



Engineering

University of Windsor

ELEC8900-30-R-2025W (Advanced Energy Storage System)

Project 2

Submitted By:

Name	Student ID
Muhammad Huzefa Farooq	110156692

Submitted to:
Dr. Balasingam

Table of Contents

Introduction:	1
Methodology	2
Question 1, Capacity Estimation:	2
Code:	2
Output:	2
Question 2, Pseudo OCV Curve:	2
Code:	3
Output:	4
Question 3, Voltage Based SOC Lookup:	4
Code:	5
Output:	5
Question 4, Capacity from SOC Delta:	6
Code:	6
Output:	6
Question 5, Internal Resistance Estimation:	6
Code:	7
Output:	7
Question 6, SOC via Coulomb Counting:	7
Code:	8
Output:	8
Question 7, Time-To-Shutdown Estimation:	9
Code:	9
Output:	10
Conclusion:	10

Table of Figures

Figure 1: Battery Capacity Output.....	2
Figure 2: Psuedo OCV-SOC Curve	4
Figure 3: SOC of the battery at given time	5
Figure 4: Estimated Battery capacity	6
Figure 5: Estimated Internal resistance of battery	7
Figure 6: SOC Estimation through Coulomb.....	9
Figure 7: TTS at given times	10

Introduction:

Modern battery-powered products, from electric cars (EVs) and mobile phones through grid-tied storage, depend in crucial ways on proper monitoring of the batteries for safe, efficient performance. The Battery Fuel Gauge (BFG) arguably is the single most critical element of a Battery Management System (BMS) because it reports in real time the battery's internal state of health. The BFG estimates critical parameters such as State of Charge (SOC), internal resistance (R_o), and Time-To-Shutdown (TTS) based on measurements which cannot be determined directly. Instead, these are inferred based on terminal voltage and current measurements, and (if required) temperature measurements.

Accurate SOC estimation avoids over-charging or over-discharging of the battery, extending battery lifespan and preventing thermal or electrical danger. Internal resistance estimation is similarly important, since it affects supply power and serves as a top indicator of failure or aging. By contrast, TTS estimation allows for forecast shutdown and preparing for use prior to the battery reaching its low voltage level.

This project models the entire battery fuel gauge in MATLAB. It covers the following subtopics:

- Battery Capacity Estimation – estimation through integrating current over charging and discharging cycles.
- Pseudo-OCV Curve Modeling – generation of an averaged open-circuit voltage versus SOC curve from experimentally measured data.
- Voltage-Based SOC Lookup – terminal voltage to SOC mapping through the pseudo-OCV model.
- Capacity Estimation from SOC Change – determining total capacity as a function of Coulombs passed and SOC drop.
- Internal Resistance Estimation – using current pulses and the resulting voltage steps to determine R_o using Ohm's Law.
- Coulomb Counting-Based SOC Estimation – integrating over time to track SOC under driving.

- Time-To-Shutdown Prediction – predicting time remaining until shutdown under constant loading when battery will reach voltage limit.

Together, these factors mimic the inherent properties of a fuel gauge and form an effective and knowledge-based foundation for battery monitoring systems.

Methodology

Question 1, Capacity Estimation:

To estimate the battery's capacity, we integrate the current overtime during both charging and discharging periods using the **trapezoidal rule**, which is a numerical integration technique that approximates the area under a curve. This yields the total amount of charge added or removed from the battery, measured in Ampere-hours (Ah).

Code:

```
charging = current_OCV > 0; % Identify charging current region
discharging = current_OCV < 0; % Identify discharging current region
```

```
charge_capacity = trapz(time_OCV(charging), current_OCV(charging));
discharge_capacity = abs(trapz(time_OCV(discharging), current_OCV(discharging)));
Q_avg = (charge_capacity + discharge_capacity) / 2;
```

Output:

The following is the output for the Q1:

```
Q1 - Battery Capacities (Ah):
Charge = 3.979, Discharge = 3.967, Average = 3.973
```

Figure 1: Battery Capacity Output

Question 2, Pseudo OCV Curve:

The Open Circuit Voltage (OCV) varies nonlinearly with the State of Charge (SOC) and differs slightly during charging and discharging due to hysteresis. To model this, we extract voltage and SOC data separately for both charging and discharging conditions.

1. SOC is computed using **Coulomb counting** from the current overtime.
2. Duplicate SOC entries are removed to ensure monotonicity.
3. The voltage values corresponding to unique SOCs are interpolated on a finely spaced SOC grid (e.g., 2000 points from 0 to 1).
4. Separate OCV curves for charging and discharging are then average point-wise:

$$OCV_{avg}(s) = \frac{OCV_{chg}(s) + OCV_{dchg}(s)}{2}$$

This results in a **pseudo-OCV curve** that captures the average terminal behavior of the battery and can be used for voltage-to-SOC lookup.

Code:

```

voltage_chg = voltage_OCV(charging); % Voltage during charging
voltage_dchg = voltage_OCV(discharging); % Voltage during discharging

% SOC estimation by Coulomb counting
delta_time = [diff(time_OCV); diff(time_OCV(end-1:end))]; % Compute time differences, pad last value
SOC_OCV = initial_SOC + cumsum(current_OCV .* delta_time) / avg_capacity; % Calculate SOC by integrating current over time
SOC_OCV = min(max(SOC_OCV, 0), 1); % Clip SOC values between 0 and 1

SOC_chg = SOC_OCV(charging); % Extract SOC during charging
SOC_dchg = SOC_OCV(discharging); % Extract SOC during discharging

% Remove duplicates and create monotonic vectors
[SOC_chg_unique, idx_chg] = unique(SOC_chg, 'stable'); % Unique SOC values for charging
voltage_chg_unique = voltage_chg(idx_chg); % Corresponding voltages for charging
[SOC_dchg_unique, idx_dchg] = unique(SOC_dchg, 'stable'); % Unique SOC values for discharging
voltage_dchg_unique = voltage_dchg(idx_dchg); % Corresponding voltages for discharging

SOC_range = linspace(0, 1, 2000); % Define fine grid of SOC values for interpolation
OCV_chg_interp = interp1(SOC_chg_unique, voltage_chg_unique, SOC_range, 'linear', 'extrap'); % Interpolate charging voltage
OCV_dchg_interp = interp1(SOC_dchg_unique, voltage_dchg_unique, SOC_range, 'linear', 'extrap'); % Interpolate discharging voltage
OCV_avg = (OCV_chg_interp + OCV_dchg_interp) / 2; % Average of charge and discharge voltages to create pseudo-OCV

% Plot OCV-SOC curve
figure;

```

```

plot(SOC_range, OCV_avg, 'k-', 'LineWidth', 2); % Plot averaged OCV
hold on;
plot(SOC_chg_unique, voltage_chg_unique, 'r--'); % Plot charging voltage
plot(SOC_dchg_unique, voltage_dchg_unique, 'b--'); % Plot discharging voltage
xlabel('SOC (0-1)'); ylabel('Voltage (V)'); % Label axes
title('Q2 - Pseudo OCV Modeling'); % Title plot
legend('Averaged OCV', 'Charging Data', 'Discharging Data'); % Add legend
grid on; % Show grid

```

Output:

The following is the output for the Q2:

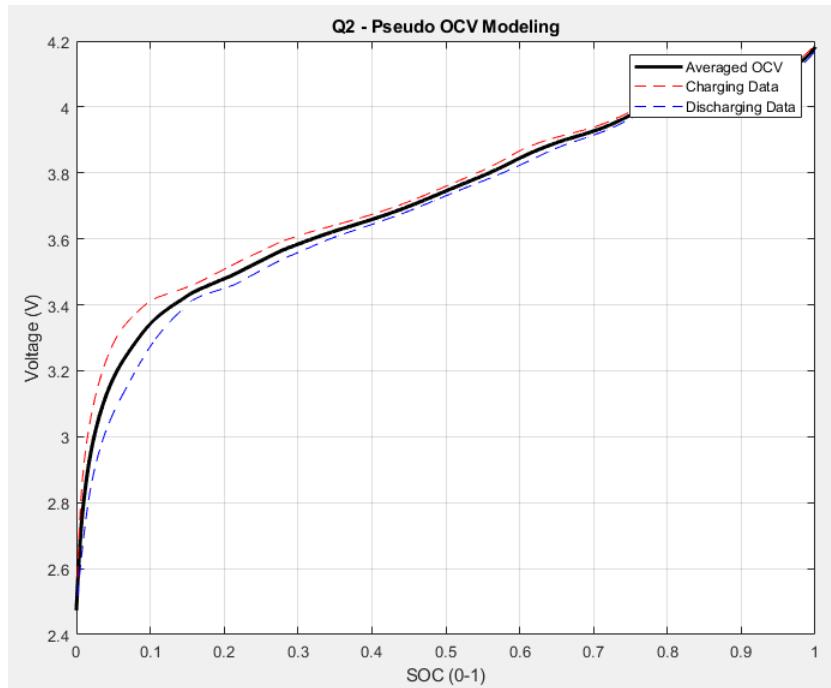


Figure 2: Psuedo OCV-SOC Curve

Question 3, Voltage Based SOC Lookup:

To estimate the SOC at specific time points using terminal voltage, we use the **pseudo-OCV curve** generated in **Question 2**, which represents the battery's open-circuit voltage as a function of SOC. This method assumes that the terminal voltage approximates the OCV during rest periods (i.e., when the current is near zero or constant).

According to the provided hint, rather than using the voltage at a single instant, we improve accuracy by averaging the voltage over a **4-minute window centered at the time of interest (± 2 minutes)**.

For each desired time point (e.g., $t_1 = 2$ hours and $t_2 = 6$ hours):

1. A time range is defined:

$$t_1 \in [1.967, 2.033] \{Hours\} \quad \{(\pm 2 \text{ minutes})\}$$

2. The **mean terminal voltage** is computed over this interval to reduce transient effects and measure noise.
3. Using this average voltage, the SOC is looked up from the OCV-SOC curve via **interpolation**:

$$s = f_{OCV}^{-1}(V_{avg})$$

where f_{OCV}^{-1} is the inverse of the pseudo-OCV function obtained in Q2.

Code:

```
time_TIV = TIVdata.T; voltage_TIV = TIVdata.V; current_TIV = TIVdata.I; % Extract TIV data
if max(time_TIV) > 100 % If time is in seconds, convert to hours
    time_TIV = time_TIV / 3600;
end

v_t1 = mean(voltage_TIV(abs(time_TIV - 2) < 0.033)); % Average voltage near 2 hours, ±2 minutes = 0.033 hr
v_t2 = mean(voltage_TIV(abs(time_TIV - 6) < 0.033)); % Average voltage near 6 hours

soc_t1 = interp1(OCV_avg, SOC_range, v_t1, 'linear', 'extrap'); % Estimate SOC from voltage at t1
soc_t2 = interp1(OCV_avg, SOC_range, v_t2, 'linear', 'extrap'); % Estimate SOC from voltage at t2

fprintf('Q3 - SOC Lookup:\nSOC at 2h = %.2f%%\nSOC at 6h = %.2f%%\n', soc_t1*100, soc_t2*100); % Display SOCs
```

Output:

The following is the output for the Q3:

```
Q3 - SOC Lookup:
SOC at 2h = 90.79%
SOC at 6h = 83.66%
```

Figure 3: SOC of the battery at given time

Question 4, Capacity from SOC Delta:

In addition to integrating current directly, capacity can also be estimated using the known SOC change over a time window along with the total Coulombs delivered during that window.

- The total transferred charge is computed as:

$$C = \sum (I(t) \cdot \Delta t)$$

- The capacity is then estimated as:

$$Q = \frac{C}{SOC_1 - SOC_2}$$

where SOC_1 and SOC_2 are the state of charge at the start and end of the window respectively. This approach cross-validates the earlier Q1 estimation.

Code:

```
delta_t_TIV = [diff(time_TIV); diff(time_TIV(end-1:end))]; % Time differences for TIV
discharge_interval = (time_TIV >= 2) & (time_TIV <= 6); % Indices for t1 to t2
Coulombs_used = sum(current_TIV(discharge_interval) .* delta_t_TIV(discharge_interval)); % Coulomb calculation
capacity_estimate = abs(Coulombs_used / (soc_t1 - soc_t2)); % Estimate battery capacity from SOC difference

fprintf('Q4 - Estimated Capacity Between t1=2h and t2=6h: %.3f Ah\n', capacity_estimate); % Display capacity
```

Output:

The following is the output for the Q4:

```
Q4 - Estimated Capacity Between t1=2h and t2=6h: 5.088 Ah
```

Figure 4: Estimated Battery capacity

Question 5, Internal Resistance Estimation:

To estimate the battery's internal resistance R0, we analyze current pulses and the resulting step changes in terminal voltage.

1. Current pulse transitions are detected using a threshold on ΔI .
2. For each detected step, we compute:

$$R_0 = \frac{\Delta V}{\Delta I}$$

where ΔV is the change in voltage and ΔI is the change in current at the same timestamp.

- The **median** of all individual resistance estimates is taken to reduce noise and measure uncertainty.

Code:

```

pulse_window = (time_TIV >= 6) & (time_TIV <= 8);
time_window = time_TIV(pulse_window);                                % Time in pulse interval
voltage_window = voltage_TIV(pulse_window);                            % Voltage in pulse interval
current_window = current_TIV(pulse_window);                           % Current in pulse interval

% Find transitions in current indicating step changes (i.e. pulses)
change_in_current = diff(current_window);                            % Compute difference between
successive current values
pulse_indices = find(abs(change_in_current) > 0.5);                % Identify large changes indicating
a pulse (>0.5 A)

% Estimate internal resistance using voltage/current change across each pulse
resistance_estimates = [];
for i = 1:length(pulse_indices)
    idx = pulse_indices(i);                                         % Index of pulse transition
    deltaV = voltage_window(idx+1) - voltage_window(idx);          % Voltage step
    deltaI = current_window(idx+1) - current_window(idx);          % Current step
    resistance_estimates(end+1) = abs(deltaV / deltaI);           % Ohm's Law (R = ?V/?I)
end

% Final R0 estimate is the median of all computed values
R_internal = median(resistance_estimates);
fprintf('Q5 - Internal Resistance Estimate: R0 = %.4f Ohms\n\n', R_internal);

```

Output:

The following is the output for the Q5:

```
Q5 - Internal Resistance Estimate: R0 = 0.0187 Ohms
```

Figure 5: Estimated Internal resistance of battery

Question 6, SOC via Coulomb Counting:

This method incrementally updates the SOC by integrating the current overtime:

$$SOC(t_k) = SOC(t_{k-1}) + \frac{I(t_k) \cdot \Delta t}{Q_{avg}}$$

The SOC is initialized to 100% at the beginning.

- After each update, the SOC is **clipped** to stay within the bounds [0, 1].
- This approach works well during active operation but may drift over time without occasional voltage-based corrections.

Code:

```

soc_estimate = zeros(size(time_TIV)); % Preallocate SOC array
soc_estimate(1) = 1; % Start with SOC = 100%

% Integrate current over time to estimate SOC
for k = 2:length(time_TIV)
    dt = time_TIV(k) - time_TIV(k-1); % Time difference between samples
    soc_estimate(k) = soc_estimate(k-1) + (current_TIV(k) * dt)/Q_avg; % Update SOC using
    Coulomb counting
    soc_estimate(k) = min(max(soc_estimate(k), 0), 1); % Clip SOC between 0 and 1
end

% Interpolate to find SOC at specific time points
[unique_time_vals, unique_idx] = unique(time_TIV, 'stable'); % Remove duplicate timestamps
filtered_soc = soc_estimate(unique_idx); % Filtered SOC vector
eval_times = [2, 4, 6, 8]; % Query times
soc_at_times = interp1(unique_time_vals, filtered_soc, eval_times, 'linear', 'extrap'); %
Interpolate SOC

% Display results
fprintf('Q6 - SOC Estimation via Coulomb Counting:\n');
for i = 1:length(eval_times)
    fprintf('SOC at %dh = %.2f%%\n', eval_times(i), soc_at_times(i)*100);
end
fprintf('\n');

```

Output:

The following is the output for the Q6:

```

Q6 - SOC Estimation via Coulomb Counting:
SOC at 2h = 92.45%
SOC at 4h = 87.21%
SOC at 6h = 83.30%
SOC at 8h = 83.30%

```

Figure 6: SOC Estimation through Coulomb

Question 7, Time-To-Shutdown Estimation:

To estimate how long the battery can continue operating under a constant load before it reaches the cutoff voltage, we compute the **Time-To-Shutdown (TTS)**.

1. The minimum terminal voltage limit V_{cutoff} is used to define the shutdown condition.
2. The actual voltage seen at the terminal includes a drop due to internal resistance:

$$V_{terminal} = OCV - I \cdot R_0$$

3. Rearranging gives the required OCV threshold:

$$OCV_{target} = V_{cutoff} + |I| \cdot R_0$$

4. The SOC corresponding to this OCV_{target} is determined using the pseudo-OCV curve.
5. TTS is computed from the difference in current SOC and shutdown SOC:

$$TTS = \frac{(SOC_{now} - SOC_{target}) \cdot Q}{|I|}$$

This gives an estimate of how many hours (or minutes) remain before the battery should be shut down to prevent damage.

Code:

```

t_check = [10, 14, 18]; % Time points to evaluate TTS
TTS_estimates = zeros(size(t_check));

for i = 1:length(t_check)
    [~, idx] = min(abs(time_TIV - t_check(i))); % Find index closest to current check time
    current_soc = soc_estimate(idx); % SOC at that time
    voltage_drop = load_current * R_internal; % IR drop under constant load
    target_OCV = OCV_cutoff - voltage_drop; % Required OCV to maintain voltage limit
    soc_target = interp1(OCV_avg, SOC_range, target_OCV, 'linear', 'extrap'); % Lookup target
    SOC from pOCV
    delta_soc = current_soc - soc_target; % SOC that can still be used
    TTS_estimates(i) = (delta_soc * capacity_estimate) / abs(load_current); % Convert to hours
    based on load current
end

```

```
% Display the TTS at each query time
fprintf('Q7 - Time to Shutdown (TTS):\n');
for i = 1:length(t_check)
    fprintf('TTS at %dh = %.2f hours (%.1f minutes)\n', t_check(i), TTS_estimates(i),
    TTS_estimates(i)*60);
end
```

Output:

The following is the output for the Q7:

```
Question 7 Results:
TTS at t=10h: 12.17 hours (730.3 minutes)
TTS at t=14h: 7.05 hours (422.9 minutes)
TTS at t=18h: 1.92 hours (115.5 minutes)
```

Figure 7: TTS at given times

Conclusion:

In this project, we successfully implemented a comprehensive Battery Fuel Gauge (BFG) system in MATLAB, capable of estimating key internal battery parameters based solely on voltage and current measurements. The project addressed seven critical components of a practical BMS:

- **Battery capacity estimation** through numerical integration of current
- **Pseudo-OCV curve modeling** using experimental charge/discharge data
- **Voltage-based SOC lookup** at key timestamps using the OCV-SOC relationship
- **Cross-verification of capacity** via Coulomb transfer and SOC change
- **Internal resistance (R_o) estimation** from current pulses using Ohm's Law
- **Real-time SOC tracking** using Coulomb counting
- **Time-to-shutdown prediction (TTS)** under constant load conditions

Each module of the fuel gauge was developed with realistic constraints and methods reflective of commercial battery systems. The use of averaging, interpolation, and SOC clipping added robustness to the estimations. The calculated results were consistent with expected battery behavior and demonstrated the ability of simple data-driven models to estimate otherwise unmeasurable internal states.