Lecture 3: Tutorial on Battery State of Charge Estimation

The **State of Charge (SoC)** of a battery represents the percentage of the battery's remaining charge relative to its full capacity, essentially acting as a 'fuel gauge' for batteries. It can be expressed as:

$$\text{SoC} = \frac{\text{Remaining Capacity}}{\text{Full Capacity}} \times 100\%$$

State of Charge (SoC) estimation is important in optimizing battery performance, lifetime, and cost-efficiency. Accurate SoC measurements enable aggressive use of the battery capacity without over-engineering, resulting in smaller, lighter, and more cost-effective battery systems. Proper SoC estimation also prevents overcharging or over-discharging, which can cause permanent damage and reduce the battery's lifetime. Also, reliable SoC estimation minimizes warranty service costs and enhances the overall system's reliability.

SoC cannot be directly measured, so it must be estimated using different methods (e.g., voltage-based, current-based, model-based, or data-driven approaches). Accurate estimation is challenging due to factors like:

- Nonlinear battery behavior.
- Temperature variations.
- Battery aging effects.

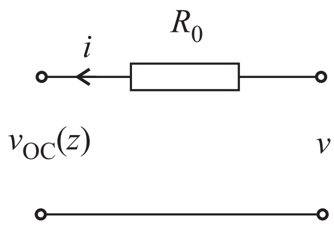

Figure 1: Static battery model. v_{OC} is the open circuit voltage, which depends on the state of charge z, i is the current and v is the terminal voltage.

Voltage-based method:

Based on the Rint model (as shown in Figure 1), we can calculate the OCV as:

$$v_{\text{OC}}(z) = v(t) + i(t)R_0$$

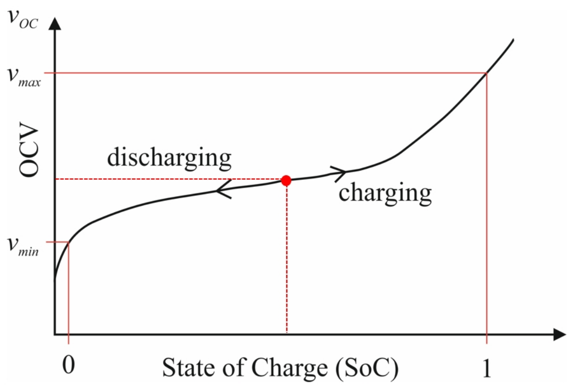

Figure 2: The OCV-SOC relationship.

After get the OCV, then we can look up SOC on "SOC-OCV" curve to determine SOC (as shown in Figure 2).

Coulomb Counting:

The Coulomb Counting method is a widely used technique for estimating the SoC of a battery. It involves tracking the charge entering or leaving the battery over time by integrating the battery's current:

$$SoC(t) = SoC(t_0) + \frac{\eta}{Q} \int_{t_0}^{t} i(\tau)\, d\tau$$

where $SoC(t_0)$ is initial SoC at time t_0. Q is nominal capacity of the battery. $i(\tau)$ is current of the battery at time τ. We can have a quite reliable voltage measurement of the OCV when the batteries have been in rest for sufficiently long. Set the time to t_0 when we measure the OCV, and we can get the initial SoC by:

$$SoC(t_0) = f_{\mathrm{OC}}^{-1}(v_{\mathrm{OC}}(t_0))$$

Question 1

Run the file `OCV-Method.m` *to access dynamic test profiles. The resulting voltage and current profiles are displayed in Figure 3. Also, the defined function* `OCV-SOC` *establishes the SoC to OCV relationship, represented as* `OCV-SOC-25C`*. This function takes low-current measurement data as input. The output consists of two columns: the first column represents the SoC, and the second column represents the corresponding OCV.*

(a) Plot the SoC estimation result. Compare the estimated SoC with the true SoC. Identify the major differences between the estimated and true SoC values and try to explain for these discrepancies.

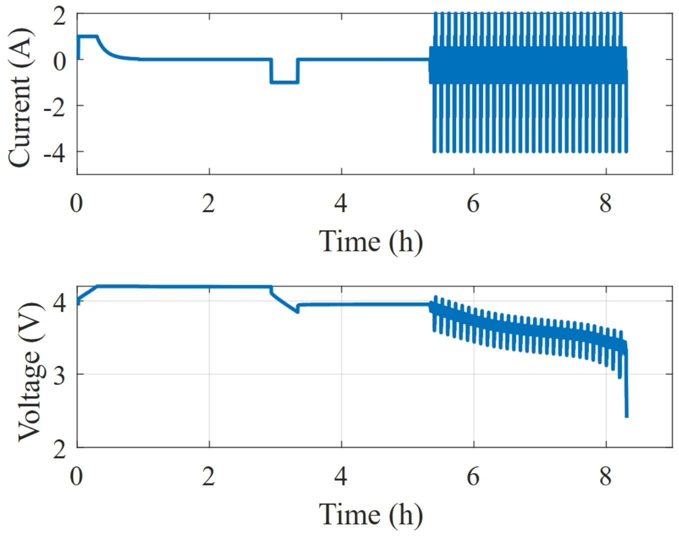

Figure 3: The dynamic test profiles.

(b) *The model parameter R_0 here is identified using pulse test method. What alternative methods can be used to identify R_0? Additionally, if the R_0 value is increased or decreased, how will it affect the estimated SoC, and why?*

Question 2

To explore the Coulomb Counting Method for battery SoC estimation, run the file ***`CoulombCountingMethod.m`***. *Use the same data provided in Exercise 1 to ensure consistent measurements.*

(a) *Plot the SoC estimation results and compare the estimated SoC with the true SoC. Analyze the error characteristics by plotting the error as a function of time. Additionally, manually adjust the value of the estimated initial SoC (increase or decrease it) and observe how this adjustment affects the estimated SoC over time.*

(b) *In real-world applications, current sensors are often less accurate than those used in laboratory settings, and the measured current signals may include noise or bias. In this code, you can manually simulate these imperfections by adjusting the Standard Deviation (`noise-std`) and Mean (`noise-mean`) of the noise. Gradually increase or decrease the values of `noise-std` and `noise-mean` separately to observe how each parameter affects the estimated SoC.*

(c) *Based on the results from (a) and (b), summarize the key factors that influence the accuracy of SoC estimation when using the Coulomb Counting method. For example the effects of measurement noise, sensor bias, initial SoC estimation, and etc.*

Model Based Method:

Model-based methods for SoC estimation utilize sensor measurements to infer the internal (hidden) state of a battery, as shown in Figure 4. These approaches typically involve applying the same input (e.g., current) to both the actual battery system and a mathematical battery model. The method compensates for discrepancies between the model and the real battery by injecting modeling errors into the system and closing the estimation loop. By leveraging techniques like the **Kalman Filter**, the model iteratively updates the state estimate ($\hat{x}$) using system dynamics ($A\hat{x} + Bu$) and a correction term based on the difference between measured output (y) and predicted output ($\hat{y}$). This feedback loop helps account for process noise and measurement noise, enabling robust and accurate SoC estimation even under uncertain conditions.

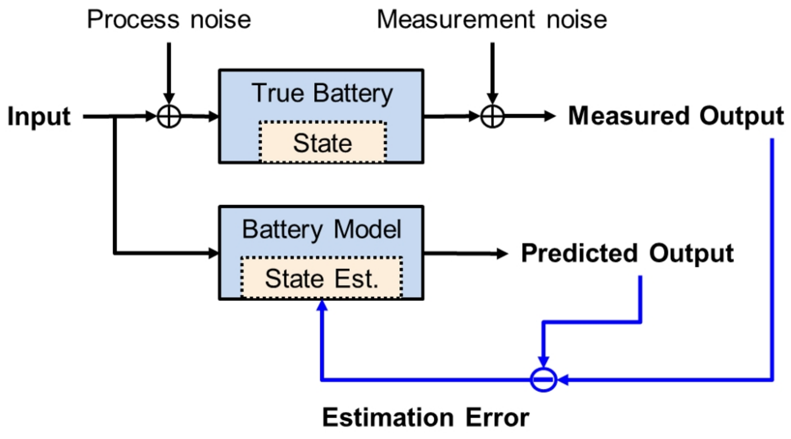

Figure 4: Model based method for battery state estimation.

The **Kalman Filter** is a mathematical algorithm that provides an efficient way to estimate the state of a dynamic system from noisy measurements. It operates using a state-space representation and a two-step process: prediction and update.

State-Space Representation:

$$x(k+1) = Ax(k) + Bu(k) + v(k) \tag{1}$$

$$y(k) = g(x(k), u(k)) + w(k) \tag{2}$$

If the output equation is linear, then

$$y(k) = Cx(k) + Du(k) + w(k) \tag{3}$$

- $x(k)$: State vector at time k
- $u(k)$: Control input vector at time k
- $v(k)$: Process noise (Gaussian, mean 0, covariance Q)
- $y(k)$: Measurement vector at time k

- $w(k)$: Measurement noise (Gaussian, mean 0, covariance R)

Prediction Step:

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_k \tag{4}$$
$$P_{k|k-1} = AP_{k-1|k-1}A^T + Q \tag{5}$$

Update Step:

$$K_k = P_{k|k-1}C^T(CP_{k|k-1}C^T + R)^{-1} \tag{6}$$
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(y_k - C\hat{x}_{k|k-1} - Du_k) \tag{7}$$
$$P_{k|k} = (I - K_kC)P_{k|k-1} \tag{8}$$

The Kalman gain K_k adjusts the state estimate $\hat{x}_{k|k}$ based on the measurement error, ensuring optimal estimation accuracy. This recursive process refines the state estimates by accounting for both system dynamics and noisy observations.

Question 3

To study the Kalman filter method for battery SoC estimation, please run the file `Main_EKF.m` *to get a initial SoC estimation result. In this code, the First Order Model (1 RC model) is used. The function* `SOC_model_EKF.m` *is the defined function to implement the EKF algorithm.*

To study the Kalman filter method for battery SoC estimation, run the file `Main_EKF.m` *to obtain an initial SoC estimation result. This code uses the First-Order Model (1 RC model, see Figure 5)) to simulate the battery dynamics. The Extended Kalman Filter (EKF) algorithm is implemented in the defined function* `SOC_model_EKF.m`*.*

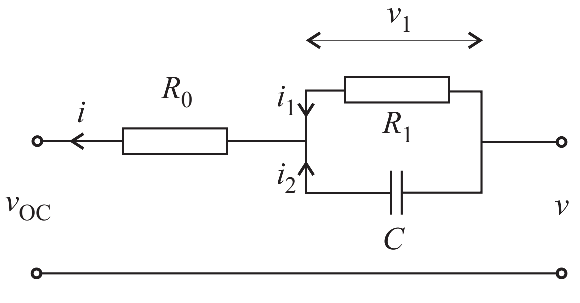

Figure 5: The first order equivalent circuit model structure.

(a) *To study the effect of the initial SoC error on the performance of SoC estimation using the EKF, manually change the initial SoC guess (the value* `SOC_initial`*) in the* `SOC_model_EKF.m` *file. Observe how the change in the initial guess affects the estimated SoC.*

The true initial SoC is 0.8, and the SoC range is from 0 to 1. Do you think the EKF requires an accurate initial SoC to perform effectively? Provide an explanation for your conclusion.

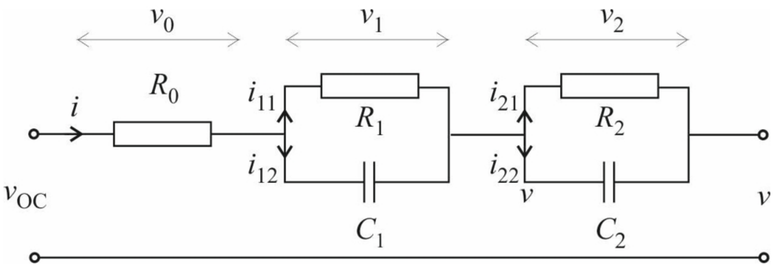

Figure 6: The second order equivalent circuit model structure.

(b) *As mentioned in Exercise 2, measured current signals may include noise or bias. Similarly, manually adjust the Standard Deviation (`noise_std`) and Mean (`noise_mean`) of the noise in the `SOC_model_EKF.m` file. Gradually increase or decrease the values of `noise_std` and `noise_mean` separately to analyze how these parameters affect the estimated SoC.*

Compare the EKF's performance with the Coulomb Counting Method under noisy conditions. Which method is better at handling noise, and why?

(c) *In the EKF algorithm, the Initial Estimation Error Covariance (P_0), Process Noise Covariance (Q), and Measurement Noise Covariance (R) are three critical matrices that need to be tuned.*

Manually adjust the values of these matrices (`P0`, `Q`, `R`) separately in the `SOC_model_EKF.m` file and observe the impact on the performance of the estimated SoC. Summarize how each of these matrices affects SoC estimation and the trade-offs involved in their tuning.

(d) *Consider using a second-order equivalent circuit model (2RC model, as shown in Figure 6) in the EKF. Implement this model by modifying the provided code for the first-order model. The parameters for the 2RC model are provided as:*

$$\begin{aligned} &R_0 = 0.06962\,\Omega, \\ &R_1 = 0.00949\,\Omega, \quad C_1 = 987.42575\,\mathrm{F}, \\ &R_2 = 0.00111\,\Omega, \quad C_2 = 655.93623\,\mathrm{F}. \end{aligned}$$

Compare the SoC estimation results obtained with the 2RC model to those from the 1RC model. What differences do you observe, and what conclusions can you draw regarding the impact of using a more complex model on SoC estimation accuracy?