REAL-TIME ESTIMATION OF STATE-OF-CHARGE USING PARTICLE SWARM OPTIMIZATION ON THE ELECTRO-CHEMICAL MODEL OF A SINGLE CELL

A Thesis

Submitted to the Faculty

of

Purdue University

by

Arun Chandra Shekar

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Mechanical Engineering

May 2017

Purdue University

Indianapolis, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL

STATEMENT OF THESIS APPROVAL

Dedicating this work to my parents Shantha and Chandrashekar and to my wife, Chaitra Chandrashekara. Their love and support is the impetus for my accomplishments.

ACKNOWLEDGMENTS

I am grateful to Dr. Sohel Anwar for his guidance, advice and patience throughout the course of this work.I would like to thank my friends Sourav and Bibin for their inputs and support.I would like to thank my supervisor Mr. David O'Dell who has always encouraged and supported me to pursue my goals. A special thanks to my colleagues for their support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

SYMBOLS

a	inter-facial surface area
A	cross-sectional area of an electrochemical cell
$C_{e,i}$	concentration in electrolyte phase
$C_{s,i}$	concentration in solid phase
D_e	diffusion coefficient of electrolyte
D_s	diffusion coefficient of solid
$f_{+/-}$	mean molar activity coefficient of electrolyte
F	Faraday's constant
i_n	transfer current at the surface of active material
i_o	exchange current density
L_i	thickness of electrode/separator
Q	charge capacity
R	universal gas constant
T	temperature
α_a, α_c	anodic and cathodic transfer coefficient
ϵ_i	volume fraction
κ_i	ionic conductivity in electrolyte
σ_i	ionic conductivity in solid matrix
$\Phi_{s,i}$	potential in solid matrix
$\Phi_{e,i}$	potential in electrolyte
ρ	material density

ABBREVIATIONS

SOC	State Of Charge
OCV	Open Circuit Voltage
CNG	Compressed Natural Gas
SPM	Single Particle Model
P2D	Pseudo-Two Dimensional
PSO	Particle Swarm Optimization
RMS	Root Mean Square Error
GA	Genetic Algorithm
Li-Ion	Lithium Ion

ABSTRACT

Chandra Shekar, Arun. M.S.M.E., Purdue University, May 2017. Real-Time Estimation of State-Of-Charge using Particle Swarm Optimization on the Electro-chemical Model of a Single Cell. Major Professor: Sohel Anwar.

Accurate estimation of State of Charge (SOC) is crucial. With the ever-increasing usage of batteries, especially in safety critical applications, the requirement of accurate estimation of SOC is paramount. Most current methods of SOC estimation rely on data collected and calibrated offline, which could lead to inaccuracies in SOC estimation as the battery ages or under different operating conditions. This work aims at exploring the real-time estimation and optimization of SOC by applying Particle Swarm Optimization (PSO) to a detailed electrochemical model of a single cell. The goal is to develop a single cell model and PSO algorithm which can run on an embedded device with reasonable utilization of CPU and memory resources and still be able to estimate SOC with acceptable accuracy. The scope is to demonstrate the accurate estimation of SOC for 1C charge and discharge for both healthy and aged cell.

1. INTRODUCTION

1.1 Overview

Environmental Protection agency of USA estimated that production of Electricity and Transportation contribute to 56% of greenhouse gases in 2014 as shown in Figure 1.1. Additionally, Environmental Protection Agency in Europe (Decision No 406/2009/EU) estimates that by 2020 the emissions just due to transportation may increase by 16% and may still increase by 10% even if the targets for additional measures, which sets the target of 10% renewable fuel in transport, are met.

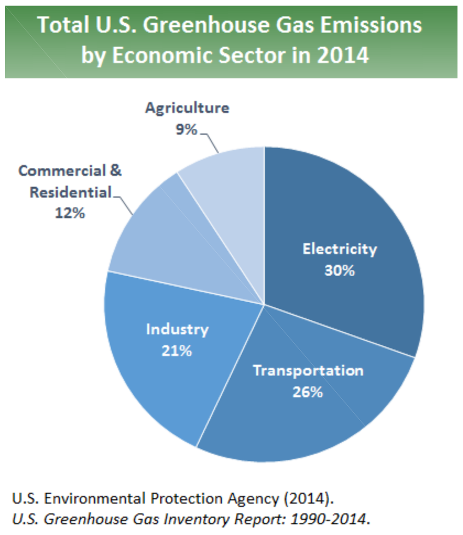

Fig. 1.1. US Greenhouse Gas Emission by Economic Sector

One of the measures proposed by the Environmental Protection Agency of USA, to reduce these emissions, is Fuel Switching. Use of public transport fueled by Compressed Natural Gas (CNG), increased use of hybrid and electrical vehicles are some of the examples of fuel switching. It is clear from this directive that usage of portable energy sources in transportation, like battery, will dramatically increase in the coming years. Apart from transportation, batteries are used in many devices we use in our daily lives in everything from consumer electronics, like cell phones, laptops etc., to safety critical medical devices, like pacemaker. Thus, the effective utilization and management of battery energy becomes paramount.

There are a lot of challenges in the field of battery management system, from choosing the right chemistry for an application, to understanding the operating conditions of a battery.

State of Charge is one of the most important parameters of a battery. It is analogous to the fuel gauge in an automobile. SOC indicates the amount of usable energy left in the battery at a given time. Accurately knowing SOC enables effective management of operations like range estimation, fast charging, diagnostics to name a few. SOC is also imperative in the safe operation of batteries, like Li-Ion, where overcharging and over discharging can prove hazardous.

However, SOC is not a quantity that can be measured directly in real time and must be estimated with the knowledge of other measurable quantities like voltage, current etc. There are a wide range of options available to estimate SOC depending on the application. This work explores Particle Swarm Optimization to accurately estimate the SOC under varying conditions of the battery. Another goal of this work is to be able to run this battery and estimator model on an embedded device with reasonable performance.

1.2 Major Contribution of Thesis Work

Particle Swarm Optimization is employed on a Single Cell Physics model with the goal of optimizing the SOC estimation. Particle Swarm optimization is a population based algorithm where the best solution is explored in a solution space. Historically optimization algorithms have been computationally demanding and seldom used in real time application but with the advancement in embedded computers in the recent years, the goal of running optimization in real time is reasonable The contributions of this work are listed below:

- Development of Single Cell model in Matlab Simulink.
- Develop PSO algorithm and verify operation in simulation with experimental data.
- Verification of operation in real time, particularly on a Raspberry Pi 3.

1.2.1 Organization of Thesis

The document is organized into 6 chapters. Chapter 1 provides a brief introduction and the motivation for this work. Chapter 2 explores the different SOC estimation techniques and provides an overview of battery chemistries. It also lists the different battery parameters and definitions used. Chapter 3 deals with the overview of battery modeling techniques and details the battery model used in this work.The latter part of chapter 3 details the implementation of the chosen battery model in Simulink©and presents the data of theoretical validation. Chapter 4 covers the PSO algorithm overview and implementation. Chapter 5 deals with the validation of developed algorithm, it explains the experimental setup to collect data for validation. It presents the simulation and real-time validation data. Finally Chapter 6 offers conclusion and proposal of future work.

2. LITERATURE SURVEY

2.1 Battery Overview

2.1.1 Definition

A battery is an electrochemical device which converts chemical energy to electrical energy by a reduction-oxidation reaction [1]. It would be imperative to discuss the definition and distinction of a cell and a battery here. A Cell is a basic unit which delivers power through the process of electrochemical reaction. Most non-rechargeable batteries that we use in our daily life are in fact cells. Battery is a more popular term that is used commercially [1]. A Battery or a Battery Pack is the arrangement of cells, either in series, parallel or both, to achieve desired performance as per requirement. A popular example of this is Lead Acid Battery in automobiles. Electrochemical Cells and Batteries are broadly classified as Primary and Secondary.

2.1.2 Classification

Primary cells or batteries are non-rechargeable and they are discarded once they are discharged. They tend to be inexpensive, lightweight, portable with a long shelf life [1]. These features make them useful in consumer electronics like watches, toys, cameras etc. Secondary cells or batteries are rechargeable and they are cycled through many charge-discharge cycles thought the lifespan of their application. They are characterized by high power density, high discharge rate but tend to have a lower retention rate or shelf life. These features make them useful in a wide range of application from consumer electronics to Electric Vehicle applications. There are other battery types like Reserve Battery and Fuel Cells although they still fall under the same broad classification of Primary and Secondary cells or batteries.

2.1.3 Electrochemical Process and Battery Chemistry

A cell primarily consists of three parts the anode or the negative electrode, the cathode or the positive electrode and the electrolyte which is the medium in which the ions are exchanged between the cathode and the anode [1]. During the discharge process, anode gives up electrons and thus oxidized. These electrons travel through the external circuit and reach the cathode. Cathode accepts the electrons and thus getting reduced. The negative ions called anions and positive ion called cation are exchanged through the electrolyte. The reverse of this process occurs during charging, if the cell is rechargeable. The selection of the Anode, Cathode and Electrolyte material, which also referred to as the battery chemistry, determine the performance parameters of a battery. Typical characteristics considered in choosing the materials are as follows [1]:

Anode:

- Efficient reducing agent
- High Coloumbic Output.
- Low cost to produce and fabricate.
- Good conductivity and stability.

Cathode:

- Efficient Oxidizing agent
- Stability in operation.
- Low cost to produce and fabricate.

Electrolyte:

- Good ionic conductivity.
- Stability in operation.
- Low cost to produce and fabricate.

Table 2.1 lists the Capacity, atomic/molecular weight and type(cathode or anode) of some electrodes. Lithium (Li), with an atomic weight of 6.94 g, is the lightest

Table 2.1.
Characteristics of Electrodes [1]

Material	Type	Atomic/Molecular Weight (g)	Capacity (Ah/g)
H_2	Anode	2.01	26.59
Li		6.94	3.68
Na		23.0	1.16
Mg		24.3	2.20
Pb		207.2	0.26
O_2	Cathode	32.0	3.35
MnO_2		86.9	0.308
Li_xCoO_2		98	0.137

metal but still has one of the highest capacity at 3.86 Ah/g. Thus, Li-Ion chemistry is one of the most popular chemistry used especially in the transport industry.

The chart 2.1 shows comparison of specific energy of some of the popular battery chemistry.

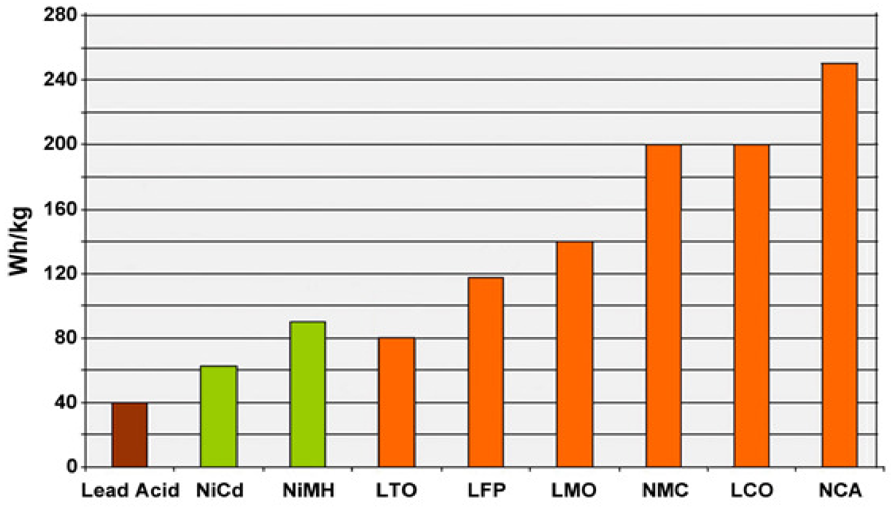

Fig. 2.1. Typical Energy Densities of Lead, Nickel and Lithium-based Batteries. [2]

Additionally, Li-Ion battery has the following advantages:

- High specific energy.
- Does not have the memory effect between charge/discharge cycles.
- Wider operating temperatures.
- Higher Open Circuit Voltage.

Some of the drawbacks of Li-Ion are:

- Hazardous if not operated under safe conditions.
- Does not age gracefully.

Some of the popular Li-Ion chemistries are along with their abbreviation is summarized in the table 2.2

Table 2.2.
Popular Li-ion Battery Chemistries. [2]

Chemical Name	Material	Abbreviation	Short Form
Lithium Cobalt Oxide	$LiCoO_2$	LCO	Li-cobalt
Lithium Manganese Oxide	$LiMn_2O_4$	LMO	Li-manganese
Lithium Iron Phosphate	$LiFePO_4$	LFP	Li-phosphate
Lithium Nickel manganese Cobalt Oxide	$LiNiMnCoO_2$	NMC	NMC
Lithium Nickel Cobalt Aluminium Oxide	$LiNiCoAlO_2$	NCA	NCA
Lithium Titanate	$Li_4Ti_5O_{12}$	LTO	Li-titanate

Referring to the Figure 2.1 it is noted that Li-Cobalt (LCO) has one of the highest energy density thus the equation of Lithium Cobalt Oxide battery are further explored:

$$C + LiCoO_2 \leftrightarrow LiC_6 + Li_{0.5}CoO_2 \tag{2.1}$$

At the Cathode:

$$LiCoO_2 - Li^+ - e^- \leftrightarrow Li_{0.5}CoO_2 \Rightarrow 143mAh/g \tag{2.2}$$

At the Anode:

$$6C + Li^+ + e^- \leftrightarrow LiC_6 \Rightarrow 372mAh/g \tag{2.3}$$

2.1.4 Battery Parameter

Batteries are identified by the amount of energy they can hold and deliver. Some of the definitions that are required in the study of battery are:

Rated Capacity: Capacity of a battery is its ability to deliver power to an external circuit. Manufacturer specifies the Rated Capacity in Ah (Ampere Hours).
Energy Density: It is energy that can be delivered by the battery expressed in Wh/L (Watthour/liter).
Specific Energy: It is the energy that can be delivered by the battery expressed in Wh/kg (Watthour/kilogram).
State-Of-Charge: It is measure of capacity or charge contained in a battery. It is expressed as a ratio of available capacity to the rated capacity. It is also expressed as a percentage of the rated capacity.
State-Of-Health: It is a measurement that determines the condition or health of the battery. SOC is measure in percentage where 100% denotes a battery that can deliver 100% to the rated specification of the manufacturer.

2.2 State-Of-Charge Estimation Techniques

SOC is analogous to the fuel gauge in an automobile, which makes it a very crucial measurement in designing any device that uses a battery. However, there is no direct and easy method to measure SOC. In all cases, it must be estimated or determined by establishing SOC as a function of other measurable signals like Voltage, Current, Cell Temperature etc. This paradigm makes SOC estimation one of the most researched topics. There are a wide variety of techniques available for SOC estimation [3]. They can be broadly classified into three categories [4], which are:

- Non-model based method like Ampere hour counting.
- Computational Intelligence and Optimization based methods, like Fuzzy logic, Particle Swarm Optimization.
- Estimation based methods, like variations of Kalman filter using equivalent circuit and state space models.

Both the Computational Intelligence & Optimization and Estimation based methods are considered as online estimation methods as the SOC is estimated in real-time. Some of these techniques are discussed in this section.

2.2.1 Ampere Hour Counting

Ampere hour counting or Coulomb Counting is the most popular and easy technique to determine SOC [3]. The charge is directly proportional to the current supplied during charging and the current withdrawn during discharge operation. The current can be integrated as shown in the equation 2.4 to determine the SOC

$$SOC = SOC_0 + \frac{1}{C_N} \int_{t_o}^{t} (I_{batt} - I_{loss}) dt \tag{2.4}$$

Where SOC_0 is the initial SOC, C_N the rated capacity. I_{batt} is the current and I_{loss} is the current loss in the system. From the equation it is clear that this technique is subject to inaccuracies in charge/discharge current measurement, knowledge of accurate initial SOC and rated capacity. If the accurate initial SOC and Capacity are unknown, then it could lead to offset in estimated SOC. Inaccurate measurement of current can add up over time, due to integration, leading to the drift of estimated SOC from the actual SOC. This can be overcome by resetting the SOC when certain conditions like full charge are reached [3]. But to achieve such a condition, for example, in an electric vehicle application would be impractical.

2.2.2 Open Circuit Voltage

Open Circuit Voltage (OCV) is the voltage across the battery/cell terminal under no load condition. SOC bears a linear relationship to OCV [3]. This property makes OCV a suitable candidate to directly determine SOC. However, this technique would be suitable only in applications where there are rest periods [5] to take the OCV readings. Lead acid battery, having a very linear relation with SOC, is a good candidate for this technique but for batteries like Li ion (LiFePO4) where the OCV is flat in lower operating voltages, 2.0 to 3.65V which corresponds to 20% - 80%, even a small error in measurement of OCV would lead to large estimation errors [6]. For the aforementioned reasons this technique would not be suitable for online estimation.

2.2.3 Estimation Based

There are a good number of online estimation techniques proposed for the estimation of SOC. Kalman Filter method and its variants are one of the important methods. The series of papers by Plett [7] [8] [9] proposes Extended Kalman filter method to estimate battery states like SOC, power fade, capacity fade and instantaneous available power. The general requirement for such an estimation technique is the definition of SOC as a function of measurable signals like Voltage or Current. However, the parameters of the battery for different chemistries, different operating conditions and different health condition must be determined. A wide variety of techniques are employed to determine the battery parameters, from off-line least squared method to on-line PSO based method. More modern methods like Moving Horizon Estimator [10] and Particle Swarm Optimization [11] have been proposed for the estimation of SOC.

2.2.4 Computational Intelligence & Optimization

Computational intelligence like Genetic Algorithm and Fuzzy logic are proposed to estimate SOC [12] [13] [14]. The models need to be trained off-line to learn various operating conditions and scenarios. The model then estimates the states,in real-time, with the knowledge of measured inputs, both current and historical.

3. BATTERY MODEL

3.1 Modeling Techniques

There is a wide variety of battery models available. The right model must be chosen depending on the application and its performance requirements. Some applications may need an elaborate model to accurately predict the states of the battery while others may need a model for fast computation in real time. A few modeling techniques are discussed in this section.

One of the most popular modeling technique used is the Equivalent Circuit or RC model, here the battery is constructed using resistors(R) and Capacitors(C) to model the battery behavior. The order of the model, that is the number of R and C cells in the circuit, determines the accuracy. The higher the order, greater is the ability of the model to account for different physical phenomenon like, Internal Polarization [15]. However, more parameters are involved with higher order model, which makes identifying and tuning them more difficult therefore a trade-off must be reached between acceptable accuracy and model performance. Even with a good RC model, it must be tuned for every operating condition, like healthy battery, aged battery, overcharged battery etc. In a dynamic system, such as a Hybrid Vehicle, the application of such a model would be less practical.

Another technique, a purely mathematical one, is Empirical Modeling. Here the experimentally obtained historical data is used to realize a model where the different battery parameters and characteristics are represented using mathematical relations like, exponential, polynomial etc. Evidently these models are computationally efficient. However, they must be re-tuned to represent different operating conditions, which means that the data for every such scenario should be available.

Fuzzy Logic and Neural Network techniques are also used to model battery behav-

ior [12] [13] [14]. These techniques involve a learning process where the models are trained on experimental data to learn the relationship between the different battery parameters. These models are then used to predict various quantities and states, like voltage, SOC etc. Similar to Empirical models, these models require large amounts of data sets to be efficient. Additionally, the learning process could be complex and time consuming.
Electrochemical or Physics based models incorporate both the chemical and the electrical behavior to accurately model a battery. There are many models available with varying complexities. The Single Particle Model (SPM) models the battery as constituent particles in the active material. A model considering the diffusion phenomenon of such particles and intercalation within the particle was developed by Zhang et al [16]. Doyle et al [17] developed a more comprehensive model based on concentrated solution theory. This model, also known as Pseudo 2-Dimensional (P2D) model, aims to represent the behavior of wide range of Lithium Ion batteries with different combination of electrolytes, separators, anodes and cathodes.

3.2 Electrochemical Single Cell Model

The goal of this thesis is to develop a PSO algorithm for Li-Ion battery which shall, efficiently and seamlessly, work for different operating conditions and at the same time it shall be computationally efficient to run on an embedded device in real time. The equivalent circuit model is computationally efficient but needs re-tuning for every operating condition. The Empirical model and the Genetic Algorithm models need a lot of experimental data to be efficient. The electrochemical model developed by Doyle et al [17] encompasses the battery physics to give more accurate estimation of the states and the parameters but this model is not suitable for running in real-time. However, there has been a lot of research done based on this model. One such example is the model developed by Subramanian et al. [18], derived from first principles as an isothermal pseudo-two-dimensional model. One model of interest is the model developed by Smith et al. [19] which is a control oriented, one-dimensional electrochemical model. Based on this 1D model, Sourav et al. [20] developed and validated a discrete model solved using Finite Difference Method (FDM). This model demonstrates both the detail of battery physics and the computational efficiency to run in real-time. This model shall be used in this thesis work.

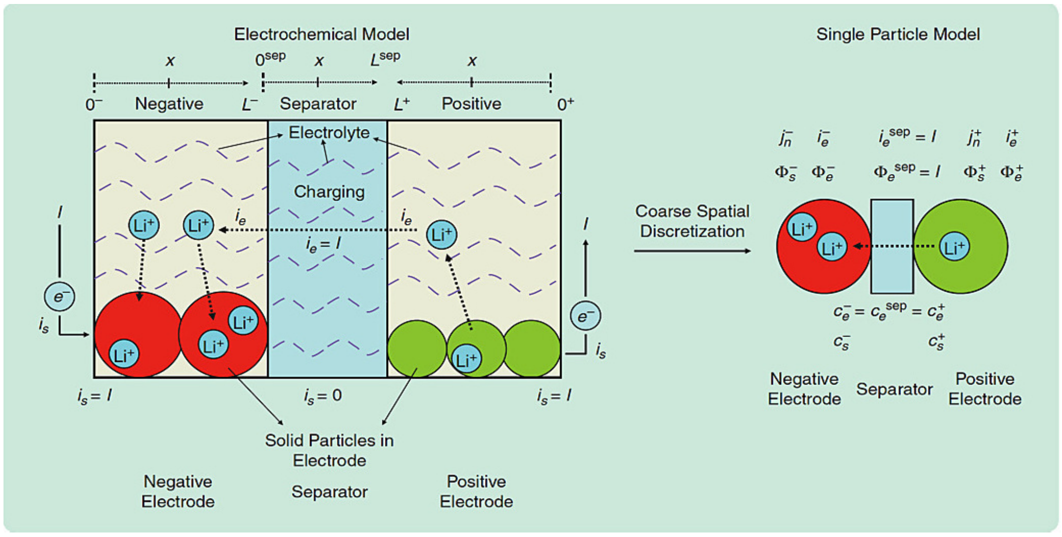

Fig. 3.1. Single Particle Model. [21]

Figure 3.1 gives a diagrammatic view of the modeling approach. Here the lithium ions are assumed to be constituted in spherical particles of radius R [21] along the X-axis. It is a one-dimensional model where only the reactions along the X-axis are considered.

3.3 Constituent Equations

Figure 3.2 gives an overview of a battery model based on 1-D Electrochemical model along with its governing equations.

This model has six states, which are :

Electrolyte lithium concentration $C_e(x,t)$

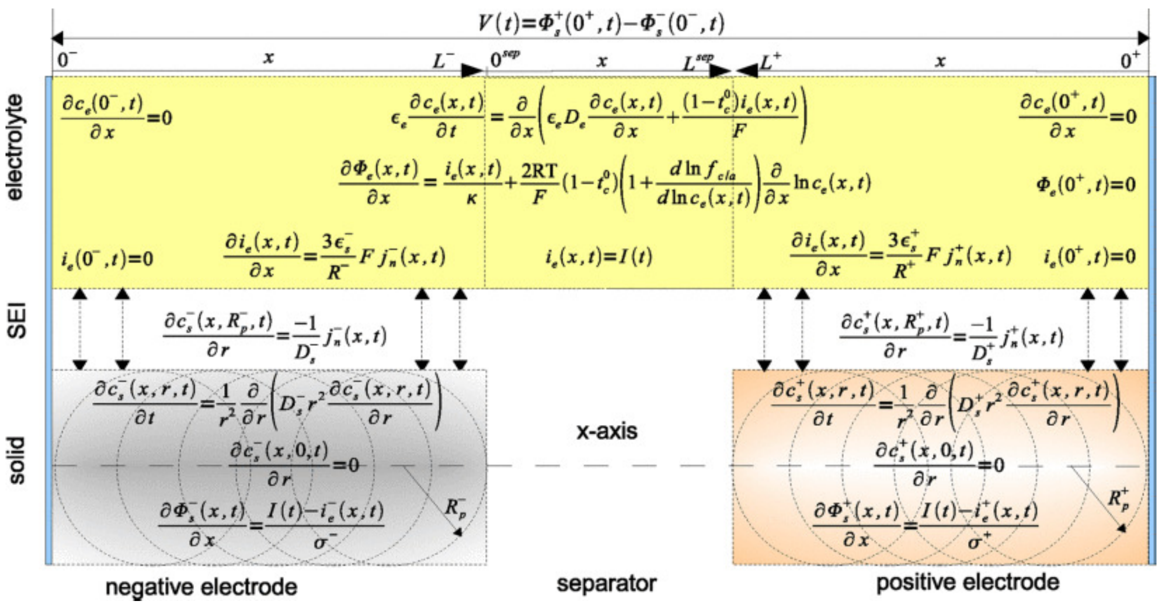

Fig. 3.2. Lithium Ion Cell Structure. [22]

Electrode lithium concentration $C_s(x,r,t)$ both anode and cathode

Electrolyte potential $\phi_e(x,t)$

Electrode potential $\phi_s(x,t)$ both anode and cathode

Electrolyte ionic current $i_e(x,t)$

Molar ionic flux $j_n(x,t)$

The constituent equations are as follows ([17] - [23])

$$\epsilon_p \frac{\partial C(x,t)}{\partial t} = D_{eff,p} \frac{\partial C(x,t)^2}{\partial x} + a_p(1-t_+^0)j_p \tag{3.1}$$

Reordering the equation we get:

$$\epsilon_e \frac{\partial C_e(x,t)}{\partial t} = \frac{\partial}{\partial x}(\epsilon_e D_e \frac{\partial C_e(x,t)}{\partial x} + \frac{1-t_+^0}{F} i_e(x,t)) \tag{3.2}$$

$$\frac{\partial C_{s,i}(x,r,t)}{\partial t} = \frac{1}{r^2}\frac{\partial}{\partial r}(D_{s,i} r^2 \frac{\partial C_{s,i}(x,r,t)}{\partial r}) \tag{3.3}$$

This is further simplified as:

$$\frac{\partial C_s(t)}{\partial t} = -\frac{3}{R_p} J(t), \tag{3.4}$$

$$\frac{\partial \phi_e(x,t)}{\partial x} = -\frac{i_e(x,t)}{\kappa} + \frac{2RT}{F}(1 - t_+^0) \times (1 + \frac{dlnf_{c/a}(x,t)}{dlnC_e(x,t)})\frac{\partial lnC_e(x,t)}{\partial x} \tag{3.5}$$

$$\frac{\partial \phi_s(x,t)}{\partial x} = \frac{i_e(x,t) - I(t)}{\sigma} \tag{3.6}$$

$$\frac{\partial i_e(x,t)}{\partial x} = \frac{3\epsilon_s}{R_p} F j_n(x,t) \tag{3.7}$$

$$j_n(x,t) = \frac{i_0(x,t)}{F}(e^{\frac{\alpha_a F}{RT}\eta(x,t)} - e^{\frac{\alpha_c F}{RT}\eta(x,t)}) \tag{3.8}$$

In Equation 3.8, the exchange current density $i_0(x,t)$ and the over potential $\eta(x,t)$ for the main reaction are modeled as:

$$i_0(x,t) = r_{eff} C_e(x,t)^{\alpha_c}(C_s^{max} - C_{ss}(x,t))^{\alpha_a} C_s(x,t)^{\alpha_c}, \tag{3.9}$$

$$\eta(x,t) = \phi_s(x,t) - \phi_e(x,t) - U(C_{ss}) - FR_f j_n(x,t), \tag{3.10}$$

where $c_{ss}(x,t) \approx c_s(x, R_p, t), U(c_{ss}(x,t))$ is the open circuit potential of the active material and c_s^{max} is the maximum concentration in the active material of each electrode. The internal temperature is described by:

$$\begin{aligned} \rho^{avg} c_p \frac{dT(t)}{dt} &= h_{cell}(T_{amb}(t) - T(t)) + I(t)V(t) \\ &\quad - \int_{0^-}^{0^+} \frac{3\epsilon_s}{R_p} F j_n(t)(U(\bar{c}_s(x,t)) - T(t)\frac{\partial U(\bar{c}_s(x,t))}{\partial T})dx, \end{aligned} \tag{3.11}$$

where, T_{amb} is the ambient temperature and $\bar{c}(x,t)$ represents the volume averaged concentration of a particle in the solid phase defined as

$$\bar{c}_s(x,t) = \frac{3}{R_p^3}\int_0^{R_p} r^2 c_s(x,r,t)dr, \tag{3.12}$$

The initial conditions of the battery model are given by:

$c_e(x,0) = c_e^0(x)$, $c_s(x,r,0) = c_s^0(x,r)$, $T(0) = T^0$,

and the boundary conditions are given by:

$$\frac{\partial c_e(0^-,t)}{\partial x} = \frac{\partial c_e(0^+,t)}{\partial x} = 0, \tag{3.13}$$

$$c_e(L^-,t) = c_e(0^{sep},t), c_e(L^{sep},t) = c_e(L^+,t), \tag{3.14}$$

$$\epsilon_e^- D_e \frac{\partial c_e(L^-,t)}{\partial x} = \epsilon_e^{sep} D_e \frac{\partial c_e(0^{sep},t)}{\partial x}, \tag{3.15}$$

$$\epsilon_e^{sep} D_e \frac{\partial c_e(L^{sep},t)}{\partial x} = -\epsilon_e^+ D_e \frac{\partial c_e(L^+,t)}{\partial x}, \tag{3.16}$$

$$\frac{\partial c_s(x,0,t)}{\partial r} = 0, \frac{\partial c_s(x,R_p,t)}{\partial r} = -\frac{j_n(x,t)}{D_s}, \tag{3.17}$$

$$\phi_e(L^-,t) = \phi_e(0^{sep},t), \phi_e(L^{sep},t) = \phi_e(L^+,t), \tag{3.18}$$

$$\phi_e(0^+,t) = 0, \tag{3.19}$$

$$i_e(0^-,t) = i_e(0^+,t) = 0, i_e(x^{sep},t) = -I(t), \tag{3.20}$$

$$i_e(L^-,t) = -i_e(L^+,t) = -I(t), \tag{3.21}$$

where, $x^{sep} \in [0^{sep}, L^{sep}]$ represents the entire separator domain of the battery.

In the above equations, ϵ_e, ϵ_s, σ, R, R_p, F, α_a, α_c, ρ_{avg}, c_p, h_{cell} and t_c^0 are model parameters and are constant in each region of the cell. κ, $f_{c/a}$ and D_e are known functions of the electrolyte concentration.

The voltage is represented as:

$$V(t) = \phi_s(0^+, t) - \phi_s(0^-, t), \tag{3.22}$$

This model has a nominal capacity of 3.5Ah. The SOC is directly calculated from the Lithium Ion concentration as shown in the equation 3.23

$$SOC = \frac{\overline{Cs}}{Cs_{max}} \tag{3.23}$$

where

$$\overline{Cs} = \frac{\sum_{i=1}^{N} Cs(i,t)}{N} \tag{3.24}$$

N being the number of nodes in the electrode.

The Lithium ion concentration of the Negative Electrode is used to estimate the SOC. It is observed that the lithium ion concentration in the negative electrode demonstrates the range and direction of the actual SOC, meaning SOC increases with charge and decreases with discharge. Lithium ion concentration of the positive electrode lacks the range and is also opposite in direction.

3.4 Reformulation of Battery model

The two main goals of this work are:

- Application of PSO for SOC estimation.
- Ability to run the algorithm in real time on an embedded target.

To achieve the above goals, the model needs to be re-formulated and the requirement for each of the above goals are discussed here.
An optimization algorithm like PSO, evaluates the function with a set of parameters to determine an optimal solution. Additionally, this process could be repeated at every time instance. In the case of a battery model, it would have to be run with

a set values for a given state, evaluating the fitness of the output, at each value, to the given optimization criteria. This requirement demands the battery model be modular, where it can be run repeatedly within a given time-step.

The equations for positive and negative electrodes, Eq. 3.4, 3.6, 3.8, 3.9 and 3.10, are similar. Hence all these equations can grouped together into a function. Similarly, the equations for the electrolyte and temperature calculation can be grouped into corresponding functions. Also, to run the model on an embedded target, it is preferred to be in a high-level language like C, C++, where the code can be compiled into real time executables.

Matlab©Simulink©offers a platform where we can not only develop the modular model but we can also simulate and analyze the results. Additionally, it offers rapid prototyping on embedded hardware, like Raspberry Pi. Once the Battery Model and PSO algorithm are developed and validated in Simulink©it can generate auto code and can also be run on an embedded device with a click of a button.

3.4.1 Simulink Implementation

The core of the battery model is implemented as a Simulink library for modularity. Figure 3.3 shows the blocks *UpdatePosElectrode* , *UpdateNegElectrode* , *UpdateElectrolyte* and *UpdateTemperature*, which update the states of Positive Electrode, Negative Electrode, Electrolyte and the cell temperature correspondingly at every time step.

The blocks in the Figure 3.3 are used to create the single cell model as shown in the Figure 3.4. The top level modular implementation of a single cell is shown in 3.5. It takes Ambient temperature(T_Amb) and Measured Current (I) as Inputs. Voltage, Open Circuit Voltage(opc), State of Charge at the Negative Electrode(SOC_Neg) and State of Charge of Positive Electrode (SOC_Pos) are the model outputs. The states are calculated and updated internally. This modular approach makes the model efficient for prototyping and analysis.

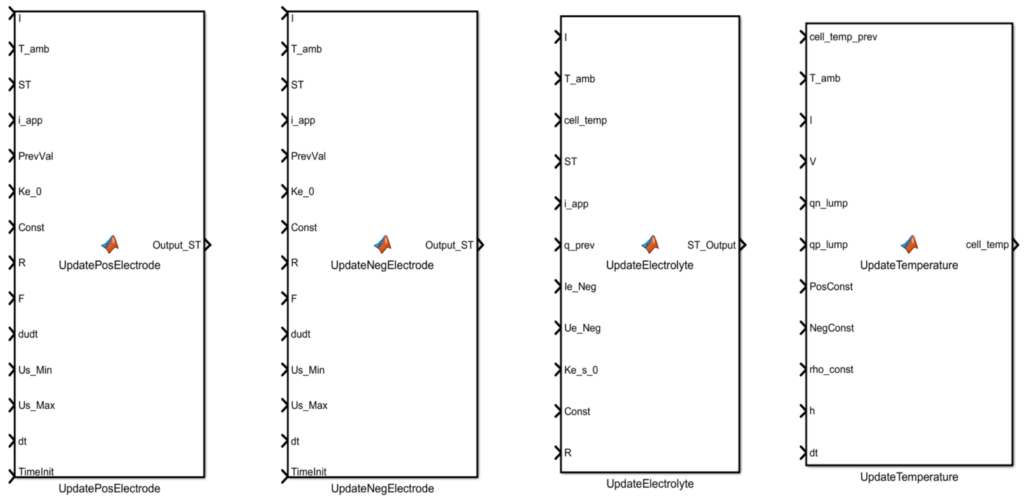

Fig. 3.3. Single Cell Battery Core Library.

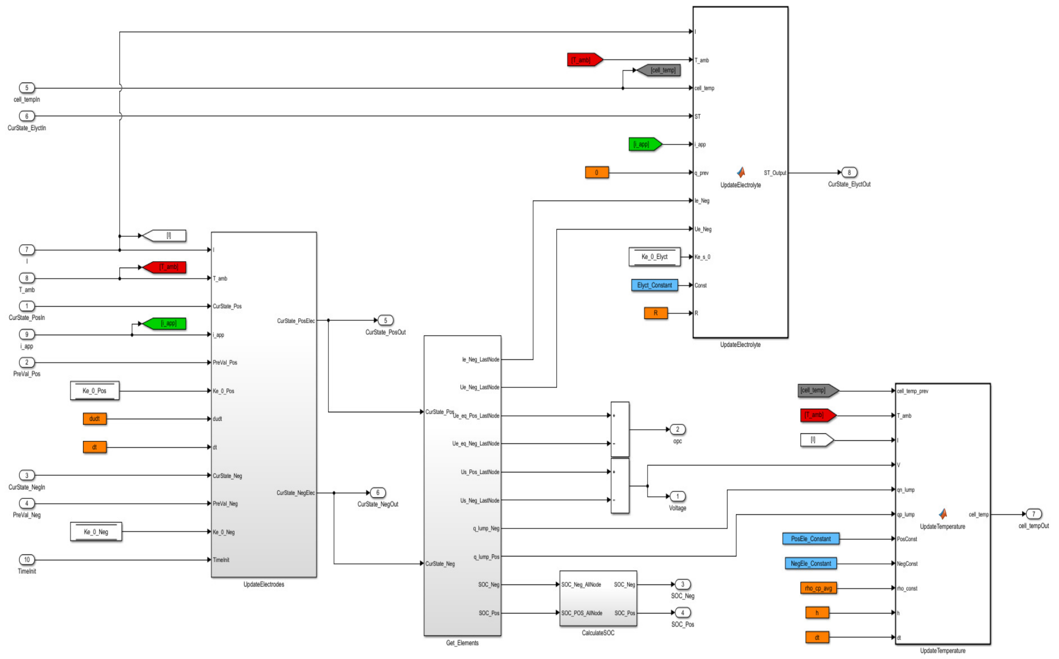

Fig. 3.4. Single Cell Battery Core Library.

3.4.2 Theoretical Simulation Validation

This model was subjected to a standard discharge rate of 1C (30 A/m^2) and 0.5C (15 A/m^2) in order to verify the battery operation in theory. The battery parameters

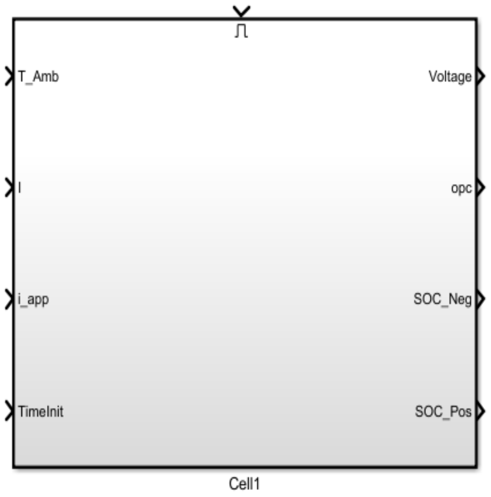

Fig. 3.5. Single Cell Battery Model.

used in this work are listed in the Table 3.1. The response in Figure 3.6 indicates that at 1C the battery discharges in 3500s and at 0.5C it takes 7000s to completely discharge. This is in tune with the expected response of a battery with a nominal capacity of 3.5Ah.

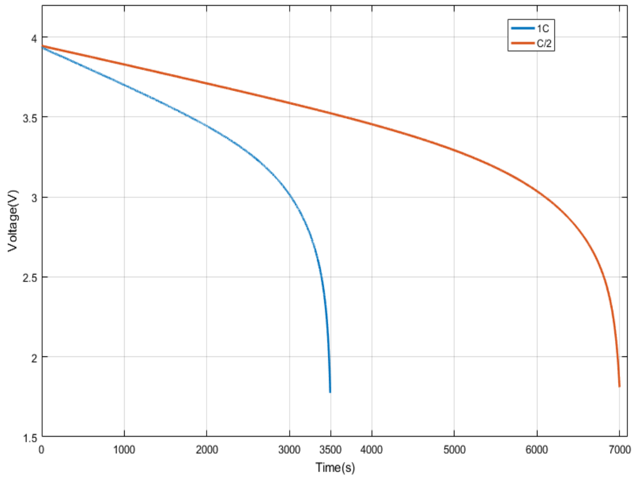

Fig. 3.6. Single Cell Battery Model.

Table 3.1.
Battery Parameters. [18]

Parameter	Units	Negative Electrode	Separator	Positive Electrode
L	m	88×10^{-6}	35×10^{-6}	80×10^{-6}
N	-	88	35	80
R_i	m	2×10^{-6}	-	2×10^{-6}
ϵ_s	-	0.4824	-	0.5
ϵ_e	-	0.485	0.724	0.385
σ	$\Omega^{-1}m^{-1}$	100	-	100
${t_+}^0$	-	0.363	0.363	0.363
$Cs_{max,i}$	$molm^{-3}$	30555	-	51554
$c_{e,i}$	$molm^{-3}$	1000	1000	1000
α_a, α_c	-	0.5, 0.5	-	0.5, 0.5
R	$jmol^{-1}K^{-1}$	8.314	8.314	8.314
F	$Cmol^{-1}$	96487	96487	96487
A	m^2	30	-	30

4. SOC ESTIMATION

4.1 Particle Swarm Optimization

Particle Swarm Optimization, inspired by movement of a flock of birds or a school of fish, is a population based stochastic optimization technique. It was developed by Dr. Kennedy and Dr. Eberhart in 1995. The idea of the algorithm is that a set of random solutions, called particles, are initialized in the solution space and the function is evaluated at each of this solution for fitness. The velocity of the best particle is then updated so that the other particles fly with the same velcity of the best solution. This is analogous to flock of birds flying around looking for optimal path for food. PSO had several advantages over GA, namely, PSO had fewer parameters to tune, the algorithm is computationally efficient and has a higher degree of convergence. After the advent of PSO it has been used in a wide variety of industries. It has been applied in robotics for controller optimization [24], it has been used to optimize the estimation of global solar radiation for regions where they cannot be measured [25], a biological application where PSO is used in detection of Ovarian cancer can also be cited [26], it has also been used in manufacturing industry to optimize machine loads [27]. PSO has also been used in the automotive applications. One of the most popular use of PSO is battery application is to identify battery parameters [28] [29] [30]. Another popular application is in optimization of energy management in Hybrid Vehicles [31] [32] [33]. A particular work of interest is the PSO based SOC estimation of Li-Ion battery by Ismail, N.H.F. et al [11],in which a Thevenin model was employed for the estimation of SOC of LiFePO4 battery. A Mean Square Error of 0.2633 was demonstrated using the PSO estimation. Thevenin model or the Equivalent circuit model has its drawback for SOC estimation as discussed in the literature survey.

4.2 PSO Algorithm

A global version of PSO algorithm is used in this thesis work. In this algorithm, a swarm of solution, called particles, are initialized randomly in the solution space. These particles are then flown through the solution space. The particles keep track of its coordinates with respect to the best solution in the problem space. This is termed as pbest. Similarly, a global best, termed as gbest, is tracked which is the best location or solution obtained by any particle so far. At every time step the velocities of the particles are updated towards the pbest and the gbest values until the required optimization goals are met or the algorithm termination condition is reached.

A basic outline of the algorithm is presented here. It describes a systematic approach of the algorithm:
Step 1: Initialize the particles
Initialize the position array with random numbers having uniform distribution

$$X = U_{rand}(r_{lowerlim}, r_{upperlim}) \tag{4.1}$$

Assign this initial positon to best known position array.

$$P = X \tag{4.2}$$

Initialize particle Velocity

$$V = X \tag{4.3}$$

If the number of particles are Num_p then, X is a Num_p size array of particle position, similarly P is a Num_p size array of *pbest* positions and V is a Num_p size array particle velocities.
Step 2: Evaluate the optimization fitness function-

$$E_x = F(X) \; and \; E_p = F(P) \; and \; e_g = f(gbest) \tag{4.4}$$

Where E_x and E_p are the fitness evaluation array for X and P correspondingly.
e_g is the function evaluation at *gbest*
Step 3: Update *pbest* value for each particle of the population-

$$if\ E_x(i)\ <\ E_p(i)\ then\ P(i)\ =\ X(i) \tag{4.5}$$

Where $i = 1, 2, ...Num_p$.
Step 4: Update *gbest* value for the entire population -

$$if\ E_p(i)\ <\ e_g\ then\ gbest\ =\ P(i) \tag{4.6}$$

Step 5: Update the velocity and position of the particles-

$$V(i) = wV(i)\ +\ c_1u_{rand}(0,1)(P(i) - X(i))\ +\ c_2u_{rand}(0,1)(gbest - X(i)) \tag{4.7}$$

$$X(i)\ =\ X(i)\ +\ V(i) \tag{4.8}$$

Where w is the inertial weight, c_1 is the cognitive parameter and c_2 is the social parameter.
Step 6: If criteria is met then exit else loop to Step 2. Exit criteria is usually a fitness threshold or maximum number of iterations completed.

4.3 SOC Estimation Approach

An overview of the approach of SOC estimation using PSO is show in the Figure 4.1. The measured voltage and current is supplied to the PSO algorithm, which optimizes the states in order to produce minimum error between the Measured Voltage and the Estimated Voltage from the battery model for the given current value. The battery model has six states but for simplicity only the lithium ion concentration, $C_s(x, r, t)$, of the electrode is used as particles of PSO in this work. C_s is the amount of charge in the electrode, which is intuitively the SOC, also the charge in the electrodes is directly proportional the Open Circuit Voltage. Thus optimizing the estimation based on this state is expected to be efficient. C_s shall be used as the particles of

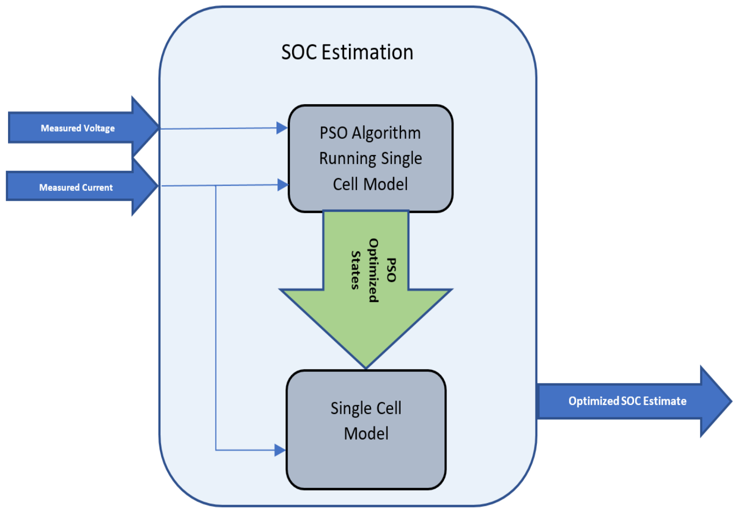

Fig. 4.1. SOC Estimation Overview.

the PSO algorithm, where they are initialized to uniformly random values around the previous C_s and the Voltage is estimated and optimized based on these states for the measured current value. A pictorial view of this process is shown in Figure 4.2.

One point to note is that at the initial value of C_s could be chosen at random. However, for faster convergence, best known value that would represent the current battery condition is used instead. To achieve this a linear interpolation function is employed, which takes the measured voltage and current as input and provides stoichiometric value as output, which is used in the calculation of c_s as shown in the equation 4.9.

$$C_s = stoichiometric\ value * C_s max, i \tag{4.9}$$

This is done only once at the beginning of the algorithm. For subsequent time steps the optimized states, including C_s, from the previous step are fed back.

4.4 PSO Algorithm Parameters

From the equation 4.7 it is clear that the PSO algorithm uses only three parameters, $c1$ & $c2$, which are learning coefficients of the stochastic acceleration terms and w

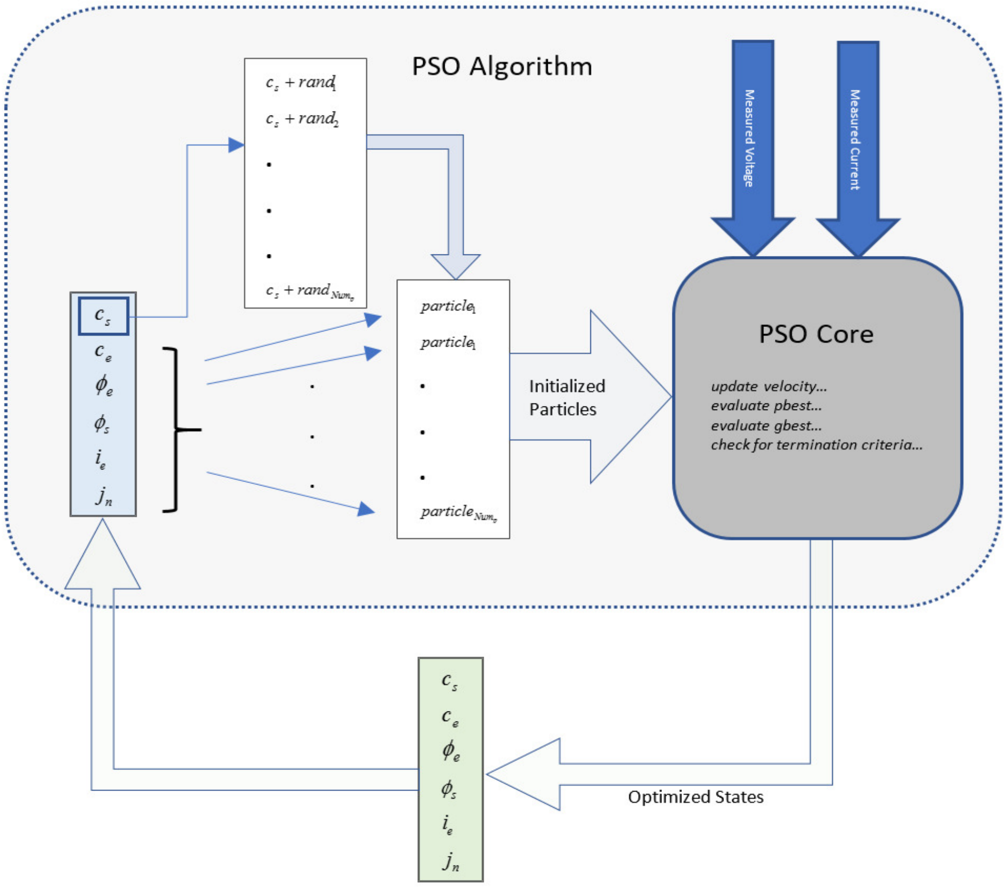

Fig. 4.2. Internal Working of PSO for SOC Estimation.

the inertial coefficient. Generally the values of $c1$ and $c2$ are chosen to be 2.0 [34] [35], which is mainly based on early experience and trail and error. However Clerc [36] introduced a constriction factor to ensure convergence. This constricting factor can be applied to equation 4.7 as shown in equation 4.10. Here K is represented as shown in equation 4.11 [35].

$$V(i) = K * [V(i) \; + \; c_1 u_{rand}(0,1)(P(i) - X(i)) \; + \; c_2 u_{rand}(0,1)(gbest - X(i))] \quad (4.10)$$

$$K \;=\; \frac{2}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|} \; where \; \phi \;=\; c1 + c2, \;\; \phi > 4 \qquad (4.11)$$

A typical number used for ϕ while using this constriction method is 4.1 [35], which gives us the value of K to be 0.729. Thus $c1$ and $c2$ is 1.494. A slight adjustment was

Table 4.1.
PSO Parameters.

Parameter	Value
w	$[0.5 + \frac{u_{rand}}{2}]$
c1	1.494
c2	1.494
Number of Particles	8
Max Number of Iteration	20

proposed by Eberhart and Shi [35] for application of PSO for dynamic systems. In this method the inertial weight w in set as $[0.5+\frac{u_{rand}}{2}]$, which gives a randomly varying number between 0.5 and 1.0 with a mean of 0.75, which is in line with the Clerc‘s constriction factor. This improved inertial method along with Clec‘s constriction coefficients is used in this work because the estimation of SOC is a dynamic process where PSO is applied at every time instance. The values of the PSO parameters w, $c1$, $c2$, Number of Particles and Maximum Iteration chosen for this work are shown in the table 4.1. Figure 4.3 shows the Simulink implementation of the PSO estimation.

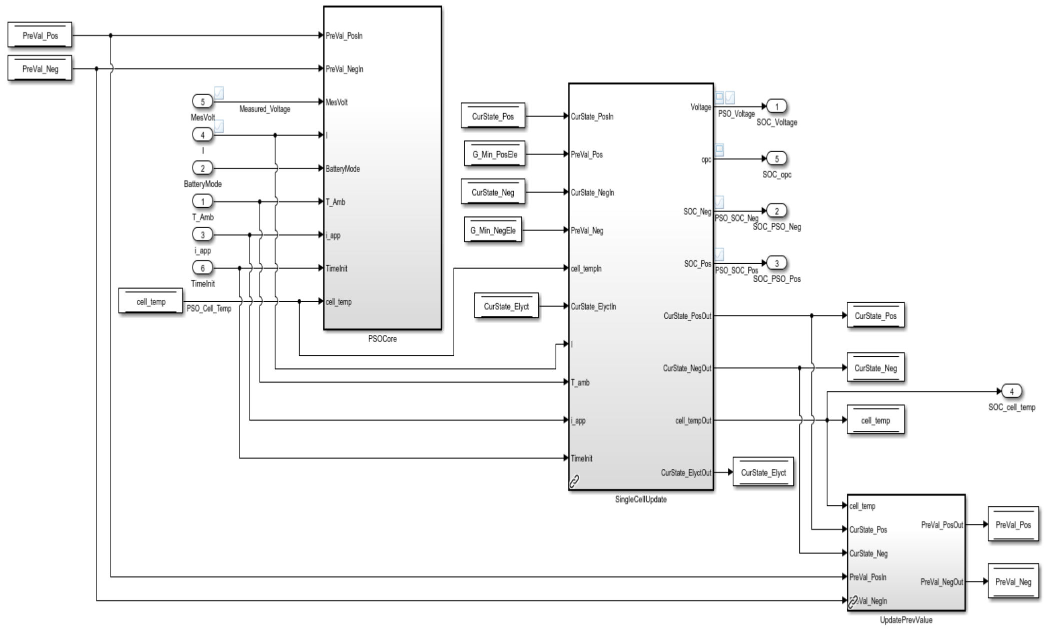

Fig. 4.3. Simulink implementation of SOC Estimation using PSO.

5. ALGORITHM VALIDATION AND RESULTS

5.1 Experimental Setup

In order to validate the PSO based estimation of SOC, battery testing was conducted to collect experimental data. Cadex C8000 was used to collect the battery data as shown in the Figure 5.1. The battery used was Panasonic NCR 18650B. It has a rated capacity of 3200 mAh at a nominal voltage of 3.6 V. Charging and discharging data, at 1C, of a new healthy battery was collected. Charging was performed using the built in NCR18650B charging profile of the CADEX software. Discharging was performed at constant current of 3.3 A.This battery was then subjected to a 100-cycle aging and the discharging and charging data, at 1C, was collected. The data was logged at 1 Hz.
Figure 5.2 and 5.3 show the charging voltage and current of both healthy and aged battery. Similarly, Figure 5.4 and 5.5 show the discharging voltage and current of both healthy and aged battery. These graphs clearly show the degradation of battery performance with aging.

5.2 Simulation Validation

The data collected from the experimental set up was run on the Single Cell model with PSO estimation for charging. The model was set to run with a discrete solver with a step time of 0.2s. The model was set to 'Normal Mode' in order to verify the algorithm in simulation. To evaluate the performance of the algorithm, Root Mean Square Error (RMS) as per equation 5.1 is employed on Voltage and SOC values. The

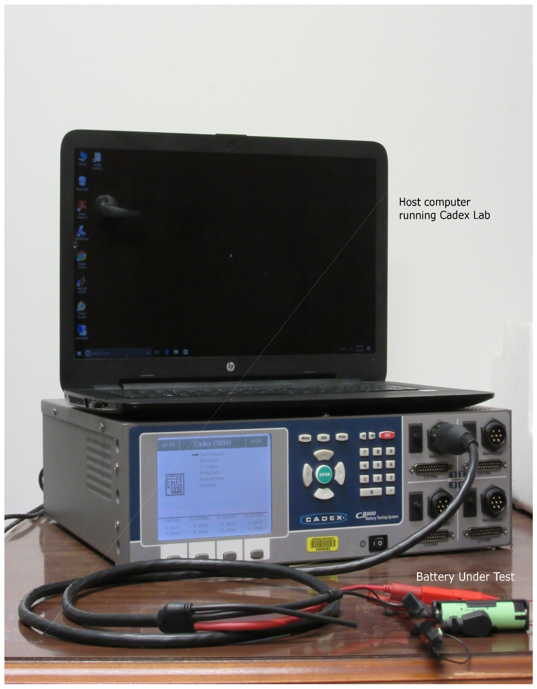

Fig. 5.1. CADEX C8000 experimental setup.

simulation data for 1C Charging and Discharging for both Healthy and Aged batter is presented in this section.

$$RMS = \sqrt{\frac{\sum_{i=1}^{M} |x_i|^2}{M}} \tag{5.1}$$

5.2.1 Healthy Charging

Figure 5.6 compares the Measured Voltage against the PSO optimized voltage. Although the measured and the estimated voltage start with different values the PSO algorithm optimizes the voltage in less than 45s. Figure 5.7 provides a zoomed in view of voltage convergence. The average RMS error between the estimated and measured is 0.0221. The graph of running RMS of PSO Voltage and measured voltage is shown in Figure 5.9.

Figure 5.8 compares the experimental SOC against the PSO optimized SOC. The curve for SOC estimation from the single cell battery without optimization is also

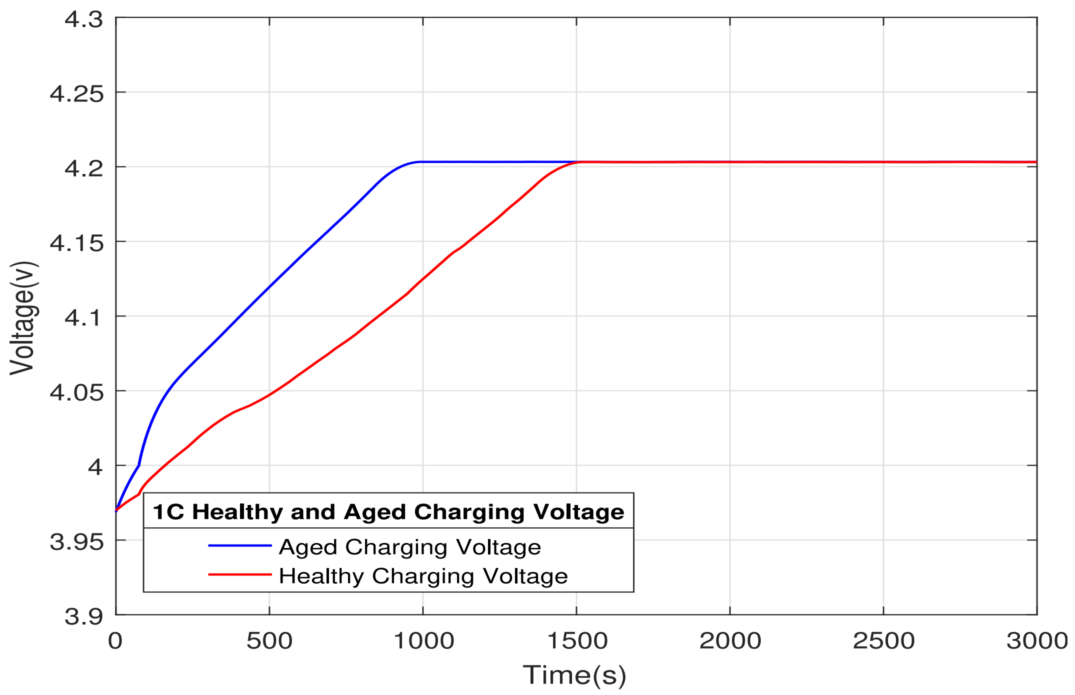

Fig. 5.2. Experimental Aged and Healthy Charging Voltage.

plotted to provide a reference for analysis purposes.The estimated SOC converges within a absolute difference of less than 2% within the first 800s and then continues to follow the experimental SOC with a running RMS Of 0.0355 as shown in Figure 5.10. It is clear from the graph that the SOC estimated without PSO diverges from the experimental SOC.

5.2.2 Healthy Discharging

Figure 5.11 compares the Measured Voltage against the PSO optimized voltage for discharging. The measured and the estimated voltage start with different values but the PSO algorithm optimizes the voltage in less than 15s. The short convergence time is because the initial value of the PSO Voltage is closer to measured voltage, with an absolute difference of 0.025 V for the given conditions. Figure 5.12 provides a zoomed in view of voltage convergence. The average RMS error between the estimated and measured is 0.0200.

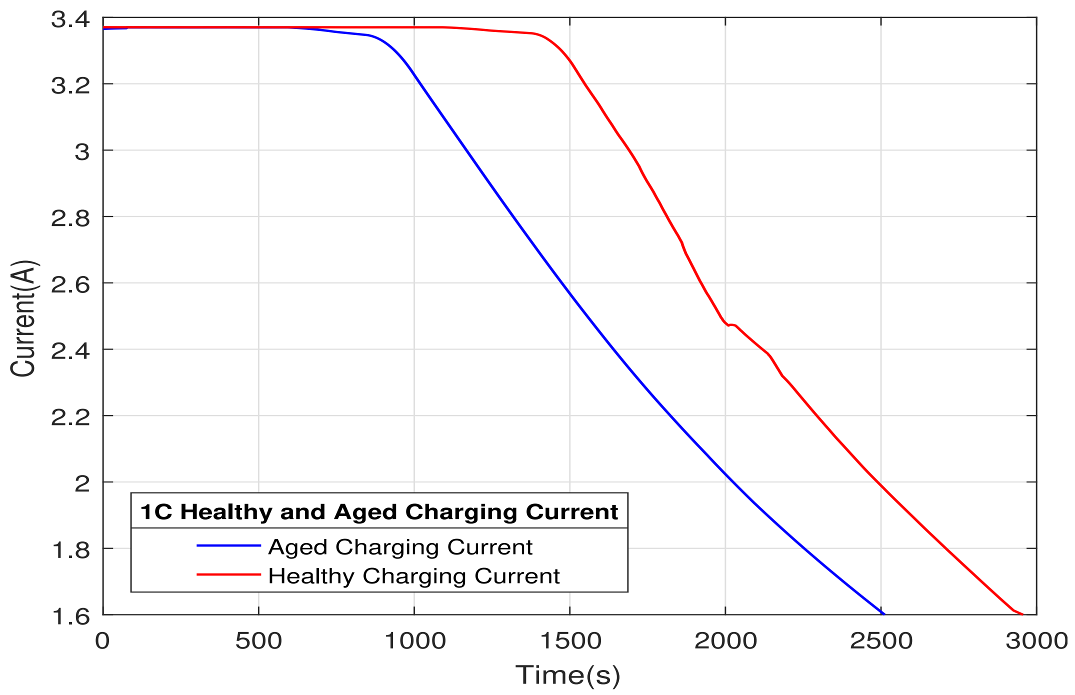

Fig. 5.3. Experimental Aged and Healthy Charging Current.

The graph of running RMS of PSO Voltage and measured voltage is shown in Figure 5.14.

Figure 5.13 compares the experimental SOC against the PSO optimized SOC. The curve for SOC estimation from the single cell battery without optimization is also plotted to provide a reference for analysis purposes. The estimated SOC converges within a absolute difference of less than 2% within the first 1160s and then continues to follow the experimental SOC with a running RMS Of 0.03186 as shown in Figure 5.15. The SOC of Single cell without optimization, SOC with PSO and Experimental SOC converge to within 3% absolute difference withing 1500s.

5.2.3 Aged Charging

Figure 5.16 compares the Measured Voltage against the PSO optimized voltage for charging. Although the measured and the estimated voltage start with different values the PSO algorithm optimizes the voltage in less than 30s. The average RMS

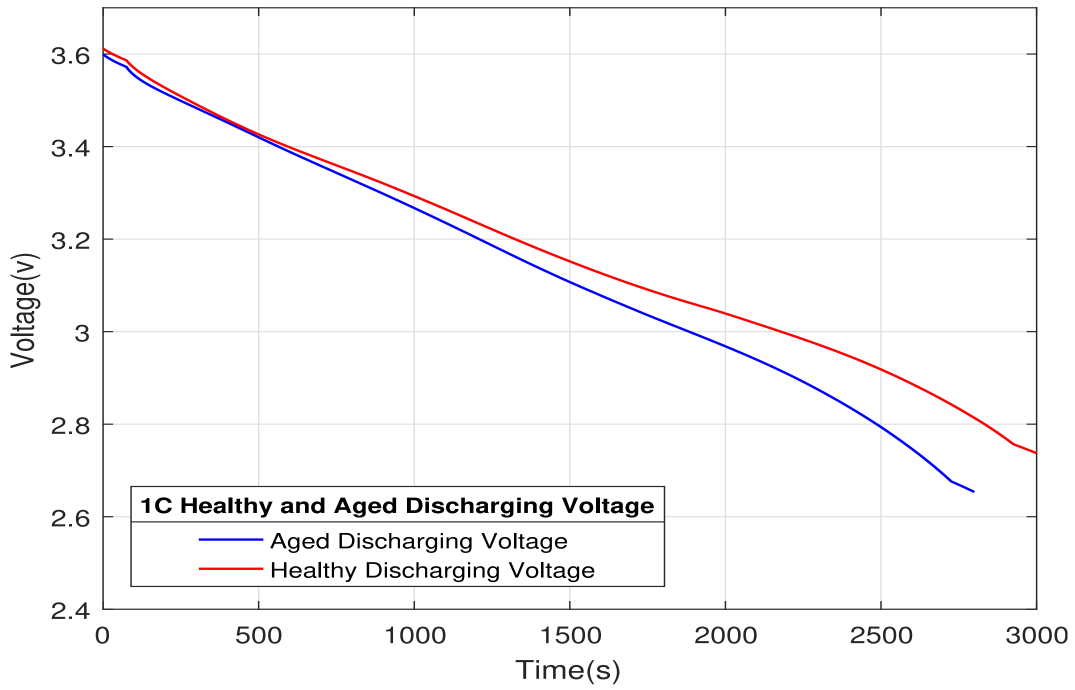

Fig. 5.4. Experimental Aged and Healthy Discharging Voltage.

error between the estimated and measured is 0.01729. Figure 5.17 provides a zoomed in view of voltage convergence. The graph of running RMS of PSO Voltage and measured voltage is shown in Figure 5.19.

Figure 5.18 compares the experimental SOC against the PSO optimized SOC. The curve for SOC estimation from the single cell battery without optimization is also plotted to provide a reference for analysis purposes.Similar to the healthy charging, the estimated SOC converges within a absolute difference of less than 2% within the first 800s and then continues to follow the experimental SOC with a running RMS Of 0.03016 as shown in Figure 5.20.Just like in the case of Healthy Battery, it is clear from the graph that the SOC estimated without PSO diverges from the experimental SOC.

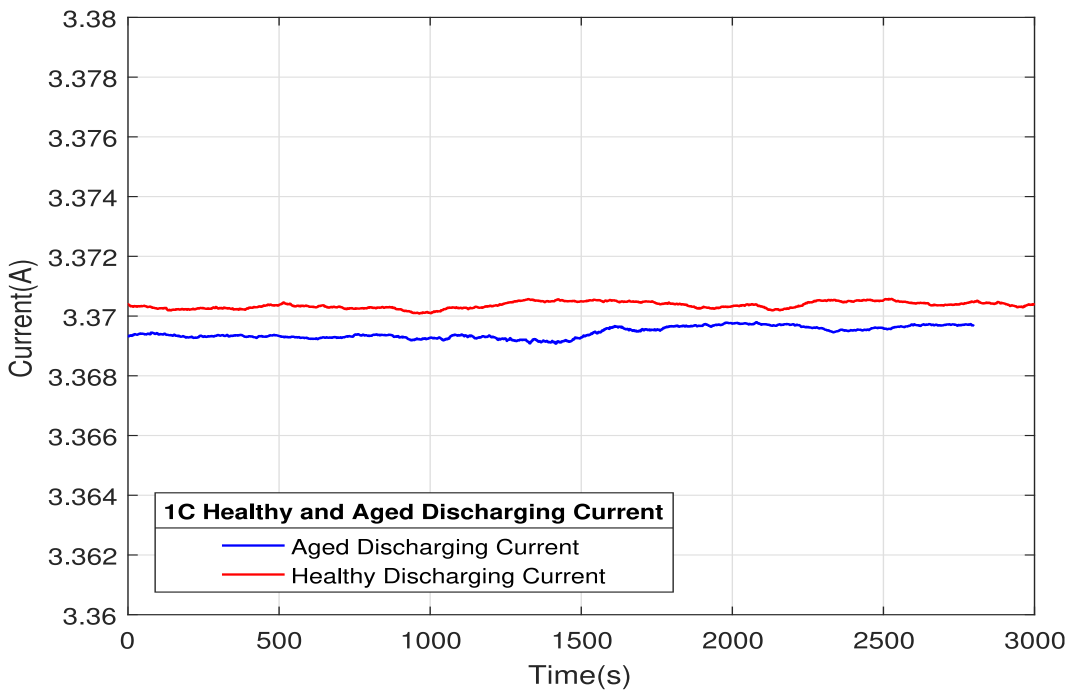

Fig. 5.5. Experimental Aged and Healthy Discharging Current.

5.2.4 Aged Discharging

Figure 5.21 compares the Measured Voltage against the PSO optimized voltage for discharging. The measured and the estimated voltage start with different values but the PSO algorithm optimizes the voltage in less than 12s. The short convergence time is because the initial value of the PSO Voltage is closer to measured voltage, with an absolute difference of 0.0212 V for the given conditions. The average RMS error between the estimated and measured is 0.0196. Figure 5.22 provides a zoomed in view of voltage convergence. The graph of running RMS of PSO Voltage and measured voltage is shown in Figure 5.24.

Figure 5.23 compares the experimental SOC against the PSO optimized SOC. The curve for SOC estimation from the single cell battery without optimization is also plotted to provide a reference for analysis purposes. The estimated SOC converges within a absolute difference of less than 1% within the first 830s and then continues to follow the experimental SOC with a running RMS Of 0.02515 as shown in Figure

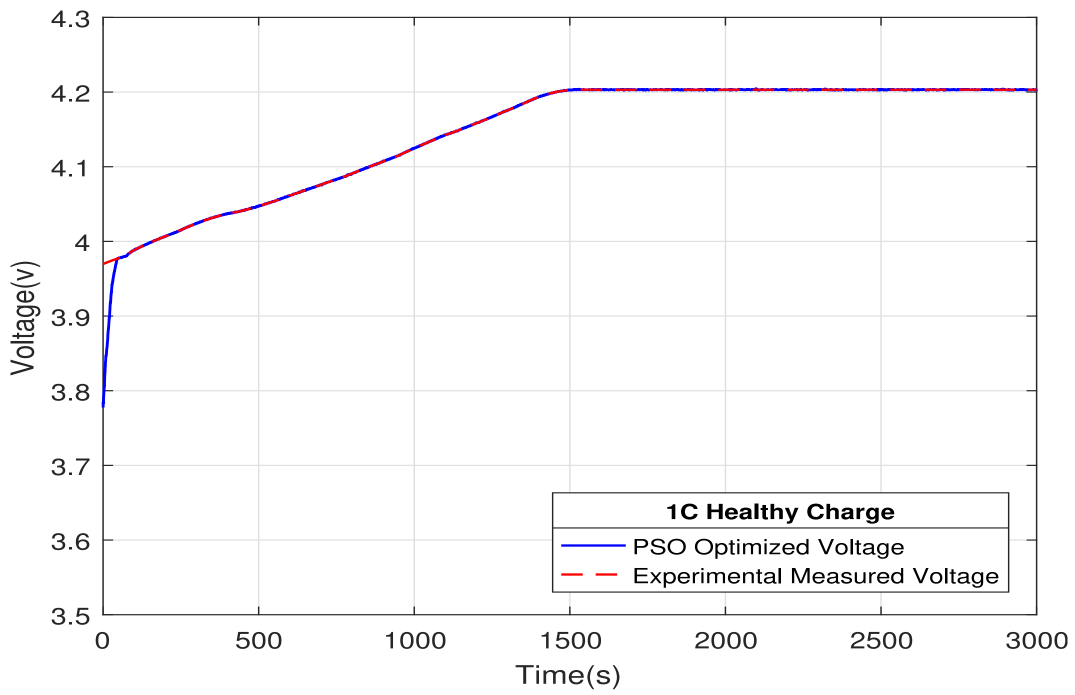

Fig. 5.6. Healthy Battery: Experimental Vs PSO Voltage.

5.25. The SOC of Single cell without optimization, SOC with PSO and Experimental SOC converge to within 3% absolute difference withing 1500s. However it is clear that the PSO optimized SOC closely follows the experimentally calculated SOC.

5.2.5 Charge Discharge Cycle

Figure 5.26 compares the Measured Voltage against the PSO optimized voltage for a single Charge Discharge cycle. The measured and the estimated voltage start with different values but the PSO algorithm optimizes the voltage in less than 45s. The battery is charged to 40% and then a discharging current of 3.37 V is applied at 1300s. Figure 5.27 shows the applied Charging/Discharging Current. Figure 5.28 compares the Experimental SOC and PSO optimized SOC for the Charge/Discharge cycle. The maximum error is 8.9% at the beginning of the estimation where the PSO algorithm is trying to converge to the experimental values. However, the estimated SOC converges within 2% of Experimental SOC in the first 800s and continues to

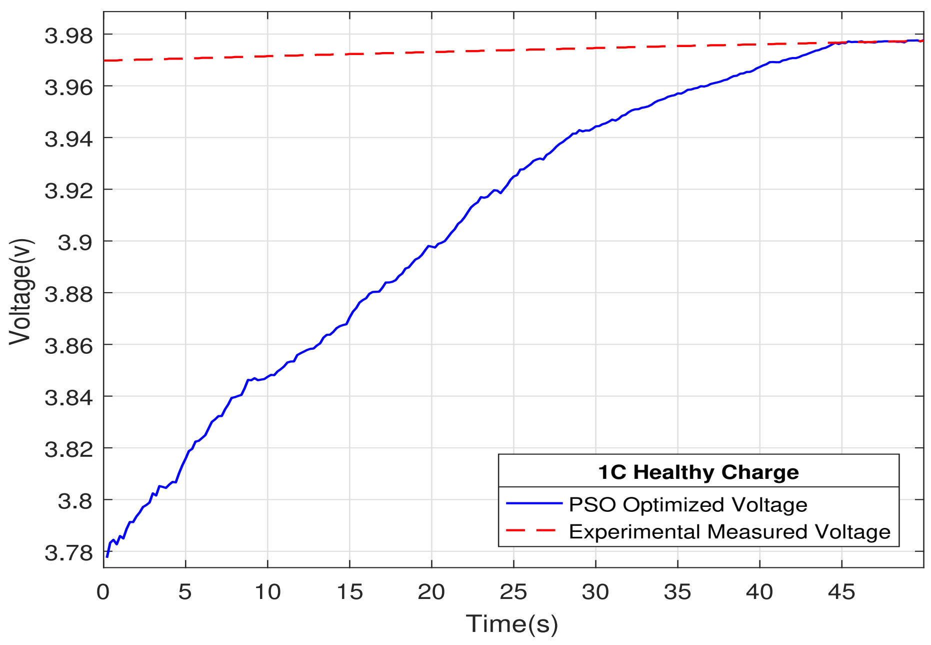

Fig. 5.7. Healthy Battery: Experimental Vs PSO Voltage - Zoomed In View.

follow the experimental SOC within 1% error even when the battery switches to discharging. The average RMS between the estimated and experimental SOC is 0.0366.

5.3 Real Time Validation

The PSO estimator is run on Raspberry Pi 3 by using the external mode in Simulink. Raspberry Pi 3 is a single board computer with a 64-bit quad core processer and 1GB internal RAM.Being a low-cost device, it is ideal for prototyping computationally intense algorithms like the PSO Estimator presented in this work. Simulink provides a seamless interface to Raspberry Pi. Once the models are validated in pure simulation the same can be run on Raspberry Pi by changing the simulation mode to 'External Mode' in the model. If the Pi has been been set up correctly (as per instruction in Matlab Help) then by clicking the simulate button,

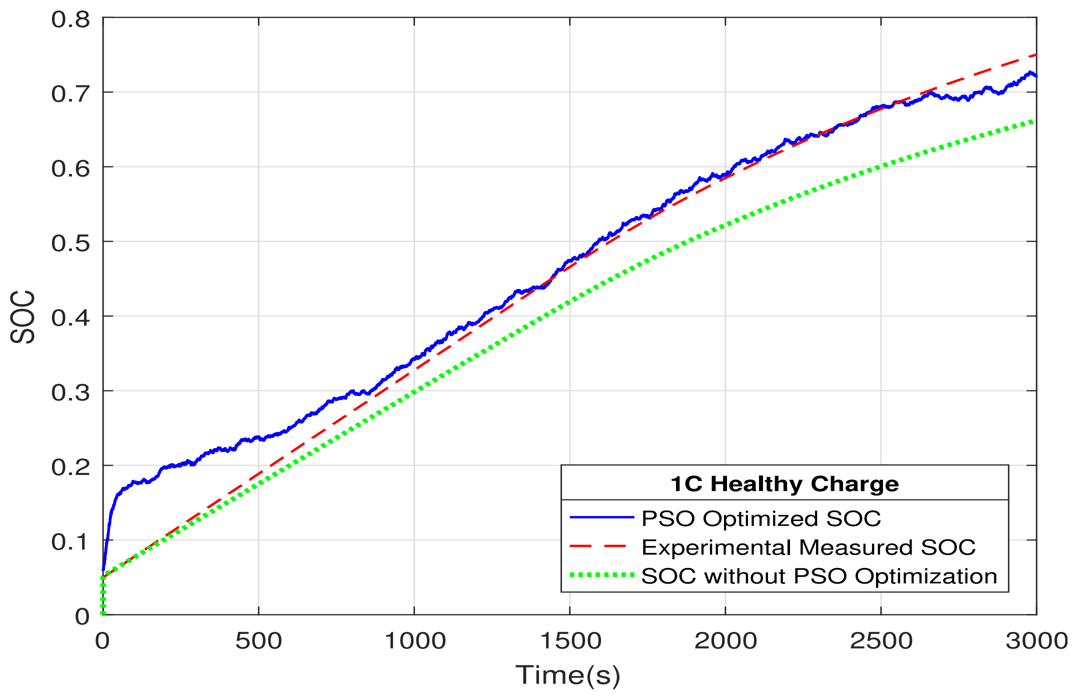

Fig. 5.8. Healthy Battery: Experimental Vs PSO SOC.

Simulink generates code, compiles it and starts running it in real time on Raspberry pi. The real time data is displayed on the same scopes that were set up for pure simulation. Figure 5.29 shows the block diagram of the raspberry pi setup and Figure 5.30 shows the actual picture of the setup.

Figures 5.31 to 5.38, show the comparison of Voltage and SOC obtained from Simulation and from Real-Time processor (Raspberry Pi). The results match 100%, which shows that the PSO estimator is efficient on the real-time processor, thereby giving the exact same results. The CPU utilization time of Raspberry Pi was collected to evaluate the performance. A maximum of 26% CPU utilization was observed at PSO Estimator initialization and during convergence, while it ran at 3% average utilization for the rest of the execution.

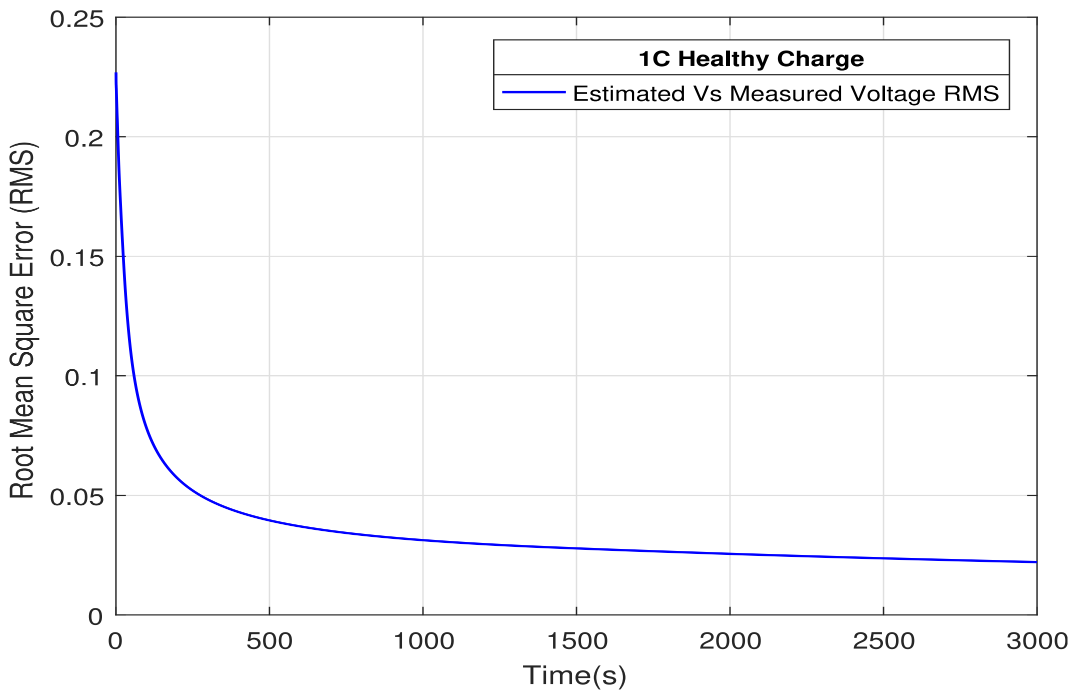

Fig. 5.9. Healthy Battery: Experimental Vs PSO Voltage RMS.

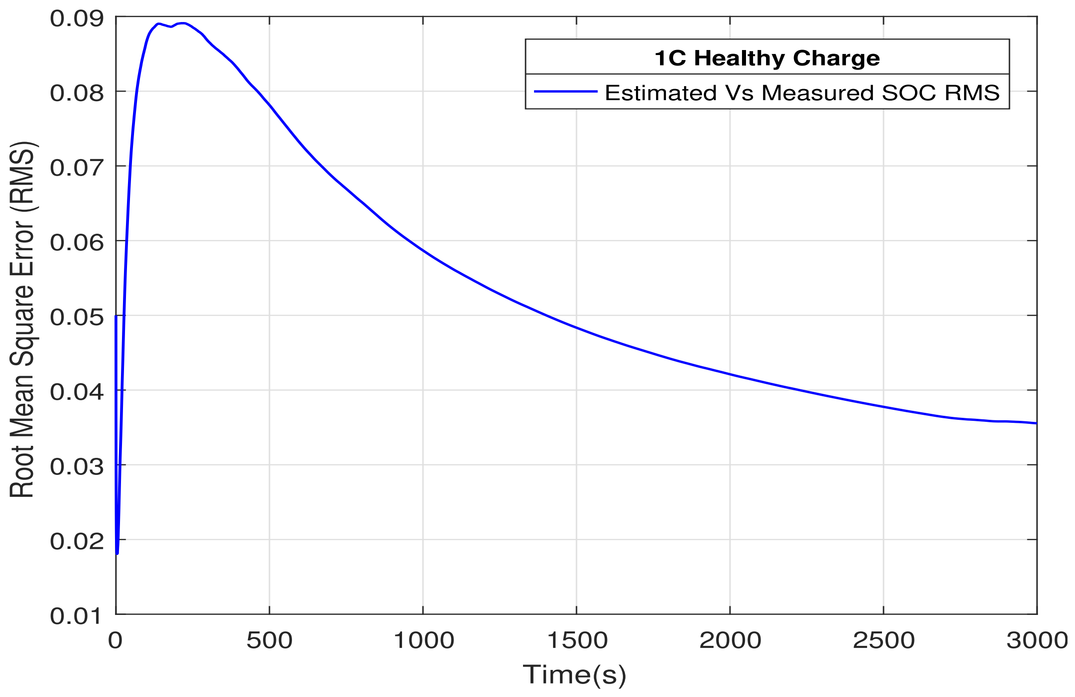

Fig. 5.10. Healthy Battery: Experimental Vs PSO SOC RMS.

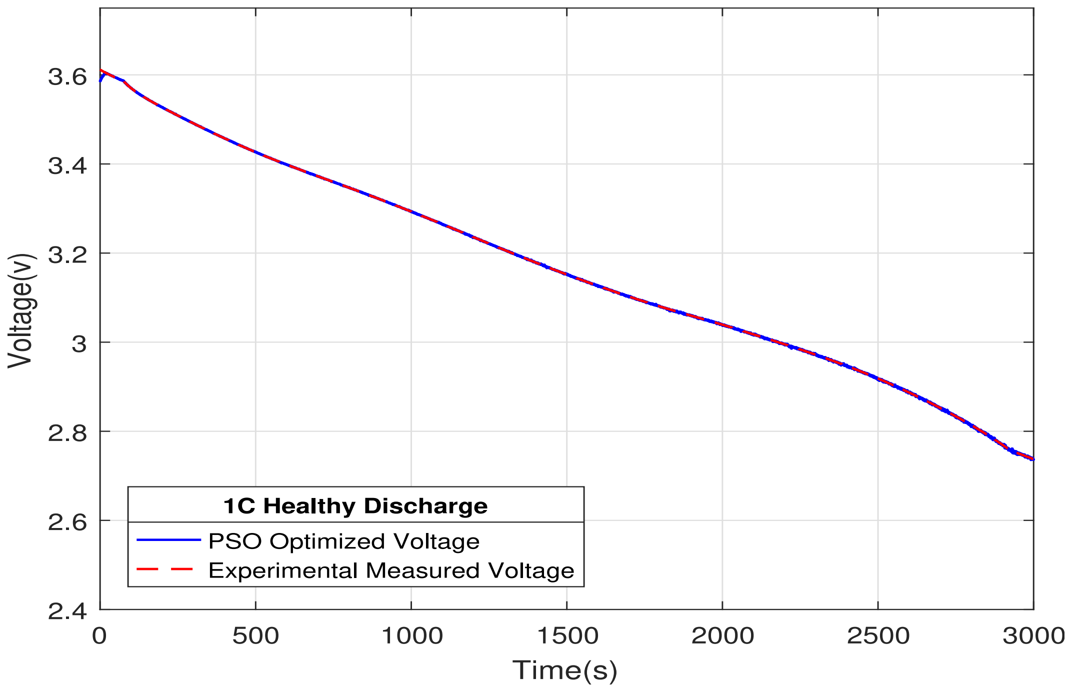

Fig. 5.11. Healthy Battery: Experimental Vs PSO Voltage.

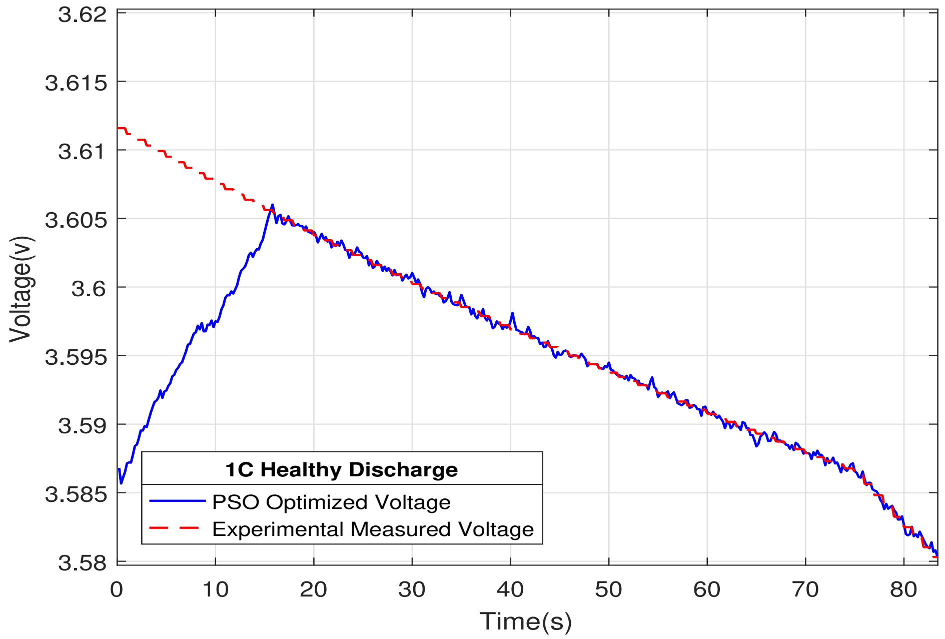

Fig. 5.12. Healthy Battery: Experimental Vs PSO Voltage - Zoomed In View.

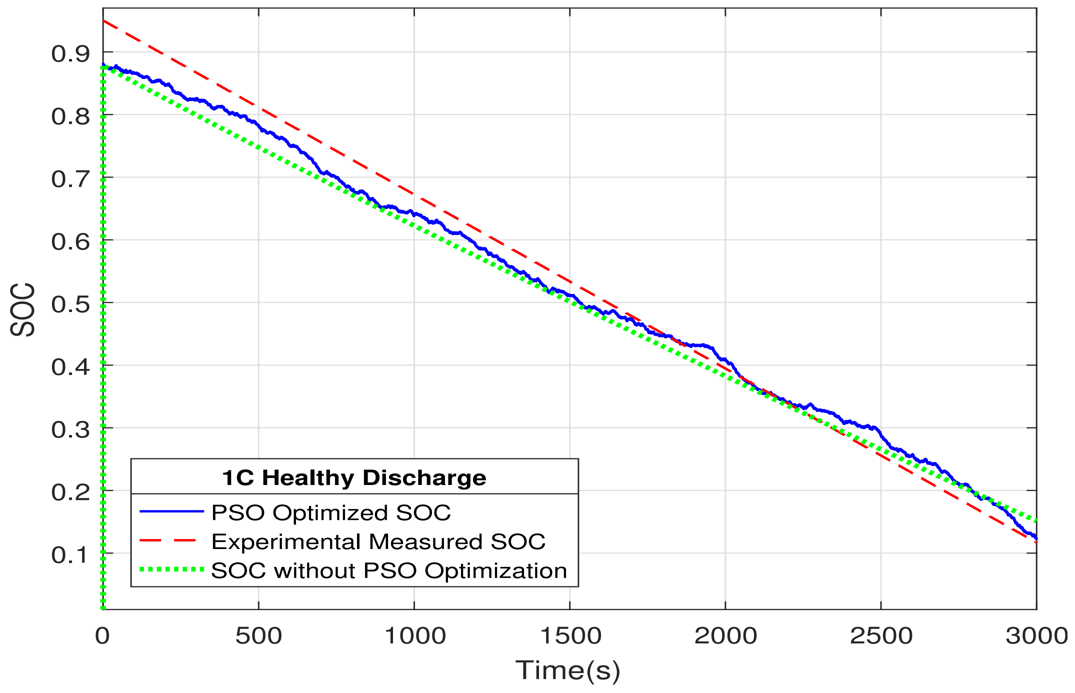

Fig. 5.13. Healthy Battery: Experimental Vs PSO SOC.

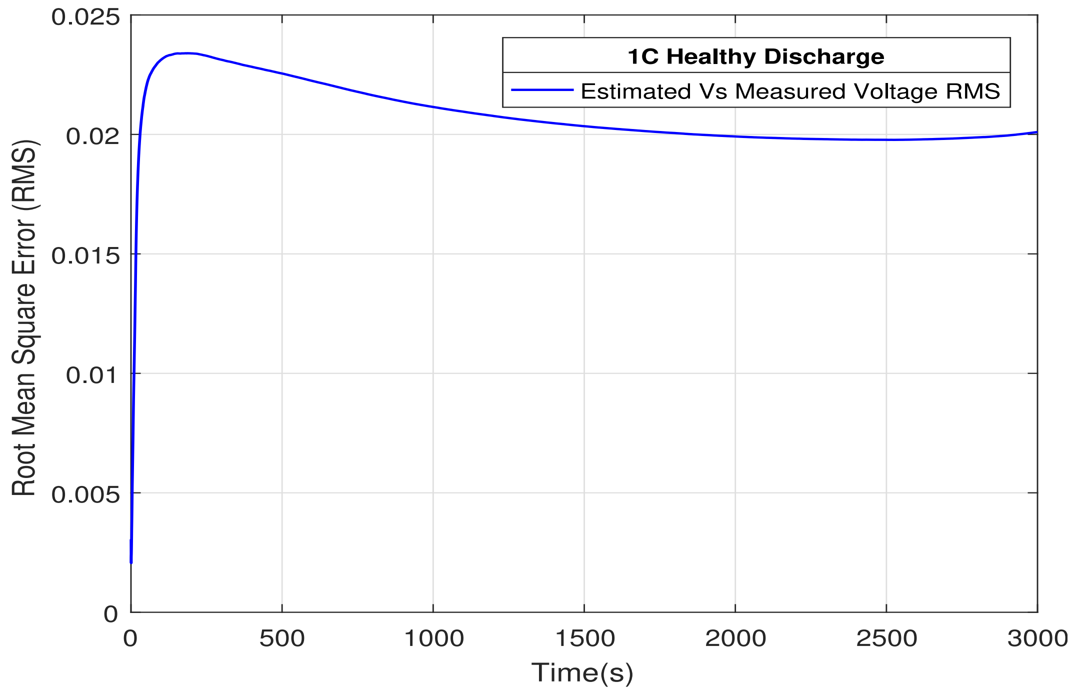

Fig. 5.14. Healthy Battery: Experimental Vs PSO Voltage RMS.

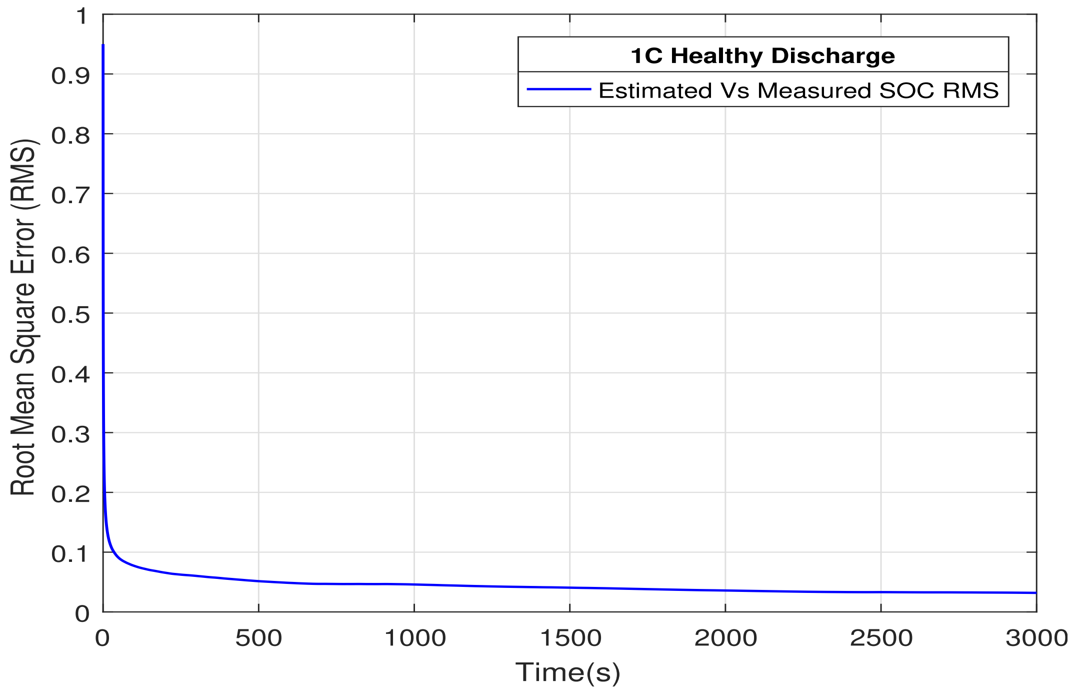

Fig. 5.15. Healthy Battery: Experimental Vs PSO SOC RMS.

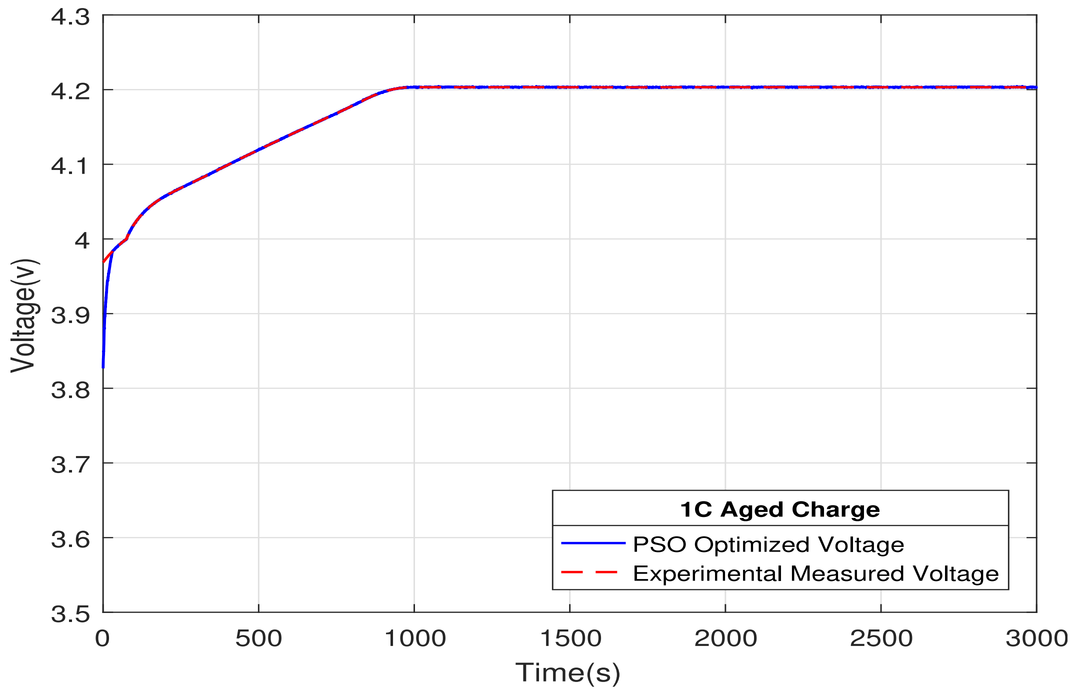

Fig. 5.16. Aged Battery: Experimental Vs PSO Voltage.

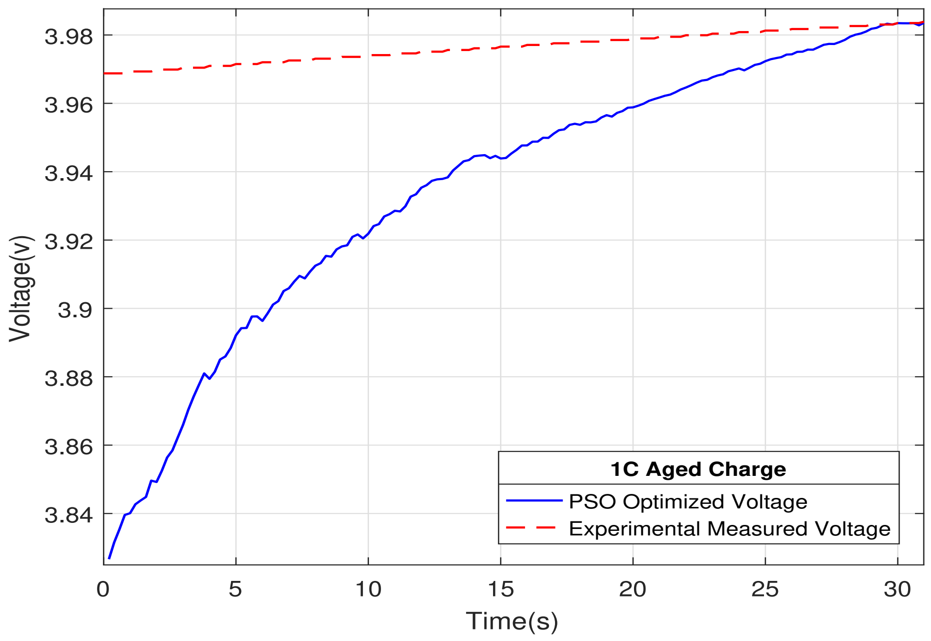

Fig. 5.17. Aged Battery: Experimental Vs PSO Voltage - Zoomed In View.

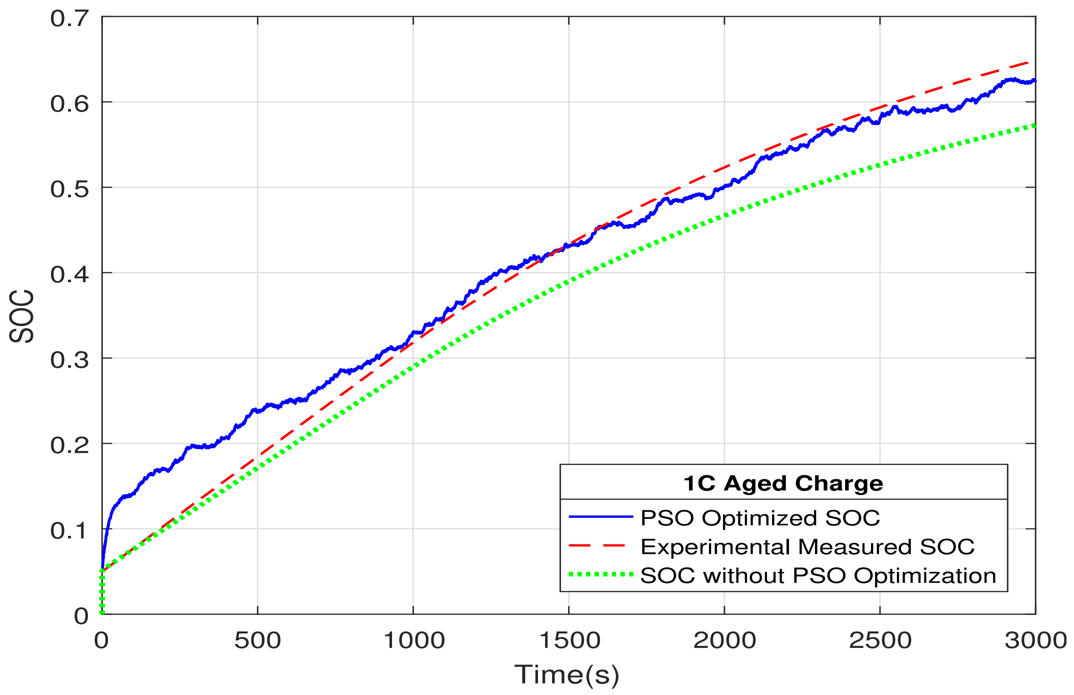

Fig. 5.18. Aged Battery: Experimental Vs PSO SOC.

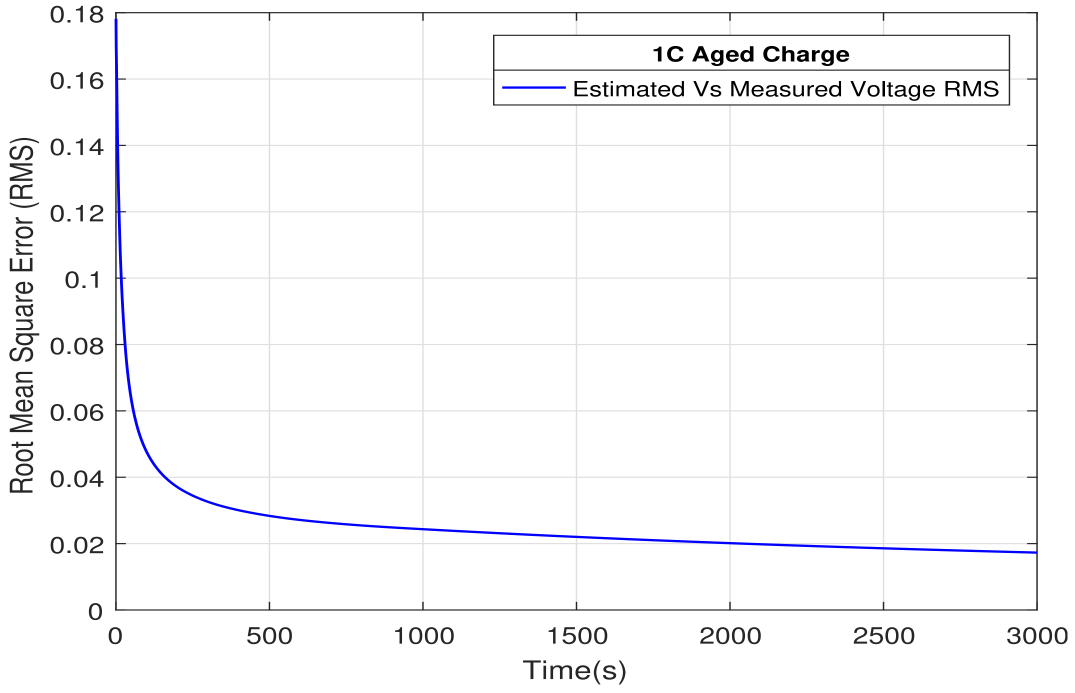

Fig. 5.19. Aged Battery: Experimental Vs PSO Voltage RMS.

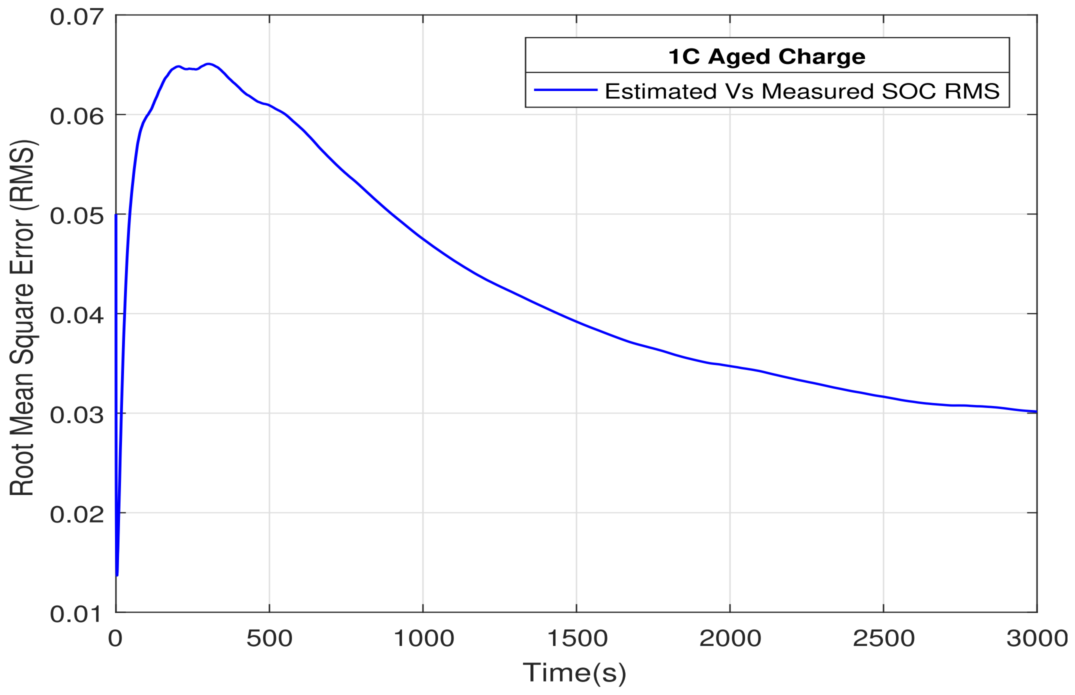

Fig. 5.20. Aged Battery: Experimental Vs PSO SOC RMS.

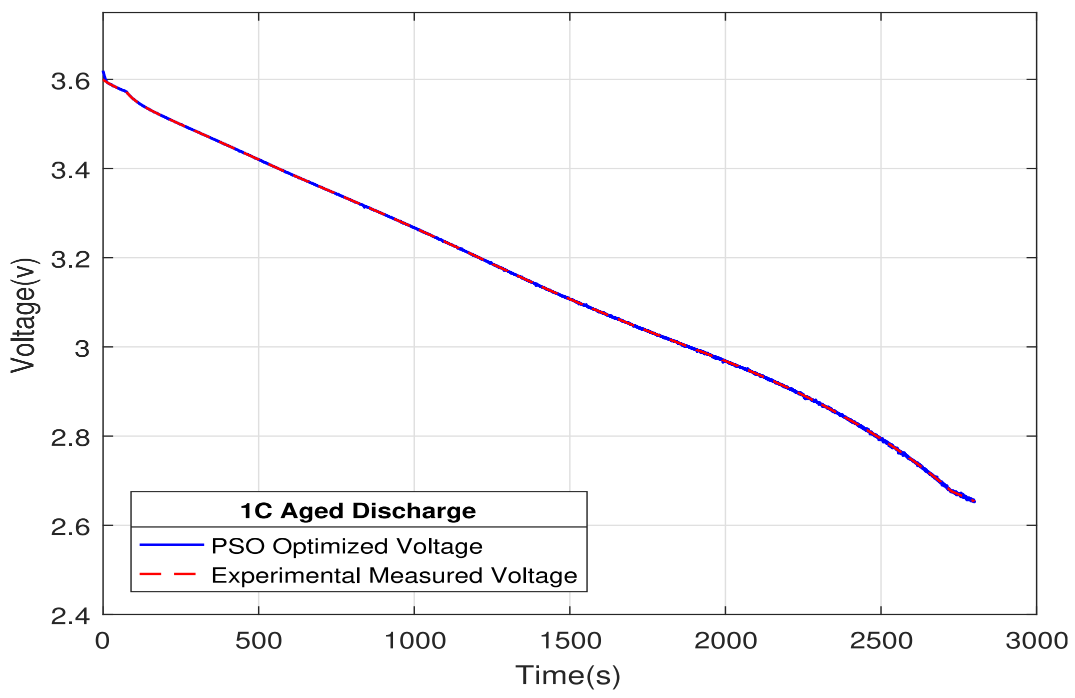

Fig. 5.21. Aged Battery: Experimental Vs PSO Voltage.

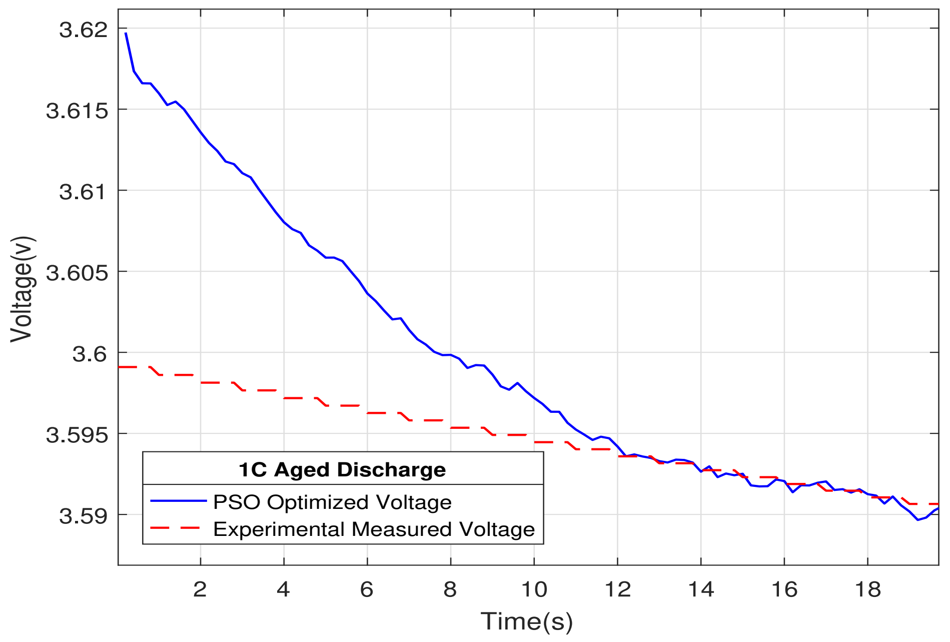

Fig. 5.22. Aged Battery: Experimental Vs PSO Voltage - Zoomed In View.

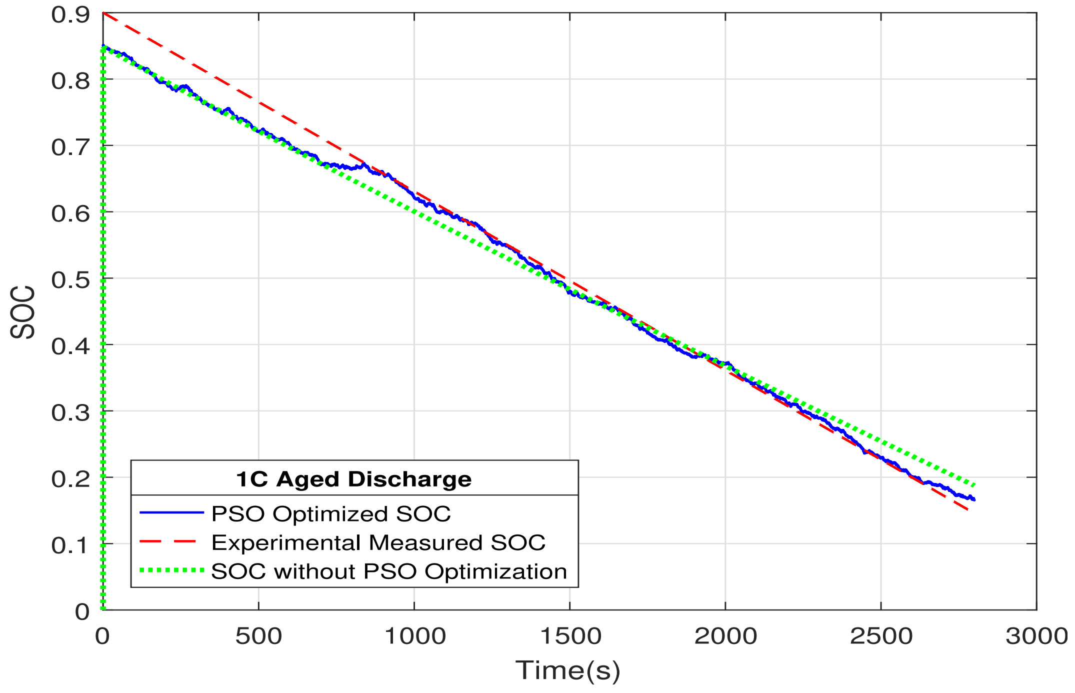

Fig. 5.23. Aged Battery: Experimental Vs PSO SOC.

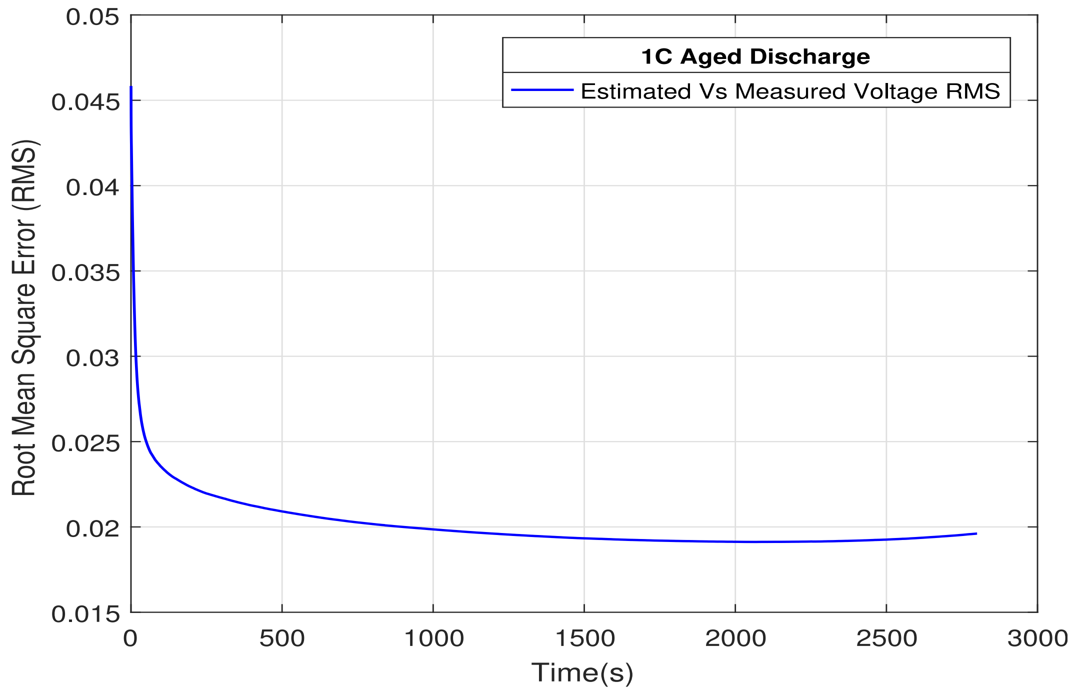

Fig. 5.24. Aged Battery: Experimental Vs PSO Voltage RMS.

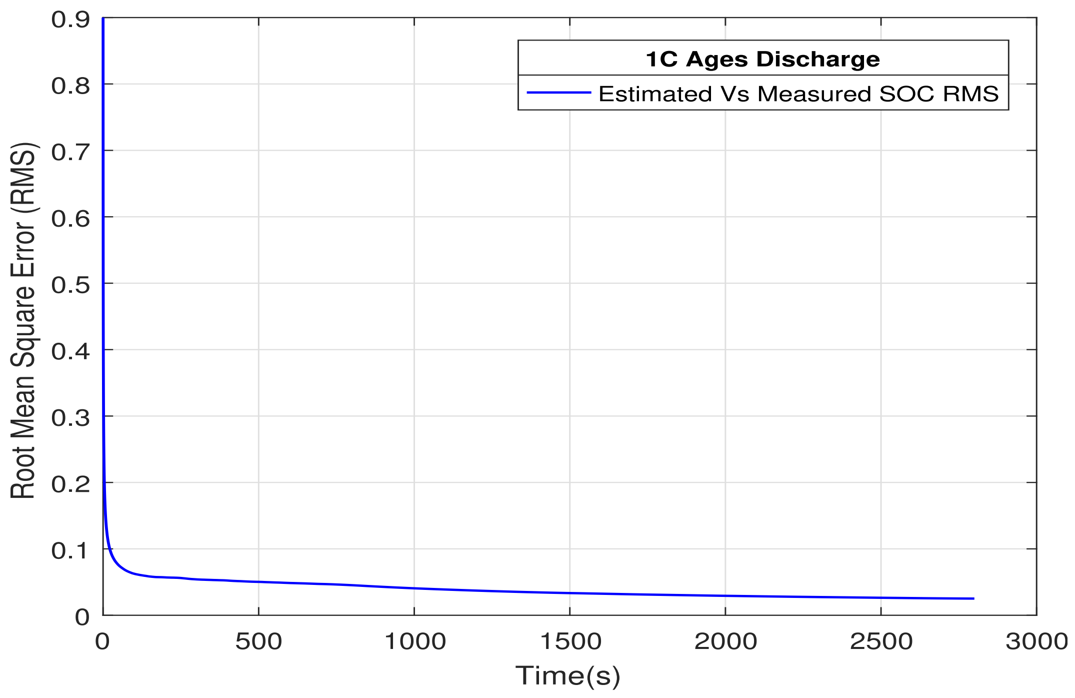

Fig. 5.25. Aged Battery: Experimental Vs PSO SOC RMS.

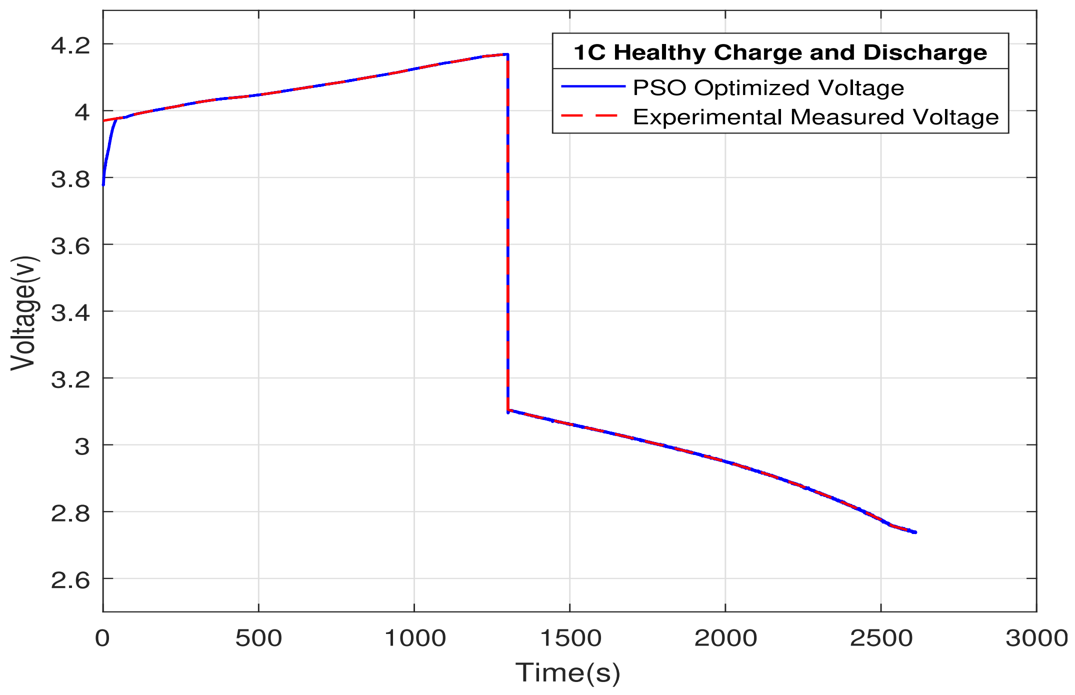

Fig. 5.26. Healthy Battery: Experimental Vs PSO Voltage - Charge-Discharge Cycle.

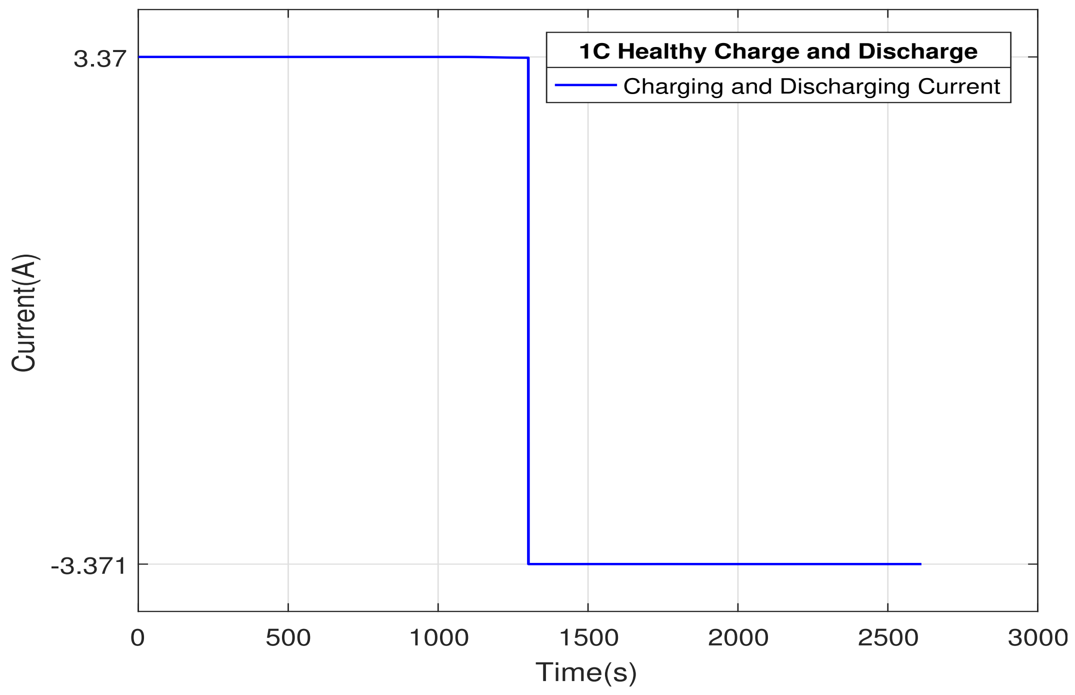

Fig. 5.27. Healthy Battery: Current - Charge-Discharge Cycle.

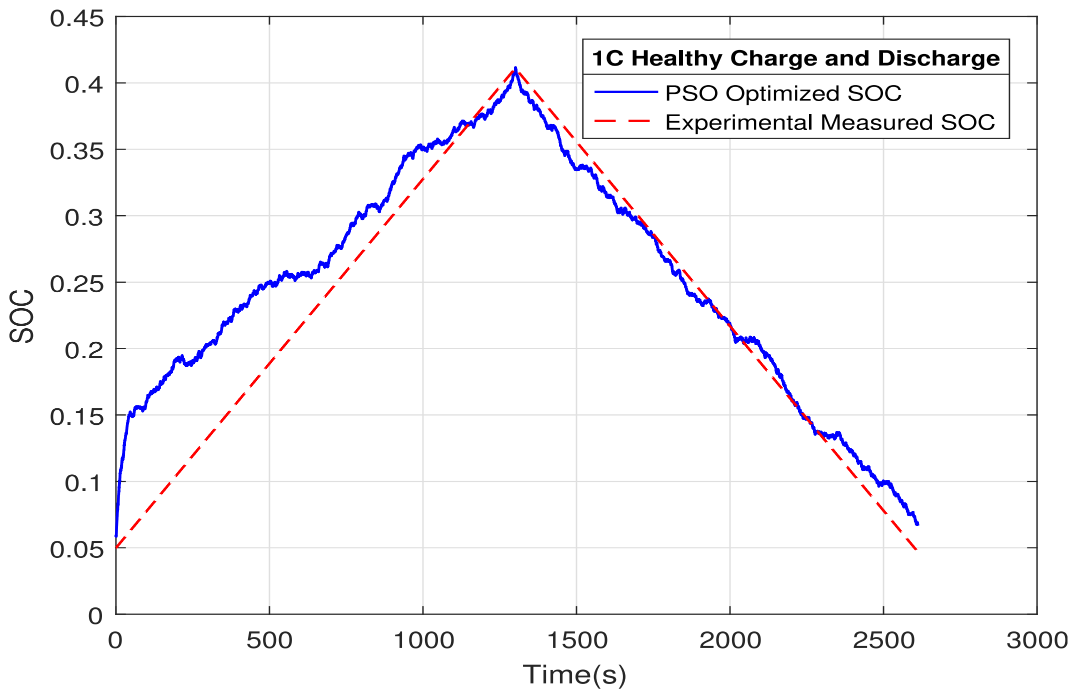

Fig. 5.28. Healthy Battery: Experimental Vs PSO SOC - Charge-Discharge Cycle.

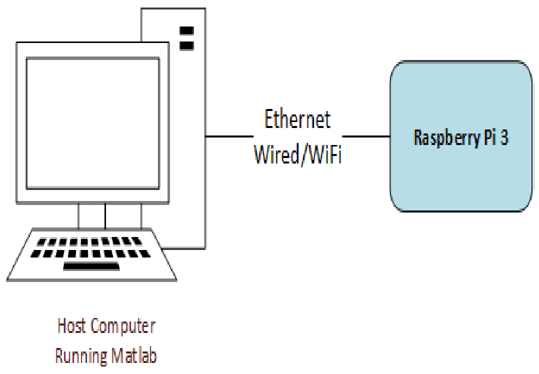

Fig. 5.29. Raspberry Pi3 Setup Overview.

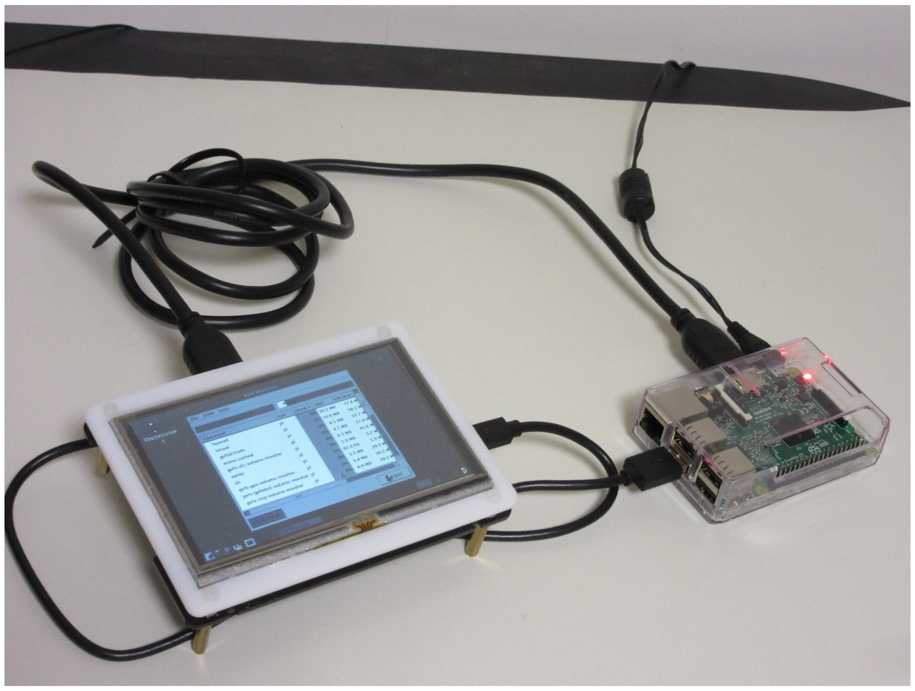

Fig. 5.30. Raspberry Pi3 Experimental Setup.

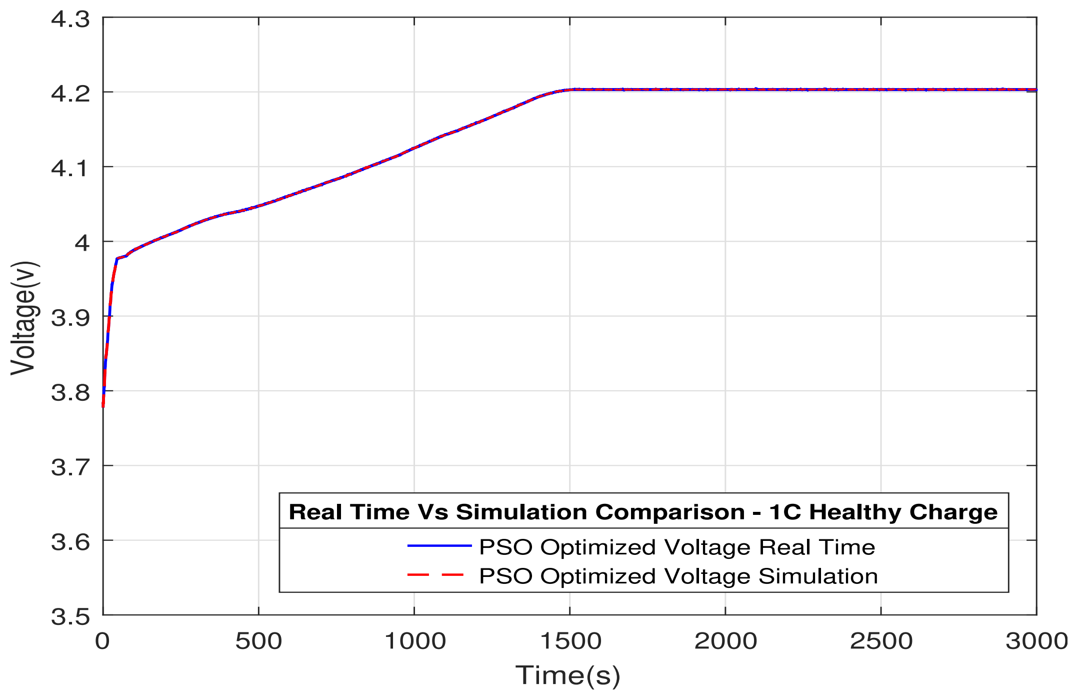

Fig. 5.31. Simulation Vs Real Time Voltage.

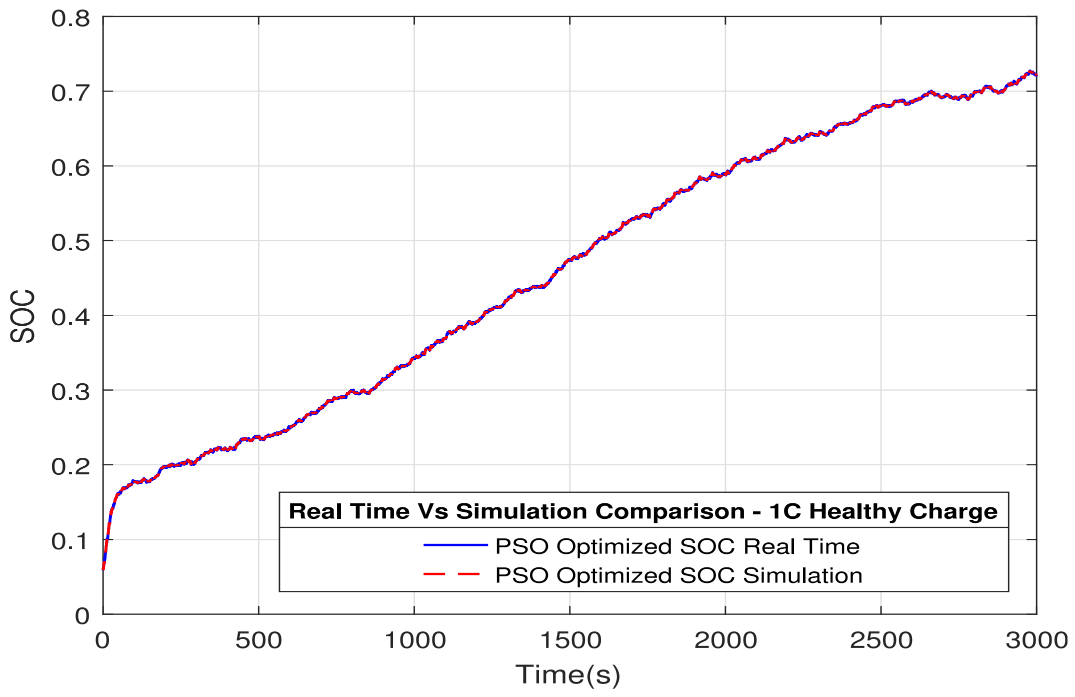

Fig. 5.32. Simulation Vs Real Time SOC.

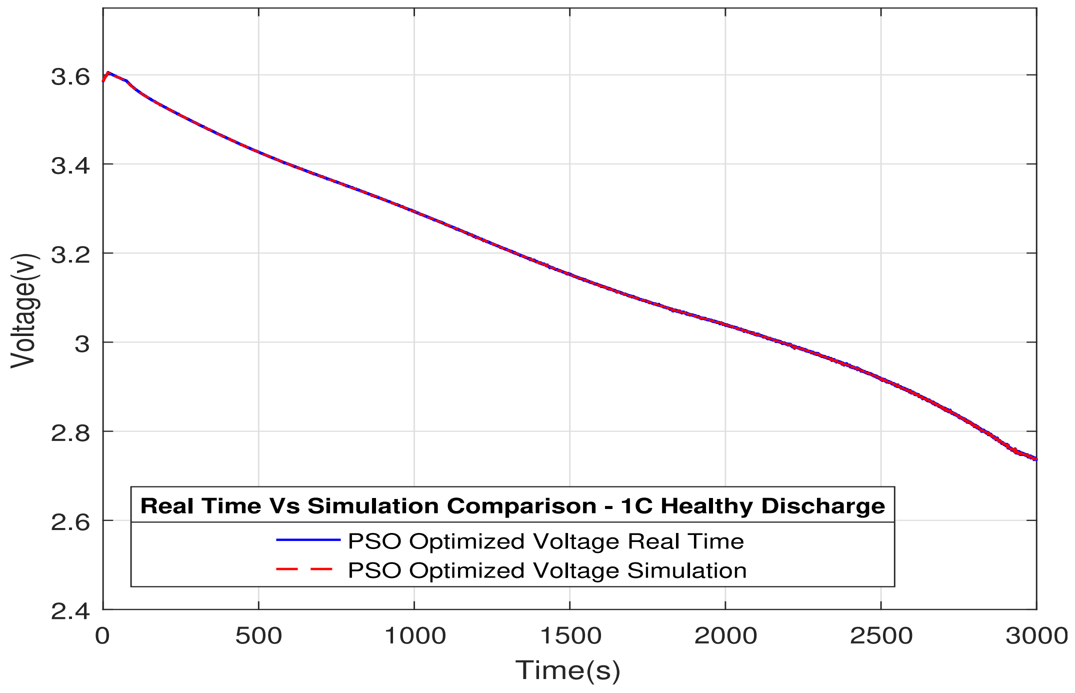

Fig. 5.33. Simulation Vs Real Time Voltage.

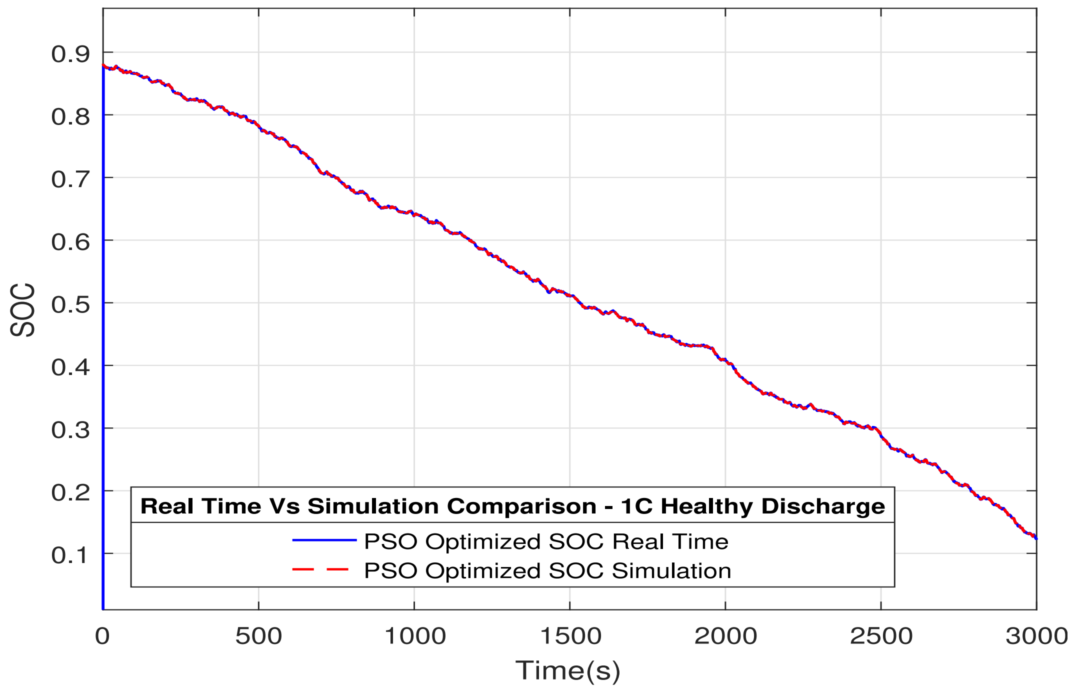

Fig. 5.34. Simulation Vs Real Time SOC.

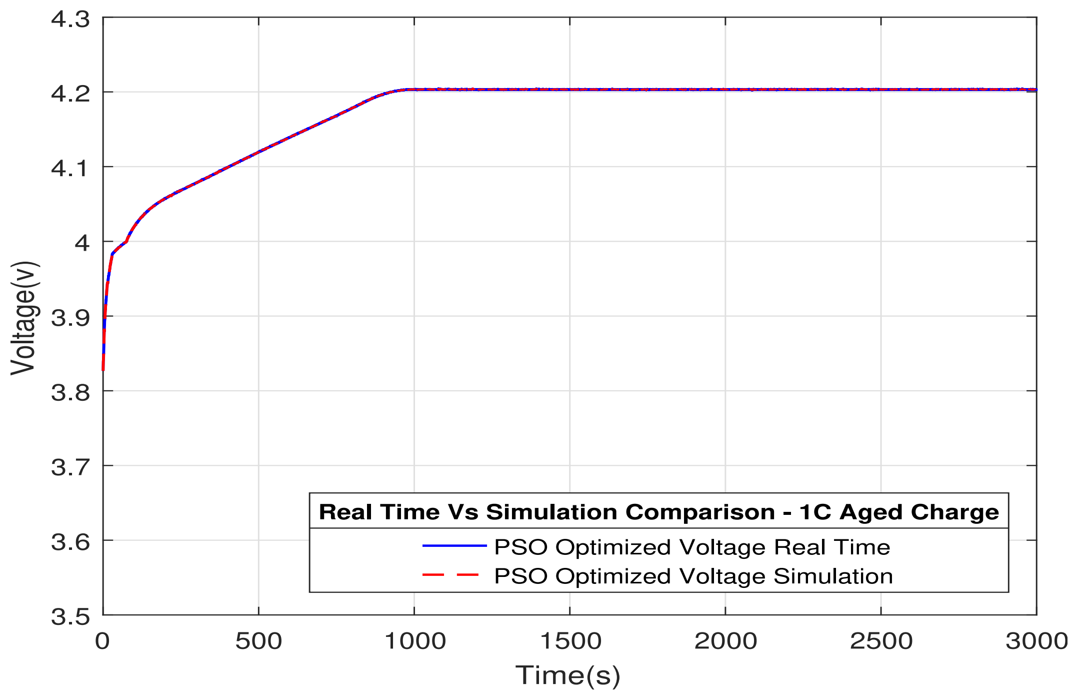

Fig. 5.35. Simulation Vs Real Time Voltage.

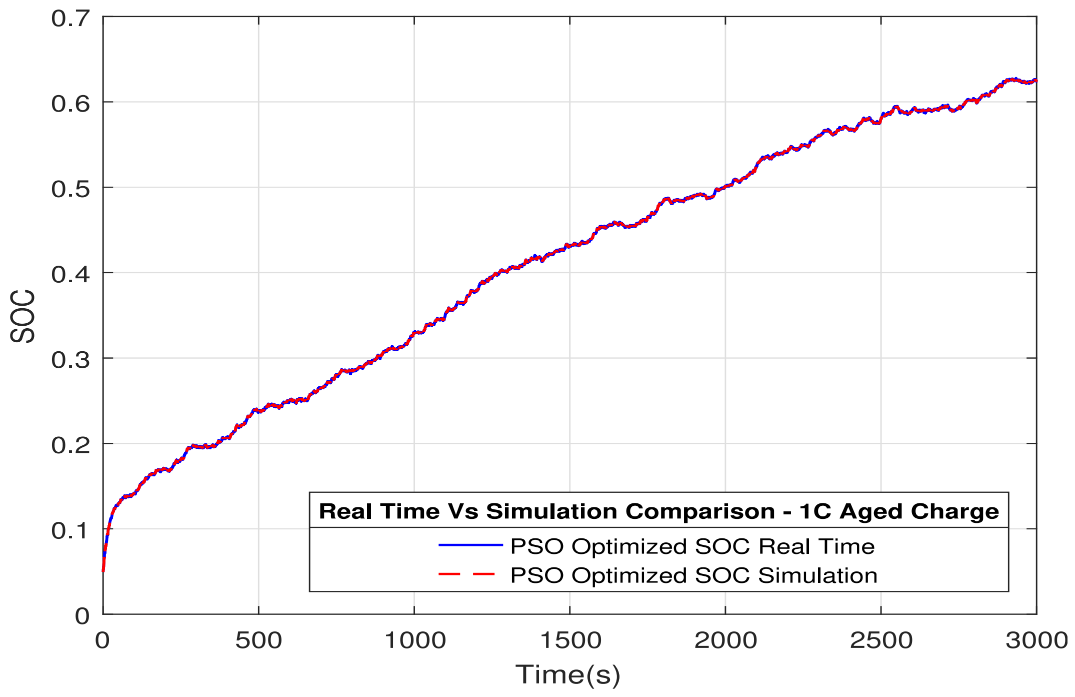

Fig. 5.36. Simulation Vs Real Time SOC.

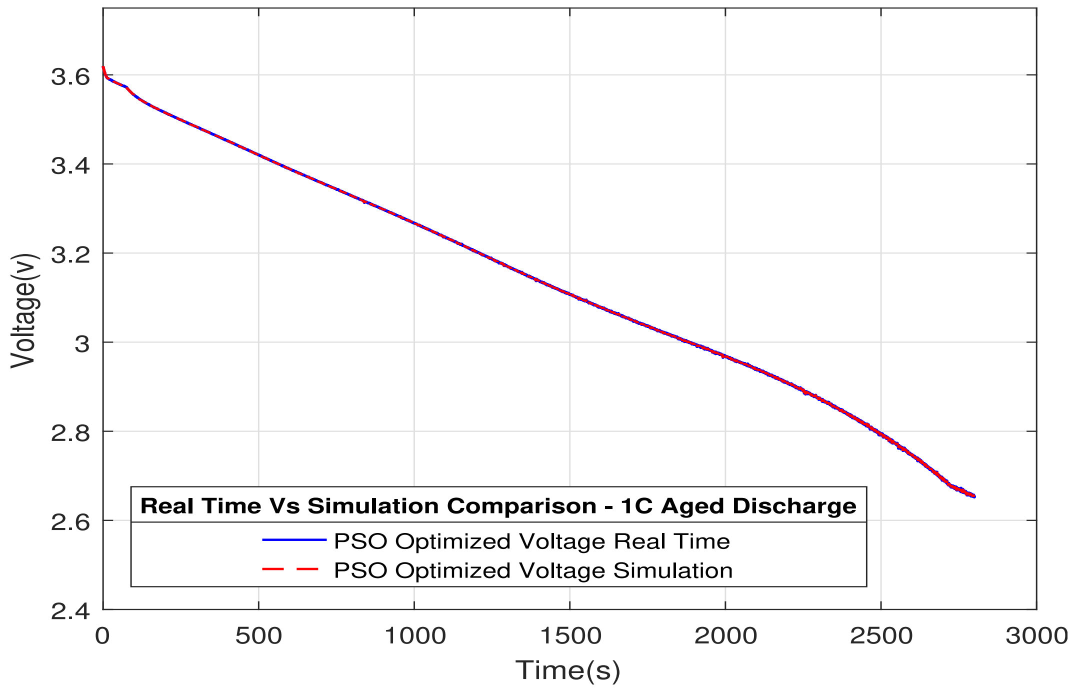

Fig. 5.37. Simulation Vs Real Time Voltage.

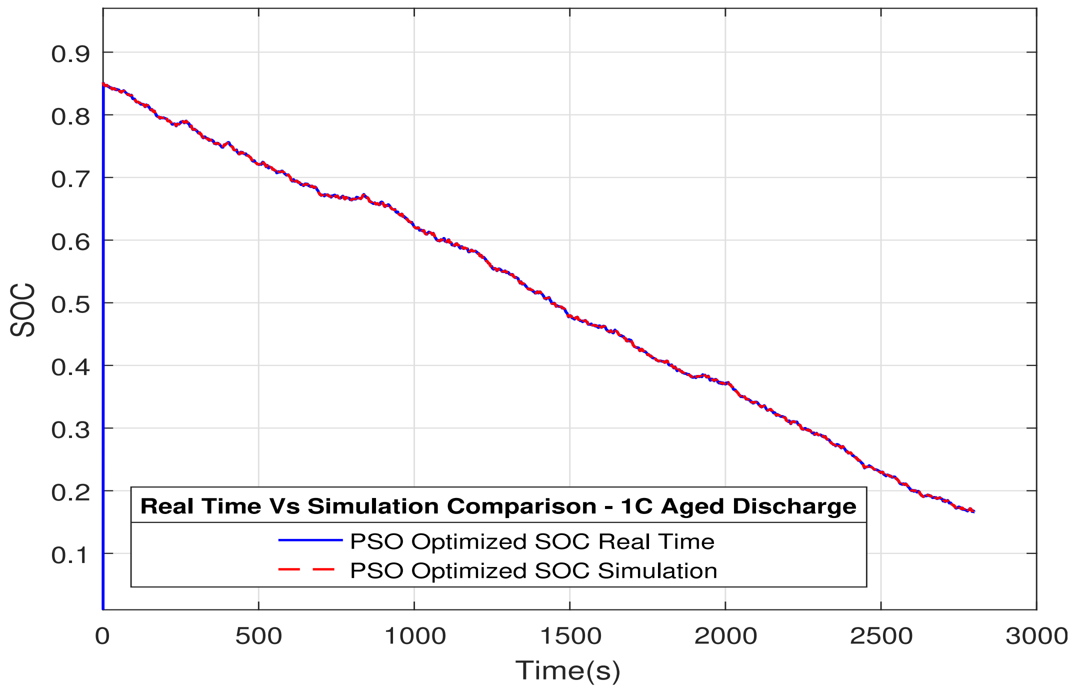

Fig. 5.38. Simulation Vs Real Time SOC.

6. CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusion

The major goals of this work, as stated earlier, is to develop an a battery model and a PSO algorithm to accurately estimate the SOC, which is also capable of running in real-time.

A modular single cell electrochemical model was developed in Simulink and its performance was theoretically verified. A basic version of PSO with constricting factor was proposed and the algorithm was developed in Simulink. Experimental data was collected for Healthy and Aged battery in order to validate the PSO algorithm.

Particle Swarm Optimization is a promising technique for estimation of State of Charge. Results show a close agreement between the estimated value and experimental values:

- The SOC estimation for charging of a Healthy battery is 0.0355.
- The SOC estimation for discharging of a Healthy battery is 0.03186.
- The SOC estimation for charging of a Aged battery is 0.03016.
- The SOC estimation for discharging of a Aged battery is 0.0251.

It was demonstrated the proposed implementation of the PSO estimation is capable of effectively running on Raspberry Pi in real-time processor with reasonable performance.

There were some limitations observed:

- Initial conditions needs to be closer to the actual value while starting the estimation to ensure faster convergence.
- Only one out of the six states has been used for optimization in the proposed method.

- The single cell battery models were not optimized against the experimental data. A more accurate and robust estimation of SOC may be achieved with an optimized battery model.

In conclusion, with the availability of a modular battery model in Simulink, like the one proposed in this paper, it gives the ability to perform faster analysis along with the added capability of rapid prototyping on real-time processor.

6.2 Recommendations for Future Work

Simulink based Single Cell Model and PSO algorithm developed in this work has a potential for wide variety of application.

- This work uses only one battery state for PSO, a natural progression is to use all six states to achieve robustness.
- Only voltage has been used in this work for fitness evaluation of the optimization algorithm. This can be extended to include cell temperature in the fitness evaluation.
- PSO algorithm developed here may be used to estimate other battery states like, State of Health(SOH).
- Incorporate advanced PSO algorithms to address dynamic nature of the system.
- Raspberry Pi provides the ability to connect Voltage and current sensors from a real battery. One of the next steps would be to run and verify the PSO estimation in autonomous mode where the signals from an actual battery can be directly used.

REFERENCES

REFERENCES

[1] Linden and Reddy, *Handbook of Batteries,Third Edition.* McGraw Hill, 2014.

[2] (Last Accessed Feb 11, 2017) Battery university. [Online]. Available: http://batteryuniversity.com/learn/article

[3] C. Moo, K. Ng, Y. Chen, and Y. Hsieh, “State-of-charge estimation with open-circuit-voltage for lead-acid batteries,” *Fourth Power Conversion Conference-NAGOYA, PCC-NAGOYA 2007 - Conference Proceedings*, pp. 758 – 762, 2007.

[4] X. Hu, F. Sun, and Y. Zou, “Estimation of state of charge of a lithium-ion battery pack for electric vehicles using an adaptive luenberger observer,” *Energies*, vol. 3, no. 9, pp. 1586 – 1603, 2010.

[5] C. Moo, K. Ng, Y. Chen, and Y. Hsieh, “State-of-charge estimation with open-circuit-voltage for lead-acid batteries,” *Journal of power sources*, vol. 96, no. 1, pp. 113–120, 2001.

[6] G. Dong, J. Wei, C. Zhang, and Z. Chen, “Online state of charge estimation and open circuit voltage hysteresis modeling of lifepo4battery using invariant imbedding method,” *Applied Energy*, vol. 162, pp. 163 – 171, 2016.

[7] G.L.Plett, “Extended kalman filtering for battery management systems of lipb-based hev battery packs: Part 1. background,” *Journal of power sources*, vol. 134, no. 2, pp. 252–261, 2004.

[8] G.L.Plett, “Extended kalman filtering for battery management systems of lipb-based hev battery packs: Part 2. modeling and identification,” *Journal of power sources*, vol. 134, no. 2, pp. 262–276, 2004.

[9] G.L.Plett, “Extended kalman filtering for battery management systems of lipb-based hev battery packs: Part 3. state and parameter estimation,” *Journal of power sources*, vol. 134, no. 2, pp. 277–292, 2004.

[10] B. Pattel, S. Anwar, and H. Borhan, “An evaluation of the moving horizon estimation algorithm for online estimation of battery state of charge and state of health,” *ASME International Mechanical Engineering Congress and Exposition, Proceedings (IMECE)*, vol. 4B, 2014.

[11] N. H. F. Ismail and S. F. Toha, “State of charge estimation of a lithium-ion battery for electric vehicle based on particle swarm optimization,” *2013 IEEE International Conference on Smart Instrumentation, Measurement and Applications, ICSIMA 2013*, 2013.

[12] A. J. Salkind, C. Fennie, P. Singh, T. Atwater, and D. E. Reisner, “Determination of state-of-charge and state-of-health of batteries by fuzzy logic methodology,” *Journal of Power Sources*, vol. 80, pp. 293–300, 1999.

[13] S. R. Bhatikar, R. L. Mahajan, K. Wipke, and V. Johnson, "Artificial neural network based energy storage system modeling for hybrid electric vehicles," *SAE Technical Papers*, 2000.

[14] P. Singh and D. Reisner, "Fuzzy logic-based state-of-health determination of lead acid batteries," *INTELEC, International Telecommunications Energy Conference (Proceedings)*, pp. 583 – 590, 2002.

[15] G. Paganelli, Y. G. Guezennec, H. Kim, and A. Brahma, "Battery dynamic modeling and real-time state-of-charge estimation in hybrid electric vehicle application," *ASME International Mechanical Engineering Congress and Exposition, Proceedings*, vol. 2, pp. 1101 – 1107, 2001.

[16] D. Zhang, B. N. Popov, and R. E. White, "Modeling lithium intercalation of a single spinel particle under potentiodynamic control," *Journal of The Electrochemical Society*, vol. 147, no. 3, p. 831, 2000.

[17] M. Doyle, T. F. Fuller, and J. Newman, "Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell," *Journal of The Electrochemical Society*, vol. 140, no. 6, pp. 1526–1533, 1993.

[18] V. R. Subramanian, V. Boovaragavan, V. Ramadesigan, and M. Arabandi, "Mathematical model reformulation for lithium-ion battery simulations: Galvanostatic boundary conditions," *Journal of The Electrochemical Society*, vol. 156, no. 4, pp. 260–271, 2009.

[19] K. A. Smith, C. D. Rahn, and C.-Y. Wang, "Control oriented 1d electrochemical model of lithium ion battery," *Science Direct*, vol. 48, no. 9, pp. 2565–2578, 2007.

[20] S. Pramanik and S. Anwar, "Electrochemical model based charge optimization for lithium-ion batteries," *Journal of Power Sources*, vol. 313, pp. 164 – 177, 2016.

[21] N. A. Chaturvedi, R. Klein, J. Christensen, J. Ahmed, and A. Kojic, "Algorithms for advanced battery-management systems," *IEEE Control Systems Magazine*, vol. 30, no. 3, pp. 49 – 68, 2010.

[22] R. Klein, N. A. Chaturvedi, J. Christensen, J. Ahmed, R. Findeisen, and A. Kojic, "Electrochemical model based observer design for a lithium-ion battery," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 2, pp. 289 – 301, 2013.

[23] R. Klein, N. A. Chaturvedi, J. Christensen, J. Ahmed, R. Findeisen, and A. Kojic, "Optimal charging strategies in lithium-ion battery," *Proceedings of the American Control Conference*, pp. 382 – 387, 2011.

[24] M. T. Das, L. C. Dulger, and G. S. Das, "Robotic applications with particle swarm optimization (pso)," *2013 International Conference on Control, Decision and Information Technologies, CoDIT 2013*, pp. 160 – 165, 2013.

[25] M. A. Mohandes, "Modeling global solar radiation using particle swarm optimization (pso)," *Solar Energy*, vol. 86, no. 11, pp. 3137 – 3145, 2012.

[26] Y. Meng, “A swarm intelligence based algorithm for proteomic pattern detection of ovarian cancer,” *Proceedings of the 2006 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, CIBCB’06*, pp. 44 – 50, 2006.

[27] F. Zhao, A. Zhu, D. Yu, and Y. Yang, “A hybrid particle swarm optimization (pso) algorithm schemes for integrated process planning and production scheduling,” *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)*, vol. 2, pp. 6772 – 6776, 2006.

[28] M. A. Rahman, S. Anwar, and A. Izadian, “Electrochemical model based fault diagnosis of a lithium ion battery using multiple model adaptive estimation approach,” *Proceedings of the IEEE International Conference on Industrial Technology*, vol. 2015-June, pp. 210 – 217, 2015.

[29] H. H. Afshari, M. Attari, R. Ahmed, M. Farag, and S. Habibi, “Modeling, parameterization, and state of charge estimation of li-ion cells using a circuit model,” *2016 IEEE Transportation Electrification Conference and Expo, ITEC 2016*, 2016.

[30] T. Kim, W. Qiao, and L. Qu, “Real-time state of charge and electrical impedance estimation for lithium-ion batteries based on a hybrid battery model,” *Conference Proceedings - IEEE Applied Power Electronics Conference and Exposition - APEC*, pp. 563 – 568, 2013.

[31] C. K. Samanta, S. K. Padhy, S. P. Panigrahi, and B. K. Panigrahi, “Hybrid swarm intelligence methods for energy management in hybrid electric vehicles,” *IET Electrical Systems in Transportation*, vol. 3, no. 1, pp. 22 – 29, 2013.

[32] Z. Chen, R. Xiong, K. Wang, and B. Jiao, “Optimal energy management strategy of a plug-in hybrid electric vehicle based on a particle swarm optimization algorithm,” *Energies*, vol. 8, no. 5, pp. 3661 – 3678, 2015.

[33] Z. Chen, R. Xiong, C. Wang, and J. Cao, “An on-line predictive energy management strategy for plug-in hybrid electric vehicles to counter the uncertain prediction of the driving cycle,” *Applied Energy*, vol. 185, pp. 1663 – 1672, 2017.

[34] P. X. Hu. (last accessed February 11, 2017) Swarm intelligence. [Online]. Available: http://www.swarmintelligence.org/

[35] D. R. Eberhart and D. Y. Shi, *Computational Intelligence,First Edition.* Elsevier, 2007.

[36] M. Clerc and J. Kennedy, “The particle swarm-explosion, stability, and convergence in a multidimensional complex space,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58 – 73, 2002.

APPENDIX

A. APPENDIX

One of the goals for the battery model is the ability to run on an embedded device. Additional steps taken to generate efficient code is explained in this section.

A.1 Structure definition for Constants

The structures associated with the electrodes and the electrolyte are defined as Simulink structures. This is a compact way of passing all the constants to the function. Additionally, this produces compact code for running in real-time. The structure definitions for the constants are shown here.

```
   Positive Electode Constant
**************************************************************

PosEle_Const = struct('Length',80.0*1e-6,... %Thickness of Electrode
  'Nodes',81,...       %Number of nodes
  'Grid',1.0000e-06,... %Spacing of Grid
  'SoildDiff',1.0*1e-14,... %Soild Phase diffusivity
   'ElyteDiff',1.6478*1e-11,...%Electrolyte Solid Phase
                        Diffusivity
   'Sigma',100,...
   'REff',2.164e-11,...
   'Max_SP_Conc',51554,...%Max Solid Phase concentration
                          51554
   'Epss',0.5,...%Volume of Solid Fraction in the Electrode
   'Epsl',0.385,...%Volume of Electrolyte in positive
                        Electrode
   'Radius',2e-6);%Raduis of particle

   Negative Electode Constant
***************************************************************

NegEle_Const = struct('Length',88.0*1e-6,...%Thickness of Electrode
  'Nodes',89,...    %Number of nodes
  'Grid',1.0000e-06,...  %Spacing of Grid
  'SoildDiff',3.9*1e-14,...%Soild Phase diffusivity
  'ElyteDiff',4.1498*1e-11,...%Electrolyte Solid Phase
                          Diffusivity
  'Sigma',100,...
  'REff',1.942e-11,...
  'Max_SP_Conc',30555,... %Max Solid Phase concentration
                          30555
  'Epss',0.4824,...%Volume of Solid Fraction in the
                         Electrode
                      0.4824
'Epsl',0.485,...%Volume of Electrolyte in Negative
                          Electrode
'Radius',2e-6); %Raduis of particle

   Negative Electode Constant
***************************************************************

Elyct_Const = struct('Length',35.0*1e-6,...%Thickness of Electrolyte
'Nodes',36,...%Number of nodes
'Grid',1.0000e-06,...%Spacing of Grid
'ElyteDiff',7.5*1e-10,...%Electrolyte Phase Diffusivity
'Epss',0.1,...%Volume of Solid Fraction in the Seperator
'Epsl',0.724);%Volume of Electrolye Fraction in the Seperator
```

A.2 Simulink Bus definition for States

The states for electrodes and electrolytes are defined as Simulink Bus. This is an efficient and compact way of handling the multi-dimensional signals with heterogeneous data type. The bus definitions are shown here.

```
   Electrode Simulink Bus
***************************************************************

function Electrode_BusCreate()
% Electrode_BusCreate initializes a set of bus objects in the
MATLAB base workspace

% Bus object: slBus7
clear elems;
elems(1) = Simulink.BusElement;
elems(1).Name = ’Ce’;
elems(1).Dimensions = [1 1];
elems(1).DimensionsMode = ’Fixed’;
elems(1).DataType = ’double’;
elems(1).SampleTime = -1;
elems(1).Complexity = ’real’;
elems(1).SamplingMode = ’Sample based’;
elems(1).Min = [];
elems(1).Max = [];
elems(1).DocUnits = ’’;
elems(1).Description = ’’;

elems(2) = Simulink.BusElement;
elems(2).Name = ’Cs’;
elems(2).Dimensions = [1 81];
elems(2).DimensionsMode = 'Fixed';
elems(2).DataType = 'double';
elems(2).SampleTime = -1;
elems(2).Complexity = 'real';
elems(2).SamplingMode = 'Sample based';
elems(2).Min = [];
elems(2).Max = [];
elems(2).DocUnits = '';
elems(2).Description = '';

elems(3) = Simulink.BusElement;
elems(3).Name = 'Css';
elems(3).Dimensions = [1 81];
elems(3).DimensionsMode = 'Fixed';
elems(3).DataType = 'double';
elems(3).SampleTime = -1;
elems(3).Complexity = 'real';
elems(3).SamplingMode = 'Sample based';
elems(3).Min = [];
elems(3).Max = [];
elems(3).DocUnits = '';
elems(3).Description = '';

elems(4) = Simulink.BusElement;
elems(4).Name = 'Ds';
elems(4).Dimensions = [1 81];
elems(4).DimensionsMode = 'Fixed';
elems(4).DataType = 'double';
elems(4).SampleTime = -1;
elems(4).Complexity = 'real';
elems(4).SamplingMode = 'Sample based';
elems(4).Min = [];
elems(4).Max = [];
elems(4).DocUnits = '';
elems(4).Description = '';

elems(5) = Simulink.BusElement;
elems(5).Name = 'Ie';
elems(5).Dimensions = [1 81];
elems(5).DimensionsMode = 'Fixed';
elems(5).DataType = 'double';
elems(5).SampleTime = -1;
elems(5).Complexity = 'real';
elems(5).SamplingMode = 'Sample based';
elems(5).Min = [];
elems(5).Max = [];
elems(5).DocUnits = '';
elems(5).Description = '';

elems(6) = Simulink.BusElement;
elems(6).Name = 'Us';
elems(6).Dimensions = [1 81];
elems(6).DimensionsMode = 'Fixed';
elems(6).DataType = 'double';
elems(6).SampleTime = -1;
elems(6).Complexity = 'real';
elems(6).SamplingMode = 'Sample based';
elems(6).Min = [];
elems(6).Max = [];
elems(6).DocUnits = '';
elems(6).Description = '';

elems(7) = Simulink.BusElement;
elems(7).Name = 'Ke';
elems(7).Dimensions = [1 81];
elems(7).DimensionsMode = 'Fixed';
elems(7).DataType = 'double';
elems(7).SampleTime = -1;
elems(7).Complexity = 'real';
elems(7).SamplingMode = 'Sample based';
elems(7).Min = [];
elems(7).Max = [];
elems(7).DocUnits = '';
elems(7).Description = '';

elems(8) = Simulink.BusElement;
elems(8).Name = 'Ue';
elems(8).Dimensions = [1 81];
elems(8).DimensionsMode = 'Fixed';
elems(8).DataType = 'double';
elems(8).SampleTime = -1;
elems(8).Complexity = 'real';
elems(8).SamplingMode = 'Sample based';
elems(8).Min = [];
elems(8).Max = [];
elems(8).DocUnits = '';
elems(8).Description = '';
elems(9) = Simulink.BusElement;
elems(9).Name = ’qs’;
elems(9).Dimensions = [1 81];
elems(9).DimensionsMode = ’Fixed’;
elems(9).DataType = ’double’;
elems(9).SampleTime = -1;
elems(9).Complexity = ’real’;
elems(9).SamplingMode = ’Sample based’;
elems(9).Min = [];
elems(9).Max = [];
elems(9).DocUnits = ’’;
elems(9).Description = ’’;

elems(10) = Simulink.BusElement;
elems(10).Name = ’U_eq’;
elems(10).Dimensions = [1 81];
elems(10).DimensionsMode = ’Fixed’;
elems(10).DataType = ’double’;
elems(10).SampleTime = -1;
elems(10).Complexity = ’real’;
elems(10).SamplingMode = ’Sample based’;
elems(10).Min = [];
elems(10).Max = [];
elems(10).DocUnits = ’’;
elems(10).Description = ’’;

elems(11) = Simulink.BusElement;
elems(11).Name = ’eta’;
elems(11).Dimensions = [1 81];
elems(11).DimensionsMode = 'Fixed';
elems(11).DataType = 'double';
elems(11).SampleTime = -1;
elems(11).Complexity = 'real';
elems(11).SamplingMode = 'Sample based';
elems(11).Min = [];
elems(11).Max = [];
elems(11).DocUnits = '';
elems(11).Description = '';

elems(12) = Simulink.BusElement;
elems(12).Name = 'q';
elems(12).Dimensions = [1 81];
elems(12).DimensionsMode = 'Fixed';
elems(12).DataType = 'double';
elems(12).SampleTime = -1;
elems(12).Complexity = 'real';
elems(12).SamplingMode = 'Sample based';
elems(12).Min = [];
elems(12).Max = [];
elems(12).DocUnits = '';
elems(12).Description = '';

elems(13) = Simulink.BusElement;
elems(13).Name = 'J';
elems(13).Dimensions = [1 81];
elems(13).DimensionsMode = 'Fixed';
elems(13).DataType = 'double';
elems(13).SampleTime = -1;
elems(13).Complexity = 'real';
elems(13).SamplingMode = 'Sample based';
elems(13).Min = [];
elems(13).Max = [];
elems(13).DocUnits = '';
elems(13).Description = '';

elems(14) = Simulink.BusElement;
elems(14).Name = 'q_lump';
elems(14).Dimensions = [1 81];
elems(14).DimensionsMode = 'Fixed';
elems(14).DataType = 'double';
elems(14).SampleTime = -1;
elems(14).Complexity = 'real';
elems(14).SamplingMode = 'Sample based';
elems(14).Min = [];
elems(14).Max = [];
elems(14).DocUnits = '';
elems(14).Description = '';

elems(15) = Simulink.BusElement;
elems(15).Name = 'SOC';
elems(15).Dimensions = [1 81];
elems(15).DimensionsMode = 'Fixed';
elems(15).DataType = 'double';
elems(15).SampleTime = -1;
elems(15).Complexity = 'real';
elems(15).SamplingMode = 'Sample based';
elems(15).Min = [];
elems(15).Max = [];
elems(15).DocUnits = '';
elems(15).Description = '';

elems(16) = Simulink.BusElement;
elems(16).Name = 'i_0';
elems(16).Dimensions = [1 81];
elems(16).DimensionsMode = 'Fixed';
elems(16).DataType = 'double';
elems(16).SampleTime = -1;
elems(16).Complexity = 'real';
elems(16).SamplingMode = 'Sample based';
elems(16).Min = [];
elems(16).Max = [];
elems(16).DocUnits = '';
elems(16).Description = '';

elems(17) = Simulink.BusElement;
elems(17).Name = 'Type';
elems(17).Dimensions = [1 1];
elems(17).DimensionsMode = 'Fixed';
elems(17).DataType = 'double';
elems(17).SampleTime = -1;
elems(17).Complexity = 'real';
elems(17).SamplingMode = 'Sample based';
elems(17).Min = [];
elems(17).Max = [];
elems(17).DocUnits = '';
elems(17).Description = '';

temp = Simulink.Bus;
temp.HeaderFile = '';
temp.Description = '';
temp.DataScope = 'Auto';
temp.Alignment = -1;
temp.Elements = elems;
clear elems;
  assignin('base','Electrode_Bus', temp);

  Electrolyte Simulink Bus
***************************************************************

function temp = Elyct_BusCreate()
% Elyct_BusCreate initializes a set of bus objects in the
MATLAB base workspace

% Bus object: slBus9
clear elems;
elems(1) = Simulink.BusElement;
elems(1).Name = 'Ie';
elems(1).Dimensions = [1 36];
elems(1).DimensionsMode = 'Fixed';
elems(1).DataType = 'double';
elems(1).SampleTime = -1;
elems(1).Complexity = 'real';
elems(1).SamplingMode = 'Sample based';
elems(1).Min = [];
elems(1).Max = [];
elems(1).DocUnits = '';
elems(1).Description = '';

elems(2) = Simulink.BusElement;
elems(2).Name = 'Ke';
elems(2).Dimensions = [1 36];
elems(2).DimensionsMode = 'Fixed';
elems(2).DataType = 'double';
elems(2).SampleTime = -1;
elems(2).Complexity = 'real';
elems(2).SamplingMode = 'Sample based';
elems(2).Min = [];
elems(2).Max = [];
elems(2).DocUnits = '';
elems(2).Description = '';

elems(3) = Simulink.BusElement;
elems(3).Name = 'Ue';
elems(3).Dimensions = [1 36];
elems(3).DimensionsMode = 'Fixed';
elems(3).DataType = 'double';
elems(3).SampleTime = -1;
elems(3).Complexity = 'real';
elems(3).SamplingMode = 'Sample based';
elems(3).Min = [];
elems(3).Max = [];
elems(3).DocUnits = '';
elems(3).Description = '';
elems(4) = Simulink.BusElement;
elems(4).Name = 'q';
elems(4).Dimensions = [1 36];
elems(4).DimensionsMode = 'Fixed';
elems(4).DataType = 'double';
elems(4).SampleTime = -1;
elems(4).Complexity = 'real';
elems(4).SamplingMode = 'Sample based';
elems(4).Min = [];
elems(4).Max = [];
elems(4).DocUnits = '';
elems(4).Description = '';

temp = Simulink.Bus;
temp.HeaderFile = '';
temp.Description = '';
temp.DataScope = 'Auto';
temp.Alignment = -1;
temp.Elements = elems;
clear elems;           %Volume of Electrolye Fraction in the Seperator
```