > 1)
                if (statenum(cnt) == '2') && (statenum(cnt-1) == '1')
                    pulse_num = pulse_num + 1;
                end
            end
        case 3
            TempE(k,:) = All_Data(i,:);
        case 4
            TempF(k,:) = All_Data(i,:);
            Temp(cnt) = interpret(TempE(k,:),TempF(k,:));
        case 5
            ICHE(k,:) = All_Data(i,:);
        case 6
            ICHF(k,:) = All_Data(i,:);
            ICH(cnt) = interpret(ICHE(k,:),ICHF(k,:));
        case 7
```

```
            IDISE(k,:) = All_Data(i,:);
        case 8
            IDISF(k,:) = All_Data(i,:);
            IDIS(cnt) = interpret(IDISE(k,:),IDISF(k,:));
        case 9
            ICAPE(k,:) = All_Data(i,:);
        case 10
            ICAPF(k,:) = All_Data(i,:);
            ICAP(cnt) = interpret(ICAPE(k,:),ICAPF(k,:));
        case 11
            VcapacitorE(k,:) = All_Data(i,:);
        case 12
            VcapacitorF(k,:) = All_Data(i,:);
            Vcapacitor(cnt) = interpret(VcapacitorE(k,:),VcapacitorF(k,:));
        case 13
            VbatteryE(k,:) = All_Data(i,:);
        case 14
            VbatteryF(k,:) = All_Data(i,:);
            Vbattery(cnt) = interpret(VbatteryE(k,:),VbatteryF(k,:));
        case 15
            VBATE(k,:) = All_Data(i,:);
        case 16
            VBATF(k,:) = All_Data(i,:);
            VBAT(cnt) = interpret(VBATE(k,:),VBATF(k,:));
        case 17
            VCAPE(k,:) = All_Data(i,:);
        case 0
            VCAPF(k,:) = All_Data(i,:);
            VCAP(cnt) = interpret(VCAPE(k,:),VCAPF(k,:));
            k = k + 1;
        otherwise
    end
end
pulse_num = pulse_num + 1;

%calculate efficiency from data
%scale data
startcnt = cnt/pulse_num*(pulse_num - 10) + 1; %startcnt is the start number of
last ten pulses
for kk = 1:(cnt - startcnt + 1)
    i = kk + startcnt - 1;
```

```
    VBAT_s(kk) = VBAT(i) * CA_VBAT;
    VCAP_s(kk) = VCAP(i) * CA_VCAP;
    ICH_s(kk) = ICH(i) * CA_ICH;
    IDIS_s(kk) = IDIS(i) * CA_IDIS;
    ICAP_s(kk) = ICAP(i) * CA_ICAP;
    ICAP_s(kk) = ICAP(i) * CA_ICAP;
    Vbattery_s(kk) = Vbattery(i) * CA_Vbattery;
    Vcapacitor_s(kk) = Vcapacitor(i) * CA_Vcapacitor;

    PDIS(kk) = VCAP_s(kk)*IDIS_s(kk);  %power dissipate on the discharge branch
    PCAP(kk) = VCAP_s(kk)*ICAP_s(kk);  %power dissipate on the cap branch
    PR2(kk) = R2*(ICAP_s(kk))^2;       %power dissipate on R2
    PR3(kk) = R3*(ICH_s(kk))^2;        %power dissipate on R3
    PowerBatt(kk) = Vbattery_s(kk)*ICH_s(kk); %power generate by battery
end

%Integrate LAST 10 PULSES to get Energy and Current
for i = 1:(cnt - startcnt)
    EDIS = EDIS + (PDIS(i) + PDIS(i+1))*dt/2;
    ECAP = ECAP + (PCAP(i) + PCAP(i+1))*dt/2;
    ER2 = ER2 + (PR2(i) + PR2(i+1))*dt/2;
    ER3 = ER3 + (PR3(i) + PR3(i+1))*dt/2;
    Ebatt = Ebatt + (PowerBatt(i) + PowerBatt(i+1))*dt/2;
    current_integralBatt = current_integralBatt + (ICH_s(i) + ICH_s(i+1))*dt/2;
    current_integralDis = current_integralDis + (IDIS_s(i) + IDIS_s(i+1))*dt/2;
    current_integralCap = current_integralCap + (ICAP_s(i) + ICAP_s(i+1))*dt/2;
end

EDIS = EDIS + (PDIS(1) + PDIS(cnt - startcnt + 1))*dt/2;
ECAP = ECAP + (PCAP(1) + PCAP(cnt - startcnt + 1))*dt/2;
ER2 = ER2 + (PR2(1) + PR2(cnt - startcnt + 1))*dt/2;
ER3 = ER3 + (PR3(1) + PR3(cnt - startcnt + 1))*dt/2;
Ediss = EDIS + ER2 + ER3;
Ebatt = Ebatt + (PowerBatt(1) + PowerBatt(cnt - startcnt + 1))*dt/2;
current_integralBatt = current_integralBatt + (ICH_s(1) + ICH_s(cnt - startcnt
+ 1))*dt/2;
current_integralDis = current_integralDis + (IDIS_s(1) + IDIS_s(cnt - startcnt
+ 1))*dt/2;
current_integralCap = current_integralCap + (ICAP_s(1) + ICAP_s(cnt - startcnt
+ 1))*dt/2;
```

```
%[EDIS,ECAP,ER2,ER3]/10

Ediss = Ediss/10
Ebatt = Ebatt/10
current_integralBatt = current_integralBatt/10
current_integralDis = current_integralDis/10
current_integralCap = current_integralCap/10
effB = -Ediss/current_integralBatt

%plot the data
t = dt*(0:1:(cnt - startcnt));  %because at the end, there is a "shift", so
recordnums counts one more.
figure(2);plot(t,ICH_s,'-b.','LineWidth',2,'MarkerEdgeColor','r','MarkerFaceC
olor','r','MarkerSize',10);grid;hold on;
figure(3);plot(t,IDIS_s,'-b.','LineWidth',2,'MarkerEdgeColor','r','MarkerFace
Color','r','MarkerSize',10);grid;hold on;
figure(4);plot(t,ICAP_s,'-b.','LineWidth',2,'MarkerEdgeColor','r','MarkerFace
Color','r','MarkerSize',10);grid;hold on;
figure(5);plot(t,VBAT_s,'-b.','LineWidth',2,'MarkerEdgeColor','r','MarkerFace
Color','r','MarkerSize',5);grid;hold on;
figure(5);plot(t,Vbattery_s,'-c.','LineWidth',2,'MarkerEdgeColor','g','Marker
FaceColor','g','MarkerSize',5);grid;hold on;
figure(6);plot(t,VCAP_s,'-b.','LineWidth',2,'MarkerEdgeColor','r','MarkerFace
Color','r','MarkerSize',10);grid;hold on;
figure(7);plot(VBAT*3,'-c.','LineWidth',2);hold on;
save(savefilename,'Temp','ICH','IDIS','ICAP','Vbattery','Vcapacitor','VBAT','
VCAP','pulse_num');
```

Appendix 6 Matlab codes for sweeping simulation, compare and plot results

This appendix contains 8 MATLAB code files, as listed below:

- ucapnonlin_ode.m – ode function of the nonlinear capacitors, wrote based on the analysis in appendix 3
- hybridnonlin_ode.m – ode function of the hybrid system, wrote based on the analysis in appendix 5
- GenNORM.m – calculate the difference between simulation waveform and experimental waveform, generate 2-norm or infinite norm of the difference
- OptimizeRC_GA.m – use GA optimization to optimize the function GenNORM.m, find the optimized RC parameters
- BatteryEnergy_ode.m – calculate the energy dissipation, current integration, $\emptyset$ and $\delta\emptyset$ in the battery-only system for given pulse load profiles.
- HybridEnergy_ode.m – calculate the energy dissipation, current integration, discharge capacity and normalized energy gain in the hybrid system at given pulse load profile.
- SweepSimuAll_ode.m – sweep simulation along different pulse amplitude, duty cycle and pulse period, calculate $\emptyset$ and $\delta\emptyset$
- PlotResult_Liion.m – plot the $\emptyset$ and $\delta\emptyset$ for simulation and experimental data of Li-ion batteries. The process of plotting data for Ni-MH batteries is similar.

Ucapnonlin_ode.m

```
%THIS FUNCTION IS UCAP'S MODEL FUNCTION for ODE
%Variable is Q: ----------------------------------
%Q = [Q1;Q2;..;Qpairnum], Qi is charge inside branch #i;
%V = [V1;V2;..;Vpairnum], Vi is voltage across C in branch #i;
%V is a nonlinear function of Q, where
%V1 = [(sqrt(C1^2 + 4*kv*Q1) - C1)/2/kv]; and rest Vi = Qi/Ci;
%PARAMETERS:---------------------------------------
%pairnum: how many RC branches in parallel, including the nonlinear branch
%matrix r (given): r contains all resistance value in each branch;
%matrix c (given): c contains all capacitance value in each branch;
%INPUT:--------------------------------------------
%I, is the current input to terminal of UCAP.
%Function:-----------------------------------------
%dQ/dt = Mqv*V(Q) + Mqi*I(t)
function dx = ucapnonlin_ode(tx,x)
global t loadi CAPpairnum cc kv Mqv Mqi

current = interp1(t,loadi,tx,'spline','extrap'); % Interpolate the data set (ft,f)
at time t
voltage = zeros(CAPpairnum,1);
for i = 1:CAPpairnum
   if (i == 1)
      if (kv == 0)
         voltage(i) = x(i)/cc(i);
      else
         voltage(i) = (sqrt(cc(i)^2 + 4*kv*x(i)) - cc(i))/2/kv;
      end
   else voltage(i) = x(i)/cc(i);
   end
end
%voltage
%pause
%solve F(x) = 0;
dx = Mqv*voltage + Mqi*current;
end
```

Hybridnonlin_ode.m

```
%THIS FUNCTION IS HYBRID SYSTEM'S MODEL FUNCTION FOR ODE
%Variable is Q: ------------------------------------
%Q = [Qb1;Qb2;..;Qc1;Qc2;..;Qpairnum], Qi is charge inside UNKNOWN branch #i (B
+ CAP);
%V = [Vb1;Vb2;..;Vc1;Vc2;...;Vpairnum], Vi is voltage across C in branch #i (B
+ CAP);
%V is a nonlinear function of Q, where
%Vc1 = [(sqrt(C1^2 + 4*kv*Q1) - C1)/2/kv]; and rest Vi = Qi/Ci;
%PARAMETERS:----------------------------------------
%Bpairnum: how many RC branches in series in the battery model
%CAPpairnum: how many RC branches in parallel in the UCAP model
%matrix c (given): c contains all capacitance value in each branch;
%c = [Cb1,Cb1,...,Cc1,Cc2,...];
%kv: nonlinear capacitance
%INPUT:---------------------------------------------
%current, is the load current;Voc: open circuit voltage
%Function:------------------------------------------
%[x_dot] = statestep(ti,x)
%global Mx Mv Mi C Voc u t
%ui = interp1(t,u,ti,'spline','extrap');
%x_dot = Mx * (C(:).*x(:)) + Mv * Voc + Mi*ui;
%dQ/dt = Mx*V(Q) + Mv*Voc + Mi*I(t);
%OUTPUT:Q(n)
function x_dot = hybridnonlin_ode(ti,x)
global LBpairnum cx kv Mx Mv Mi Voc_intp Ild_intp

current = Ild_intp(ti);
Vopen = Voc_intp(ti);

voltage = x(:)./cx(:);
if kv ~= 0
   i = LBpairnum+1;
   voltage(i) = (sqrt(cx(i)^2 + 4*kv*x(i)) - cx(i))/2/kv;
end

x_dot = Mx*voltage + Mv*Vopen + Mi*current;
end
```

GenNORM.m

```
%THIS FUNCTION IS FOR OPTIMIZATION RC VALUE FOR LIION BATTERY or UCAP MODEL
%Generate 2-norm or inf-norm to compare experimental and simulation differences
%INPUT: two voltage waveform V1, V2; two current waveform I1,I2;
%variable: RC value
%flag: flag = 1, ucap; 2, liion;
function y =
GenNORM(x,Voc1,Voc2,V1ac,V2ac,I1in,I2in,t1,t2,pulsenum1,pulsenum2,N1_1,N1_2,N
2_1,N2_2,Rshunt,flag,pairnum)
global t loadi CAPpairnum cc kv Mqv Mqi
%x is the matrix of RC value
%For battery: x = [Resr,Rs,Cs,Rm,Cm,Rl,Cl]
%For capacitor, x = [rleak,r0,c0,r1,c1,r2,c2,kv];
R1 = Rshunt(1);R2 = Rshunt(2);R3 = Rshunt(3);
predict_voltage1(1) = Voc1;
if (flag ~= 1) predict_voltage2(1) = Voc2;end
switch flag
    case 1  %if the model is UCAP
        rleak = x(1);  %leakage resistance
        for i = 1:pairnum
            r(i) = x(2*i);
            c(i) = x(2*i+1);
        end
        kv = x(length(x));
        %PREPARE MATRIX FOR UCAP: form matrix for differential equation
        gama = r(1)/rleak;
        for i = 1:pairnum
            gama = gama + r(1)/r(i);
        end
        %form matrix Mqv
        Mqv = [];
        for i = 1:pairnum
            for j = 1:pairnum
                if (i ~= j)
                    Mqv(i,j) = 1/r(i)/gama*r(1)/r(j);
                else Mqv(i,j) = 1/r(i)*(r(1)/r(i)/gama - 1);
                end
            end
        end
        %form matrix Mqi
```

```
        Mqi = ones(pairnum,1);
        for i = 1:pairnum
            Mqi(i) = r(1)/r(i)/gama;
        end

        %Define Q,V
        V0 = V1ac(1)*ones(pairnum,1); %initial state of V
        for i = 1:pairnum    %initial state of Q
            if (i == 1)
                %Q(i,1) = c(i)*V0(i) + kv*V0(i)^2;
                Q(i,1) = c(i)*V0(i) + (kv*(V0(i) - 0.41) + 21.88)*V0(i);
            else Q(i,1) = c(i)*V0(i);
            end
        end

        %calculate voltage based on input current and t
        for kk = 2:length(t1) %every time step
            dts = t1(2) - t1(1);
            %USE FSOLVE, WORK----------------------------------------
            options = optimset('Display','off','TolFun',1e-6,'MaxIter',300);
            fucap =
@(x)ucapnonlin(x,pairnum,c,kv,Mqv,Mqi,I1in(kk),Q(:,kk-1),dts);
            Q(:,kk) = fsolve(fucap,Q(:,kk-1),options);
            ic1(kk) = (Q(1,kk) - Q(1,kk-1))/dts;
        end

        for i = 1:pairnum
            if (i == 1)
                if (kv == 0)
                    V(i,:) = Q(i,:)/c(i);
                else
                    B = 21.886 + c(i) - kv*0.41;
                    V(i,:) = (sqrt(B^2*ones(size(Q(i,:))) + 4*kv*Q(i,:)) -
B*ones(size(Q(i,:))))/2/kv;
                end
            else V(i,:) = Q(i,:)/c(i);
            end
        end
        predict_voltage1 = (ic1*r(1) + V(1,:))';
    case 2
        Resr = x(1);          %series resistance
```

```
        Hs = Resr;
        for j = 1:pairnum
            r(j) = x(2*j);
            c(j) = x(2*j+1);
            tau(j) = r(j)*c(j);
            Hs = Hs + tf(r(j),[tau(j),1]);
        end
        delta_v1 = lsim(Hs,I1in,t1);  %voltage subtraction on RC network,waveform1
        delta_v2 = lsim(Hs,I2in,t2);  %voltage subtraction on RC network, waveform2
        predict_voltage1 = Voc1*ones(size(delta_v1)) - delta_v1;
        %predict output votage 1, open circuit V comes from real data
        predict_voltage2 = Voc2*ones(size(delta_v2)) - delta_v2; %predict vout 2
    otherwise
end
%only take last 100 or 10
startcnt1 = length(I1in)/pulsenum1*(pulsenum1 - N1_1) + 1;
endcnt1 = length(I1in)/pulsenum1*(pulsenum1 - N2_1);

startcnt2 = length(I2in)/pulsenum2*(pulsenum2 - N1_2) + 1;
endcnt2 = length(I2in)/pulsenum2*(pulsenum2 - N2_2);

voltage = [predict_voltage1(startcnt1:endcnt1) - V1ac;...
    predict_voltage2(startcnt2:endcnt2) - V2ac];
y = norm(voltage,inf);
end
```

OptimizeRC_GA.m

```
%THIS .M FILE IS FOR CALCULATING THE OPTIMIZED RC PARAMETERS FOR BATTERY/UCAP
clc,clf,clear all,close all;
format long;

%FIRST READ TWO SETS OF EXPERIMENTAL DATA
BatteryType = '_Li_';  %'_Li_' meanse Li-Ion battery;'_' means NIMH
BID = 'B';
BatteryID = '23';
CHDISID = 'DIS';

DateBID = '031114';%'050314';
IID = '8A';
PeriodID = '200ms';
DutyID = 'D50';

I1 = 8;
T1 = 200*1e-3;
D1 = 10*0.01;

DateBID2 = '042014';
IID2 = '16A';
PeriodID2 = '200ms';
DutyID2 = 'D10';

I2 = 16;
T2 = 200*1e-3;
D2 = 10*0.01;

flag = 1; %flag indicate to simulate which device, 1 is ucap, 2 is liion

N1_1 = 10;%410;%210;%1510;
N2_1 = 0;%400;%1200;%1500;
N1_2 = 2000;%1760;
N2_2 = 1999;%1750;
%Differential Amp Coefficiency
R1 = 0.01;R2 = 0.01;R3 = 0.01;
AMP1 = 4.984; AMP2 = 4.984;AMP3 = 4.984; %for data after Jan, 2014

CA_VBAT = 3;
```

```
CA_VCAP = 3;
CA_Vbattery = 3;
CA_Vcapacitor = 3;
CA_ICH = 1/(R3*AMP3);
CA_IDIS = 1/(R1*AMP1);
CA_ICAP = 1/(R2*AMP2);
dt = 1/(14.7654*1e6)*14720;
%%
%READ FIRST FILE B-ONLY/UCAP-ONLY EXPERIMENTAL DATA
if (flag == 1)
    filename = '041714_CAP25_SHORTDISCHARGE.mat';
else
    filename =
strcat(DateBID,BatteryType,IID,PeriodID,DutyID,BID,'_',BatteryID,'_',CHDISID,
'.mat');
end

load(filename);
cnt = length(Vbattery);

%startcnt1 = 1;
%endcnt1 = cnt;
startcnt1 = cnt/pulse_num*(pulse_num - N1_1) + 1;
endcnt1 = cnt/pulse_num*(pulse_num - N2_1);
for kk = 1:(endcnt1 - startcnt1 + 1)
    i = kk + startcnt1 - 1;
    VBAT_s(kk) = VBAT(i) * CA_VBAT;
    VCAP_s(kk) = VCAP(i) * CA_VCAP;
    ICH_s(kk) = ICH(i) * CA_ICH;
    IDIS_s(kk) = IDIS(i) * CA_IDIS;
    ICAP_s(kk) = ICAP(i) * CA_ICAP;
    ICAP_s(kk) = ICAP(i) * CA_ICAP;
    Vbattery_s(kk) = Vbattery(i) * CA_Vbattery;
    Vcapacitor_s(kk) = Vcapacitor(i) * CA_Vcapacitor;
end

%calculate integration of current till startcnt2
inteIb = 0;
for kk = 2:startcnt1
    inteIb = inteIb + dt*(ICH(kk) + ICH(kk - 1))*CA_ICH/2/3600/2.3;
end
```

```
inteIb = abs(inteIb);
Voc1 = LiionSOCOCV(inteIb);

%form group1 data
t1 = dt*(0:1:(endcnt1 - startcnt1))';
V1ac = Vbattery_s';
I1ac = -ICH_s';

[I1_input,t1_input] = pulsecurrentgen(pulse_num,T1,D1,I1,dt);
pulsenum1 = pulse_num;
%READ SECOND FILE EXPERIMENTAL DATA
if (flag ~= 1)
    clear filename All_Data recordnums state statenum Temp ICH IDIS ICAP Vcapacitor
VBAT VCAP pulse_num
    clear VBAT_s VCAP_s ICH_s IDIS_s ICAP_s Vbattery_s Vcapacitor_s t V I

    filename =
strcat(DateBID2,BatteryType,IID2,PeriodID2,DutyID2,BID,'2000_',BatteryID,'_',
CHDISID,'.mat');
    load(filename);
cnt = length(Vbattery);

EDIS = 0;
ECAP = 0;
ER2 = 0;
ER3 = 0;
Ebatt = 0;
current_integralBatt = 0;
current_integralDis = 0;
current_integralCap = 0;

%scale data
startcnt2 = cnt/pulse_num*(pulse_num - N1_2) + 1; %startcnt is the start number
of last ten pulses
endcnt2 = cnt/pulse_num*(pulse_num - N2_2);
for kk = 1:endcnt2 - startcnt2 + 1
    i = kk + startcnt2 - 1;
    VBAT_s(kk) = VBAT(i) * CA_VBAT;
    VCAP_s(kk) = VCAP(i) * CA_VCAP;
    ICH_s(kk) = ICH(i) * CA_ICH;
    IDIS_s(kk) = IDIS(i) * CA_IDIS;
```

```
    ICAP_s(kk) = ICAP(i) * CA_ICAP;
    ICAP_s(kk) = ICAP(i) * CA_ICAP;
    Vbattery_s(kk) = Vbattery(i) * CA_Vbattery;
    Vcapacitor_s(kk) = Vcapacitor(i) * CA_Vcapacitor;
end
%calculate integration of current till startcnt2
inteIb = 0;
for kk = 2:startcnt2
    inteIb = inteIb + dt*(ICH(kk) + ICH(kk - 1))*CA_ICH/2/3600/2.3;
end
inteIb = abs(inteIb);
Voc2 = LiionSOCOCV(inteIb);

%form group 2 data
t2 = dt*(0:1:(endcnt2 - startcnt2))';
V2ac = Vbattery_s';
I2ac = -ICH_s';

[I2_input,t2_input] = pulsecurrentgen(pulse_num,T2,D2,I2,dt);
pulsenum2 = pulse_num;
else
    t2_input = [];
    I2_input = [];
    pulsenum2 = [];
    Voc2 = [];
    V2ac = [];
end
%%
%Ucap boundary conditions
Rshunt = [R1,R2,R3];
CAPlowerbound = [10000,1e-6,1,1e-5,1e-5,1e-5,1e-5,1e-5,1e-5,1];
CAPupperbound = [100000,1,50,10,10,3000,3000,1,1,10];
CAPpairnum = 4;
CAPvaranum = 2*CAPpairnum + 2;

%Li-ion boundary conditions
LBlowerbound = [0.06,0,0,0,10,0,1,0,0,0,0];
LBupperbound = [0.2,0.1,5,0.1,150,0.1,700,0.1,20];
LBpairnum = 4;
LBvaranum = 2*LBpairnum + 1;
```

```
switch flag
    case 1
        pairnum = CAPpairnum;
        lowerbound = CAPlowerbound;
        upperbound = CAPupperbound;
        varanum = CAPvaranum;
    case 2
        pairnum = LBpairnum;
        lowerbound = LBlowerbound;
        upperbound = LBupperbound;
        varanum = LBvaranum;
    otherwise
end
options = optimset('Display','iter','TolFun',1e-6,'MaxIter',300);
fGEN =
{@GenNORM,Voc1,Voc2,V1ac,V2ac,I1_input,I2_input,t1_input,t2_input,pulsenum1,p
ulsenum2,N1_1,N1_2,N2_1,N2_2,Rshunt,flag,pairnum};
[OPTIMIZERC,fv] =
ga(fGEN,varanum,[],[],[],[],lowerbound,upperbound,[],options);
OPTIMIZERC
%%
simu_scnt1 = length(I1_input)/pulsenum1*(pulsenum1 - N1_1) + 1; %startcnt is the
start number of last ten pulses
simu_ecnt1 = length(I1_input)/pulsenum1*(pulsenum1 - N2_1);

simu_scnt2 = length(I2_input)/pulsenum2*(pulsenum2 - N1_2) + 1; %startcnt is the
start number of last ten pulses
simu_ecnt2 = length(I2_input)/pulsenum2*(pulsenum2 - N2_2);

Predict =
GenCURVE(OPTIMIZERC,Voc1,Voc2,V1ac,V2ac,I1_input,I2_input,t1_input,t2_input,p
ulsenum1,pulsenum2,N1_1,N1_2,N2_1,N2_2,Rshunt,flag,pairnum);
Predict_v1 = Predict(1:simu_ecnt1 - simu_scnt1 + 1);
%%
%plot the data
figure(2);plot(t1(startcnt1:endcnt1) -
t1(startcnt1),I1ac,'-b.','LineWidth',3,'MarkerEdgeColor','c','MarkerFaceColor
','c','MarkerSize',10);grid;
figure(3);subplot(1,2,1);
```

```
plot(t1(startcnt1:endcnt1) -
t1(startcnt1),V1ac,'-b','LineWidth',3,'MarkerEdgeColor','b','MarkerFaceColor'
,'b','MarkerSize',5);hold on;
plot(t1_input(simu_scnt1:simu_ecnt1) -
t1_input(simu_scnt1),Predict_v1,'-rs','MarkerEdgeColor','r','MarkerFaceColor'
,'r','MarkerSize',5);hold off;legend('EXPERIMENTS','SIMULATION');
title('Simulation & Experimental Comparision  - 8A200msD50','FontSize',16);
ylabel('Battery Voltage/V','FontSize',16);
xlabel('t/second','FontSize',16);set(gca,'fontsize',16);grid;

figure(3);subplot(1,2,2);
plot(t1(startcnt1:endcnt1) -
t1(startcnt1),V1ac,'-b','LineWidth',3,'MarkerEdgeColor','b','MarkerFaceColor'
,'b','MarkerSize',5);hold on;plot(t1_input(simu_scnt1:simu_ecnt1) -
t1_input(simu_scnt1),Predict_v1,'-rs','MarkerEdgeColor','r','MarkerFaceColor'
,'r','MarkerSize',5);hold off;legend('EXPERIMENTS','SIMULATION');
title('Zoom-in Transient Performance','FontSize',16);
ylabel('Battery Voltage/V','FontSize',16);
xlabel('t/second','FontSize',16);set(gca,'fontsize',16);grid;

if (flag ~=1)
    Predict_v2 = Predict(simu_ecnt1 - simu_scnt1 + 2:length(Predict));

figure(4);plot(t2,I2ac,'-b','MarkerEdgeColor','b','MarkerFaceColor','b','Mark
erSize',5);legend('Bonly','Cap');xlabel('t/s','FontSize',16);ylabel('ICH/A','
FontSize',16);title('R3/Battery current','FontSize',16);hold
off;set(gca,'fontsize',16);

figure(5);subplot(1,2,1);
plot(t2,V2ac,'-b','LineWidth',3,'MarkerEdgeColor','b','MarkerFaceColor','b','
MarkerSize',5);hold on;plot(t2_input(simu_scnt2:simu_ecnt2) -
t2_input(simu_scnt2),Predict_v2,'s','MarkerEdgeColor','r','MarkerFaceColor','
r','MarkerSize',5);hold off;legend('EXPERIMENTS','SIMULATION');
title('Simulation & Experimental Comparision - 16A200msD10','FontSize',16);
ylabel('Battery Voltage/V','FontSize',16);
xlabel('t/second','FontSize',16);set(gca,'fontsize',16);grid;
subplot(1,2,2);
plot(t2,V2ac,'-b','LineWidth',3,'MarkerEdgeColor','b','MarkerFaceColor','b','
MarkerSize',5);hold on;plot(t2_input(simu_scnt2:simu_ecnt2) -
t2_input(simu_scnt2),Predict_v2,'s','MarkerEdgeColor','r','MarkerFaceColor','
r','MarkerSize',5);hold off;legend('EXPERIMENTS','SIMULATION');
```

```
title('Zoom-in Transient Performance','FontSize',16);
ylabel('Battery Voltage/V','FontSize',16);
xlabel('t/second','FontSize',16);set(gca,'fontsize',16);grid;
end
```

BatteryEnergy_ode.m

```
%THIS FUNCTION IS FOR CALCULATING ENERGY FROM BATTERY-ONLY SYSTEM
%OUTPUT: Energy to Load, Energy from Battery, battery current integration,
capacitor current integration, discharge capacity
function y = BatteryEnergy_ode(Hs)
global loadi t Voc Rshunt Rpad simu_pulsenum
R3 = Rshunt(3);
deltabv = lsim(Hs,loadi,t);  %voltage subtraction on RC network, waveform 1
battv = Voc - deltabv;
Vout = battv - loadi*(R3 + Rpad(1));

%CALCULATE THE ENERGY
if (simu_pulsenum >= 10)
   startcnt = length(t)*(simu_pulsenum - 10)/simu_pulsenum + 1;
else startcnt = 1;
end
endcnt = length(t);
PR3 = R3*(loadi(startcnt:endcnt).^2);      %power dissipate on R3
PowerBatt = battv(startcnt:endcnt).*loadi(startcnt:endcnt); %power generate by
battery
Powerdiss = Vout(startcnt:endcnt).*loadi(startcnt:endcnt) + PR3;

Ebatt = trapz(t(startcnt:endcnt),PowerBatt);
Ediss = trapz(t(startcnt:endcnt),Powerdiss);
current_integralBatt = trapz(t(startcnt:endcnt),loadi(startcnt:endcnt));
DC = Ediss/current_integralBatt;
y = [Ebatt;Ediss;current_integralBatt;DC];
end
```

HybridEnergy_ode.m

```
%This function is to simulate hybrid system and gives back:
%Energy to Load, Energy from Battery, battery current integration, capacitor
current integration, discharge capacity
function y = HybridEnergy_ode()
global LBpairnum CAPpairnum loadi t D
global cx kv m Mbi alpha beta A
global Voc Rshunt Rpad Resr simu_pulsenum

R1 = Rshunt(1);
R2 = Rshunt(2);
R3 = Rshunt(3);
rpadb = Rpad(1);
rpadc = Rpad(2);

%set initial state
totpairnum = LBpairnum + CAPpairnum;
V0(1:LBpairnum) = zeros(LBpairnum,1);
V0(LBpairnum + 1:totpairnum) = Voc(1)/m*ones(CAPpairnum,1); %initial state of V
capv(:,1) = V0(LBpairnum + 1:totpairnum);  %no current yet

Q0 = cx(:).*V0(:);
Q0(LBpairnum) = Q0(LBpairnum) + kv*V0(LBpairnum)^2;

options = odeset('RelTol',1e-5,'AbsTol',1e-5);
Qdata = ode23tb(@hybridnonlin_ode,[0,t(length(t))],Q0);
odetime = (Qdata.x)';
Q_ode = Qdata.y;

for i = 1:totpairnum
   Q(i,:) = interp1(odetime,Q_ode(i,:),t);
   V(i,:) = Q(i,:)/cx(i);
   if (i == LBpairnum + 1)
      if (kv ~= 0)
         V(i,:) = (sqrt(cx(i)^2*ones(size(Q(i,:))) + 4*kv*Q(i,:)) -
cx(i)*ones(size(Q(i,:))))/2/kv;
       end
   end
end
Ib = alpha*Voc + beta*loadi - (A*V)';
```

```
Ic = Ib - loadi;
battv = Voc - Ib*Resr - ([Mbi',zeros(1,CAPpairnum)]*V)';
batti = Ib;
Vout = battv - Ib*(R3 + rpadb);
capv = Vout - Ic*(R2 + rpadc);
capi = Ic;

sectionnumber = round(simu_pulsenum * 0.01);
if (sectionnumber < 1)
    sectionnumber = 1;
else if (sectionnumber > 10)
        sectionnumber = 10;
    end
end
startcnt = length(t)*(simu_pulsenum - sectionnumber)/simu_pulsenum + 1;
endcnt = length(t);

PR2 = (R2 + Rpad(1))*(capi(startcnt:endcnt).^2);
PR3 = (R3 + Rpad(2))*(batti(startcnt:endcnt).^2);      %power dissipate on R3
PowerBatt = battv(startcnt:endcnt).*batti(startcnt:endcnt); %power generate by
battery
PowerOut = Vout(startcnt:endcnt).*loadi(startcnt:endcnt);
Ebatt = trapz(t(startcnt:endcnt),PowerBatt);
Ediss = trapz(t(startcnt:endcnt),PowerOut) + trapz(t(startcnt:endcnt),PR2) +
trapz(t(startcnt:endcnt),PR3);
current_integralI = trapz(t(startcnt:endcnt),loadi(startcnt:endcnt));
current_integralBatt = trapz(t(startcnt:endcnt),batti(startcnt:endcnt));
current_integralCAP = trapz(t(startcnt:endcnt),capi(startcnt:endcnt));
DC = Ediss/current_integralI;
y = [Ebatt;Ediss;current_integralBatt;current_integralCAP;DC];
end
```

SweepSimuAll_ode.m

```
%This file is for sweep along pulse amplitude I, duty cycle D, pulse period T for
both the Li-ion battery only model and 8.33 F Hybrid model, and calculate discharge
capacity and normalized energy efficiency gain.
%-------------2014.05.04-----------By YH
clc,clf,clear all
close all
format long;
%%
global LBpairnum CAPpairnum
global cx kv m Mbi alpha beta A Mx Mv Mi
global Voc_intp Rshunt Rpad Resr simu_pulsenum
global Ild_intp t Voc loadi

%DEFINE PARAMETERS-------------------------------------------------
R1 = 10e-3;R2 = 10e-3;R3 = 10e-3;
Rshunt = [R1;R2;R3];
dts = .001;
Drange = 0.01:0.01:0.91; %%
Irange = 2:0.5:18;        %A
Trange = 40:20:420;  %ms
Vrange = 5:0.5:7.5;      %v
Crang = 0.5:0.5:30;

state_flag = 1:1:3; %The flag indicates the state of sweep which variable;
%state_flag = 1:I; 2:D; 3:T;

%Battery model paras:-----------------------------------------------
zb = [0.074499766
0.001215743
1.878365419
0.000860477
65.73597844
0.074815159
186.4688713];

Resr = zb(1);        %series resistance
Re = Resr + R3;      %including R3
LBpairnum = (length(zb) - 1)/2;
Hs = Re;
```

```
Mbv = eye(LBpairnum);
Mbi = ones(LBpairnum,1);
for i = 1:LBpairnum
    rb(i) = zb(2*i);
    cb(i) = zb(2*i + 1);
    taub(i) = rb(i)*cb(i);
    Hs = Hs + tf(rb(i),[taub(i),1]);
    Mbv(i,i) = -1/rb(i);
end

%Ultracap model paras:-------------------------------------------------
zc = [74737
0.0261
20.28
0.1313
1.5374
186.4
1.05
0.06
0.417
0.0%3.51
];
m = 3;     %how many UCAP in series
rleak = zc(1); %leakage resistance
kv = zc(length(zc));
CAPpairnum = (length(zc) - 2)/2;
for i = 1:CAPpairnum
    rc(i) = zc(2*i);
    cc(i) = zc(2*i + 1);
    tauc(i) = rc(i)*cc(i);
end

%ucap: form matrix for differential equation
gama = rc(1)/rleak;
for i = 1:CAPpairnum
    gama = gama + rc(1)/rc(i);
end
%form matrix Mqv
Mqv = [];
for i = 1:CAPpairnum
    for j = 1:CAPpairnum
```

```
        if (i ~= j)
            Mqv(i,j) = 1/rc(i)/gama*rc(1)/rc(j);
        else Mqv(i,j) = 1/rc(i)*(rc(1)/rc(i)/gama - 1);
        end
    end
end
%form matrix Mqi
Mqi = ones(CAPpairnum,1);
for i = 1:CAPpairnum
    Mqi(i) = rc(1)/rc(i)/gama;
end

%hybrid: form matrix for differential equation
%pad resistance
rpadb = 0;%0.015;
rpadc = 0;%.015;              %pad resistance, connect ucaps
Rpad = [rpadb;rpadc];

Rp = rpadc + R2;
alpha = 1/(m*rc(1)*Mqi(1) + Resr + R3 + rpadb + R2 + rpadc);
beta = alpha*(m*rc(1)*Mqi(1) + R2 + rpadc);
A = alpha*[ones(1,LBpairnum),m*(rc(1)*Mqv(1,:) + [1,zeros(1,CAPpairnum - 1)])];

Mx = blkdiag(Mbv,Mqv) - [Mbi;Mqi]*A;
Mv = alpha*[Mbi;Mqi];
Mi = [beta*Mbi;(beta - 1)*Mqi];

cx = [cb,cc];

%PARAMETERS FOR EXPERIMENTS
filename = 'ResultCompare_latest.xlsx';
sheetname = 'Liion_SOC_99%';
savefilenameI = '051914_SimulationResult_I.mat';
savefilenameD = '051914_SimulationResult_D.mat';
savefilenameT = '051914_SimulationResult_T.mat';
savefilenameV = '051914_SimulationResult_V.mat';
%%
%SWEEP SIMULATION BONLY----------------------------------------
for i = 2:3 %1:length(state_flag)
    switch state_flag(i)
        case 1 %sweep pulse amplitude
```

```
        D = 10*0.01;
        T = 200*1e-3;
        for j = 1:length(Irange)
            clear BONLYDATA HYBRIDDATA
            loadi = [];t = [];Voc = [];Voc_intp = [];Ild_intp = [];
            I = Irange(j);
            %generate pulse load current
            simu_pulsenum = round(0.00966*3600*2.3/(T*D*I));
            [loadi,t] = pulsecurrentgen(simu_pulsenum,T,D,I,dts);

            %calculate voc
            deltat = t(2) - t(1);
            inteIb = zeros(size(loadi));
            for kk = 2:length(t)
                if (loadi(kk) > 0)
                    inteIb(kk) = inteIb(kk-1) + (loadi(kk) +
loadi(kk-1))/2*deltat/3600/2.3;
                else inteIb(kk) = inteIb(kk-1);
                end
            end
            Voc = LiionSOCOCV(inteIb); %changing with interIb
            Voc_intp = griddedInterpolant(t,Voc);
            Ild_intp = griddedInterpolant(t,loadi);

            %CALCULATE BONLY BEHAVIOR-------------------------
            BONLYDATA = BatteryEnergy_ode(Hs);
            BEbatt_I(j) = BONLYDATA(1);
            BEdiss_I(j) = BONLYDATA(2);
            Bcurrent_integralBatt_I(j) = BONLYDATA(3);
            BDC_I(j) = BONLYDATA(4);
            %CALCULATE HYBRID BEHAVIOR------------------------
            HYBRIDDATA = HybridEnergy_ode();
            HEbatt_I(j) = HYBRIDDATA(1);
            HEdiss_I(j) = HYBRIDDATA(2);
            Hcurrent_integralBatt_I(j) = HYBRIDDATA(3);
            Hcurrent_integralCAP_I(j) = HYBRIDDATA(4);
            HDC_I(j) = HYBRIDDATA(5);
            %CALCULATE EFF GAIN AND PLOT
            EFFGAIN_I(j) = HDC_I(j)/BDC_I(j) - 1;
        end
    case 2 %sweep pulse duty cycle
```

```
        I = 8;
        T = 200*1e-3;
        for j = 1:length(Drange)
            clear BONLYDATA HYBRIDDATA
            loadi = [];t = [];Voc = [];Voc_intp = [];Ild_intp = [];
            D = Drange(j);
            simu_pulsenum = round(0.00966*3600*2.3/(T*D*I));
            %generate pulse load current
            [loadi,t] = pulsecurrentgen(simu_pulsenum,T,D,I,dts);

            %calculate voc
            deltat = t(2) - t(1);
            inteIb = zeros(size(loadi));
            for kk = 2:length(t)
                if (loadi(kk) > 0)
                    inteIb(kk) = inteIb(kk-1) + (loadi(kk) +
loadi(kk-1))/2*deltat/3600/2.3;
                else inteIb(kk) = inteIb(kk-1);
                end
            end
            Voc = LiionSOCOCV(inteIb); %changing with interIb
            Voc_intp = griddedInterpolant(t,Voc);
            Ild_intp = griddedInterpolant(t,loadi);

            %CALCULATE BONLY BEHAVIOR-------------------------
            BONLYDATA = BatteryEnergy_ode(Hs);
            BEbatt_D(j) = BONLYDATA(1);
            BEdiss_D(j) = BONLYDATA(2);
            Bcurrent_integralBatt_D(j) = BONLYDATA(3);
            BDC_D(j) = BONLYDATA(4);
            %CALCULATE HYBRID BEHAVIOR------------------------
            HYBRIDDATA = HybridEnergy_ode();
            HEbatt_D(j) = HYBRIDDATA(1);
            HEdiss_D(j) = HYBRIDDATA(2);
            Hcurrent_integralBatt_D(j) = HYBRIDDATA(3);
            Hcurrent_integralCAP_D(j) = HYBRIDDATA(4);
            HDC_D(j) = HYBRIDDATA(5);
            %CALCULATE EFF GAIN AND PLOT
            EFFGAIN_D(j) = HDC_D(j)/BDC_D(j) - 1;
        end
    case 3 %sweep pulse period
```

```
            I = 8;
            D = 10*0.01;
            for j = 1:length(Trange)
                clear BONLYDATA HYBRIDDATA
                loadi = [];t = [];Voc = [];Voc_intp = [];Ild_intp = [];
                T = Trange(j) * 1e-3;
                simu_pulsenum = round(0.00966*3600*2.3/(T*D*I));
                %generate pulse load current
                [loadi,t] = pulsecurrentgen(simu_pulsenum,T,D,I,dts);
                %calculate voc
                deltat = t(2) - t(1);
                inteIb = zeros(size(loadi));
                for kk = 2:length(t)
                    if (loadi(kk) > 0)
                        inteIb(kk) = inteIb(kk-1) + (loadi(kk) +
loadi(kk-1))/2*deltat/3600/2.3;
                    else inteIb(kk) = inteIb(kk-1);
                    end
                end
                Voc = LiionSOCOCV(inteIb); %changing with interIb
                Voc_intp = griddedInterpolant(t,Voc);
                Ild_intp = griddedInterpolant(t,loadi);

                %CALCULATE BONLY BEHAVIOR-------------------------
                BONLYDATA = BatteryEnergy_ode(Hs);
                BEbatt_T(j) = BONLYDATA(1);
                BEdiss_T(j) = BONLYDATA(2);
                Bcurrent_integralBatt_T(j) = BONLYDATA(3);
                BDC_T(j) = BONLYDATA(4);
                %CALCULATE HYBRID BEHAVIOR------------------------
                HYBRIDDATA = HybridEnergy_ode();
                HEbatt_T(j) = HYBRIDDATA(1);
                HEdiss_T(j) = HYBRIDDATA(2);
                Hcurrent_integralBatt_T(j) = HYBRIDDATA(3);
                Hcurrent_integralCAP_T(j) = HYBRIDDATA(4);
                HDC_T(j) = HYBRIDDATA(5);
                %CALCULATE EFF GAIN AND PLOT
                EFFGAIN_T(j) = HDC_T(j)/BDC_T(j) - 1;
            end
        otherwise
    end
```

```
end
%%
save(savefilenameI,'Irange','BEbatt_I','BEdiss_I','Bcurrent_integralBatt_I','
BDC_I','HEbatt_I','HEdiss_I','Hcurrent_integralBatt_I','Hcurrent_integralCAP_
I','HDC_I','EFFGAIN_I');
save(savefilenameD,'Drange','BEbatt_D','BEdiss_D','Bcurrent_integralBatt_D','
BDC_D','HEbatt_D','HEdiss_D','Hcurrent_integralBatt_D','Hcurrent_integralCAP_
D','HDC_D','EFFGAIN_D');
save(savefilenameT,'Trange','BEbatt_T','BEdiss_T','Bcurrent_integralBatt_T','
BDC_T','HEbatt_T','HEdiss_T','Hcurrent_integralBatt_T','Hcurrent_integralCAP_
T','HDC_T','EFFGAIN_T');
figure(2);plot(Irange,EFFGAIN_I,'LineWidth',2);hold on;
figure(3);plot(Drange,EFFGAIN_D,'LineWidth',2);hold on;
figure(4);plot(Trange,EFFGAIN_T,'LineWidth',2);hold on;
figure(6);plot(Irange,BDC_I,Irange,HDC_I,'LineWidth',2);
figure(7);plot(Irange,Bcurrent_integralBatt_I,Irange,Hcurrent_integralBatt_I,
'LineWidth',2);
figure(8);plot(Irange,BEbatt_I,Irange,BEdiss_I,Irange,HEbatt_I,Irange,HEdiss_
I,'LineWidth',2);
```

PlotResults_Liion.m

```
%THIS SCRIPT READS RESULTS OF NIMH BATTERIES AND PLOTS OUT
clc,clf,clear all
close all

filename = 'ResultCompare_latest.xlsx';
sheetname = 'Liion_SOC_99%';
sheetname2 = 'Liion_SOC_85%';
filenameI = '051914_SimulationResult_I_SAM3.mat';
filenameD = '051914_SimulationResult_D_SAM3.mat';
filenameT = '051914_SimulationResult_T_SAM3.mat';

load(filenameI);
load(filenameD);
load(filenameT);
%%
%READ EXPERIMENTAL DATA 99%-----------------------------------------------
Experi_result = xlsread(filename,sheetname);
%Experi_result = [pulsenum Capacitor(F) Current(A) DutyCycle (%)
% Period(ms) Pulse Width(ms) Ediss Ebatt Curr_Batt Curr_Dis dc Ediss/Curr_Batt];
bki = 1;bkd = 1;bkt = 1;
hki = 1;hkd = 1;hkt = 1;
hki_1 = 1;hkd_1 = 1;hkt_1 = 1;
hki_2 = 1;hkd_2 = 1;hkt_2 = 1;

DCNO = 16;
EFFIBNO = 17;
EFFNO = 20;
ERRORNO = 18;

[xn,yn] = size(Experi_result);

for i = 1:xn
      if (Experi_result(i,2) == 0) %Bonly data
         if ((Experi_result(i,4) == 0.1) && (Experi_result(i,5) == 200))
            sBDC_Irange(bki) = Experi_result(i,3);
            sBDC_I(bki) = Experi_result(i,DCNO);
            sBDCE_I(bki) = 0.152/(Experi_result(i,3)*49.84) +
0.008/Experi_result(i,3) + 0.015;
            bki = bki + 1;
```

```
        end
        if ((Experi_result(i,3) == 8) && (Experi_result(i,5) == 200))
            sBDC_Drange(bkd) = Experi_result(i,4);
            sBDC_D(bkd) = Experi_result(i,DCNO);
            sBDCE_D(bkd) = 0.152/(Experi_result(i,3)*49.84) +
0.008/Experi_result(i,3) + 0.015;
            bkd = bkd + 1;
        end
        if ((Experi_result(i,3) == 8) && (Experi_result(i,4) == 0.1))
            sBDC_Trange(bkt) = Experi_result(i,5);
            sBDC_T(bkt) = Experi_result(i,DCNO);
            sBDCE_T(bkt) = 0.152/(Experi_result(i,3)*49.84) +
0.008/Experi_result(i,3) + 0.015;
            bkt = bkt + 1;
        end
    end

    if (Experi_result(i,2) == 1.667) %C5 data
        if ((Experi_result(i,4) == 0.1) && (Experi_result(i,5) == 200))
            sHDC5_Irange(hki_1) = Experi_result(i,3);
            sHDC5_I(hki_1) = Experi_result(i,DCNO);
            sEFFIB5_I(hki_1) = Experi_result(i,EFFIBNO);
            sEFFG5_I(hki_1) = Experi_result(i,EFFNO);
            sHDC5E_I(hki_1) = 0.152/(Experi_result(i,3)*49.84) +
0.008/Experi_result(i,3) + 0.015;
            hki_1 = hki_1 + 1;
        end
        if ((Experi_result(i,3) == 8) && (Experi_result(i,5) == 200))
            sHDC5_Drange(hkd_1) = Experi_result(i,4);
            sHDC5_D(hkd_1) = Experi_result(i,DCNO);
            sEFFIB5_D(hkd_1) = Experi_result(i,EFFIBNO);
            sEFFG5_D(hkd_1) = Experi_result(i,EFFNO);
            sHDC5E_D(hkd_1) = 0.152/(Experi_result(i,3)*49.84) +
0.008/Experi_result(i,3) + 0.015;
            hkd_1 = hkd_1 + 1;
        end

        if ((Experi_result(i,3) == 8) && (Experi_result(i,4) == 0.1))
            sHDC5_Trange(hkt_1) = Experi_result(i,5);
            sHDC5_T(hkt_1) = Experi_result(i,DCNO);
            sEFFIB5_T(hkt_1) = Experi_result(i,EFFIBNO);
```

```
            sEFFG5_T(hkt_1) = Experi_result(i,EFFNO);
            sHDC5E_T(hkt_1) = 0.152/(Experi_result(i,3)*49.84) +
0.008/Experi_result(i,3) + 0.015;
            hkt_1 = hkt_1 + 1;
        end
    end

    if (Experi_result(i,2) == 8.333) %C25 data
        if ((Experi_result(i,4) == 0.1) && (Experi_result(i,5) == 200))
            sHDC25_Irange(hki) = Experi_result(i,3);
            sHDC25_I(hki) = Experi_result(i,DCNO);
            sEFFIB25_I(hki) = Experi_result(i,EFFIBNO);
            sEFFG25_I(hki) = Experi_result(i,EFFNO);
            sHDC25E_I(hki) = 0.152/(Experi_result(i,3)*49.84) +
0.008/Experi_result(i,3) + 0.015;
            hki = hki + 1;
        end
        if ((Experi_result(i,3) == 8) && (Experi_result(i,5) == 200))
            sHDC25_Drange(hkd) = Experi_result(i,4);
            sHDC25_D(hkd) = Experi_result(i,DCNO);
            sEFFIB25_D(hkd) = Experi_result(i,EFFIBNO);
            sEFFG25_D(hkd) = Experi_result(i,EFFNO);
            sHDC25E_D(hkd) = 0.152/(Experi_result(i,3)*49.84) +
0.008/Experi_result(i,3) + 0.015;
            hkd = hkd + 1;
        end

        if ((Experi_result(i,3) == 8) && (Experi_result(i,4) == 0.1))
            sHDC25_Trange(hkt) = Experi_result(i,5);
            sHDC25_T(hkt) = Experi_result(i,DCNO);
            sEFFIB25_T(hkt) = Experi_result(i,EFFIBNO);
            sEFFG25_T(hkt) = Experi_result(i,EFFNO);
            sHDC25E_T(hkt) = 0.152/(Experi_result(i,3)*49.84) +
0.008/Experi_result(i,3) + 0.015;
            hkt = hkt + 1;
        end
    end

    if (Experi_result(i,2) == 16.667) %C50 data
        if ((Experi_result(i,4) == 0.1) && (Experi_result(i,5) == 200))
            sHDC50_Irange(hki_2) = Experi_result(i,3);
```

```
            sHDC50_I(hki_2) = Experi_result(i,DCNO);
            sEFFIB50_I(hki_2) = Experi_result(i,EFFIBNO);
            sEFFG50_I(hki_2) = Experi_result(i,EFFNO);
            sHDC50E_I(hki_2) = 0.152/(Experi_result(i,3)*49.84) +
0.008/Experi_result(i,3) + 0.015;
            hki_2 = hki_2 + 1;
        end
        if ((Experi_result(i,3) == 8) && (Experi_result(i,5) == 200))
            sHDC50_Drange(hkd_2) = Experi_result(i,4);
            sHDC50_D(hkd_2) = Experi_result(i,DCNO);
            sEFFIB50_D(hkd_2) = Experi_result(i,EFFIBNO);
            sEFFG50_D(hkd_2) = Experi_result(i,EFFNO);
            sHDC50E_D(hkd_2) = 0.152/(Experi_result(i,3)*49.84) +
0.008/Experi_result(i,3) + 0.015;
            hkd_2 = hkd_2 + 1;
        end

        if ((Experi_result(i,3) == 8) && (Experi_result(i,4) == 0.1))
            sHDC50_Trange(hkt_2) = Experi_result(i,5);
            sHDC50_T(hkt_2) = Experi_result(i,DCNO);
            sEFFIB50_T(hkt_2) = Experi_result(i,EFFIBNO);
            sEFFG50_T(hkt_2) = Experi_result(i,EFFNO);
            sHDC50E_T(hkt_2) = 0.152/(Experi_result(i,3)*49.84) +
0.008/Experi_result(i,3) + 0.015;
            hkt_2 = hkt_2 + 1;
        end
    end
end
%%
%PLOT data
%calculate experi eff error
sBDCE_I = sBDCE_I;sBDCE_D = sBDCE_D;sBDCE_T = sBDCE_T;
sHDC5E_I = sHDC5E_I;sHDC5E_D = sHDC5E_D;sHDC5E_T = sHDC5E_T;
sHDC25E_I = sHDC25E_I;sHDC25E_D = sHDC25E_D;sHDC25E_T = sHDC25E_T;
sHDC50E_I = sHDC50E_I;sHDC50E_D = sHDC50E_D;sHDC50E_T = sHDC50E_T;
%5F
for kk = 1:length(sHDC5_Irange)
    for jj = 1:length(sBDC_Irange)
        if (sHDC5_Irange(kk) == sBDC_Irange(jj))
            maxsEFFG5E_I(kk) = ((1 + sHDC5E_I(kk))*sHDC5_I(kk) - (1 -
sBDCE_I(jj))*sBDC_I(jj))/((1 - sBDCE_I(jj))*sBDC_I(jj));
```

```
            minsEFFG5E_I(kk) = ((1 - sHDC5E_I(kk))*sHDC5_I(kk) - (1 +
sBDCE_I(jj))*sBDC_I(jj))/((1 + sBDCE_I(jj))*sBDC_I(jj));
        end
    end
end
for kk = 1:length(sHDC5_Drange)
    for jj = 1:length(sBDC_Drange)
        if (sHDC5_Drange(kk) == sBDC_Drange(jj))
            maxsEFFG5E_D(kk) = ((1 + sHDC5E_D(kk))*sHDC5_D(kk) - (1 -
sBDCE_D(jj))*sBDC_D(jj))/((1 - sBDCE_D(jj))*sBDC_D(jj));
            minsEFFG5E_D(kk) = ((1 - sHDC5E_D(kk))*sHDC5_D(kk) - (1 +
sBDCE_D(jj))*sBDC_D(jj))/((1 + sBDCE_D(jj))*sBDC_D(jj));
        end
    end
end
for kk = 1:length(sHDC5_Trange)
    for jj = 1:length(sBDC_Trange)
        if (sHDC5_Trange(kk) == sBDC_Trange(jj))
            maxsEFFG5E_T(kk) = ((1 + sHDC5E_T(kk))*sHDC5_T(kk) - (1 -
sBDCE_T(jj))*sBDC_T(jj))/((1 - sBDCE_T(jj))*sBDC_T(jj));
            minsEFFG5E_T(kk) = ((1 - sHDC5E_T(kk))*sHDC5_T(kk) - (1 +
sBDCE_T(jj))*sBDC_T(jj))/((1 + sBDCE_T(jj))*sBDC_T(jj));
        end
    end
end
%%
%25 F
for kk = 1:length(sHDC25_Irange)
    for jj = 1:length(sBDC_Irange)
        if (sHDC25_Irange(kk) == sBDC_Irange(jj))
            maxsEFFG25E_I(kk) = ((1 + sHDC25E_I(kk))*sHDC25_I(kk) - (1 -
sBDCE_I(jj))*sBDC_I(jj))/((1 - sBDCE_I(jj))*sBDC_I(jj));
            minsEFFG25E_I(kk) = ((1 - sHDC25E_I(kk))*sHDC25_I(kk) - (1 +
sBDCE_I(jj))*sBDC_I(jj))/((1 + sBDCE_I(jj))*sBDC_I(jj));
        end
    end
end
for kk = 1:length(sHDC25_Drange)
    for jj = 1:length(sBDC_Drange)
        if (sHDC25_Drange(kk) == sBDC_Drange(jj))
```

```
            maxsEFFG25E_D(kk) = ((1 + sHDC25E_D(kk))*sHDC25_D(kk) - (1 -
sBDCE_D(jj))*sBDC_D(jj))/((1 - sBDCE_D(jj))*sBDC_D(jj));
            minsEFFG25E_D(kk) = ((1 - sHDC25E_D(kk))*sHDC25_D(kk) - (1 +
sBDCE_D(jj))*sBDC_D(jj))/((1 + sBDCE_D(jj))*sBDC_D(jj));
        end
    end
end
for kk = 1:length(sHDC25_Trange)
    for jj = 1:length(sBDC_Trange)
        if (sHDC25_Trange(kk) == sBDC_Trange(jj))
            maxsEFFG25E_T(kk) = ((1 + sHDC25E_T(kk))*sHDC25_T(kk) - (1 -
sBDCE_T(jj))*sBDC_T(jj))/((1 - sBDCE_T(jj))*sBDC_T(jj));
            minsEFFG25E_T(kk) = ((1 - sHDC25E_T(kk))*sHDC25_T(kk) - (1 +
sBDCE_T(jj))*sBDC_T(jj))/((1 + sBDCE_T(jj))*sBDC_T(jj));
        end
    end
end
%%
%50 F
for kk = 1:length(sHDC50_Irange)
    for jj = 1:length(sBDC_Irange)
        if (sHDC50_Irange(kk) == sBDC_Irange(jj))
            maxsEFFG50E_I(kk) = ((1 + sHDC50E_I(kk))*sHDC50_I(kk) - (1 -
sBDCE_I(jj))*sBDC_I(jj))/((1 - sBDCE_I(jj))*sBDC_I(jj));
            minsEFFG50E_I(kk) = ((1 - sHDC50E_I(kk))*sHDC50_I(kk) - (1 +
sBDCE_I(jj))*sBDC_I(jj))/((1 + sBDCE_I(jj))*sBDC_I(jj));
        end
    end
end
for kk = 1:length(sHDC50_Drange)
    for jj = 1:length(sBDC_Drange)
        if (sHDC50_Drange(kk) == sBDC_Drange(jj))
            maxsEFFG50E_D(kk) = ((1 + sHDC50E_D(kk))*sHDC50_D(kk) - (1 -
sBDCE_D(jj))*sBDC_D(jj))/((1 - sBDCE_D(jj))*sBDC_D(jj));
            minsEFFG50E_D(kk) = ((1 - sHDC50E_D(kk))*sHDC50_D(kk) - (1 +
sBDCE_D(jj))*sBDC_D(jj))/((1 + sBDCE_D(jj))*sBDC_D(jj));
        end
    end
end
for kk = 1:length(sHDC50_Trange)
    for jj = 1:length(sBDC_Trange)
```

```
        if (sHDC50_Trange(kk) == sBDC_Trange(jj))
            maxsEFFG50E_T(kk) = ((1 + sHDC50E_T(kk))*sHDC50_T(kk) - (1 -
sBDCE_T(jj))*sBDC_T(jj))/((1 - sBDCE_T(jj))*sBDC_T(jj));
            minsEFFG50E_T(kk) = ((1 - sHDC50E_T(kk))*sHDC50_T(kk) - (1 +
sBDCE_T(jj))*sBDC_T(jj))/((1 + sBDCE_T(jj))*sBDC_T(jj));
        end
    end
end
%%
%trend line for EFF
f25I_EFF = polyfit(sHDC25_Irange,sEFFG25_I*100,2);
f25D_EFF = polyfit(sHDC25_Drange,sEFFG25_D*100,1);
f25T_EFF = polyfit(sHDC25_Trange,sEFFG25_T*100,1);

f50I_EFF = polyfit(sHDC50_Irange,sEFFG50_I*100,2);
f50D_EFF = polyfit(sHDC50_Drange,sEFFG50_D*100,1);
f50T_EFF = polyfit(sHDC50_Trange,sEFFG50_T*100,1);
%Trend line for DC
fBI_DC = polyfit(sBDC_Irange,sBDC_I,1);
fBD_DC = polyfit(sBDC_Drange,sBDC_D,1);
fBT_DC = polyfit(sBDC_Trange,sBDC_T,1);

f5I_DC = polyfit(sHDC5_Irange,sHDC5_I,1);
f5D_DC = polyfit(sHDC5_Drange,sHDC5_D,1);
f5T_DC = polyfit(sHDC5_Trange,sHDC5_T,1);

f25I_DC = polyfit(sHDC25_Irange,sHDC25_I,1);
f25D_DC = polyfit(sHDC25_Drange,sHDC25_D,1);
f25T_DC = polyfit(sHDC25_Trange,sHDC25_T,1);

f50I_DC = polyfit(sHDC50_Irange,sHDC50_I,1);
f50D_DC = polyfit(sHDC50_Drange,sHDC50_D,1);
f50T_DC = polyfit(sHDC50_Trange,sHDC50_T,1);

%%
%Plot
%Simulation Data
%SIMU EFF VS EXPERI EFF
figure(2);plot(Irange,EFFGAIN_I*100,':r','LineWidth',3);hold on;
figure(3);plot(Drange(2:length(Drange)),EFFGAIN_D(2:length(Drange))*100,':r',
'LineWidth',3);hold on;
```

```
figure(4);plot(Trange,EFFGAIN_T*100,':r','LineWidth',3);hold on;

figure(2);errorbar(sHDC25_Irange,sEFFG25_I*100,minsEFFG25E_I*100,maxsEFFG25E_
I*100,'rs','MarkerEdgeColor','r','MarkerFaceColor','r','MarkerSize',5,'LineWi
dth',3);hold on;
figure(3);errorbar(sHDC25_Drange,sEFFG25_D*100,minsEFFG25E_D*100,maxsEFFG25E_
D*100,'rs','MarkerEdgeColor','r','MarkerFaceColor','r','MarkerSize',5,'LineWi
dth',3);hold on;
figure(4);errorbar(sHDC25_Trange,sEFFG25_T*100,minsEFFG25E_T*100,maxsEFFG25E_
T*100,'rs','MarkerEdgeColor','r','MarkerFaceColor','r','MarkerSize',5,'LineWi
dth',3);hold on;

%title,label,...
figure(2);hold off;grid;title('Li-ion Simulation vs Experiments - Normalized EG
vs Pulse Amplitude','FontSize',16);
xlabel('Pulse Amplitude/A','fontsize',16);ylabel('Normalized
EG/%','fontsize',16);
legend('Simu - B & Hybrid 8.33F','Expe - B & Hybrid
8.33F');set(gca,'fontsize',16);ylim([0 30]);
figure(3);hold off;grid;title('Li-ion Simulation vs Experiments - Normalized EG
vs Duty Cycle','FontSize',16);
xlabel('Duty Cycle/%','fontsize',16);ylabel('Normalized EG/%','fontsize',16);
legend('Simu - B & Hybrid 8.33F','Expe - B & Hybrid
8.33F');set(gca,'fontsize',16);ylim([0 30]);
figure(4);hold off;grid;title('Li-ion Simulation vs Experiments - Normalized EG
vs Pulse Period','FontSize',16);
xlabel('Pulse Period/ms','fontsize',16);ylabel('Normalized
EG/%','fontsize',16);
legend('Simu - B & Hybrid 8.33F','Expe - B & Hybrid
8.33F');set(gca,'fontsize',16);ylim([0 30]);
%%
%SIMU DC VS EXPERI DC
figure(5);plot(Irange,BDC_I,':b','LineWidth',2);hold
on;plot(Irange,HDC_I,':r','LineWidth',3);hold on;
figure(6);plot(Drange,BDC_D,':b','LineWidth',2);hold
on;plot(Drange(2:length(Drange)),HDC_D(2:length(Drange)),':r','LineWidth',3);
hold on;
figure(7);plot(Trange,BDC_T,':b','LineWidth',2);hold
on;plot(Trange,HDC_T,':r','LineWidth',3);hold on;
figure(5);errorbar(sBDC_Irange,sBDC_I,sBDCE_I.*sBDC_I,'b*','MarkerEdgeColor',
'b','MarkerFaceColor','b','MarkerSize',7,'LineWidth',3);hold on;
```

```
figure(5);errorbar(sHDC25_Irange,sHDC25_I,sHDC25E_I.*sHDC25_I,'rs','MarkerEdg
eColor','r','MarkerFaceColor','r','MarkerSize',5,'LineWidth',3);hold on;
figure(6);errorbar(sBDC_Drange,sBDC_D,sBDCE_D.*sBDC_D,'b*','MarkerEdgeColor',
'b','MarkerFaceColor','b','MarkerSize',7,'LineWidth',3);hold on;
figure(6);errorbar(sHDC25_Drange,sHDC25_D,sHDC25E_D.*sHDC25_D,'rs','MarkerEdg
eColor','r','MarkerFaceColor','r','MarkerSize',5,'LineWidth',3);hold on;
figure(7);errorbar(sBDC_Trange,sBDC_T,sBDCE_T.*sBDC_T,'b*','MarkerEdgeColor',
'b','MarkerFaceColor','b','MarkerSize',7,'LineWidth',3);hold on;
figure(7);errorbar(sHDC25_Trange,sHDC25_T,sHDC25E_T.*sHDC25_T,'rs','MarkerEdg
eColor','r','MarkerFaceColor','r','MarkerSize',5,'LineWidth',3);hold on;

figure(5);hold off;grid;title('Li-ion Simulation vs Experiments - Discharge
Capacity vs Pulse Amplitude','FontSize',16);
xlabel('Pulse Amplitude/A','fontsize',16);ylabel('Discharge
Capacity/V','fontsize',16);
legend('Simu - B','Simu - Hybrid 8.33F','Expe - B','Expe - Hybrid 8.33F');%,'Expe
85% - B','Expe 85% - Hybrid 8.33F','Expe 85% - Hybrid 16.67F');
set(gca,'fontsize',16);ylim([4.5 8]);
figure(6);hold off;grid;title('Li-ion Simulation vs Experiments - Discharge
Capacity vs Duty Cycle','FontSize',16);
xlabel('Duty Cycle/%','fontsize',16);ylabel('Discharge
Capacity/V','fontsize',16);
legend('Simu - B','Simu - Hybrid 8.33F','Expe - B','Expe - Hybrid 8.33F');%,'Expe
85% - B','Expe 85% - Hybrid 8.33F','Expe 85% - Hybrid 16.67F');
set(gca,'fontsize',16);ylim([4 8]);
figure(7);hold off;grid;title('Li-ion Simulation vs Experiments - Discharge
Capacity vs Pulse Period','FontSize',16);
xlabel('Pulse Period/ms','fontsize',16);ylabel('Discharge
Capacity/V','fontsize',16);
legend('Simu - B','Simu - Hybrid 8.33F','Expe - B','Expe - Hybrid 8.33F');%,'Expe
85% - B','Expe 85% - Hybrid 8.33F','Expe 85% - Hybrid 16.67F');
set(gca,'fontsize',16);ylim([4.5 8]);
%%
%Experimental Data
%EXPERI EFF
clear Irange Drange Trange
Irange = 2:1:25;
Drange = 0:0.1:1;
Trange = 25:5:450;
```

```
figure(8);errorbar(sHDC5_Irange,sEFFG5_I*100,minsEFFG5E_I*100,maxsEFFG5E_I*10
0,'co','MarkerEdgeColor','c','MarkerFaceColor','c','MarkerSize',8,'LineWidth'
,2);hold on;
errorbar(sHDC25_Irange,sEFFG25_I*100,minsEFFG25E_I*100,maxsEFFG25E_I*100,'rs'
,'MarkerEdgeColor','r','MarkerFaceColor','r','MarkerSize',8);hold on;
errorbar(sHDC50_Irange,sEFFG50_I*100,minsEFFG50E_I*100,maxsEFFG50E_I*100,'g^'
,'MarkerEdgeColor','g','MarkerFaceColor','g','MarkerSize',8,'LineWidth',2);ho
ld on;
figure(9);errorbar(sHDC5_Drange,sEFFG5_D*100,minsEFFG5E_D*100,maxsEFFG5E_D*10
0,'co','MarkerEdgeColor','c','MarkerFaceColor','c','MarkerSize',8,'LineWidth'
,2);hold on;
errorbar(sHDC25_Drange,sEFFG25_D*100,minsEFFG25E_D*100,maxsEFFG25E_D*100,'rs'
,'MarkerEdgeColor','r','MarkerFaceColor','r','MarkerSize',8,'LineWidth',2);ho
ld on;
errorbar(sHDC50_Drange,sEFFG50_D*100,minsEFFG50E_D*100,maxsEFFG50E_D*100,'g^'
,'MarkerEdgeColor','g','MarkerFaceColor','g','MarkerSize',8,'LineWidth',2);ho
ld on;
figure(10);errorbar(sHDC5_Trange,sEFFG5_T*100,minsEFFG5E_T*100,maxsEFFG5E_T*1
00,'co','MarkerEdgeColor','c','MarkerFaceColor','c','MarkerSize',8,'LineWidth
',2);hold on;
errorbar(sHDC25_Trange,sEFFG25_T*100,minsEFFG25E_T*100,maxsEFFG25E_T*100,'rs'
,'MarkerEdgeColor','r','MarkerFaceColor','r','MarkerSize',8,'LineWidth',2);ho
ld on;
errorbar(sHDC50_Trange,sEFFG50_T*100,minsEFFG50E_T*100,maxsEFFG50E_T*100,'g^'
,'MarkerEdgeColor','g','MarkerFaceColor','g','MarkerSize',8,'LineWidth',2);ho
ld on;

figure(8);
plot(Irange,polyval(f25I_EFF,Irange),'-.r','LineWidth',1.5);hold on;
plot(Irange,polyval(f50I_EFF,Irange),'-.g','LineWidth',1.5);hold on;
figure(9);
plot(Drange,polyval(f25D_EFF,Drange),'-.r','LineWidth',1.5);hold on;
figure(10);
plot(Trange,polyval(f25T_EFF,Trange),'-.r','LineWidth',1.5);hold on;
plot(Trange,polyval(f50T_EFF,Trange),'-.g','LineWidth',1.5);hold on;

figure(8);hold off;grid;title('Li-ion Experiments - Normalized EG vs Pulse
Amplitude','FontSize',16);
xlabel('Pulse Amplitude/A','fontsize',16);ylabel('Normalized
EG/%','fontsize',16);
```

```
legend('Expe - B & Hybrid 1.667F','Expe - B & Hybrid 8.33F','Expe - B & Hybrid
16.67F');%,'Expe 85% - B & Hybrid 8.33F','Expe 85% - B & Hybrid 16.67F');
set(gca,'fontsize',16);ylim([0 40]);
figure(9);hold off;grid;title('Li-ion Experiments - Normalized EG vs Duty
Cycle','FontSize',16);
xlabel('Duty Cycle/%','fontsize',16);ylabel('Normalized EG/%','fontsize',16);
legend('Expe - B & Hybrid 1.667F','Expe - B & Hybrid 8.33F','Expe - B & Hybrid
16.67F');%,'Expe 85% - B & Hybrid 8.33F','Expe 85% - B & Hybrid 16.67F');
set(gca,'fontsize',16);ylim([0 30]);
figure(10);hold off;grid;title('Li-ion Experiments - Normalized EG vs Pulse
Period','FontSize',16);
xlabel('Pulse Period/ms','fontsize',16);ylabel('Normalized
EG/%','fontsize',16);
legend('Expe - B & Hybrid 1.667F','Expe - B & Hybrid 8.33F','Expe - B & Hybrid
16.67F');%,'Expe 85% - B & Hybrid 8.33F','Expe 85% - B & Hybrid 16.67F');
set(gca,'fontsize',16);ylim([0 30]);
%%
%Experi DC
figure(11);errorbar(sBDC_Irange,sBDC_I,sBDCE_I.*sBDC_I,'b*','MarkerEdgeColor'
,'b','MarkerFaceColor','b','MarkerSize',8,'LineWidth',2);hold on;
figure(11);errorbar(sHDC5_Irange,sHDC5_I,sHDC5E_I.*sHDC5_I,'co','MarkerEdgeCo
lor','c','MarkerFaceColor','c','MarkerSize',5);hold on;
figure(11);errorbar(sHDC25_Irange,sHDC25_I,sHDC25E_I.*sHDC25_I,'rs','MarkerEd
geColor','r','MarkerFaceColor','r','MarkerSize',8,'LineWidth',2);hold on;
figure(11);errorbar(sHDC50_Irange,sHDC50_I,sHDC50E_I.*sHDC50_I,'g^','MarkerEd
geColor','g','MarkerFaceColor','g','MarkerSize',8,'LineWidth',2);hold on;

figure(12);errorbar(sBDC_Drange,sBDC_D,sBDCE_D.*sBDC_D,'b*','MarkerEdgeColor'
,'b','MarkerFaceColor','b','MarkerSize',8,'LineWidth',2);hold on;
figure(12);errorbar(sHDC5_Drange,sHDC5_D,sHDC5E_D.*sHDC5_D,'co','MarkerEdgeCo
lor','c','MarkerFaceColor','c','MarkerSize',5);hold on;
figure(12);errorbar(sHDC25_Drange,sHDC25_D,sHDC25E_D.*sHDC25_D,'rs','MarkerEd
geColor','r','MarkerFaceColor','r','MarkerSize',8,'LineWidth',2);hold on;
figure(12);errorbar(sHDC50_Drange,sHDC50_D,sHDC50E_D.*sHDC50_D,'g^','MarkerEd
geColor','g','MarkerFaceColor','g','MarkerSize',8,'LineWidth',2);hold on;

figure(13);errorbar(sBDC_Trange,sBDC_T,sBDCE_T.*sBDC_T,'b*','MarkerEdgeColor'
,'b','MarkerFaceColor','b','MarkerSize',8,'LineWidth',2);hold on;
figure(13);errorbar(sHDC5_Trange,sHDC5_T,sHDC5E_T.*sHDC5_T,'co','MarkerEdgeCo
lor','c','MarkerFaceColor','c','MarkerSize',5);hold on;
```

```
figure(13);errorbar(sHDC25_Trange,sHDC25_T,sHDC25E_T.*sHDC25_T,'rs','MarkerEd
geColor','r','MarkerFaceColor','r','MarkerSize',8,'LineWidth',2);hold on;
figure(13);errorbar(sHDC50_Trange,sHDC50_T,sHDC50E_T.*sHDC50_T,'g^','MarkerEd
geColor','g','MarkerFaceColor','g','MarkerSize',8,'LineWidth',2);hold on;

figure(11);
plot(Irange,polyval(fBI_DC,Irange),'-.b','LineWidth',1.5);hold on;
plot(Irange,polyval(f25I_DC,Irange),'-.r','LineWidth',1.5);hold on;
plot(Irange,polyval(f50I_DC,Irange),'-.g','LineWidth',1.5);hold on;
figure(12);
plot(Drange,polyval(fBD_DC,Drange),'-.b','LineWidth',1.5);hold on;
plot(Drange,polyval(f25D_DC,Drange),'-.r','LineWidth',1.5);hold on;
figure(13);
plot(Trange,polyval(fBT_DC,Trange),'-.b','LineWidth',1.5);hold on;
plot(Trange,polyval(f25T_DC,Trange),'-.r','LineWidth',1.5);hold on;
plot(Trange,polyval(f50T_DC,Trange),'-.g','LineWidth',1.5);hold on;

figure(11);hold off;grid;title('Li-ion Experiments - Discharge Capacity vs Pulse
Amplitude','FontSize',16);
xlabel('Pulse Amplitude/A','fontsize',16);ylabel('Discharge
Capacity/V','fontsize',16);
legend('Expe - B','Expe - Hybrid 1.667F','Expe - Hybrid 8.33F','Expe - Hybrid
16.67F');%,'Expe 85% - B','Expe 85% - Hybrid 8.33F','Expe 85% - Hybrid 16.67F');
set(gca,'fontsize',16);ylim([4.5 8]);
figure(12);hold off;grid;title('Li-ion Experiments - Discharge Capacity vs Duty
Cycle','FontSize',16);
xlabel('Duty Cycle/%','fontsize',16);ylabel('Discharge
Capacity/V','fontsize',16);
legend('Expe - B','Expe - Hybrid 1.667F','Expe - Hybrid 8.33F','Expe - Hybrid
16.67F');%,'Expe 85% - B','Expe 85% - Hybrid 8.33F','Expe 85% - Hybrid 16.67F');
set(gca,'fontsize',16);ylim([4 8]);
figure(13);hold off;grid;title('Li-ion Experiments - Discharge Capacity vs Pulse
Period','FontSize',16);
xlabel('Pulse Period/ms','fontsize',16);ylabel('Discharge
Capacity/V','fontsize',16);
legend('Expe - B','Expe - Hybrid 1.667F','Expe - Hybrid 8.33F','Expe - Hybrid
16.67F');%,'Expe 85% - B','Expe 85% - Hybrid 8.33F','Expe 85% - Hybrid 16.67F');
set(gca,'fontsize',16);ylim([4.5 8]);
```

Reference

[1] Cymbet Corporation Application Note AN-1025, "Use the EnerChip in Pulse Current Applications, http://www.cymbet.com/pdfs/AN-1025.pdf

[2] Maxwell Inc. Report, http://www.maxwell.com/products/ultracapacitors/docs/superior_tools_casestudy.pdf

[3] Maxwell Inc. Ultracapacitors for engine Start. http://www.maxwell.com/products/ultracapacitors/docs/200904_whitepaper_enginecoldstarting.pdf

[4] Donghwa Shin; Younghyun Kim; Jaeam Seo; Naehyuck Chang; Yanzhi Wang; Pedram, M., "Battery-supercapacitor hybrid system for high-rate pulsed load applications," Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011 , vol., no., pp.1,4, 14-18 March 2011

[5] P. Mindl and Z. Cerovsky, "Regenerative braking by electric hybrid vehicles using super capacitor and power splitting generator," Power Electronics and Applications, 2005 European Conference on, 2006.

[6] Alireza Khaligh, Zhihao Li, IEEE "Battery, Ultracapacitor, Fuel Cell, and Hybrid Energy Storage Systems for Electric, Hybrid Electric, Fuel Cell, and Plug-In Hybrid Electric Vehicles: State of the Art", IEEE Transactions on Vehicular Technology, Vol. 59, No. 6, July 2010.

[7] J.M. Miller, M. Everett, T. Bohn, and T.J. Dougherty, "Ultracapacitor plus Lithium-ion for PHEV: Technical and Economic Analysis," 26th International Battery Seminar & Exhibition, San Diego, CA: Maxwell Technologies, Inc, 2009.

[8] Arjan van Voorden, Laura Ramirez Elizondo, Gerard Paap, Jody Verboomen, Lou van der Sluis, "The application of super capacitors to relieve battery-storage systems in autonomous renewable energy systems", Power Tech, 2007 IEEE Lausanne

[9] R. Dougal, S. Liu, and R. White, "Power and life extension of battery-ultra capacitor hybrids," Components and Packaging Technologies, IEEE Transactions. Components, Packaging and Manufacturing Technology, vol. 25, 2002, pp. 120-131.

[10] T.A. Smith, G.A. Turner, and J.P. Mars, "Using supercapacitors to improve battery performance", Power Electronics Specialists Conference, 2002 (PESC 02), IEEE 33rd Annual, 2002, vol.1, pp. 124-128.

[11] Palma, L., Enjeti, P.; Howze, J. W., "An approach to improve battery run-time in mobile applications with supercapacitors" Power Electronics Specialist Conference, 2003. PESC '03. 2003 IEEE 34th Annual , vol.2, no., pp.918,923 vol.2, 15-19 June 2003

[12] Ian Smith, John G. Kassakian, Joel Schindall, "Benefits of Battery-Ultracapacitor Hybrid Energy Storage Systems", Master Degree Thesis, submitted in May 2012.

[13] Sherry Boschert, "Plug-in Hybrids: The Cars that will Recharge America", New Society Publishers, Gabriola Island, Canada., 2006

[14] I. Buchmann, Cadex Electronics Inc., "Batteries in a Portable World", Second Edition, 2000

[15] Battery and Energy Technologies, http://www.mpoweruk.com/lithiumS.htm

[16] Sharma, Pawan, and T. S. Bhatti., "A review on electrochemical double-layer capacitors", Energy Conversion and Management, 51.12 (2010): 2901-2912.
[17] Bullard, G. L.; Sierra-Alcazar, H.B.; Lee, H.-L.; Morris, J. L., "Operating principles of the ultracapacitor," Magnetics, IEEE Transactions on , vol.25, no.1, pp.102,106, Jan 1989
[18] J. Bockris, A.K.N. Reddy, Modern Electrochemistry, Vol. 2, Ch. 7, Plenum, 1
[19] D. A. New, "Double layer capacitors: automotive applications and modeling", M.S. Thesis, Dept. Elect. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, Feb. 2004.
[20] Signorelli, R.; Ku, D.C.; Kassakian, J.G.; Schindall, J.E., "Electrochemical Double-Layer Capacitors Using Carbon Nanotube Electrode Structures," Proceedings of the IEEE , vol.97, no.11, pp.1837,1847, Nov. 2009
[21] Riccardo Signorelli, John G. Kassakian, Joel Schindall, "High Energy and Power Density nanotube-enhanced ultracapacitor design, modeling, testing, and predicted performance", Ph.D Degree Thesis, submitted in June 2009.
[22] Wikipedia, Nickel-metal hydride battery, http://en.wikipedia.org/wiki/Nickel%E2%80%93_metal_hydride_battery
[23] Spyker, R.L.; Nelms, R.M., "Classical equivalent circuit parameters for a double-layer capacitor," Aerospace and Electronic Systems, IEEE Transactions on , vol.36, no.3, pp.829,836, Jul 2000
[24] Ariyoshi, G.; Harada, K.; Yamasaki, K.; Murita, K., "Load leveling using EDLCs under PLL control," Applied Power Electronics Conference and Exposition, 2000. APEC 2000. Fifteenth Annual IEEE , vol.2, no., pp.774,780 vol.2, 2000
[25] Global vehicles,'' Automotive Eng. Int., pp. 14–16, Dec. 2002.
[26] Dixon, J.W.; Ortuzar, M.E., "Ultracapacitors + DC-DC converters in regenerative braking system," Aerospace and Electronic Systems Magazine, IEEE , vol.17, no.8, pp.16,21, Aug 2002
[27] LaMonica, M, "Supercapacitor-Enhanced Hybrid Storage to Earn Cash for Subways", IEEE Spectrum, April, 2014
[28] Lisheng Shi; Crow, M.L., "Comparison of ultracapacitor electric circuit models," Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE , vol., no., pp.1,6, 20-24 July 2008
[29] Matsushita Electronic Components Co., Ltd., "Technical Guide of Electric Double-layer Capacitor", Matsushita Electronic Components Co., Ltd., 1992.
[30] Zubieta, L.; Bonert, Richard, "Characterization of double-layer capacitors for power electronics applications," Industry Applications, IEEE Transactions on , vol.36, no.1, pp.199,205, Jan/Feb 2000
[31] Dougal, R. A., L. Gao, and S. Liu. "Ultracapacitor model with automatic order selection and capacity scaling for dynamic system simulation." Journal of Power Sources 126.1 (2004): 250-257.
[32] L. Zubieta, "Characterization of double-layer capacitors for power electronics applications", M.A.Sc. thesis, Dep. of Elect. and Comput. Eng., Univ. of Toronto, Toronto, Ont., Canada, 1997.

[33] Rafik, Fouad, et al. "Frequency, thermal and voltage supercapacitor characterization and modeling." Journal of power sources 165.2 (2007): 928-934.
[34] Chia-Jui Chiang; Jing-Long Yang; Wen-Chin Cheng, "Dynamic modeling of the electrical and thermal behavior of ultracapacitors," Control and Automation (ICCA), 2013 10th IEEE International Conference on , vol., no., pp.1839,1844, 12-14 June 2013
[35] Min Chen; Rincon-Mora, G.A., "Accurate electrical battery model capable of predicting runtime and I-V performance," Energy Conversion, IEEE Transactions on , vol.21, no.2, pp.504,511, June 2006
[36] Einhorn, M.; Conte, F.V.; Kral, C.; Fleig, J., "Comparison, Selection, and Parameterization of Electrical Battery Models for Automotive Applications," Power Electronics, IEEE Transactions on , vol.28, no.3, pp.1429,1437, March 2013
[37] Baronti, F.; Zamboni, W.; Femia, N.; Rahimi-Eichi, H.; Roncella, R.; Rosi, S.; Saletti, R.; Chow, M.-Y., "Parameter identification of Li-Po batteries in electric vehicles: A comparative study," Industrial Electronics (ISIE), 2013 IEEE International Symposium on , vol., no., pp.1,7, 28-31 May 2013
[38] Kroeze, R.C.; Krein, P.T., "Electrical battery model for use in dynamic electric vehicle simulations," *Power Electronics Specialists Conference, 2008. PESC 2008. IEEE* , vol., no., pp.1336,1342, 15-19 June 2008
[39] Low Wen Yao; Aziz, J.A.; Ramli, N., "Detail analysis of RC parallel network-based model for high capacity Lithium Ferro Phosphates battery," Power Electronics, Machines and Drives (PEMD 2012), 6th IET International Conference on , vol., no., pp.1,6, 27-29 March 2012
[40] Baronti, F.; Zamboni, W.; Femia, N.; Roncella, R.; Saletti, R., "Experimental analysis of open-circuit voltage hysteresis in lithium-iron-phosphate batteries," Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE , vol., no., pp.6728,6733, 10-13 Nov. 2013
[41] Suleiman, A. and Doerffel, D., "Rapid test and non-linear model characterization of solid-state lithium-ion batteries." Journal of Power Sources130.1 (2004): 266-274.
[42] Panasonic HHR370AH battery datasheet, http://www.panasonic.com/industrial/includes/pdf/Panasonic_NiMH_HHR370AH.pdf
[43] A123 ANR26650-M1 battery datasheet, http://liionbms.com/pdf/a123/charging.pdf
[44] Maxwell HC series Ultracapacitors Datasheet, http://www.maxwell.com/products/ultracapacitors/docs/hcseries_ds_1013793-9.pdf
[45] Analog Device Inc. Circuit Note CN1051, "Versatile High Precision Programmable Current Sources Using DACs, Amps and MOSFET Transistors", http://www.analog.com/static/imported-files/circuit_notes/CN0151.pdf
[46] Microchip Application Note AN884, "Driving Capacitive Loads With Op Amps", http://ww1.microchip.com/downloads/en/appnotes/00884a.pdf
[47] James Roberge, Operational Amplifiers: Theory and Practice, published by John Wiley & Sons, Inc., 1975

[48] Analog Device Application Note, "Using High-Voltage Op Amps to drive Power MOSFETs ", 1993
[49] Atmel ATxmega192A3U Datasheet, http://www.atmel.com/Images/Atmel-8386-8-and-16-bit-AVR-Microcontroller-ATxmega64A3U-128A3U-192A3U-256A3U_datasheet.pdf
[50] AD 8671, Precision, Very Low Noise, Low Input Bias Current Operational Amplifier, http://www.analog.com/static/imported-files/data_sheets/AD8671_8672_8674.pdf
[51] Hussein, H.A.-H.; Batarseh, I., "A Review of Charging Algorithms for Nickel and Lithium Battery Chargers," Vehicular Technology, IEEE Transactions on , vol.60, no.3, pp.830,838, March 2011
[52] Panasonic, Inc. Charging Method for Nickle-Metal Hydride Batteries. https://industrial. panasonic.com/www-data/pdf/ACG4000/ACG4000PE2.pdf
[53] Chia-Hsiang Lin, Chi-Lin Chen, etc., "Fast Charging Technique for Li-Ion Battery Charger", 15th IEEE International Conference on Electronics, Circuits and Systems, 2008 (ICECS 2008).
[54] Vishay Precision Shunt Resistor FPR 2-T218, http://www.vishaypg.com/docs/64200/FPR -2-T218.pdf
[55] AD7606, 8-Channel DAS with 16-Bit Bipolar Input Simultaneous Sampling ADC, http://www.analog.com/static/imported-files/data_sheets/AD7606_7606-6_7606-4.pdf