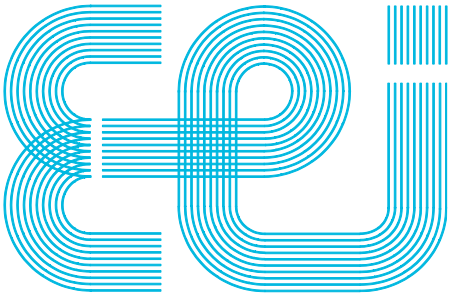


Escuela de Ingeniería Industrial

BACHELOR'S THESIS

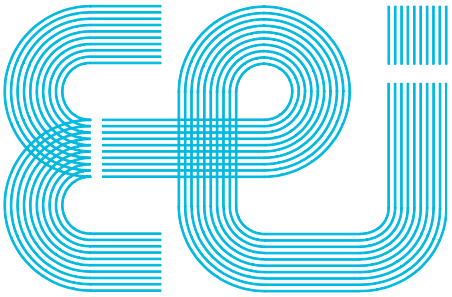
Core and surface temperature estimation of large-scale battery pack with reduced sensors

Bachelor degree in Industrial Electronics and Automation Engineering

STUDENT: Jens Kjaersgaard Larrañaga

SUPERVISORS: Juan José Rodríguez Andina

Universidad de Vigo



Escuela de Ingeniería Industrial

BACHELOR'S THESIS

Core and surface temperature estimation of large-scale battery pack with reduced sensors

Bachelor degree in Industrial Electronics and Automation Engineering

Document

REPORT

Universidad de Vigo

CONTENT

TABLE INDEX

FIGURE INDEX

1 INTRODUCTION

1.1 Motivation

The automotive industry is undergoing a transformative shift towards sustainable transportation, with electric vehicles (EVs) emerging as a key solution to reduce greenhouse gas emissions, diminish dependence on fossil fuels, and comply with increasingly rigorous environmental regulations. In this context, lithium-ion battery technology has established itself as the dominant energy storage system for EVs, offering high energy density, efficiency, and reliability.

However, the performance, safety, and longevity of lithium-ion batteries are highly sensitive to their thermal behavior. Inappropriate temperature ranges can lead to significant degradation of battery capacity, reduced operational lifespan, and in extreme cases, hazardous events such as thermal runaway. Managing the temperature of large-scale battery packs in EVs is therefore critical to ensuring optimal performance and user safety.

Traditionally, comprehensive thermal monitoring requires a large number of sensors distributed throughout the battery pack, to capture both surface and core temperatures. While this approach can yield detailed temperature profiles, it poses challenges in terms of cost, complexity, and integration, especially in commercial applications, where economic viability is crucial. Consequently, there is a strong industrial demand for developing methods that can accurately estimate internal battery temperatures using a reduced number of sensors, minimizing hardware complexity while maintaining reliable thermal monitoring.

This work addresses this challenge by investigating and developing advanced modeling and estimation techniques that enable accurate core and surface temperature estimation in large-scale battery packs with a minimized sensor setup. Such developments not only enhance safety and performance but also contribute to the scalability and commercial feasibility of EV technologies.

1.2 Objectives

The primary objective of this work is to develop and validate a methodology capable of estimating the core and surface temperatures of lithium-ion battery packs using a limited number of temperature sensors. This is achieved through the following specific objectives:

- ➢ Develop accurate electrical and thermal models of lithium-ion battery behavior under various operating conditions.
- ➢ Implement numerical simulation tools to replicate the thermal dynamics of large-scale battery packs.
- ➢ Design and apply optimization algorithms to identify internal model parameters based on experimental data.

- ➢ Develop an estimation algorithm based on the Extended Kalman Filter (EKF) to accurately estimate core and surface temperatures.

- ➢ Validate the proposed models and estimation approach using experimental data obtained from laboratory tests.

1.3 Organization of the Document

The remaining of the document is organized into four chapters, each addressing a key stage of the research and development process:

Chapter 2 - State-of-the-Art: Reviews the existing literature related to lithium-ion battery technologies, thermal modeling approaches, temperature estimation techniques, and previous works addressing sensor reduction strategies. This chapter establishes the scientific foundation and justifies the relevance of the present work.

Chapter 3 - Methodology and Work Development: Describes in detail the modeling and estimation methods developed in this work. The chapter covers the formulation of electrical and thermal models, simulation procedures in MATLAB, optimization processes using genetic algorithms in Python, the design and implementation of the EKF, and the experimental setup used to acquire validation data.

Chapter 4 - Model Validation and Results: Presents the comparison between simulation outcomes and experimental measurements, analyzing the accuracy and performance of the proposed estimation approach.

Chapter 5 - Conclusions and Future Work: Summarizes the key findings of the work and discusses potential directions for further research and development aimed at improving the presented methodologies.

2 STATE-OF-THE-ART

2.1 Introduction

The accurate estimation of internal states in lithium-ion batteries requires a solid understanding of their electrical, thermal, and dynamic behaviors. Lithium-ion cells exhibit complex coupled phenomena, where electrical loads directly influence thermal responses, and where temperature in turn affects battery performance, safety, and degradation. As such, numerous modeling strategies have been developed to describe battery operation under real-world conditions, ranging from simplified equivalent circuit models to highly detailed electrochemical representations.

In addition to physical modeling, estimation algorithms play a crucial role in reconstructing unmeasured states such as core temperature, which cannot be directly monitored in most applications. Methods such as Kalman filtering, adaptive observers, and Machine Learning-based techniques have been widely adopted to infer these internal variables using accessible measurements like current, voltage, and surface temperature.

This chapter presents the theoretical and state-of-the-art foundations required to understand the electro-thermal modeling and state estimation approach developed in this work. It covers the essential battery modeling alternatives, the thermal processes involved, and the estimation algorithms that enable real-time monitoring of internal temperature states in lithium-ion battery systems.

2.2 Lithium-Ion Battery Modeling

2.2.1 Fundamentals

A typical lithium-ion battery consists of a graphite anode, a lithium metal oxide cathode, and a liquid electrolyte that facilitates ion movement between electrodes during charge and discharge cycles. During discharging, lithium ions migrate from the anode to the cathode through the electrolyte, whereas electrons flow through the external circuit, providing electrical power. This process reverses during charging (Fig. 1).

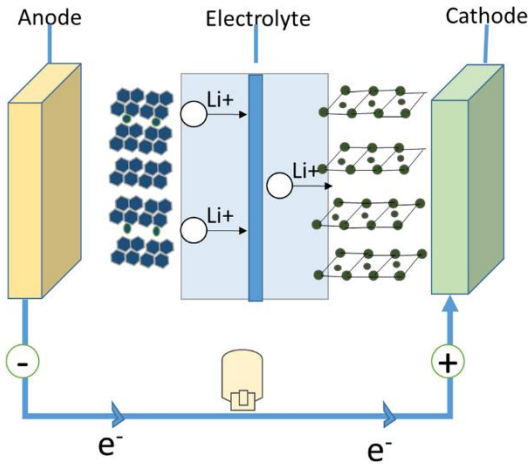


Figure 1 Diagram of lithium-ion battery structure from [1]

The electrochemical reactions that govern lithium-ion batteries are temperature-dependent. Elevated temperatures may enhance kinetics but also accelerate side reactions, whereas low temperatures hinder ion transport and increase internal resistance, severely affecting performance.

2.2.2 Reduced-Order Electrical Models

The electrical behavior of lithium-ion batteries is often modeled using Equivalent Circuit Models (ECMs). The most common ECM configurations include [2]:

- **Single-RC (Thevenin) networks** represent the cell by an open-circuit voltage source, an ohmic resistance and one RC branch. They capture the dominant voltage transient with only two or three parameters and are widely used for state-of-charge (SOC) algorithms. SOC is a dimensionless quantity that indicates the remaining capacity of a battery relative to its nominal (fully charged) capacity.

- **Dual-RC models** add a second RC branch to separate fast ohmic relaxation from slower diffusion processes. This modest increase in order halves voltage-prediction error under dynamic automotive load profiles. Dual-RC models remain observable when coupled to a two-node thermal network, an insight exploited in this work.

2.2.3 Electro-Thermal Coupling

Once the electrical model (dual-RC equivalent circuit) computes the terminal voltage and current, it immediately evaluates the net heat-generation rate, arising from both Joule (irreversible) and entropy (reversible) effects. This heat-generation rate is then **directly injected** as the heat source into the lumped two-node thermal network, where it drives the evolution of core and surface temperatures.

In turn, the updated temperatures from the thermal model are fed back into the electrical model to update temperature-dependent parameters (e.g., internal resistances, capacitances, open-circuit voltage – OCV), closing the loop and ensuring that subsequent electrical predictions account for self-heating effects.

2.3 Thermal Behavior of Lithium-Ion Cells

2.3.1 Heat-Generation Mechanisms

Heat generation in lithium-ion cells arises from two main phenomena. Ohmic (Joule) heating is the dominant irreversible source, originating in both electronic and ionic resistances as current flows through the cell's internal components. This resistive loss converts electrical energy directly into heat. In addition, entropy changes introduce a reversible heating (or cooling) effect: as the OCV varies with SOC and temperature, the associated thermodynamic entropy change contributes to the overall heat balance [3].

2.3.2 Lumped Thermal Modeling

Thermal modeling is essential for predicting the internal and external temperature dynamics of lithium-ion batteries, especially to estimate the core temperature, which cannot be directly measured in most cases. Among the various modeling approaches, lumped thermal models offer a practical balance between physical representativeness and computational simplicity, making them ideal for real-time applications such as Battery Management Systems (BMS).

In lumped thermal models, the battery pack is simplified into discrete thermal nodes representing different parts of the cell – most commonly the core, the surface, and the surrounding ambient environment. Each node is assigned a thermal capacitance, to capture its heat storage capability, whereas thermal resistances model the heat exchange between nodes and with the ambient air.

This simplified structure enables accurate prediction of both the core and surface temperature evolution with a small number of parameters, which can be experimentally identified. Furthermore, these parameters can be identified and updated online using Recursive Least Squares (RLS), allowing the model to adapt to battery ageing and varying thermal conditions without interrupting BMS operation.

The low computational cost of lumped thermal models enables their seamless integration with real-time estimation algorithms such as EKF, which is extensively used for state estimation in this work.

2.4 Temperature Estimation Problem

Accurate thermal monitoring of lithium-ion batteries is critical for ensuring safety, performance, and longevity, particularly under high-load or fast-charging conditions, where thermal gradients within the cell can become significant. However, direct measurement of the internal (core) temperature remains a major challenge. Embedding temperature sensors inside the cell's jelly-roll structure is mechanically invasive, potentially compromising cell integrity, increasing manufacturing complexity, and elevating costs.

The surface-only measurement approach fails to capture the thermal behavior at the core, where the highest temperatures typically arise due to internal heat generation and lower heat dissipation. Relying solely on surface readings can therefore mask early signs of thermal runaway. Studies on EV battery packs have shown that an error as small as 2°C in estimating core temperature can significantly reduce the safety margin by as much as 50% during aggressive charging scenarios [4]. Consequently, developing reliable core temperature estimation algorithms capable of achieving an accuracy within ±1°C using minimal external sensors becomes essential. Such estimations enable early thermal fault detection, improved thermal

management, and extended battery life without compromising structural integrity or economic feasibility.

2.5 Sensor Selection

Selecting an appropriate sensor suite balances cost, intrusiveness, and estimation performance. Surface-mounted thermocouples are typically preferred in large-scale packs due to their low cost, fast response, and ease of installation, despite only providing surface-temperature data. Resistance Temperature Detectors (RTDs) offer higher precision but respond more slowly and incur greater expense. Fiber-optic sensors, while capable of direct internal measurements, are rarely used in commercial packs owing to their high cost and complex integration. In practice, a minimal configuration pairing surface thermocouples with existing voltage and current measurements provides sufficient observability for EKF-based core-temperature estimation, achieving an optimal trade-off between sensor count and estimation accuracy [5].

2.6 State and Parameter Estimation Techniques

2.6.1 Kalman Filter Fundamentals

The Kalman Filter (KF) is a recursive estimator that fuses mathematical model's predictions with noisy sensor measurements to optimally reconstruct unmeasured system states [6]. In its standard form, the KF applies to linear systems by sequentially executing a prediction step, which projects the current state estimate forward in time via the system's state-transition model, and an update step, which corrects this prediction by incorporating new measurements weighted according to their noise characteristics. For nonlinear electro-thermal battery models, EKF is used: at each time step, the EKF linearizes the nonlinear state and output equations around the latest estimate, then proceeds with the usual predict–update routine. This approach enables real-time estimation of internal variables, such as core temperature, that cannot be instrumented directly [7].

2.6.2 RLS

RLS is used alongside the EKF to perform online parameter identification, ensuring the thermal model remains accurate as cells age or operating conditions change. RLS continually adjusts model parameters (e.g., thermal resistances or capacitances) by minimizing the cumulative squared error between the model's predicted outputs and the actual measurements. With each new data point, the algorithm updates the parameter estimate in closed-form, allowing seamless integration into the BMS without interrupting operation. This adaptability is crucial for compensating drifts in cell behavior over time [8].

2.6.3 Adaptive Kalman Filtering

Adaptive Kalman Filtering enhances traditional KF by adjusting its noise covariance matrices based on observed data, improving estimation accuracy in the presence of time-varying uncertainties. Adaptive schemes are particularly valuable when dealing with aging batteries or fluctuating ambient conditions.

2.6.4 Machine Learning and Hybrid Methods

Recent research integrates data-driven methods with physics-based models to enhance estimation robustness, leveraging historical and real-time data to correct model inaccuracies, especially under complex drive cycles or extreme environmental conditions:

- **Long Short-Term Memory (LSTM)-Based Model Fusion.** A small LSTM network learns to correct the residual between a lumped thermal model's core-temperature estimate and actual measurements. Guo *et al* model [9] achieves high accuracy, also adapting to ageing and nonlinear effects. Nevertheless, it requires large labeled datasets, resulting in a need of higher memory/CPU load than pure model-based filters [10].

- **Physics-Informed Neural Networks (PINN).** A neural network is trained with a loss function combining measured-data error and the heat-balance equation residual [11] This model requires fewer measurements but its training is more complex.

3 METHODOLOGY AND WORK DEVELOPMENT

3.1 Electrical Model

In the proposed modeling approach, each individual cell in the battery pack is represented by a first-order RC electrical network. Fig. 2 illustrates the equivalent electrical model for the first battery in the pack. This modular approach enables modeling the electro-thermal behavior of large-scale battery systems by replicating the single-cell model across all cells. The model includes the following variables and parameters:

- $\boldsymbol{V}$ (in V): Terminal voltage of the cell, which serves as the interface with the thermal model.
- $\boldsymbol{OCV}$ (in V): A nonlinear function of SOC and temperature.
- $\boldsymbol{I}$ (in A): Applied current, defined as positive during charging and negative during discharging.
- $\boldsymbol{R_1}$ (in Ω): Ohmic internal resistance, dependent on temperature, SOC, and current direction.
- $\boldsymbol{R_{11}}$ (in Ω): Polarization resistance, which models the slower voltage response due to electrochemical processes. It also varies with SOC, temperature, and current direction.
- $\boldsymbol{C_1}$ (in F): Polarization capacitance, capturing the dynamic charge storage behavior.
- $\boldsymbol{V_1}$ (in V): Polarization voltage, resulting from the interaction between R_1 and C_1, representing transient voltage effects.

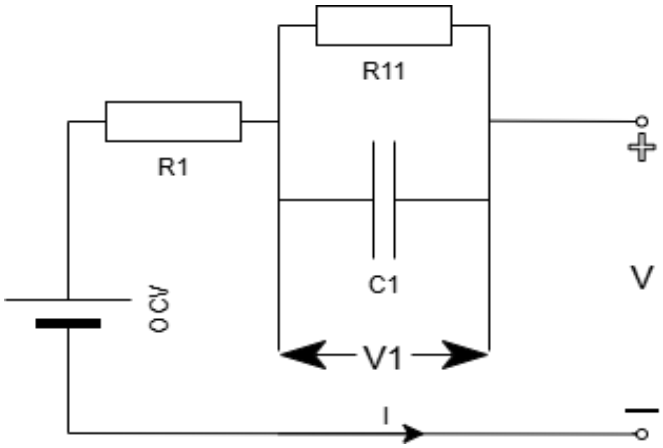


Figure 2 Electrical model of the first battery of the pack

According to Kirchhoff's voltage law, the voltage drop across the RC polarization branch can be described by the following differential equation:

$$\cdot V_1(t) = -\frac{1}{R_{11}C_1}V_1(t) + \frac{1}{C_1}I(t) \qquad (1)$$

where $\cdot V_1$ is the rate of change in V_1.

The terminal voltage of the battery cell is:

$$V = OCV + R_1 I + V_1 \quad (2)$$

The SOC evolves according to the integral of current over time, corrected by charge/discharge efficiency η (which is usually set to 1 when discharging and 0.98 when charging), and normalized by the battery's nominal capacity Q. This relationship is obtained in [12] as:

$$SOC(t) = SOC(t_0) + \int_{t_0}^{t} \frac{\eta I(\tau)}{Q} d\tau \quad (3)$$

To enable digital simulation and estimator implementation, these continuous-time equations are discretized using a fixed sampling time Δt (in s). The discretized forms for equations (1)-(3) are:

$$V_1(k) = exp(\frac{-\Delta t}{R_1 C_1}) V_1(k-1) + R_1 I(k) \left[1 - exp(\frac{-\Delta t}{R_1 C_1})\right] \quad (4)$$

$$SOC(k) = SOC(k-1) + \frac{\eta \Delta t}{Q} I(k) \quad (5)$$

The parameters to identify in this model are:

$$\theta e = [R_{11}\ C_1\ R_1\ OCV\ Q] \quad (6)$$

3.2 Thermal Model

The heat generation rate of lithium-ion batteries adopts the Bernardi heat generation model, as shown in equation (7):

$$Q = I(V - OCV) + IT\frac{dOCV}{dT} + Q_{mi} + Q_{pc} \quad (7)$$

The first term in the equation corresponds to irreversible heat generated due to the voltage drop between the OCV and the terminal voltage. The second term reflects the reversible heat arising from entropy changes during the intercalation and deintercalation processes of lithium ions. This reversible component depends on OCV's temperature derivative, $\frac{dOCV}{dT}$, known as the entropy coefficient, described in equation (8).

$$\frac{dOCV}{dT} = \gamma_1 + \gamma_2 SOC + \gamma_3 SOC^2 + \gamma_4 SOC^3 + \gamma_5 SOC^4 + \gamma_6 SOC^5 \quad (8)$$

where the γ parameters are obtained from [12] in a test scheme made to obtain the fitting curve of the temperature coefficient on SOC. The obtained values are summarized in Table 1:

Parameter	γ_6	γ_5	γ_4	γ_3	γ_2	γ_1
Value	14.56547	-35.3986	30.1126	-11.6516	2.7279	-0.3485

Table 1 Parameter values for entropy coefficient [equation (8)]

For lithium-ion chemistries, the entropy coefficient varies with SOC and its precise behavior is typically obtained through empirical testing. In this work, a data-driven fitting curve for the entropy coefficient as a function of SOC, based on the protocol described in [13], is usd to model this term.

The third term in equation (7) accounts for the heat generated due to the presence of concentration gradients within the battery during operation. These gradients come from non-uniform ion transport and reaction kinetics, leading to mixing effects that produce additional heat Q_{mi}.

The fourth term represents the heat generated by solid-phase transformations, specifically during phase changes that occur at certain SOC plateaus in some lithium-ion chemistries. These transformations involve latent heat exchange, resulting in Q_{pc}.

To reduce computational complexity and enable real-time estimation, several assumptions are made in the thermal modeling [12]:

1. All heat is generated in the core of the battery, from which it flows radially outward to the surface and then to the ambient environment. Heat transfer is considered uniform along the path battery core → battery surface → environment.

2. Radiation effects are negligible, due to their minimal contribution at standard battery operating temperatures.

3. Thermo-physical parameters, such as thermal conductivity, specific heat, and density, are treated as constants, independent of temperature.

4. Convective heat transfer is disregarded, because the internal electrolyte remains largely static, limiting any significant fluid motion that could contribute to heat transport.

5. The temperature across the battery shell is assumed to be uniformly distributed, due to the high thermal conductivity of aluminum, which is used for the shell.

6. The contributions of Q_{mi} and Q_{pc} to the overall heat generation are negligible. Hence, these components can be omitted from the thermal model and equation (7) can be simplified as follows:

$$Q = I(V - OCV) + IT\frac{dOCV}{DT} \quad (9)$$

After the heat generation is calculated from the electrical model, it is transferred to a two-state thermal model, shown in Fig. 3, including the following variables and parameters:

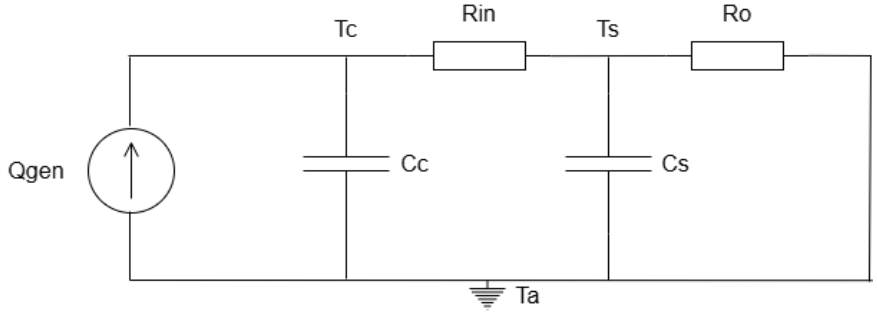


Figure 3 Thermal model of each battery of the pack

➢ $\boldsymbol{C_C}$ (in J/K): Heat capacity of the internal medium of the battery.

➢ $\boldsymbol{C_S}$ (in J/K): Heat capacity of the external medium of the battery.

➢ $\boldsymbol{R_C}$ (in K/W): Thermal resistance of the heat transfer path from core to surface.

➢ $\boldsymbol{R_S}$ (in K/W): Thermal resistance of the heat transfer path from surface to environment.

➢ $\boldsymbol{T_C}$ (in $°C$): Battery core temperature.

➢ $\boldsymbol{T_S}$ (in $°C$): Battery surface temperature.

➢ $\boldsymbol{T_a}$ (in $°C$): Environmental temperature.

The resulting mathematical expressions of the heat transfer are as follows:

$$C_C \frac{d(T_C - T_a)}{dt} = Q - \frac{T_C - T_S}{R_C}$$
$$C_S \frac{d(T_S - T_a)}{dt} = \frac{T_C - T_S}{R_C} - \frac{T_S - T_a}{R_S} \quad (10)$$

As T_a is assumed to be a constant value, it is removed from the time derivative. For discretization, the derivative is approximated by a forward difference as follows.

$$\frac{dT}{dt} \approx \frac{T(k) - T(k-1)}{\Delta t} \quad (11)$$

Equation (10) is further elaborated as:

$$\frac{Tc(k) - Tc(k-1)}{\Delta t} = \frac{Q}{Cc} - \frac{1}{CcRc}[Tc(k-1) - Ts(k-1)]$$
$$\frac{Ts(k) - Ts(k-1)}{\Delta t} = \frac{1}{CsRc}[Tc(k-1) - Ts(k-1)] - \frac{1}{CsRs}[Ts(k-1) - Ta] \quad (12)$$

Sorting out Equation (12), it yields:

$$T_C = T_C(k-1) + \frac{\Delta t}{C_C} Q - \frac{\Delta t}{C_C R_C}[T_C(k-1) - T_S(k-1)]$$
$$T_S = T_S(k-1) + \frac{\Delta t}{C_S R_C}[Tc(k-1) - T_S(k-1)] - \frac{\Delta t}{C_S R_S}[T_S(k-1) - Ta] \quad (13)$$

The parameters to be identified for the thermal model are:

$$\theta t = [Rc\ Ro\ Cc\ Cs\ \frac{dOCV}{dT}] \quad (14)$$

Among them, R_C, R_0, C_C, and C_S are identified using the genetic algorithm method, and the temperature and entropy coefficients are measured using the direct measurement method in [12].

3.3 Battery Simulation Program

This section details the software program developed to simulate the electro-thermal behavior of each battery cell within the pack, establishing the foundation for accurate temperature estimation. The significance of this simulation program lies in its ability to replicate battery performance under controlled virtual conditions, enabling the validation and refinement of temperature estimation methods prior to physical experimentation. By accurately modeling and predicting battery temperature dynamics, this program serves as an essential tool for comparing theoretical predictions with real-world measurements, thus ensuring reliability and robustness of the developed estimation techniques.

3.3.1 Electro-Thermal Model Correlation

The electrical and thermal models are tightly coupled through the mechanism of heat transfer, acting as an interface that integrates the outputs of the electrical model into the inputs of the thermal model, as illustrated in Fig. 4.

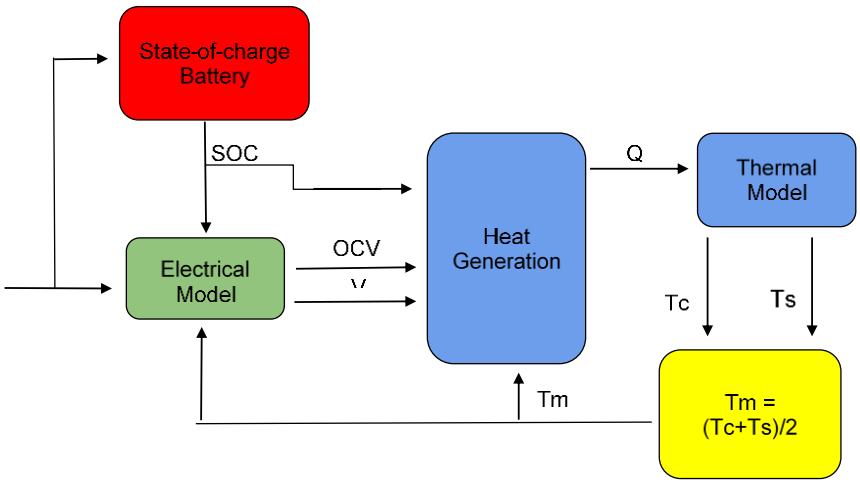


Figure 4 Coupling diagram of electrical and thermal models.

Specifically, the electrical model computes the heat generation based on parameters such as the SOC and the average temperature between the battery core and surface, commonly termed medium temperature.

The generated heat is subsequently transferred to the thermal model, enabling it to accurately estimate the evolution of core and surface temperatures. The simulation accurately captures the interactive dynamics between electrical and thermal states, ensuring robust temperature estimation, which is essential for battery management and safety analysis.

3.3.2 Thermal Parameter Identification

To determine the thermal parameters of the lumped two-state thermal model, namely internal thermal resistance (R_C), external thermal resistance (R_S), core heat capacity (C_C), and shell heat capacity (C_S), a parameter estimation approach based on the RLS algorithm is used. The thermal model is discretized into a second-order difference equation relating the internal temperature difference ($T_{Ca} = T_C - T_a$) to past values of temperature and the heat generation rate (Q), as described in [12].

The RLS algorithm incrementally updates thermal parameters estimates by minimizing the squared error between measured and predicted internal temperatures. The method relies on a regressor vector composed of past temperature and heat inputs and a parameter vector that encapsulates thermal characteristics. This adaptive estimation approach enables robust parameter identification even in the presence of measurement noise or small modeling uncertainties.

However, due to the lack of experimental instrumentation to measure the core temperature of each battery, in this work thermal parameter values have been taken directly from [12], instead of implementing the RLS algorithm as a programming script, schemed in Table 2.

Parameter	$R_c(K/W)$	$R_s(K/W)$	$C_c(K/W)$	$C_s(K/W)$
Value	1.83	4.03	67	3.115

Table 2 Thermal parameters values

3.3.3 Workflow and Structure of the Simulation Code

The software simulation program is structured around the integration of the previously discussed electrical and thermal models, following a clearly defined workflow, shown in Fig. 5, designed to emulate realistic battery behavior during charge cycles. With the target of simulating the behavior of a battery-pack of 7 batteries simultaneously.

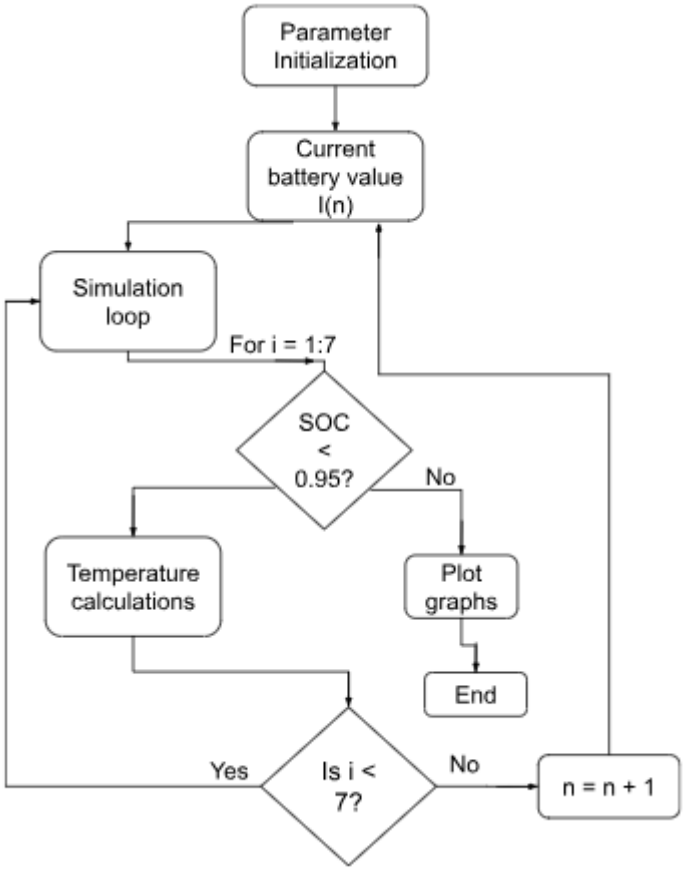


Figure 5 Workflow of battery simulation code

1. Initial parameter values

In [12], an experimental procedure is applied to extract the parameters required for the development of the electro-thermal battery model used in this work.

In the first experiment, the internal electrical parameters of the battery are characterized. An 18650 ternary lithium-ion cell is placed inside a temperature-controlled chamber, instrumented with voltage, current, and temperature sensors. The dynamic voltage response of the battery is monitored while applying charge and discharge currents under various conditions. All figures from Figure 6A to Figure 6K illustrate the dependence of the electrical parameters of the battery equivalent circuit model on both the state of charge (SOC) and the core temperature, as reported in [12]. In that experimental study, the internal ohmic resistance (Table 3), polarization resistance, and capacitance were identified under different SOC levels and temperatures, showing that these values are not constant but vary significantly with operating conditions.

Parameter	5ºC	10ºC	15ºC	20ºC	25ºC
R_0	0.06755	0.05643	0.04862	0.04613	0.04397

Table 3 Ohmic resistance R recognition results of lithium-ion batteries from [12]

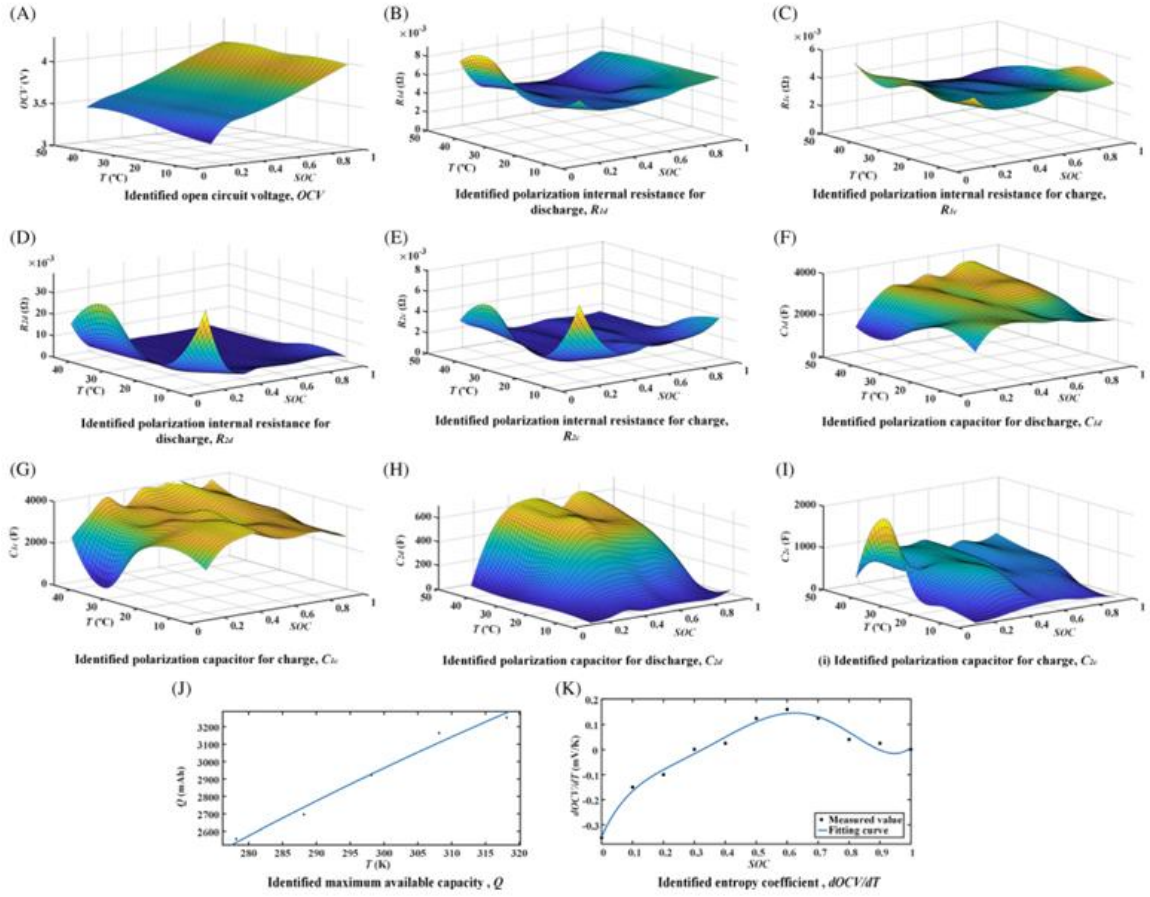


Figure 6 Parameter identification results of electrical model and entropy coefficient from [12].

Due to the lack of instrumentation to obtain the variations of the internal electrical parameters throughout the simulation. The initial parameters will remain constant during the simulation, considering the variations are minimum. In Table 4 the parameters values are given from the previous researcher of this project to match the battery-cells used.

Parameter	$R\ (\Omega)$	$R_1\ (\Omega)$	C (F)	C_1 (F)	SOC_{to}
Value	0.05	1.3	500	3000	0.1

Table 4 Electrical parameter values

SOC, differently from the other respective parameters of Table 4, does change value throughout the simulation. Therefore, the sub-index *to* refers to the initial value used.

The initial values for the thermal parameters are mentioned in previous section 3.3.2 in Table 2.

Figure 7 illustrates how randomness is implemented in the MATLAB code to each of the parameters of each battery, previously defined with the initial values already mentioned. Randomness is added to consider that not all cells from the pack are the same.

```
%Randomness added to each battery
[R1,R2,R3,R4,R5,R6,R7] = deal(R*0.85,R*1,R*1.15,R*0.9,R*0.85,R*1.1,R*1);
[R11,R12,R13,R14,R15,R16,R17] = deal(Re*1.2,Re*1.1,Re*0.8,Re*0.9,Re*1.0,Re*0.9,Re*1.2);
[C1,C2,C3,C4,C5,C6,C7] = deal(C1*1.1,C1*1.,C1*0.7,C1*0.9,C1*1.2,C1*1.,C1*1.1);
```

Figure 7 Randomness added to initial parameters values of each battery

2. Simulation loop

Simulation is based on a charging process of the batteries. Each battery is initially assumed to be partially discharged with an SOC value around 0.1. The program enters a *while* loop which will continue the simulation until one of the 7 batteries, targeted to simulate its behavior, reaches a predefined maximum SOC=0.95. This is proven in Figure 8, from the MATLAB code used for the simulation. At each simulation step, a new current input is generated, mimicking the experimental charge profile with minor random fluctuations (±0.2A), shown in Figure 8. This accounts for measurement and equipment accuracy limitations encountered in actual experimental setups.

3. Temperature calculations

For each battery cell within the pack, the electrical model dynamically computes key variables at every iteration, including the polarization voltage across the RC branch (V_1), SOC, OCV, and the terminal voltage (V_0). These calculations are based on the discrete equations outlined in Section 3.2, which describe the reduced-order electrical model used for each individual cell.

Subsequently, the thermal model processes V_0, OCV, SOC, and the average internal temperature (T_m), calculated as the average of the core and surface temperatures. These values serve as inputs to the simplified Bernardi heat generation equation, as introduced in Section 3.2. Using the resulting heat generation, the model updates the core and surface temperatures through a discretized two-node lumped thermal network. This process, also detailed in Section 3.2,

ensures that the evolving thermal dynamics of the battery are consistently informed by the electrical behavior, thus maintaining a fully coupled electro-thermal simulation framework.

In Figure 9, is illustrated how it is written both the condition loop and the temperature calculations equations in MATLAB code.

4. Output

Finally, the simulation outputs are organized into comprehensive datasets and visual representations, capturing the evolution of key parameters, including voltage profiles, SOC, and both core and surface temperatures for each battery cell, shown in Figure 10. These results are exported in structured formats (CSV and MAT files), facilitating subsequent validation against experimental measurements and supporting the development of reliable temperature estimation models.

```
% Change this value to your desired current
I_nominal = 1;
I_modified = I_nominal + ( -0.2 + 0.4*rand() );
i_battery(end+1) = I_modified;

%Current value of i_battery
Ik = i_battery(n);
```

Figure 8 Randomness added to current from MATLAB battery simulation code

```
%For loop to simulate each battery
for i = 1:nBatt
    k = num2str(i);
    if eval(['lists.soc' k '_list(n) < 0.95']) %Condition of battery almost fully charged
        % Calculate v and soc
        eval(['v' k ' = exp(-T/(R1' k '*C' k ')) * lists.v' k '_list(n) + R1' k '* Ik* (1 - exp(-T/(R1' k '*C' k ')));']);
        eval(['SOC = lists.soc' k '_list(n) + (T/C)*Ik*0.98;']);

        %Function to calculate open circuit voltage with SOC
        voc = soc_voc(SOC);

        % Store SOC and voc
        eval(['lists.soc' k '_list(end+1) = SOC;']);
        eval(['lists.voc' k '_list(end+1) = voc;']);

        %Output voltage
        eval(['vo' k ' = voc + v' k ' + R' k '* Ik;']);
        %Heat transfer
        eval(['Q = (vo' k ' - voc) * Ik + Ik * (lists.Tm' k '_list(n)+273.15) * Entrop(SOC);']);
        eval(['lists.Q' k '_list(end+1) = Q;']);
        %Core and surface temperature
        eval(['Tc_new = (lists.Tc' k '_list(end)+273.15) + T*Q/Cc-T/(Cc*Rc)*[lists.Tc' k '_list(end)-lists.Ts' k '_list(end)];']);
        eval(['Ts_new = (lists.Ts' k '_list(end)+273.15) + T/(Cs*Rc)*[lists.Tc' k '_list(end)-lists.Ts' k '_list(end)]-T/(Cs*Rs)*[lists.Ts' k '_list(end)-Ta];']);
        %Medium temperature of battery
        Tm_new = (Tc_new+Ts_new)/2;
        eval(['lists.v' k '_list(end+1) = v' k ';']);
        eval(['lists.vo' k '_list(end+1) = vo' k ';']);
        eval(['lists.Tc' k '_list(end+1) = Tc_new - 273.15;']);
        eval(['lists.Ts' k '_list(end+1) = Ts_new - 273.15;']);
        eval(['lists.Tm' k '_list(end+1) = Tm_new - 273.15;']);
    else
        stop_flag = true;
    end
```

Figure 9 While loop and temperature calculations from MATLAB battery simulation code

```
A = [x.', i_battery.', ...
    lists.vo1_list.', lists.Ts1_list.', lists.Tc1_list.', lists.soc1_list', lists.voc1_list', ...
    lists.vo2_list.', lists.Ts2_list.', lists.Tc2_list.', lists.soc2_list', lists.voc2_list', ...
    lists.vo3_list.', lists.Ts3_list.', lists.Tc3_list.', lists.soc3_list', lists.voc3_list', ...
    lists.vo4_list.', lists.Ts4_list.', lists.Tc4_list.', lists.soc4_list', lists.voc4_list', ...
    lists.vo5_list.', lists.Ts5_list.', lists.Tc5_list.', lists.soc5_list', lists.voc5_list', ...
    lists.vo6_list.', lists.Ts6_list.', lists.Tc6_list.', lists.soc6_list', lists.voc6_list', ...
    lists.vo7_list.', lists.Ts7_list.', lists.Tc7_list.', lists.soc7_list', lists.voc7_list'];
A = [["t", "ib", "vo1", "Ts1", "Tc1", "soc1", "ocv1", "vo2", "Ts2", "Tc2", "soc2", "ocv2",...
    "vo3", "Ts3", "Tc3", "soc3", "ocv3", "vo4", "Ts4", "Tc4", "soc4", "ocv4", ...
    "vo5", "Ts5", "Tc5", "soc5", "ocv5", "vo6", "Ts6", "Tc6", "soc6", "ocv6", ...
    "vo7", "Ts7", "Tc7", "soc7", "ocv7"]; A];


B = [x.', lists.Q1_list', lists.Q2_list', lists.Q3_list', lists.Q4_list', lists.Q5_list', lists.Q6_list', lists.Q7_list'];
B = [["t", "Q1", "Q2", "Q3", "Q4", "Q5", "Q6", "Q7"]; B];

%Output csv files
writematrix(A, "./Simulation_data/targetWave_elec.csv");
writematrix(B, "./Simulation_data/Qvalues_elec.csv");
```

Figure 10 Output files from MATLAB battery simulation code

3.4 Genetic Algorithm

Accurate electrical-parameter identification is critical to ensure that the reduced-order ECM faithfully reproduces the terminal-voltage response of each cell under realistic operating conditions (ambient temperature, cell ageing, instrumentation tolerances). Nominal values for parameters $\{R, R_1, C_1, C, SOC_{to}\}$ are taken from Table 4. Therefore, these must be adapted to the specific 18650 cells used in this work. To this end, we use a genetic algorithm [13] that iteratively searches the multi-dimensional parameter space to minimize the discrepancy between simulated voltage waveforms and the waveforms obtained from the proposed electrical model.

The genetic algorithm produces two complementary “best” solutions per cell:

- ➢ Best-by-loss, which minimizes mean-squared voltage error (MSE).
- ➢ Best-by-error, which minimizes the average percentage deviation from the ground-truth nominal parameters.

Selecting between these depends on whether priority lies in waveform fidelity or in recovering physically meaningful parameter values.

3.4.1 Genetic Algorithm Fundamentals

A genetic algorithm is a population-based metaheuristic inspired by natural selection. It maintains a population of candidate parameter sets (each an “individual”) and evolves them over successive generations using three main operators:

1. **Selection**: Chooses higher-fitness individuals to form a mating pool.
2. **Crossover**: Exchanges parameter “genes” between pairs of parents to generate offspring.
3. **Mutation**: Randomly perturbs genes to maintain genetic diversity.

Each individual’s fitness is defined as the negative of its loss (fitness = –MSE), so that lower-loss candidates are preferentially selected.

3.4.2 Workflow

The genetic algorithm implementation in GA.py proceeds as shown in Fig. 11:

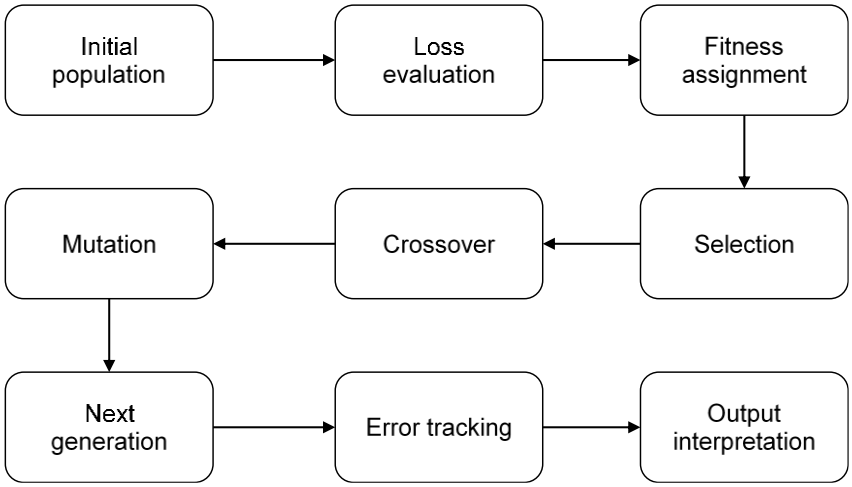


Figure 11 Workflow of genetic algorithm programming

1. Initialization (Figure 12)

In this section, the parameter nominal values are defined with their respective upper and lower limits for sampling. They are represented by working vector *x* as follows:

$$x = [R_i, R_{1i}, C_i, C_{1i}, SOC_{ito}], i \in (1,7) \qquad (1)$$

where i consists of the index number of the battery.

Other parameters are also defined for the algorithm, such as the number of generations to convert, with the population size for each of them.

```
# Synthetic true values (for error calc)
default_true = {"R": 5e-2, "R1": 1.3e0, "C1": 5e2, "C": 3e3, "SOCinit": 0.1}
self.True_ParamValue = {}
self.upper_limits = {}
self.lower_limits = {}
for i in range(num_batteries):
    for name, val in default_true.items():
        key = f"{name}{i+1}"
        self.True_ParamValue[key] = val
        if name in ["R", "R1"]:
            self.upper_limits[key] = val * 2
            self.lower_limits[key] = val * 0.2
        else:
            self.upper_limits[key] = val * 1.3
            self.lower_limits[key] = val * 0.7
```

Figure 12 Upper and lower limits defined in GA.py

2. Loss evaluation (Figure 13)

Loss is evaluated with the output parameters of the simulation of the electrical model: current value, time steps, and output voltage of each battery. The loss function is defined as the following MSE:

$$L(x) = \frac{1}{T}\sum_{k=1}^{T}(Vsim(t_k) - Vtarget(t_k))^2$$

where *x* is the population size, *T* the total number of samples, *Vsim* the simulated output voltage, and *Vtarget* the output voltage from the target waveform. *Vtarget* is calculated using the same equations as in Section 3.1, but with the differential in values added for each population.

```python
def calculate_losses(self, param):
    losses = []
    for i in range(self.num_batteries):
        R    = param[f"R{i+1}"]
        R1   = param[f"R1{i+1}"]
        C1   = param[f"C1{i+1}"]
        C    = param[f"C{i+1}"]
        SOCinit = param[f"SOCinit{i+1}"]

        ib = self.ib
        true_v = self.vb[:, i]
        sim_v = []

        v1_int = 0.0
        i_int  = 0.0
        soc_int= 0.0
        prev_v1= 0.0
        for n in range(len(ib)):
            v1 = (i_int - v1_int/R1)/C1 + prev_v1
            v1_int += v1 * self.dt
            i_int  += ib[n] * self.dt
            soc_int+= ib[n] * self.dt / (C * 3600)
            soc    = SOCinit + soc_int
            sim_v.append(self.soc_voc(soc) + v1 + R*ib[n])
            prev_v1 = v1

        loss_i = (np.sum((true_v - np.array(sim_v))**2))/len(self.ib)
        losses.append(loss_i)
    return losses
```

Figure 13 Loss function implemented in GA.py

3. Fitness assignment (Figure 14)

Since genetic algorithms maximize fitness for future generations, we define

$$f(x) = -L(x)$$

so that smaller loss yields higher fitness.

```python
def getFitness(losses):
    total = np.sum(losses)
    return -total
```

Figure 14 Fitness function in GA.py

4. Selection (Figure 15)

After every generation, each individual is ranked according to its fitness value. The population is sorted from highest to lowest fitness. Afterwards, the top 15% are carried forward to the elite set. These individuals are copied directly to the next generation, guaranteeing that those with better solutions are not lost through the process.

The remaining parents, *N,* are chosen via roulette-wheel selection on cumulative fitness percentages, which consists of the sum of all fitness values, *S,* after shifting or rescaling if necessary, so that every *Fj* >0. Then a cumulative array fitness, C_k is constructed:

$$S = \sum_{j=1}^{N} F_j\,,\; C_k = \sum_{j=1}^{k} F_j\,, C_k^{\%} = 100 * C_k/S$$

Where, k, is the index of a specific individual within the remaining parents and $C_k^{\%}$ the cumulative percentage of fitness up to individual *k*.

```
def selection(self, popRanked, eliteSize):
    df = pd.DataFrame(popRanked, columns=['Index', 'Fitness'])
    df['cum_sum'] = df.Fitness.cumsum()
    df['cum_perc'] = 100 * df.cum_sum / df.Fitness.sum()

    selectionResults = [idx for idx, _ in popRanked[:eliteSize]]
    for _ in range(len(popRanked) - eliteSize):
        pick = 100 * random.random()
        for idx, row in df.iterrows():
            if pick <= row['cum_perc']:
                selectionResults.append(int(row['Index']))
                break
    return selectionResults
```

Figure 15 Selection process in GA.py

To spin the wheel is considered as to pick one parent, draw a uniform random number $r \in [0, S]$, and locate the first index k for which $C_k \geq r$. Individual k is selected and repeated until the required number of parents has been obtained.

This process consists of each individual owning a slice of a roulette wheel whose arc length is proportional to its fitness. Better candidates therefore occupy larger slices and weaker ones still retain a non-zero chance. This helps preserve population diversity and reduce the risk of premature convergence.

5. Crossover (Breed) (Figure 16)

All non-elite individuals chosen from the previous process form the mating pool. The mating pool is shuffled at each generation to randomize who meets whom. Then the shuffled list is taken two at a time to form parent pairs $x^{(1)}, x^{(2)}$. These parents generate children via simulated binary crossover. For each gene *i,* a random $u \in [0,1]$ defines the spread factor

$$\beta_i = \begin{cases} (2u)^{\frac{1}{\eta_c+1}}, & u \leq 0.5, \\ \left(\frac{1}{2(1-u)}\right)^{\frac{1}{\eta_c+1}}, & u > 0.5, \end{cases}$$

where η_c controls the distribution of offspring around the parents. The child gene is

$$x_i^{(child)} = \frac{1}{2}[(1+\beta_i)x_i{}^{(1)} + (1+\beta_i)x_i{}^{(2)}].$$

```
def breed(self, parent1, parent2):
    child1, child2 = {}, {}
    for feature in self.parameter_names:
        u = random.random()
        beta = (1 / (2 * (1 - u)))**0.5 if u > 0.5 else (2 * u)**0.5
        val1 = 0.5*((1+beta)*parent1[feature] + (1-beta)*parent2[feature])
        val2 = 0.5*((1-beta)*parent1[feature] + (1+beta)*parent2[feature])
        lo, hi = self.lower_limits[feature], self.upper_limits[feature]
        child1[feature] = min(max(val1, lo), hi)
        child2[feature] = min(max(val2, lo), hi)
    return child1, child2

def breedPopulation(self, matingpool, eliteSize):
    children = matingpool[:eliteSize]
    pool = random.sample(matingpool, len(matingpool))
    for i in range(len(matingpool) - eliteSize):
        c1, c2 = self.breed(pool[i], pool[-i-1])
        children.append(c1)
        if len(children) < len(matingpool):
            children.append(c2)
    return children[:len(matingpool)]
```

Figure 16 Breed process implemented in GA.py

6. Mutation (Figure 17)

Each gene of a non-elite child undergoes random-reset mutation with probability defined in script as *mutation_rate.* If mutation occurs, that gene is re-sampled uniformly within its bounds. This operator preserves genetic diversity and helps escape local minima.

```
def mutate(self, individual, mutationRate):
    for feature in self.parameter_names:
        if random.random() < mutationRate:
            lo = self.lower_limits[feature]
            hi = self.upper_limits[feature]
            individual[feature] = lo + (hi - lo) * random.random()
    return individual

def mutatePopulation(self, population, mutationRate, eliteSize):
    new_pop = population[:eliteSize]
    for ind in population[eliteSize:]:
        new_pop.append(self.mutate(ind.copy(), mutationRate))
    return new_pop
```

Figure 17 Mutation step in GA.py

7. Next generation (Figure 18)

The new population comprises the elite plus the mutated children. This process repeats for as many times as the user indicates.

```
def getNextGeneration(self, pop, eliteSize, mutationRate):
    # Selection and breeding
    popRanked = self.rankRoutes(pop)
    selectionResults = self.selection(popRanked, eliteSize)
    matingpool = self.matingPool(pop, selectionResults)
    children = self.breedPopulation(matingpool, eliteSize)
    nextGen = self.mutatePopulation(children, mutationRate, eliteSize)
    # Split elites (no change) and rest for evaluation
    noChangePop = nextGen[:eliteSize]
    toBeTested = nextGen[eliteSize:]
    return noChangePop, toBeTested
```

Figure 18 Next generation function in GA.py

8. Output interpretation (Figure 19)

After each generation, the genetic algorithm evaluates the population based on the loss function, which quantifies how well the estimated parameters reproduce the measured voltage waveform. The best individual, defined as the one with the lowest loss, is selected as the generation's optimal solution. This loss directly reflects the voltage tracking accuracy: a lower loss corresponds to a better fit between the simulated and measured waveforms.

Although the percentage error of each individual's estimated parameters relative to their nominal ground-truth values can be computed, in this implementation, only the loss is used as the selection criterion. The average parameter error is not used for saving or ranking individuals. Therefore, only when the best loss in a generation is lower than in previous ones, the associated estimated parameters are saved into the output dataset for further use in the temperature estimation models.

```
for gen in range(generations):
    if gen > 0:
        noChangePop, toBeTested = GA.getNextGeneration(pop, eliteSize, mutationRate)
    toBeTested = GA.getPerformance(toBeTested)
    pop = noChangePop + toBeTested

    fitness = [getFitness(ind['losses']) for ind in pop]
    best_idx = int(np.argmax(fitness))
    best = pop[best_idx]
    # Print per-battery losses and parameters
    loss_str = ", ".join([f"B{i+1}:{l:.4e}" for i, l in enumerate(best['losses'])])
    print(f"Gen {gen}: Losses = [{loss_str}]")
    param_strs = []
    for i in range(GA.num_batteries):
        params_i = {name: best[name] for name in GA.parameter_names if name.endswith(str(i+1))}
        param_strs.append(f"B{i+1}:" + ";".join([f"{k}={v:.3e}" for k, v in params_i.items()]))


    # Check and update best per-battery files
    for i, loss_i in enumerate(best['losses']):
        if loss_i < best_losses[i]:
            best_losses[i] = loss_i
            # Save parameters for battery i+1
            params_i = {k: best[k] for k in GA.parameter_names if k.endswith(str(i+1))}
            df_i = pd.DataFrame([params_i])
            file_path = os.path.join(GA.dataDir, f"battery{i+1}_best.csv")
            df_i.to_csv(file_path, index=False)
```

Figure 19 Output interpretation process in GA.py

3.4.3 Impact on Parameter Estimation

By calibrating the ECM parameters to our specific cells, the genetic algorithm ensures that subsequent thermal simulations and EKF estimations operate on an electrical model that accurately reflects real-world behavior. A lower loss yields more faithful voltage dynamics, minimizing modeling mismatch, whereas a lower error yields parameters that align with physical expectations, reducing bias in SOC and heat generation predictions. Together, these calibrated parameters underpin reliable core and surface temperature estimation in the full electro-thermal framework.

3.5 Extended Kalman Filter

3.5.1 Introduction

The EKF is a recursive state-estimation algorithm tailored for dynamic systems whose behavior is governed by non-linear process and/or measurement equations. Starting from the classical (linear) KF, the EKF retains the same two-step structure, prediction and correction, but overcomes non-linearity by local linearization. At each time step, the non-linear functions are expanded in a first-order Taylor series around the current best estimate, yielding a set of Jacobian matrices that serve as locally valid linear models. Standard Kalman algebra is then applied to those linear surrogates, producing an updated state vector that fuses the model forecast with the newest sensor information and an evolving error-covariance matrix that quantifies real-time confidence in the estimate.

In this project the EKF is tasked with estimating core and surface temperatures of lithium-ion cells inside a large battery pack that lacks dedicated thermal sensors. The prediction model propagates these temperatures forward to the reduced-order electro-thermal model (Sections 3.1 and 3.2). During the correction step, the EKF assimilates the sparse temperature measurements available from the few instrumented reference cells. The EKF exploits the coupling, encoded in the Jacobian of the thermal model, to redistribute the information and sharpen the temperature estimates for every cell.

This EKF implementation runs on-line at the pack-management controller's sampling rate, producing continuous best-guess values for each cell's core and surface temperatures along with their uncertainties. The result is a virtual sensor network that enables early detection of thermal gradients and potential runaway conditions, without the cost or wiring complexity of a physical sensor on every cell, thereby fulfilling this work's objective of safe, scalable temperature monitoring with a minimal sensor set.

3.5.2 Electro-Thermal Model

The thermal model in Section 3.2, is represented by the following state equations [14]:

$$\underline{x}(t) = A_d x(t) + B_d u_d(t) \qquad (1)$$

where A_d and B_d come from applying Kirchoff's law into Fig. 3:

$$Q = \frac{T_c - T_s}{R_c} + C_c \frac{dT_c}{dt} \qquad (2)$$

$$\frac{T_c - T_s}{R_c} = \frac{T_s - T_a}{R_s} + C_s \frac{dT_s}{dt} \qquad (3)$$

Reconstructing them as dependent of $\frac{dT_c}{dt}$ and $\frac{dT_s}{dt}$, it yields:

$$\frac{dT_c}{dt} = \frac{T_s - T_c}{R_c C_c} + \frac{1}{C_c} \qquad (4)$$

$$\frac{dT_s}{dt} = \frac{T_c}{C_s R_c} - \frac{T_s}{C_s}\left(\frac{1}{R_c} + \frac{1}{R_s}\right) + \frac{T_a}{C_s R_s} \qquad (5)$$

State matrix: $A_d =$

$$\begin{bmatrix} \frac{-1}{R_c C_C} & \frac{1}{R_c C_c} \\ \frac{1}{C_s R_c} & \frac{-1}{C_s}\left(\frac{1}{R_c} + \frac{1}{R_s}\right) \end{bmatrix}$$

State matrix: $B_d =$

$$\begin{bmatrix} \frac{1}{C_c} & 0 \\ 0 & \frac{1}{C_s R_s} \end{bmatrix}$$

$$\hat{x} = [Tc\ Ts]^T, \hat{u}_d = [Q\ T_a]^T$$

where vector $\hat{x}$ is the augmented state vector with the parameters to estimate and $\hat{u}_d$ the input vector of known parameters value. A_d can be reconstructed into the system matrix dependent of

$\hat{x}$ and B_d of the input vector $\hat{u}_d$. The discrete state space model can be achieved by using a zero-order hold instead of the input for one sampling period:

$$x(k+1) = A_d(k)x(k) + B_d(k)u_d(k) \quad (6)$$

3.5.3 Jacobian Model

The Jacobian matrix is required for the linearization process in the model. To obtain the partial derivative it is essential to define those parameters to be identified or already identified in the state model. In this case, the parameter vector $\hat{\theta}$ is considered as:

$$\hat{\theta} = [C_c\ C_s R_c R_s]^T \quad (7)$$

where these parameters are already obtained in the previous process of thermal-model parameter identification [12]. Thus, the augmented state model (6) can be rewritten as:

$$x(k+1) = A_d(\theta(k))x(k) + B_d(\theta(k))u_d(k) \quad (8)$$

Given the augmented state vector in equation (8), the Jacobian matrix is defined as:

$$F(t) = \frac{\partial f(\cdot)}{\partial x_a}|_{\hat{x}_a(t)} = \begin{bmatrix} A_d(\hat{\theta}_{(k)}) & \frac{\partial}{\partial \theta}(A_d(\hat{\theta}_{(k)}) + B_d(\hat{\theta}_{(k)})\hat{\theta}_{(k)} \\ 0_{4x4} & I_{4x4} \end{bmatrix} \quad (9)$$

3.5.4 Filtering process

In this process, first initial guesses are made for physical state and parameters. At time step k, the predictions of the augmented state vector are calculated from the input $u_d(k)$ and the estimation of the augmented state vector.

Afterwards the Jacobian model is computed for the linearization of the augmented dynamics about the current estimate. This yields the state transition matrix for the covariance update, defined as the estimation error. The covariance of the prediction error $M(k+1)$ is calculated pushing the last covariance *P(k)* through the linearized dynamics. With the output equation of the sensed batteries, *H*, the Kalman gain is computed, which weighs the measurement residual when updating the state estimate. Then, the posterior covariance is calculated for further iteration implementation. Finally, the new estimation state vector is computed. The filtering process is summarized as follows:

a) Prediction process:
   1. Determination of the $\hat{x}(0)$, $\hat{\theta}(0), P(0)$ initial values
   2. $\underline{x}(k+1) = A_d(\hat{\theta}(k))\hat{x}(k) + B_d(\hat{\theta}(k))u_d(k)$
   3. $F(k) = \frac{\partial f(\cdot)}{\partial x_a(k)}|_{x_a(k)}$
   4. $M(k+1) = F(k)P(k)F(k)^T + Q$

b) Correction process:

5. $K(k+1) = M(k+1)H^T(HM(k+1)H^T + R)^{-1}$
6. $P(k+1) = (I - K(k+1)H)M(k+1)$
7. $\hat{x}(k+1) = \underline{x}(k+1) + K(k+1)(Z(k+1) - H\underline{x}(k+1))$
8. Go to 2.

where

$\underline{x}(k)$: Prediction of the state vector

$\hat{x}(k)$: Estimate of the state vector

M(k+1): Covariance matrix of the prediction error

P(k+1): Covariance matrix of the estimation error

I: Unit matrix

Z(k): Vector with temperature values of the batteries with sensors

This description of the filtering process applies for the case of a single battery, where the values to estimate are the ones from vectors $\hat{x}$ and $\hat{\theta}$. In this work, the temperatures of 7 batteries are guessed, resulting on these vectors defined as follows:

$\hat{x} = [T_{c1}\ T_{s1}T_{c2}\ T_{s2}T_{c3}\ T_{s3}T_{c4}\ T_{s4}T_{c6}\ T_{s6}T_{c6}\ T_{s6}T_{c7}\ T_{s7}]^T$, $\hat{\theta} = [C_c\ C_s R_c R_s]^T$.

3.5.5 Software Design

Related to the software design for the implementation of the Extended Kalman Filter. There are 2 python files created.

- In Main_7bat.py, all the parameter values are initialized and fetched parameters from TargetWave.csv file, output of the simulation program, explained in section 3.3. Also, the matrixes and equations are defined from section 3.5.2 for further use. Plus, it also creates the Jacobian Model needed on the implementation of the algorithm.
- In EKF_Estimation, once all parameters and equations are defined, this implements the prediction and correction process, in its respective order, explained in previous section.

After all the specified generations by the user to estimate the temperature is completed, the program outputs a .txt file with all the errors, between the simulated value and the estimated, of the surface and core temperature from each battery. Plus the percentage error of the parameters from the thermal model and the mean error of all thermal parameter between the values used in the battery simulation and along the estimation process. Also it plots different graphs for visual representation:

- A diagram with the representation of change of both surface and core temperature of the data collected from the simulation program and the estimated data, throughout all the number of samples.
- The error values between one of the batteries estimated surface temperature and the simulated surface temperature, alongside all the number of samples.
- Same as previous, but instead of surface with the representation of the core temperature.

3.6 Experimental Process

3.6.1 Experimental Setup and Procedure

To validate the accuracy of the proposed electro-thermal model developed in simulation, a controlled laboratory experiment has been conducted. This experiment aimed to compare the

thermal and electrical behavior of a real battery pack against the simulated results, providing a foundation for the subsequent temperature estimation implementation.

3.6.2 Instrumentation and Components

The experimental setup incorporated the following instrumentation and components:

- **Power DC Supply**, for controlled charging of the battery pack.
- **7 cylindrical 18650 Lithium-Ion Ternary Battery Cells**, configured in series to replicate a multi-cell pack.
- **Oscilloscope**, to monitor and verify the current supplied during the charging process.
- **Thermal Camera**, to capture in real time the surface temperature distribution of the battery cells.
- **DC Electronic Load**, for controlled discharging of the battery pack.
- **Multimeter**, for accurate measurement of the voltage of each battery cell during the experiment.

3.6.3 Battery Configuration

The series connection of the seven lithium-ion ternary cells (Fig. 20) ensures that all of them experience the same current flow, eliminating current imbalance across the pack and facilitating uniform thermal response analysis.

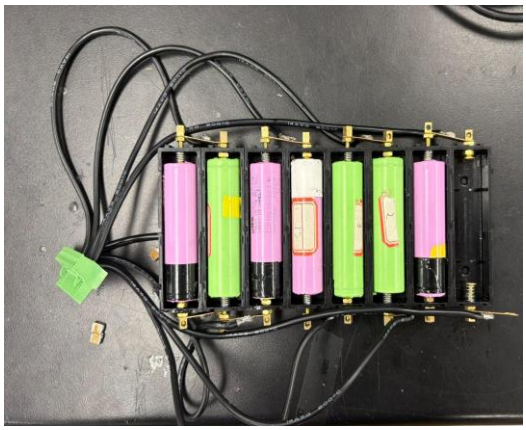


Figure 20 Battery pack configuration

3.6.4 Discharge Cycle

Prior to the experiment, the cells have been fully discharged to bring the SOC of each one of them close to zero. This has been achieved using a programmable DC electronic load (Fig. 21) operating in constant current mode. This ensures a uniform starting point for all cells in the charging cycle, which is critical for accurate thermal monitoring and modeling.

Figure 21 DC electronic load

3.6.5 Charging Process

Once discharged, the series-connected battery pack is subjected to a controlled charging process using a DC power supply (Fig. 22) configured to deliver a constant current of 1 A. The choice of this current aligns with the simulation parameters defined in the electro-thermal model. An oscilloscope was used to monitor the current waveform in real-time, helping identify any potential anomalies or deviations introduced by the power supply unit.

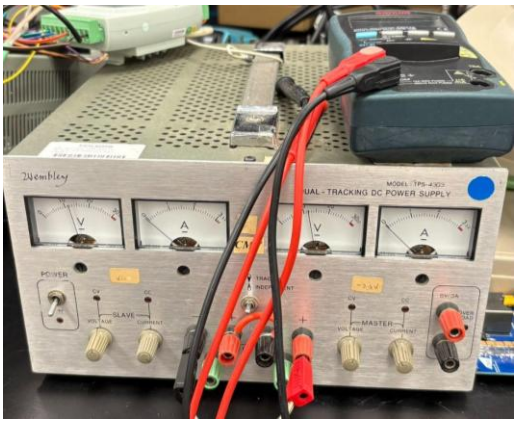


Figure 22 DC Power supply

3.6.6 Temperature and Voltage Measurement

Throughout the charging process, thermal and electrical measurements are systematically recorded. At two-minute intervals, the surface temperature of each battery cell is measured using a thermal camera (Fig. 23). Specifically, the maximum surface temperature observed on each cell is documented to establish a consistent reference for comparison with simulation outputs.

Figure 23 Thermal camera

Simultaneously, the voltage of each individual cell is measured using a digital multimeter. This ensures a high level of accuracy in tracking the charging behavior and assessing the SOC progression across all cells.

This procedure is continuously repeated until the voltage of at least one of the battery cells reaches its full charge limit, marking the end of the charging cycle.

3.6.7 Experimental Limitations

Several practical limitations influence the accuracy of the measurements:

- **Absence of Thermal Chamber**: The experiment has not been conducted in a controlled thermal environment. Ambient temperature fluctuations may have introduced variability in the battery surface temperature measurements.

- **Thermal Camera Precision**: The accuracy of temperature data depends on the resolution and calibration of the thermal camera. Inconsistencies in identifying the exact thermal hotspots may introduce minor errors.

- **Power Supply Instability**: The current output from the power supply exhibits fluctuations during the experiment, deviating from the intended constant current profile and potentially affecting both thermal and electrical responses.

These limitations are acknowledged and factored into the analysis and interpretation of the experimental data when comparing with the modeled behavior.

6.5 Program Workflow

The software workflow, illustrated in Fig. 24, integrates MATLAB and Python scripts for electrical and thermal modeling, parameter estimation, and sensor less temperature prediction using EKF.

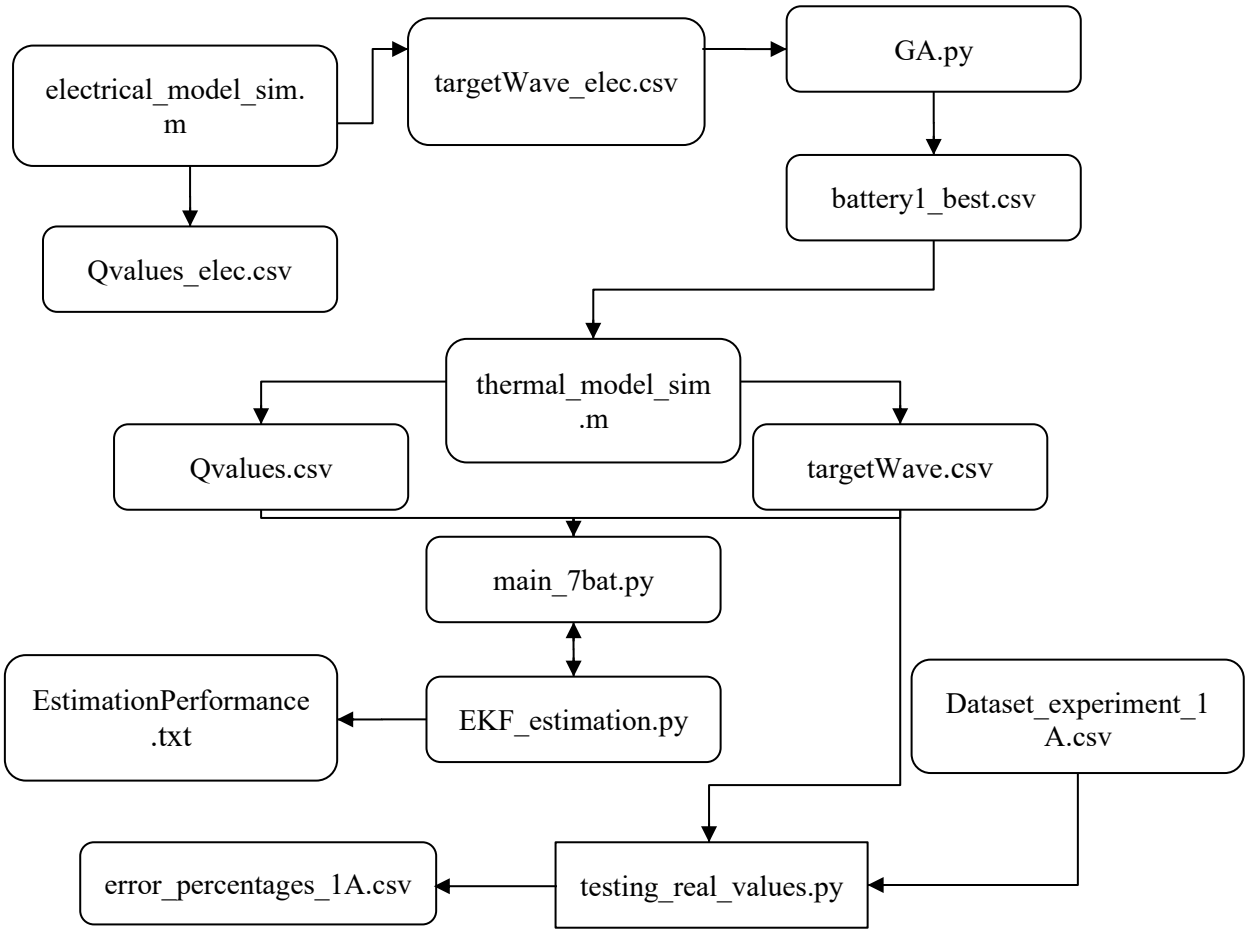


Figure 24 Workflow of the software implemented during the project

4 MODEL VALIDATION AND RESULTS

4.1 Introduction

This chapter critically assesses the effectiveness of the methods developed throughout the work. First, it quantifies how closely the coupled electro-thermal simulation reproduces the measured behavior of the battery cells obtained in laboratory tests. Second, it evaluates the performance of the EKF that infers surface temperatures on cells where no physical sensors are installed. By plotting simulated outputs with experimental data and analyzing EKF estimates against available measurements, the accuracy, limitations, and practical relevance of the entire modeling-and-estimation pipeline are evaluated.

4.2 Validation of the Electro-Thermal Simulation

4.2.1 Simulation Performance against Experimental Data

To verify that the coupled electro-thermal model reproduces the actual behavior with sufficient fidelity, the simulated temperature traces are benchmarked against the laboratory data obtained from the seven-cell 18650 pack. Because the two data sets differ markedly in time resolution (the MATLAB model outputs **N** samples per cycle (frames per sample time), whereas the thermal-camera acquisition yields only **n** samples (frames per measured time)) the comparison is carried out with a two-tier strategy.

For this observability between both simulated performance and experimental data, a python file named testing_real_values.py is created. To study the errors shown in the simulation program and deeper understanding for further modifications.

1. Qualitative trend check.

In Fig. 12, each simulated surface-temperature trajectory is plotted as a continuous curve, whereas experimental readings appear as discrete markers at the corresponding simulation time indices. Formally, the *k*-th experimental sample is positioned at index

$$index_k = \frac{N}{n}k, \qquad k = 1, 2, \ldots, n.$$

Overlaying the two series in this manner reveals whether the model captures the overall temperature *rise-and-saturation* behavior despite the sparse sampling and inevitable sensor noise. As Fig. 25 illustrates, experimental dots tightly cluster around the simulated lines for every cell, confirming that the model tracks both the transient and steady-state trends.

2. End-point error analysis

Because all cells begin the test at the same ambient temperature and follow an identical charging profile, the simplest scalar accuracy metric is the percentage deviation of the final surface temperature:

$$Error_{\%} = \frac{T_{exp,final} - T_{sim,final}}{T_{exp,final}} x100$$

These values, reported in Table 5, range from 0.29 % to 4.14 %. Five of the seven cells stay below 1 %, and even the worst case remains well within the ±2 °C tolerance typically accepted for EV BMS design.

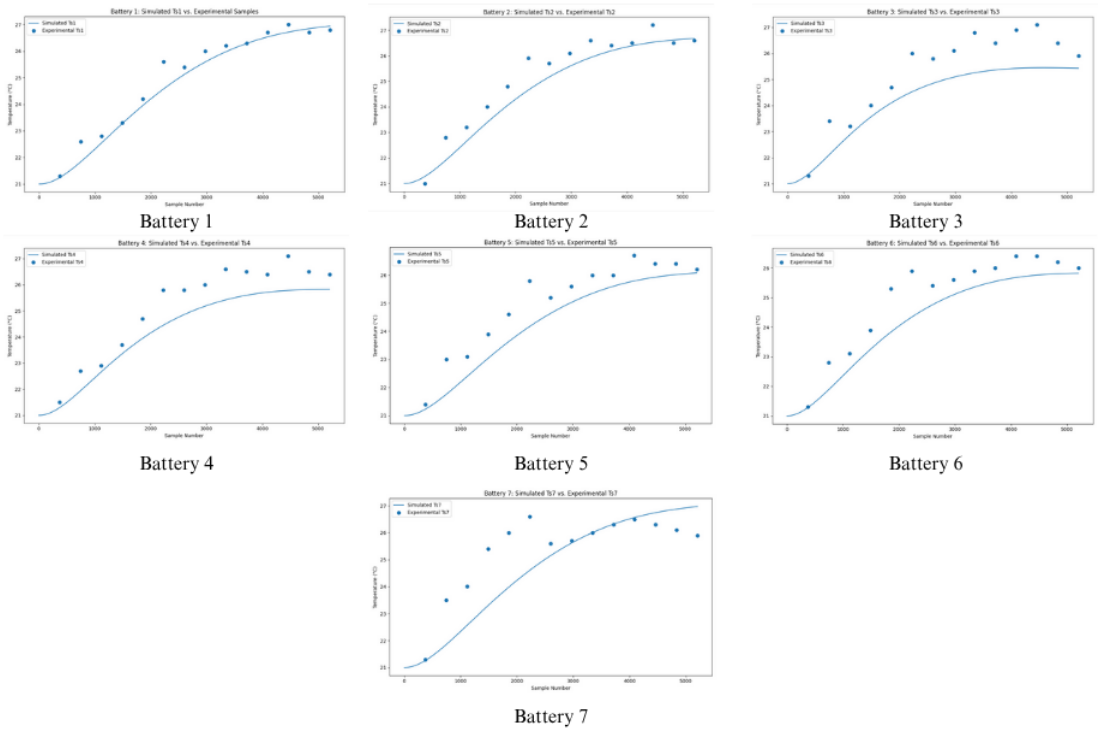


Figure 25 Surface temperature graphs of the battery pack

Cell	1	2	3	4	5	6	7
Error	0.53%	0.29%	1.80%	2.18%	0.45%	0.68%	4.14%

Table 5 Simulation and experimental surface temperatures

The trend agreement in Fig. 13 and the low end-point errors in Table 4 demonstrate that the electro-thermal model accurately predicts surface temperatures under the tested conditions, providing a solid foundation for the estimation work presented in section 4.3.

4.2.2 Parameter uncertainty and genetic algorithm implementation

Although a genetic algorithm routine for identifying the five parameters of the equivalent-circuit model has been coded and tested, it is *not* the basis of the temperature-validation plots shown earlier, for two related reasons:

1. **Absence of the right experimental signals.**

A genetic algorithm needs high-resolution current-pulse or dynamic load data, together with the resulting terminal-voltage response, to converge to realistic R, C and OCV values. In the present campaign, those signals could not be instrumented, so the genetic algorithm had to "learn" from a simulated voltage trace instead of from a real one (the *theory-over-theory* scenario described in Section 3.4).

2. **Consequent large errors.**

When a genetic algorithm is calibrated against its own source model, any mismatch introduced by sampling noise or numerical drift is amplified, producing in this case the 5–9 % discrepancies listed in Table 6. Using those parameters in the thermal simulation would therefore degrade, instead of improving, the agreement with the laboratory temperature data displayed in Fig. 13.

Cell	1	2	3	4	5	6	7
Error	6.70%	7.54%	5.15%	9.15%	7.86%	5.36%	2.71%

Table 6 Error differences between simulated and experimental surface temperatures with the genetic algorithm implementation

For these reasons, the electro-thermal results in Section 4.2 rely on literature-based parameters rather than on the genetic algorithm output. Nevertheless, retaining the genetic algorithm script is highly valuable. Once future tests capture the needed voltage-response data, the same code can be rerun with minimal modification to obtain cell-specific electrical parameters and push the overall simulation accuracy even further.

Therefore, the workflow represented in Section 3.7 of the full code has been modified as shown in Figure 26:

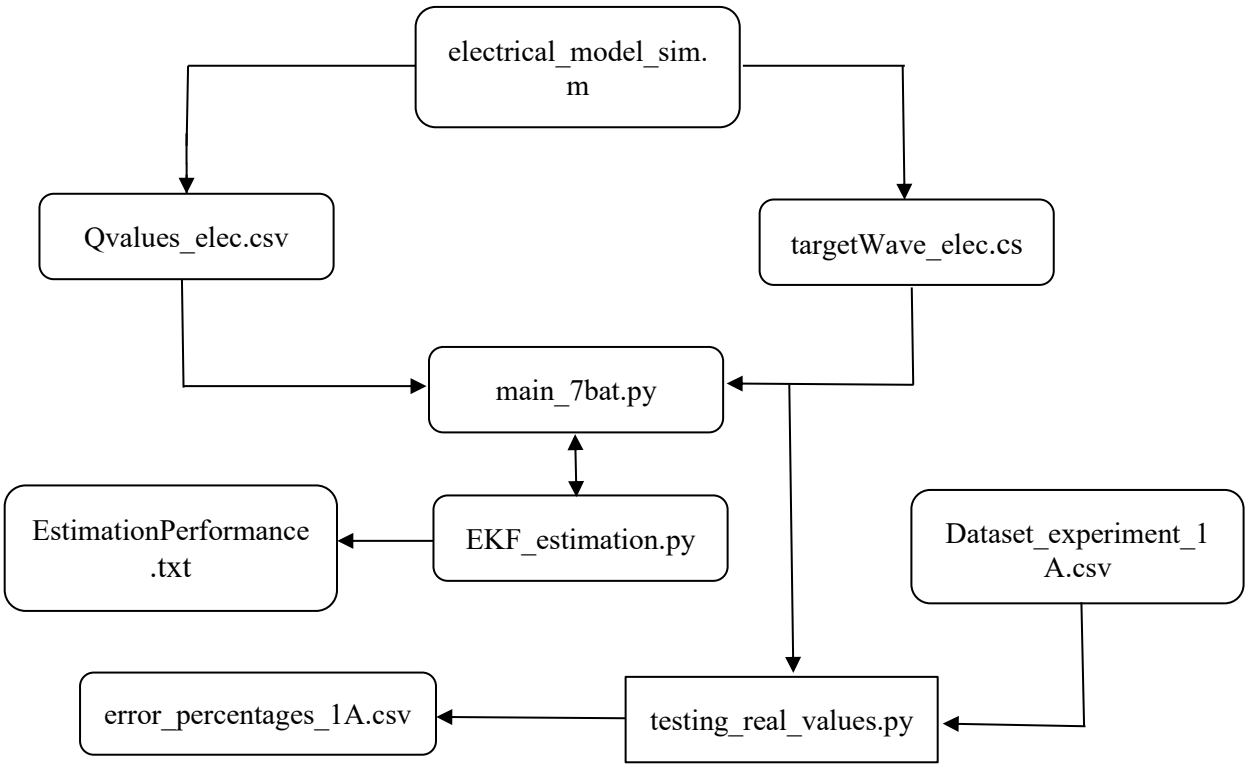


Figure 26 Workflow of the software implemented during the project without the use of the GA

4.2.3 Simulation performance vs. results in [12]

A further credibility check for the electro-thermal model has been performed by replicating the operating conditions reported in [12]: an 18650 lithium-ion cell identical to ours, tested in a 35 °C chamber and subjected to a 1.5 A square-wave current pulse profile. When these boundary conditions and load cycles are imposed to our simulation, the predicted core- and surface-temperature trajectories converge to virtually the same end-values as those in [12] (Fig. 27). In

other words, without any additional parameter tuning the model that had already matched our own laboratory data also reproduced an *independent* data set obtained under a hotter ambient and a more aggressive current regime. This cross-validation against external measurements confirms that the coupled electro-thermal framework remains accurate across a broader envelope of temperature and load conditions than those used in the initial calibration.

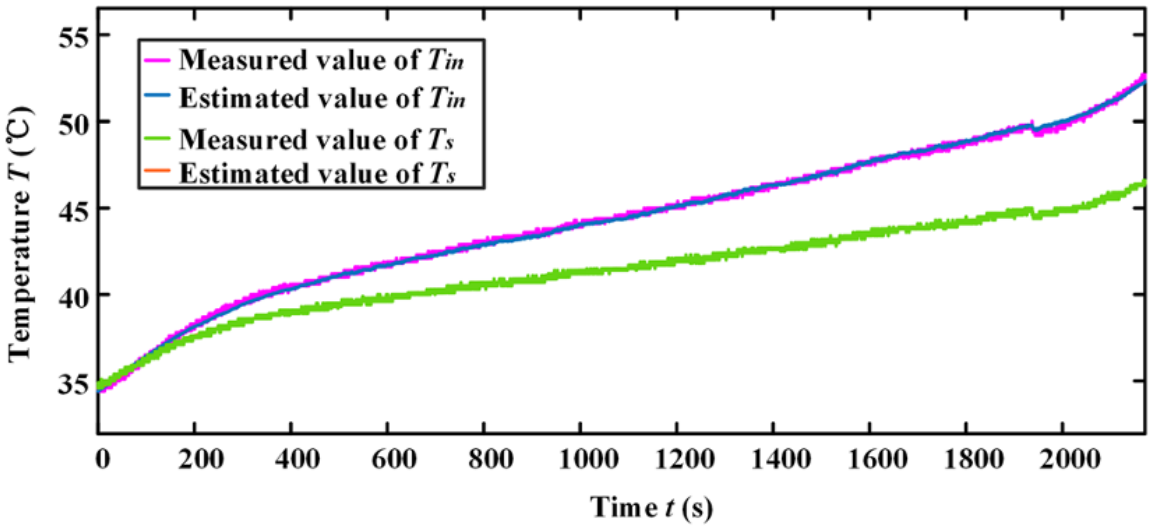


Figure 27 Results of core and surface temperature graphs from [12]

4.3 EKF Performance

To validate the EKF implementation under controlled conditions, several simulations have been conducted using surface temperature data directly extracted from the electro-thermal model. These values are treated as if they were real-time sensor readings, allowing the EKF to estimate both surface and core temperatures of non-instrumented cells. This approach isolates the performance of the estimator from measurement noise and model error.

4.3.1 Case 1: Four Surface Sensors

In this first scenario, the EKF is provided with temperature measurements from four cells (1, 3, 5, and 7). The remaining cells are non-instrumented and their temperatures are reconstructed by the observer. Fig. 28 shows the simulated surface and core temperature profiles compared with their respective EKF estimates.

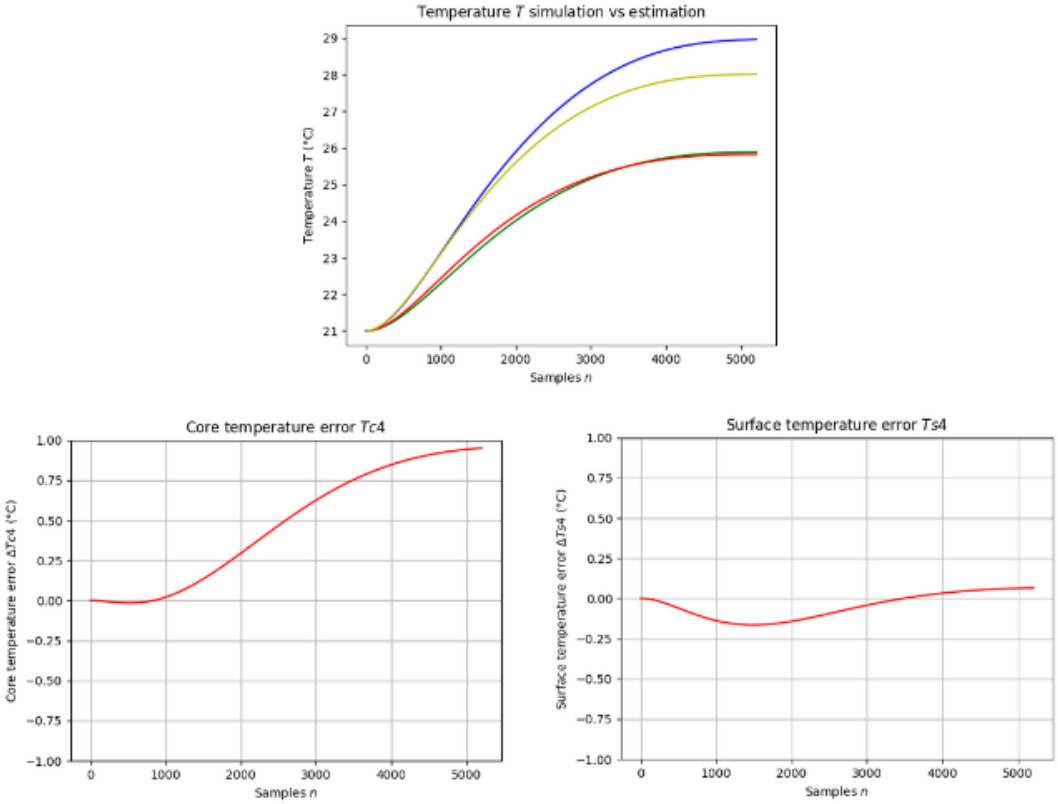


Figure 28 Surface temperature and error graphs with 4 sensors

- **Red line**: Simulated surface temperature ($T_{s,sim}$)
- **Green line**: EKF-estimated surface temperature ($T_{s,est}$)
- **Yellow line**: Simulated core temperature ($T_{c,sim}$)
- **Blue line**: EKF-estimated core temperature ($T_{c,est}$)

The surface temperature estimate closely follows the simulated ground truth, with no noticeable delay or steady-state offset. The core temperature estimate also aligns well with the simulation, despite the fact that no core temperature is directly observed by the EKF. This demonstrates the observer's ability to reconstruct unmeasured states using only the available surface data and the dynamic thermal model.

To quantify estimation performance, the instantaneous errors in both core and surface temperatures are plotted in the bottom graphs of Fig. 16. These error profiles show that:

- The **core temperature error** remains within ±1 °C throughout the entire simulation, slowly rising but stabilizing around 0.8–0.9 °C toward the end of the charging phase.
- The **surface temperature error** remains negligible, fluctuating within ±0.2 °C, confirming that the EKF effectively tracks the measured inputs even under time-varying thermal conditions.

The corresponding mean absolute errors for each of the estimated (non-instrumented) batteries (2, 4, and 6) are summarized in Table 7.

Cell	3	4	6
Surface error (ºC)	0.081	0.076	0.073
Core error (ºC)	0.478	0.473	0.461

Table 7 Mean absolute error values for EKF estimation with 4 sensors

These results confirm that with only four surface measurements, the EKF can accurately estimate the thermal behavior of the remaining cells. The surface temperature error remains well below typical thermistor precision (±0.5 °C), whereas the core temperature error consistently stays within the ±1 °C range, an important criterion for early detection of thermal deviations and prevention of thermal runaway.

This initial validation confirms that the EKF provides high-fidelity temperature estimation with partial observability, using just over half the cells as sensor inputs. The algorithm demonstrates stable convergence, low estimation error, and robustness to internal coupling effects in the thermal model. Building on this result, the next validation scenarios examine whether the estimator can retain similar performance when the number of available sensors is reduced from four to three, and eventually to only two. This progressive reduction aims at determining the minimum sensor count that still ensures safe and reliable thermal monitoring for large-scale battery packs.

4.3.2 Case 2: Three Surface Sensors

In this second test, the number of surface temperature inputs is reduced to **three**, coming from cells 1, 5, and 7. The EKF is tasked with estimating temperatures for the other four cells, including both surface and core values.

As with the previous test, the data used as measurements are directly taken from the simulation and treated as sensor inputs to evaluate the performance of the EKF. This approach isolates the effect of reduced sensor count on the estimation process while maintaining ideal (noise-free) input conditions.

Fig. 17 presents the simulated and estimated temperature profiles for one of the non-instrumented batteries. Despite the reduction in sensor inputs, the estimated surface and core temperatures retain excellent agreement with their simulated references. The filter remains stable, with no observable divergence or lag in response across the estimation window. The error curves at the bottom Fig. 29 reflect a similar trend:

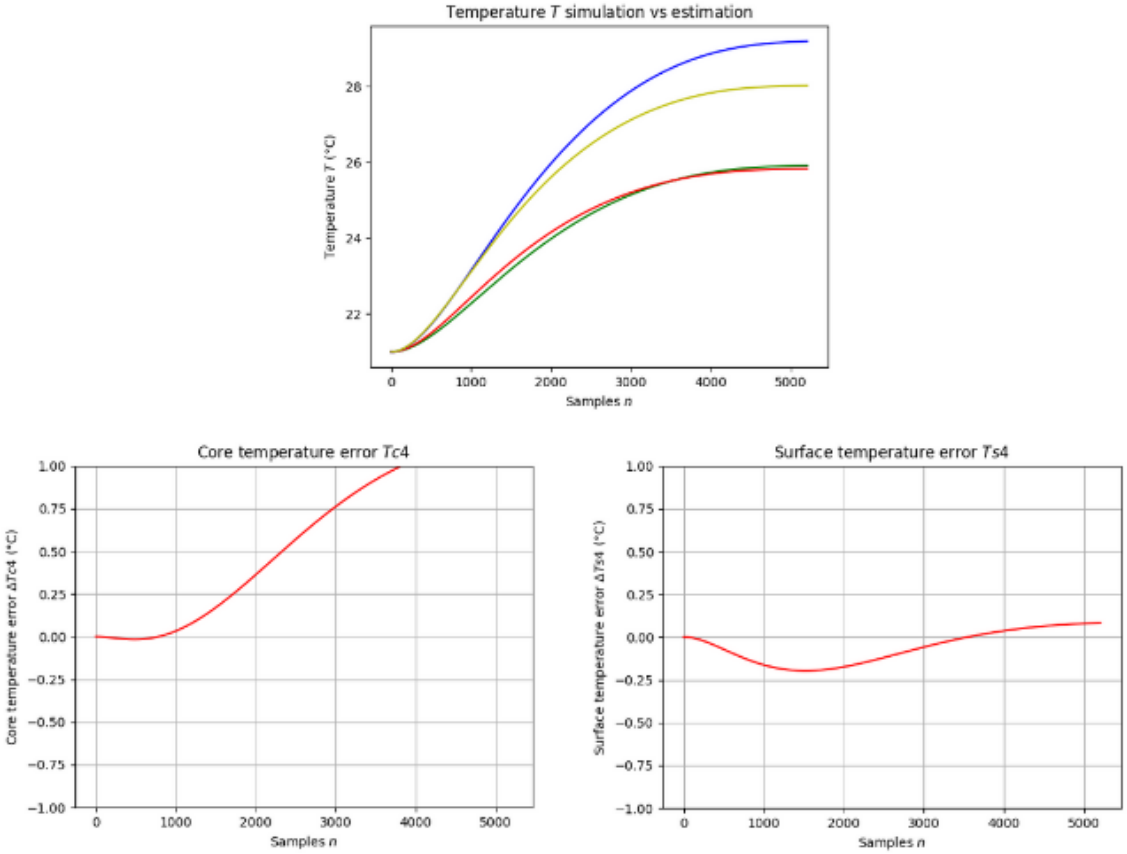


Figure 29 Surface temperature and error graphs with 3 sensors

- **Core temperature error** remains below ±1 °C, gradually increasing but staying under control across the 5,000-sample horizon.

- **Surface temperature error** remains tightly bounded, rarely exceeding ±0.2 °C.

The results summarized in Table 8, indicate only a minor degradation in estimation accuracy compared to the four-sensor case:

Cell	2	3	4	6
Surface error (ºC)	0.096	0.098	0.093	0.089
Core error (ºC)	0.627	0.586	0.580	0.566

Table 8 Mean absolute error values from EKF estimation with 3 sensors

- Surface temperature errors increased slightly but remained under **0.1 °C**, well below the resolution limits of practical thermal sensors.

- Core temperature errors increased by approximately **0.1–0.15 °C**, but still comfortably satisfy the ±1 °C safety threshold for thermal reliability.

This second test confirms that even with only three surface temperature sensors, roughly one for every two to three cells, the EKF retains strong estimation performance. The algorithm continues to provide high-fidelity reconstructions of both observable and unmeasured thermal states. These results suggest that the system remains observable and well-conditioned under reduction in available sensor data.

4.3.3 Case 3: Two Surface Sensors

Finally, in the most constrained configuration, only **two** surface temperature readings are available, from cells 1 and 5. This setup tests the EKF's ability to estimate temperatures across the remaining five cells under highly limited sensor input.

The estimation performance for cell 4 is presented in Fig. 30. The EKF continues to accurately track surface temperature, even with further reduced input data. The estimated core temperature also follows the general shape and time evolution of the simulated reference. However, as evident in the error plots below the main graph, core temperature estimate deviates more noticeably, particularly in the final third of the cycle.

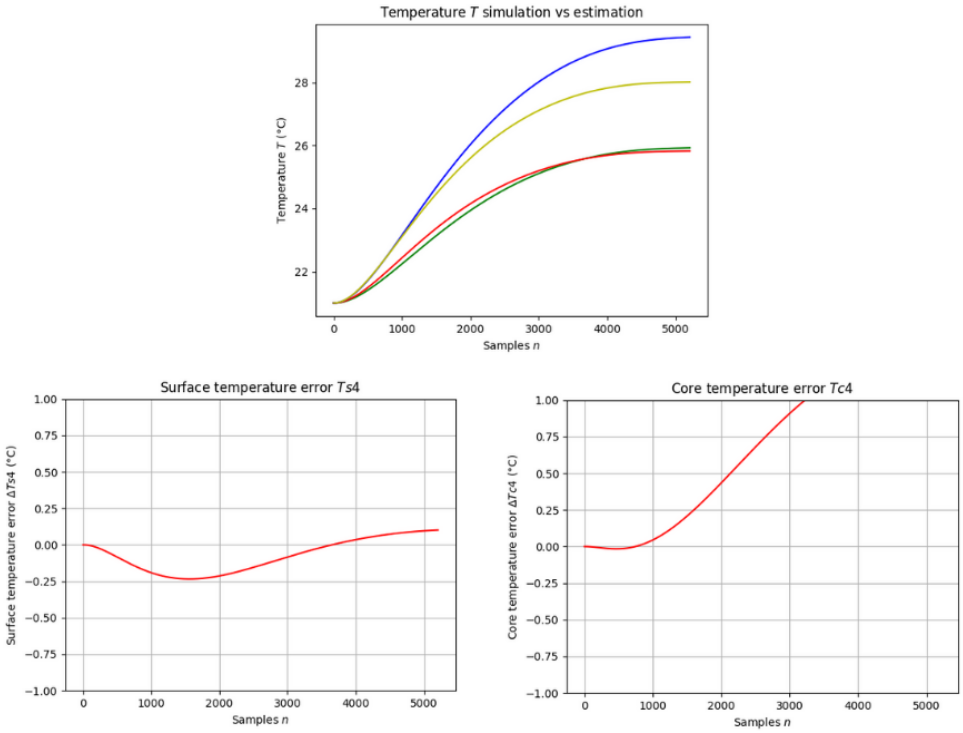


Figure 30 Surface temperature and error graphs with 2 sensors

The surface and core temperature error profiles for cell 4 remain within ±1 °C throughout the entire simulation, but their magnitudes have increased compared to the previous two configurations. In Table 9, surface temperature errors remain acceptably low (under 0.12 ºC). However, core temperature errors increase to an average of ~0.72 ºC. Even though, these values are still within the ± 1 ºC safety margin. The last samples as seen in the graph show a larger difference, which represents a more critical deviation under reduced observability.

Although EKF keeps acceptable performance with only two sensors, the increased core temperature error suggests a practical lower bound. To ensure robust estimation in real-world applications, particularly under varying ambient conditions or dynamic drive cycles, either more sensors or additional process-noise tuning may be necessary.

Cell	2	3	4	6	7
Surface error (ºC)	0.118	0.118	0.113	0.109	0.117
Core error (ºC)	0.754	0.705	0.698	0.680	0.753

Table 9 Mean absolute error values with 2 sensors

5 CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

This work has successfully developed and validated an electro-thermal model for lithium-ion battery cells, capable of estimating both surface and core temperatures using a reduced number of physical sensors.

Despite the limitations imposed by available laboratory instrumentation, such as the absence of internal thermocouples, constrained resolution of surface temperature measurements, and the lack of experimentally measured voltage profiles for precise electrical parameter fitting, the simulation results show good alignment with experimental surface temperature data. The electro-thermal model has also been benchmarked against reference data from the literature, reinforcing its physical reliability across different operating conditions.

The EKF implementation demonstrates high-fidelity temperature estimation, even for battery cells without any surface or core thermal sensors. Crucially, the EKF achieves accurate reconstruction of core temperatures by relying solely on a minimal set of surface sensors. Results show that with four or three surface sensors across a seven-cell battery pack, both surface and core temperature errors remain well below the ±1 °C industrial safety margin. When using only two sensors, estimation accuracy approaches this threshold, suggesting that there is a minimum viable configuration for safety-critical systems, which depend on the number of cells. Results in this work suggest that at least one sensor is required for each 3-4 cells.

These findings directly contribute to the scalability and cost-effectiveness of BMS architectures, allowing significant reduction in sensor count without compromising estimation quality. Fewer sensors also improve system robustness by reducing wiring complexity and minimizing points of failure, while simplifying integration in large-scale EV battery packs.

In conclusion, this work presents a validated and computationally efficient methodology for thermal state estimation in lithium-ion batteries, addressing key challenges in real-time monitoring, cost reduction, and thermal safety.

5.2 Future Work

To further enhance the performance and applicability of the proposed framework, future work could consider the following directions:

- **Incorporation of laboratory voltage-response data**, enabling robust electrical parameter identification using the genetic algorithm, which has not been possible in this work due to lack of high-resolution experimental waveforms.

- **Improved instrumentation**, i.e., deployment of high-precision thermal sensors and a controlled thermal environment (e.g., climate chamber) to increase the accuracy of both model calibration and EKF validation.

- **Online parameter adaptation**: Implementing real-time retraining of thermal and electrical parameters (e.g., via RLS or adaptive KF variants) to account for battery aging, temperature effects, and dynamic operating conditions.

- **Dynamic drive cycles**: Expanding the simulation and estimation framework to cover non-constant current profiles, typical in EV use cases, including fast charging and regenerative braking.

*All code used in this project is kept in this GitHub repository:
https://github.com/jenskjla/Temperature-estimation-battery-pack.git

BIBLIOGRAPHY

[1] Sugumar, Hemavathi & Srirama, Srinivas & Prakash, A.. (2023). *Present and Future Generation of Secondary Batteries: A Review*.

[2] Luo Y.; Chen L.; Zhang M. (2023) "Thermal Behavior Modeling of Lithium-Ion Batteries: A Comprehensive Review." *Symmetry*, 15, 1597.

[3] González-Menéndez R. *et al.* (2021) "Comparative Study of Equivalent-Circuit Model Performance in Four Li-ion Chemistries." *Batteries*, 7(3), 51.

[4] Li Y. *et al.* (2025) "Lithium-Ion Battery Thermal Modelling and Characterisation." *Journal of Energy Storage*, 66, 109565.

[5] Liang B.; Wang D.; Ren G. (2021) "Distributed Thermal Monitoring of Li-Ion Batteries with Optical Fibre Sensors." *Journal of Energy Storage*, 43, 102882.

[6] Dai H, Zhu L, Zhu J, Wei X, Sun Z. Adaptive Kalman filtering-based internal temperature estimation with an equivalent electrical-network thermal model for hard-cased batteries. *Journal of Power Sources.* 2015;293:351-365..

[7] Zhang H.; Wang P.; Li Q. (2023) "Online Core Temperature Estimation via an Aging-Integrated ECM-1D EKF." *Batteries*, 11(4), 160.

[8] Su G.; Li S. (2020) "Thermal-Parameter Identification of Li-Ion Batteries Using RLS." *Energies*, 13(9), 2281.

[9] Guo F. *et al.* (2024) "Battery Core Temperature Estimation with Numerical-Model–LSTM Fusion." *Journal of Energy Storage*, 60, 109023.

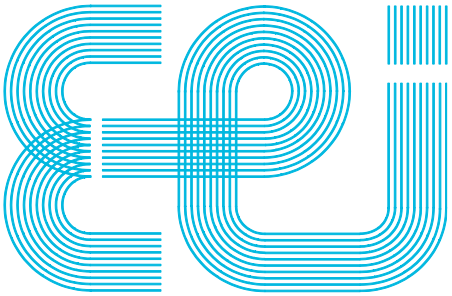
[10] Surya S, Chhetri A, Rao V, Krishna S. Kalman Filter—Machine learning fusion for core temperature estimation in Li-ion batteries. *Journal of Energy Storage.* 2025;113:115656.

[11] Zhang Y.; Tan C.; Liu K. (2023) "Physics-Informed Neural Networks for Battery Thermal Prediction." *Energy AI*, 13, 100182.

[12] Chen, L., Hu, M., Cao, K., Li, S., Su, Z., Jin, G., & Fu, C., "Core temperature estimation based on electro-thermal model of lithium-ion batteries," *International Journal of Energy Research*, vol. 44, no. 7, pp. 5320–5333, 2020.

[13] https://github.com/ezstoltz/genetic-algorithm

[14] Aksoy S, Mühürcü A, Kızmaz H. State and Parameter Estimation in Induction Motor Using the Extended Kalman Filtering Algorithm. *Proceedings of the Modern Electric Power Systems Conference (MEPS'10)*, Wrocław, Poland; 2010.



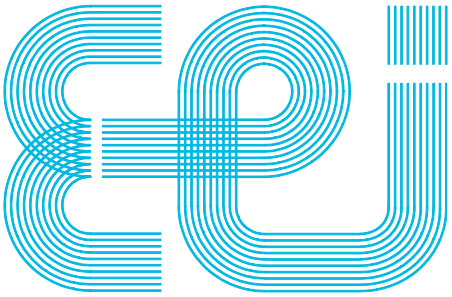
Escuela de Ingeniería Industrial

BACHELOR'S THESIS

Core and surface temperature estimation of large-scale battery pack with reduced sensors

Bachelor degree in Industrial Electronics and Automation Engineering

Universidade Vigo



Escuela de Ingeniería Industrial

BACHELOR'S THESIS

Core and surface temperature estimation of large-scale battery pack with reduced sensors

Bachelor degree in Industrial Electronics and Automation Engineering

Document

TECHNICAL SPECIFICATIONS DOCUMENT

Universidad de Vigo

CONTENT

1 GENERAL PROVISIONS

1.1 Purpose of the Document

The purpose of this document is to define the technical requirements necessary for the correct assembly, installation, and operation of the thermal estimation system developed. This system was created as part of the Bachelor's Thesis in Industrial Electronics and Automation Engineering made by Jens Kjaersgaard Larrañaga and is aimed at enabling accurate estimation of internal temperatures in lithium-ion battery packs with minimal sensor use.

1.2 Description and Scope

This project presents an experimental temperature estimation framework applied to large-scale lithium-ion battery packs. It integrates a reduced number of surface temperature sensors with electro-thermal modeling and Extended Kalman Filtering (EKF) for core and surface temperature estimation. It is not designed for immediate industrial deployment, but rather for academic and experimental analysis to serve as a foundation for future developments in Battery Management Systems (BMS).

The results presented are limited to the specific configuration, battery chemistry, and laboratory conditions tested. The system does not guarantee universal applicability beyond the experimental scope.

1.3 General Requirements fort Materials and Components

All materials and components used in this system must comply with current technical regulations and safety standards applicable to low-voltage electronic systems. Where specific brands or models are referenced, they serve as quality benchmarks and may be substituted by functionally equivalent alternatives that meet or exceed the same technical specifications.

1.4 Applicable Standards

Regardless of any specific technical regulations that are mandatory by law, the following general standards and directives must be followed at all times during the installation and operation of the system:

- **Royal Decree 842/2002, of August 2**, approving the Low Voltage Electrotechnical Regulation (Reglamento Electrotécnico para Baja Tensión).
- **Royal Decree 486/1997, of April 14**, establishing the minimum health and safety requirements in workplaces (published in the BOE on 23/04/1997).
- **Royal Decree 614/2001, of June 8**, on minimum provisions for the protection of workers' health and safety against electrical risk (published in the BOE on 21/06/2001). Encuadernación

.

2 TECHNICAL RESPONSIBILITIES

2.1 Responsibilities

The supervising engineer is responsible for ensuring correct integration and configuration of all electrical and thermal modeling subsystems, including temperature sensor placement, power supplies, and data acquisition systems.

The user or experimenter is responsible for operating the system in accordance with safety instructions outlined in this document, especially under experimental voltage and temperature conditions.

2.2 Subapartado

Before system assembly, the installer or user must confirm in writing that the documentation provided is complete and understandable. If clarification is needed, the supervisor responsible must be contacted for further explanation.

2.3 Health and Safety Plan

The user undertakes to adhere to all applicable occupational safety standards during any handling or testing of the system, including the use of thermal and electrical equipment in the laboratory environment.

2.4 Modifications and Interpretations

Any modification, clarification, or interpretation of this document must be issued in writing by the project supervisor. Such instructions must be acknowledged in writing by the user or installer before implementation.

2.5 Warranty Period

Since this is an experimental system, no commercial warranty applies. However, the author commits to maintaining the code and estimation scripts for correction or improvement upon request during the academic evaluation period.

3 BUDGETARY PROVISIONS

3.1 Budget estimation

The budget associated with this project includes only the experimental equipment, software licenses (if any), and the time invested in development tasks such as simulation, programming, and documentation.

The total amount can be seen in further document (Budget).

3.2 Testing Expenses

All testing was conducted within university facilities, and no additional cost was incurred. Repeated tests and recalibrations were done under academic supervision as part of the Bachelor's Thesis.

4 TECHNICAL CONDITIONS FOR SYSTEM IMPLEMENTATION

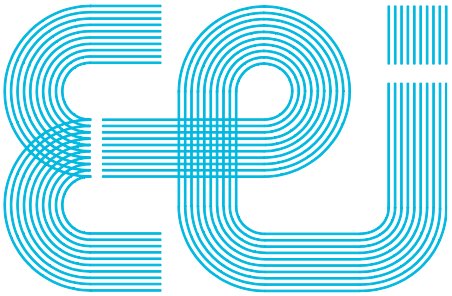
4.1 Safety Measures

The following safety precautions must be observed:

- Electrical connections must be made with the power supply turned off and verified using appropriate testing instruments.
- All test equipment must be grounded or operate at safe voltages.
- Operators should avoid metallic objects during testing and wear appropriate insulating footwear.

4.2 System Control

Before use, all components must be reviewed by the project supervisor. Defective or unapproved materials must be replaced. All models and simulation outputs must be validated against expected theoretical behavior or benchmarked against experimental results.



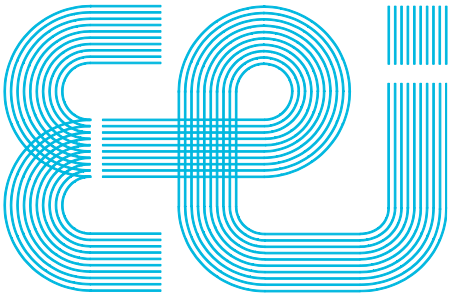
Escuela de Ingeniería Industrial

BACHELOR'S THESIS

Core and surface temperature estimation of large-scale battery pack with reduced sensors

Bachelor degree in Industrial Electronics and Automation Engineering

Universidade Vigo



Escuela de Ingeniería Industrial

BACHELOR'S THESIS

Core and surface temperature estimation of large-scale battery pack with reduced sensors

Bachelor degree in Industrial Electronics and Automation Engineering

Document

BUDGET

Universidad de Vigo

Item	Units	Unit Price (€)	Subtotal (€)
COMPONENTS			
Cylindrical Li-Ion Batteries (18650)	7	5.00	35.00
Flir E6 Pro Thermal Camera	1	1,899.00	1,899.00
DC Power Supply	1	400.00	400.00
DC Electronic Load	1	300.00	300.00
HP Pavilion Laptop 15-eg2xxx	1	650.00	650.00
Total		3284.00	
SOFTWARE LICENSES			
MATLAB	Academic License	-	
Python Environment	Free	-	
Total		-	
LABOR COST			
Task Description	**Estimated Hours**	**Rate (€/hour)**	**Subtotal (€)**
Programming	200.00	25.00	5000.00
Documentation	50.00	20.00	1000.00
Total		6000.00	
TOTAL (€)			
Components		3284.00	
Software Licenses		-	
Labor Cost		6000.00	
Total		9284.00	