A Kalman Filter Based Battery State of Charge Estimation MATLAB Function

Fauzia Khanum*, Eduardo Louback*, Federico Duperly*, Colleen Jenkins*,
Phillip J. Kollmeyer*, *Member, IEEE* Ali Emadi*, *Fellow, IEEE*
*McMaster Automotive Resource Center (MARC), McMaster University, Hamilton, ON, Canada
Email: khanumf@mcmaster.ca, barbosae@mcmaster.ca, duperlf@mcmaster.ca, jenkinsc@mcmaster.ca,
kollmeyp@mcmaster.ca, emadi@mcmaster.ca

2021 IEEE Transportation Electrification Conference & Expo (ITEC) | 978-1-7281-7583-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/ITEC51675.2021.9490163

***Abstract*—This paper proposes a Kalman filter based state-of-charge (SOC) estimation MATLAB function using a second-order RC equivalent circuit model (ECM). The function requires the SOC-OCV (open circuit voltage) curve, internal resistance, and second-order RC ECM battery parameters. Users have an option to use an extended Kalman filter (EKF) or adaptive extended Kalman filter (AEKF) algorithms as well as temperature dependent battery data. An example of the function is illustrated using the LA92 driving cycle of a Turnigy battery performed at multiple temperature ranging from -10°C to 40°C.**

***Index Terms*—Kalman Filter, Lithium-ion battery, MATLAB function, State of Charge, Temperature.**

I. Introduction

The transportation industry is the second-largest emitter of greenhouse gases in Canada and has increased by 27% between 2000 and 2017 [1]. To decrease greenhouse gases emitted by the industry, electric vehicles are one of the most viable solutions, as their emissions are up to 67% less than fuel-powered vehicles [2].

Electric vehicles require a complex battery management system responsible for monitoring the state-of-charge (SOC), state of health (SOH), and cell temperature [3]. These parameters cannot always be directly measured, so are evaluated through models. The SOC is commonly estimated through Coulomb Counting (CC), as shown in (1) [4]:

$$SOC = 1 - \frac{\int_{t_0}^{t} i\eta \, dt}{C_n} \tag{1}$$

Where C_n is the battery nominal capacity, η is the Coulombic efficiency, and i is the battery current. This method is often inaccurate due to errors in the current sensor and determination of initial SOC [5]. Another estimation technique is the SOC-Open Circuit Voltage (SOC-OCV) mapping, which approximates SOC based on the battery's OCV measurement. This method alone is not practical for dynamic applications and batteries that present a non-linear SOC-OCV relationship. They can induce considerable estimation error, primarily if they exhibit a flat region of the curve. Combining CC and SOC-OCV mapping approaches with equivalent circuit models (ECM) and Kalman filter (KF) can produce more accurate SOC estimations [6] [7].

A variety of papers have been published with different Kalman filtering techniques, battery testing conditions, and battery models. In [8], the KF was compared against an unscented KF, while in [9] an extended KF (EKF) was compared to the central difference KF. In [10], an EKF was used with a PNGV battery model using MATLAB/Simulink. In [11], several battery parameter estimation methods are compared using a Thevenin battery model and EKF. These are a handful of the hundreds of published results on SOC estimation using KF. Despite these findings, very few publicly available tools, functions, or scripts are available for researchers. One example available online is [12]. It is a MATLAB/Simulink based SOC estimation using EKF and unscented KF.

Battery parameters are significantly impacted by temperature, in cold temperatures, for example, the capacity decreases and resistance increases [13] [14]. However, there are not many SOC estimation studies that consider this impact in more detail. For example, in [15], the authors estimated the state of charge only at three different temperatures. Although the temperatures covered a wide range of 36°C, the paper did not capture the dynamics of the entire temperature range.

In this paper, we propose a MATLAB function with the objective to provide a public tool that estimates the battery SOC and terminal voltage at different temperatures using a second-order resistor-capacitor (2RC) ECM along with an extended Kalman filter (EKF). The goal of this work is to have a MATLAB function to serve as a basis for other researchers to utilize.

The paper is organized as follows: the state of charge estimation algorithm is presented in section II, while the required battery data and how to use the function are explained in section III. In section IV an example use of the function is shown, and the conclusions and future work are discussed in section V.

II. State of Charge Estimation Algorithm

One of the most common battery models seen in literature is the 2RC ECM (Fig. 1). It consists of the battery OCV, internal resistance, and two parallel RC pairs. Once these parameters have been optimized, the discrete-time state-space form of the battery model is used in the EKF algorithm. Given battery measurements (i.e. current, voltage, temperature) over time, the EKF will estimate the unknown variables in a dynamic system.

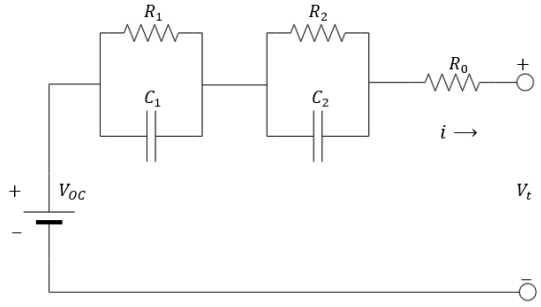

Fig. 1. Second Order Resistor-Capacitor ECM Diagram.

A. Battery Modelling

In Fig. 1, the OCV is represented by V_{OC}, the output terminal voltage by V_t, and the internal resistance of the battery by R_0. The voltage across the first RC network is V_1 and V_2 across the second network. (2) to (4) describe the ECM dynamics in state-space [16].

$$SOC(k+1) = SOC(k) - \frac{\eta \Delta t \; i(k)}{C_n} \tag{2}$$

$$V_t(k) = V_{OC}(k) - V_1(k) - V_2(k) - i(k)R_0 \tag{3}$$

$$\begin{aligned} V_1(k+1) &= \exp^{\frac{-\Delta t}{R_1 C_1}} V_1(k) + R_1(1 - \exp^{\frac{-\Delta t}{R_1 C_1}})i(k) \\ V_2(k+1) &= \exp^{\frac{-\Delta t}{R_2 C_2}} V_2(k) + R_2(1 - \exp^{\frac{-\Delta t}{R_2 C_2}})i(k) \end{aligned} \tag{4}$$

The input to the model is the current $i(k)$ at time step k. Δt is the sample time in seconds, R_1, C_1, R_2 and C_2 are the RC model parameters. V_{OC} in (3) is calculated in (5) as a function of SOC and battery surface temperature. Each SOC-OCV curve is calculated using the Hybrid Pulse Power Characterization (HPPC) test results obtained at 40°C, 25°C, 10°C, 0°C, and -10°C [17].

$$V_{OC} = f(SOC, Temperature) \tag{5}$$

The state and measurement equations can be calculated in (6) and (7) as follows:

$$x_{k+1} = A_k x_k + B_k u_k \tag{6}$$

$$z_k = C_k x_k + D_k u_k \tag{7}$$

Where x_{k+1} is the system state vector at time $k+1$, the state variables are $x = [SOC, V_1, V_2]$, system input is $u_k = i_k$, and system output is $z_k = V_t$. The A, B, C, and D matrices are given by (8) to (11) using (2) to (4):

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \exp^{\frac{-\Delta t}{R_1 C_1}} & 0 \\ 0 & 0 & \exp^{\frac{-\Delta t}{R_2 C_2}} \end{bmatrix} \tag{8}$$

$$B = \begin{bmatrix} -\frac{\Delta t}{Q}\eta[k] \\ R_1(1 - \exp^{\frac{-\Delta t}{R_1 C_1}}) \\ R_2(1 - \exp^{\frac{-\Delta t}{R_2 C_2}}) \end{bmatrix} \tag{9}$$

$$C = \begin{bmatrix} \frac{\partial V_{OC}}{\partial SOC} & \frac{\partial V}{\partial V_1} & \frac{\partial V}{\partial V_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial V_{OC}}{\partial SOC} & -1 & -1 \end{bmatrix} \tag{10}$$

$$D = -R_0 \tag{11}$$

B. Extended Kalman Filter

The EKF, a version of the regular KF, is used to estimate the states for a non-linear system. EKF uses a two-step prediction-correction algorithm as described in (12) to (16), where k denotes a discrete point in time, K is the Kalman gain, P is the covariance of the measurement error, Q is the covariance of the process, and R is the covariance of the output [18]. First, a prediction or time update is done and then the correction or measurement update. This cycle repeats until the end of the data. Note, the hat symbol, $\hat{}$, represents an estimate of a variable, $|k$ denotes predicted or a-priori estimate, and $|k+1$ denotes updated or a-posteriori estimate.

Prediction (Time Update)

1. Project the states ahead (a-priori):

$$\hat{x}_{k+1|k} = A\hat{x}_{k|k} + Bu_k \tag{12}$$

2. Project the error covariance ahead:

$$P_{k+1|k} = AP_{k|k}A^T + Q_k \tag{13}$$

Correction (Measurement Update)

1. Compute the Kalman gain:

$$K_{k+1} = P_{k+1|k}C^T(CP_{k+1|k}C^T + R_{k+1})^{-1} \tag{14}$$

2. Update the estimate with measurement z_k (a-posteriori):

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}(z_{k+1} - C\hat{x}_{k+1|k}) \tag{15}$$

3. Update the error covariance:

$$P_{k+1|k+1} = (1 - K_{k+1}C)P_{k+1|k} \tag{16}$$

Since the EKF is based on a Gaussian distribution, the states described in (8) to (11) need to be linearized for the algorithm to work properly. Matrix C as seen in (**??**) is the only matrix that requires linearization as the battery's SOC-OCV relationship is non-linear.

A fourth optional step was added to the EKF equations based on [19] in which the matrix Q is updated with each iteration using (17). Thus creating an AEKF, so the noise covariance can be updated online.

$$Q_k = K_k * Error_k * K_k^t \tag{17}$$

III. EKF SOC Estimation Function

This section will review the required battery data as well as the syntax and commands in order to utilize the EKF SOC estimation function. One can find publicly available battery test data at [17] [20] [21]. The MATLAB function, based on [18], as well as the example illustrated below can be found at [22].

A. Battery Parameters

Before using the `EKF_SOC_Estimation` function, users will need the SOC-OCV curve, R_0, and the 2RC ECM battery parameters for the specific battery that the SOC is being estimated for. This data is loaded within the function and are not passed in as function parameters.

Appropriate battery testing should be done to obtain data for OCV as a function of SOC for a desired range of battery temperatures (i.e., multiple datasets, each for a different battery temperature as shown in Fig. 2). This testing typically involves charging and discharging the battery at low currents (0.05C), however, the complete details of this testing may vary. Once finalized datasets are obtained, it is important to ensure that there are no repeated SOC points (especially if these points have differing OCV values) as this will cause an interpolation error when the function is running. This does not apply to repeated OCV values.

Typical 2RC ECM parameters are R_0, R_1, C_1, R_2 and C_2, where R_0 represents the internal resistance of the cell, and the rest of the parameters represent non-physical characteristics of the cell, which when grouped into RC branches, as shown in Fig. 3, describe the cell's dynamics. To obtain these parameters at different temperatures as functions of SOC, a model-based parameter optimization approach within MATLAB [23] was used. As described by [24], HPPC test data at different temperatures [17] was evaluated within the optimization algorithm to ensure the ECM accurately described the behaviour of the battery.

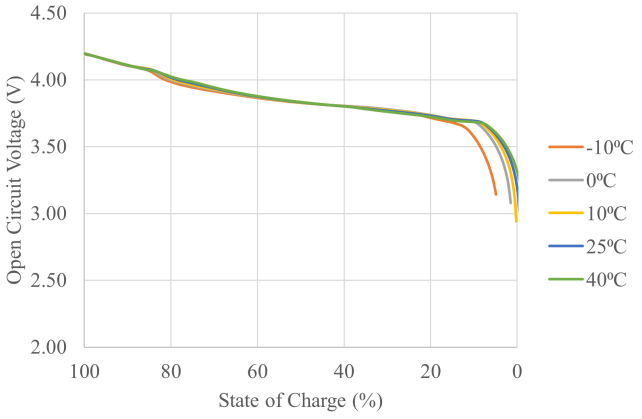

Fig. 2. OCV vs. SOC of a Turnigy Graphene 5000mAh Li-ion Battery at different temperatures.

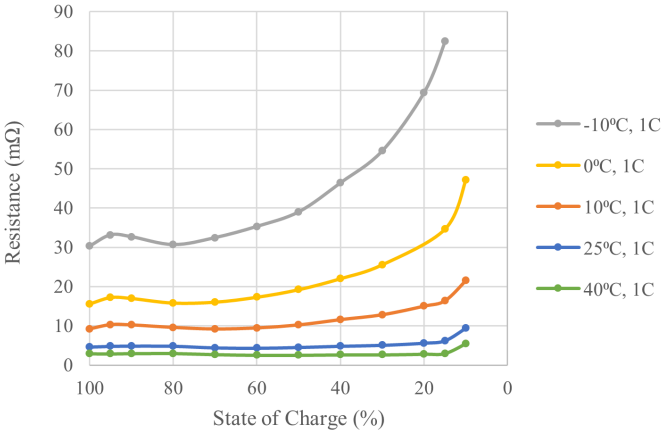

Fig. 3. Discharge resistance vs SOC of a Turnigy Graphene 5000mAh Li-ion Battery at different temperatures.

The function can be called using:

```
function [SOC_Estimated, Vt_Estimated, ...
    Vt_Error] = EKF_SOC_Estimation(Current, ...
    Vt_Actual, Temperature)
```

The function takes a drive cycle current measured in amps (`Current`), V_t measured in volts (`Vt_Actual`), and battery temperature measured in °C (`Temperature`), all as vectors of type double as input. The length of all vectors must be the same. The function outputs a vector of the estimated SOC (`SOC_Estimated`), estimated V_t (`Vt_Estimated`), and the error between V_t measured and V_t estimated (`Vt_Error`).

The function by default loads the provided `'BatteryModel.mat'` and `'SOC-OCV.mat'` files which are labeled tables. The BatteryModel table contains the SOC, R_0, R_1, C_1, R_2, C_2, and T data in columns 1 to 7, respectively. The SOC ranges from 0% to 100% by intervals of 10%, however, users can vary the intervals as needed. The SOC-OCV table contains the SOC, OCV, and T data in columns 1 to 3, respectively.

```
load 'BatteryModel.mat'; % Load the battery ...
    parameters
load 'SOC-OCV.mat'; % Load the SOC-OCV curve
```

The initial SOC is set between 0 and 1 where 0 is 0% and 1 is 100%. This can either be set to the initial SOC of the drive cycle or with a bias to test convergence and robustness. `DeltaT` is the time difference in seconds between each value in the current and X is the initial input and subsequently the system state vector from (6).

```
SOC_Init    = 1; % intial SOC
X           = [SOC_Init; 0; 0]; % state ...
    space x parameter intializations
DeltaT      = 1; % sample time in seconds
Qn_rated    = 4.81 * 3600; % Ah to Amp-seconds
```

The KF has three tunable parameters: R_x, P_x and Q_x. These will need to be adjusted for each battery either manually or through an optimization algorithm. The AEKF algorithm requires substantially different tuning from the EKF, and may be unstable when the same parameters are used.

```
R_x    = 2.5e-5;
P_x    = [0.025 0 0;
0 0.01 0;
0 0 0.01];
Q_x    = [1.0e-6 0 0;
0 1.0e-5 0;
0 0 1.0e-5];
```

The function allows for temperature dependent data and by default utilizes five different temperatures: 40°C, 25°C, 10°C, 0°C, and -10°. The function interpolates the battery parameters between temperatures at each iteration of the cycle using the output function from `scatteredInterpolant`.

```
F_R0 = scatteredInterpolant(param.T,...
    param.SOC,param.R0);
```

```
R0      = F_R0(T,SOC);
```

To use this feature, users will require internal resistance data, and battery parameters for each temperature. If the datasets available to the user is not temperature dependent or the temperature is unknown, the `scatteredInterpolant` function should be replaced with `pchip` or `interp1`.

The SOC-OCV line is curve fitted using the `polyfit` function in a least squares sense. Polynomial differentiation is then completed on the curve to be used to calculate the matrix C. Both the regular curve `SOCOCV` and the differentiated one `dSOCOCV` are then evaluated within the KF loop using `polyval`.

```
SOCOCV   = polyfit(param.SOC,param.OCV,11); % ...
      calculate 11th order polynomial for the ...
      SOC-OCV curve
dSOCOCV = polyder(SOCOCV); % derivative of ...
      SOc-OCV curve for matrix C
```

```
OCV = polyval(SOCOCV,SOC); % calculate the ...
      values of OCV at the given SOC, using ...
      the polynomial SOCOCV
```

```
      dOCV = polyval(dSOCOCV, SOC);
      C_x    = [dOCV -1 -1];
```

To use the function simply as an AEKF instead of an EKF, uncomment out the following line in the code.

```
% Q_x = KalmanGain_x * Error_x * KalmanGain_x';
```

IV. Example Simulation

To illustrate the use of the function, it was evaluated at 40°C, 25°C, 10°C, 0°C, and -10°C using EKF and the battery data available at [17].

A new Turnigy Graphene 5000mAh 65C cell was tested extensively in a thermal chamber by [17]. SOC-OCV mapping and HPPC tests were performed at 40°C, 25°C, 10°C, 0°C, and -10°C. The tests cover SOC range from 100% to 5% with four different charging and discharging currents at 1, 2, 5 and 10 C-rates. After the characterization, the battery was subjected to driving cycles UDDS, HWFET, LA92, US06 as well as combinations of these cycles. The drive cycles were sampled every 0.1 seconds, and other tests were sampled at a slower or variable rate. [17].

The SOC-OCV curve and battery model parameters were optimized for each temperature and SOC level from 100% to 0% at 10% increments using [23]. The battery data was then re-sampled to every second. The convergence and estimation of the SOC and V_t are highly dependent on the battery model parameters. The initial P, Q and R values in the EKF were manually tuned so the average root mean squared error (RMSE) of all temperatures SOC was less than 5% and V_t was less than 100mV. The results of this tuning can be seen in Fig. 4 and Fig. 5. The RMSE at 40°C was 1.75% for SOC and 1mV for V_t for the LA92 drive cycle. Fig. 6 illustrates the measured versus estimated V_t and SOC of the LA92 drive cycle at 40°C. Since there is only one set of KF parameters for all the temperatures, the lower temperatures have higher error. One solution to this is to tune different parameters for each temperature similar to the battery parameters.

To test the robustness of the function, the initial SOC was offset by 10%, and the current was offset by +/-0.1A. Fig. 7 provides details of the SOC and V_t RMSE given these conditions. The system proves to be robust at 40°C as the RMSE values do not differ from each other drastically. With the 10% initial SOC offset, the system reaches within 5% of the Coulomb counted SOC within 3 minutes (Fig. 8). A graph of the +/-0.1A current offset can be seen in Fig. 9.

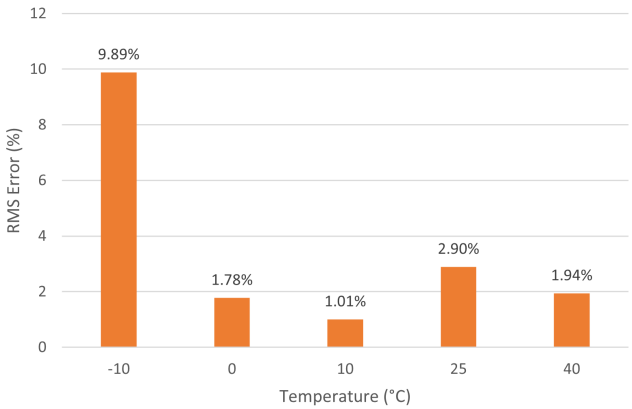

Fig. 4. RMS Error of SOC at different temperatures for LA92 drive cycle.

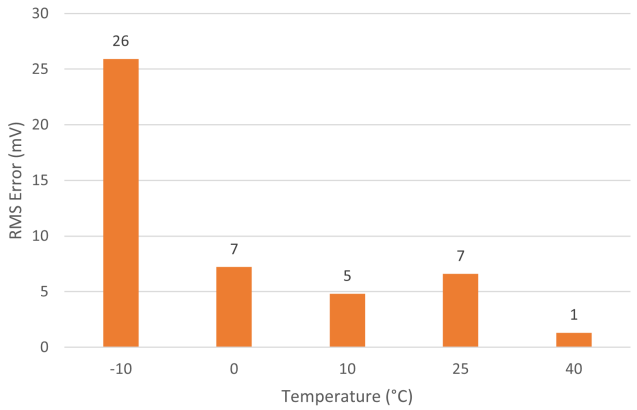

Fig. 5. RMS Error of V_t at different temperatures for LA92 drive cycle.

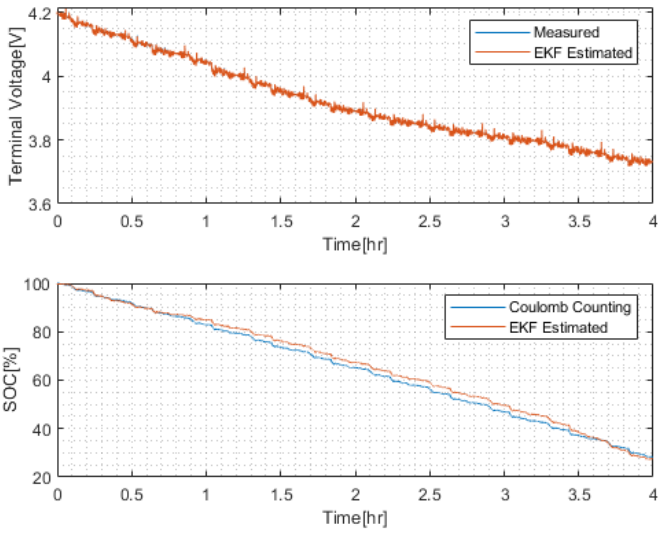

Fig. 6. (a). Measured vs. Estimated Vt at 40°C for LA92 drive cycle. (b). SOC Coloumb Counting vs. SOC EKF Estimation at 40°C for LA92 drive cycle.

The convergence rate and RMSE values of error input can be improved upon with more accurate KF tuning values P, Q, and R.

V. Conclusions

In this paper, a Kalman filter based SOC estimation MATLAB function is proposed. The function is based on a second-order ECM. It allows users to load their specific battery data including the SOC-OCV curves, internal resistance, and the battery model parameters. The function has the flexibility to be used as an EKF or AEKF as well as using battery temperature based data. An example is illustrated with publicly available Turnigy Graphene battery data where less than 2% average RMSE is achieved across the various temperatures. The users of the function have the ability to build on this function or integrate it into a more complex model.

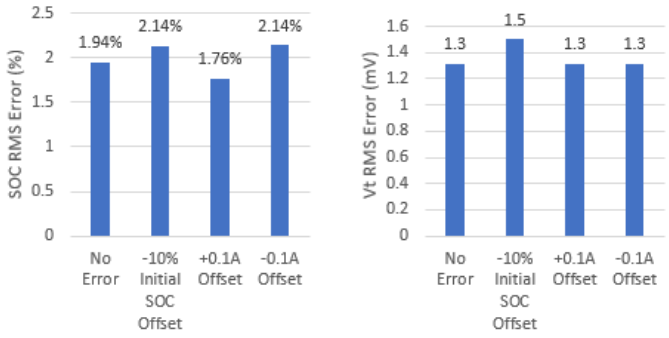

Fig. 7. SOC RMS Error and V_t RMS Error with no error, initial 10% SOC offset, and +/- 0.1A current offset at 40°C for the LA92 drive cycle.

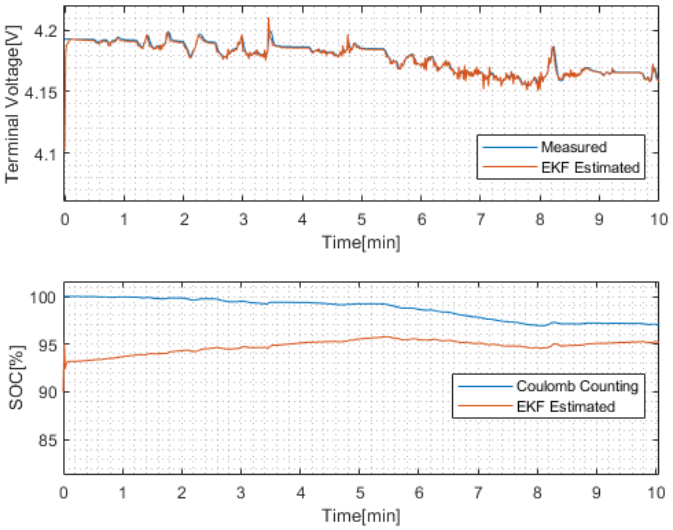

Fig. 8. (a). Measured vs. Estimated Vt with 10% initial SOC offset at 40°C for LA92 drive cycle. (b). SOC Coloumb Counting vs. SOC EKF Estimation with 10% initial SOC offset at 40°C for LA92 drive cycle.

With further tuning of the battery parameters and the KF parameters, the RMSE values can be improved further. This function can be further expanded upon to add a temperature model for more accurate estimations of the battery temperature. It can also be simplified to remove the temperature dependency as well as adaptive portion by adjusting those respective lines.

Acknowledgment

The authors would like to thank Dr. Ryan Ahmed for his contribution and constructive feedback during the development of the code.

References

[1] Environment and Climate Change Canada, "Canadian Environmental Sustainability Indicators: Greenhouse gas emissions," 2020. [Online]. Available: https://www.canada.ca/content/dam/eccc/documents/pdf/cesindicators/ghg-emissions/2020/greenhouse-gas-emissions-en.pdf. [Accessed 12 December 2020].

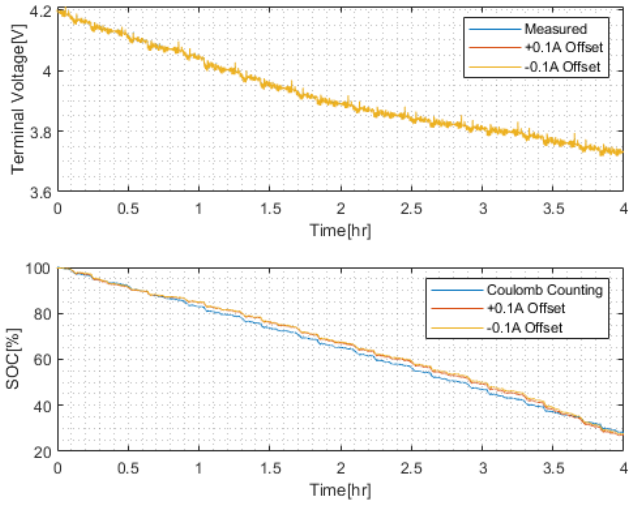

Fig. 9. (a). Measured vs. Estimated Vt with +/-0.1A current offset at 40°C for LA92 drive cycle. (b). SOC Coloumb Counting vs. SOC EKF Estimation with +/-0.1A current offset at 40°C for LA92 drive cycle.

[2] Electric Power Research Institute (EPRI), *Enviromental Assessment of Plug-In Hybrid Electric Vehicles*, vol. 1, p. 46, 2007.

[3] A. Emadi, *Advanced Electric Drive Vehicles*, CRC Press, 2014.

[4] M. Ismail, R. Dlyma, A. Elrakaybi, R. Ahmed and S. Habibi, "Battery state of charge estimation using an Artificial Neural Network," 2017 IEEE Transportation Electrification Conference and Expo (ITEC), Chicago, IL, USA, 2017, pp. 342-349.

[5] Jun Xu, Chunting Chris Mi, Binggang Cao and Junyi Cao, "A new method to estimate the state of charge of lithium-ion batteries based on the battery impedance model," *Journal of Power Sources*, vol. 233, 2013, pp. 277-284.

[6] A. Nugroho, E. Rijanto, F. D. Wijaya and P. Nugroho, "Battery state of charge estimation by using a combination of Coulomb Counting and dynamic model with adjusted gain," 2015 International Conference on Sustainable Energy Engineering and Application (ICSEEA), Bandung, Indonesia, 2015, pp. 54-58.

[7] I. Baccouche, S. Jemmali, B. Manai, R. Chaibi and N. E. Ben Amara, "Hardware implementation of an algorithm based on kalman filtrer for monitoring low capacity Li-ion batteries," 2016 7th International Renewable Energy Congress (IREC), Hammamet, Tunisia, 2016.

[8] I. Jokić, Ž. Zečević and B. Krstajić, "State-of-charge estimation of lithium-ion batteries using extended Kalman filter and unscented Kalman filter," 2018 23rd International Scientific-Professional Conference on Information Technology (IT), Zabljak, Montenegro, 2018, pp. 1-4, doi: 10.1109/SPIT.2018.8350462.

[9] V. Sangwan, R. Kumar and A. K. Rathore, "State-of-charge estimation for li-ion battery using extended Kalman filter (EKF) and central difference Kalman filter (CDKF)," 2017 IEEE Industry Applications Society Annual Meeting, Cincinnati, OH, USA, 2017, pp. 1-6, doi: 10.1109/IAS.2017.8101722.

[10] L. Haoran, L. Liangdong, Z. Xiaoyin and S. Mingxuan, "Lithium Battery SOC Estimation Based on Extended Kalman Filtering Algorithm," 2018 IEEE 4th International Conference on Control Science and Systems Engineering (ICCSSE), Wuhan, China, 2018, pp. 231-235, doi: 10.1109/CCSSE.2018.8724766.

[11] D. Yang, G. Qi and X. Li, "State-of-charge estimation of LiFePO4/C battery based on extended Kalman filter," 2013 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC), Hong Kong, China, 2013, pp. 1-5, doi: 10.1109/APPEEC.2013.6837188.

[12] Chirag (2021). Design and Test Lithium Ion Battery Management Algorithms. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/72865-design-and-test-lithium-ion-battery-management-algorithms. [Accessed: 2021].

[13] C. Vidal, O. Gross, R. Gu, P. Kollmeyer and A. Emadi, "xEV Li-Ion Battery Low-Temperature Effects—Review," in IEEE Transactions on Vehicular Technology, vol. 68, no. 5, pp. 4560-4572, May 2019.

[14] C. Chang, Y. Zheng, and Y. Yu, "Estimation for Battery State of Charge Based on Temperature Effect and Fractional Extended Kalman Filter," Energies, vol. 13, no. 22, p. 5947, Nov. 2020.

[15] Q.-Z. Zhang, Z.-Y. Wang, and H.-M. Yuan, Estimation for SOC of Li-ion battery based on two-order RC temperature model, 28-Jun-2018. [Online]. Available: https://ieeexplore-ieee-org.libaccess.lib.mcmaster.ca/document/8398150. [Accessed: 2020].

[16] G. Plett, *Battery Managment Systems*, vol. 1, 2015.

[17] Kollmeyer, Phillip; Skells, Michael (2020), "Turnigy Graphene 5000mAh 65C Li-ion Battery Data", Mendeley Data, V1, doi: 10.17632/4fx8cjprxm.1

[18] R. Ahmed, "OCV-RRC Models in State Space, Kalman Filtering and State of Charge Estimation," MECH ENG 754 Management and Control of Electric Vehicle Batteries Lecture 7, McMaster University, Hamilton, Ontario, Fall 2020.

[19] Daoming Sun, Xiaoli Yu, Chongming Wang, Cheng Zhang, Rui Huang, Quan Zhou, Taz Amietszajew, Rohit Bhagat "State of charge estimation for lithium-ion battery based on an Intelligent Adaptive Extended Kalman Filter with improved noise estimator," Energy, vol. 214, 2021.

[20] Kollmeyer, Phillip (2018), "Panasonic 18650PF Li-ion Battery Data", Mendeley Data, V1, doi: 10.17632/wykht8y7tg.1

[21] Naguib, Mina; Kollmeyer, Phillip; Skells, Michael (2020), "LG 18650HG2 Li-ion Battery Data", Mendeley Data, V1, doi: 10.17632/b5mj79w5w9.1

[22] Fauzia Khanum, Eduardo Louback, Federico Duperly, Colleen Jenkins, Phillip Kollmeyer, Ali Emadi (2021). State of Charge Estimation Function based on Kalman Filter. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/90381-state-of-charge-estimation-function-based-on-kalman-filter.

[23] J. Gazzarri, "Modeling Batteries Using Simulink and Simscape," Mathworks, [Online]. Available: https://www.mathworks.com/videos/modeling-batteries-using-simulink-and-simscape-1562930245321.html. [Accessed 19 December 2020].

[24] The Idaho National Laboratory, "Battery Test Manual for Plug-In Hybrid Electric Vehicles," U.S. Department of Energy, Idaho Falls, Idaho, 2010.

[25] Florian Knorn (2021). M-code LaTeX Package. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/8015-m-code-latex-package.