Comparison of MATLAB Scripts `tms_1.m`, `tms_2.m`, and `tms_3.m` for Battery Cooling Simulation

Matriculation Number: 5014405

May 14, 2025

Contents

1 Introduction

This document compares the MATLAB scripts `tms_1.m`, `tms_2.m`, and `tms_3.m`, which simulate the thermal behavior of a 48V, 4kWh lithium-ion battery pack with a liquid cooling system. Each script builds on its predecessor, introducing new features and improvements to enhance functionality, accuracy, and usability. Additionally, this write-up explains the decision to use MATLAB's numerical programming approach (Euler or RK4 solvers) instead of Simulink or Simscape models, driven by the availability of academic licenses and partial license limitations for Simscape.

2 Overview of the Scripts

All three scripts model a lithium-ion battery pack with a liquid cooling plate, aiming to maintain cell temperatures near 35°C under various discharge conditions. They share a common goal of simulating thermal dynamics but differ in complexity, solver methods, control strategies, and user interfaces. Below is a detailed comparison of their features, followed by a discussion of the license-driven implementation choice.

3 Comparison of Explanatory Documents

3.1 `tms_1.m`: Foundational Model

The explanatory document for `tms_1.m` (TMS_1.pdf) describes a baseline MATLAB script for simulating a 13s10p battery pack under a constant 2C discharge rate. Key features include:

- **Discharge Profile**: Fixed at 2C (171A total, 17.1A per cell), generating approximately 799.2W of heat per cell via $I^2R + 0.5$ W entropic heating.
- **Solver**: Supports Simulink (if licensed) with the `ode23t` solver or a numerical Euler method (0.01s step size) for systems without Simulink.
- **Cooling System**: Uses a liquid cooling plate (0.2 m^2 surface area, 800 W/m^2-K convective coefficient) with proportional control (75 W/m^2-K/°C gain) and a 30s coolant delay.
- **Ambient Conditions**: Nominal ambient temperature at 35°C with ±0.5°C Gaussian noise; coolant inlet at 30°C.
- **Output**: Command-line results (maximum and final temperatures, stability checks) and a traditional `figure` plot showing cell temperature, coolant, ambient, and 45°C limit lines.
- **Limitations**: Lacks a dynamic discharge profile, uses a simple Euler solver (less accurate), and has no graphical user interface (GUI).

The document emphasizes the script's simplicity and compatibility, making it suitable for users with or without Simulink. It includes troubleshooting notes for adjusting cooling parameters (e.g., increasing `h_conv_base` or reducing `tau_delay`) and checking MATLAB's configuration.

3.2 `tms_2.m`: Enhanced Modeling

The explanatory document for `tms_2.m` (TMS_2.pdf) details improvements over `tms_1.m`, introducing a more realistic and complex simulation. Key enhancements include:

- **Discharge Profile**: Dynamic drive cycle with 25 segments (0–3600s) and C-rates varying from 0.5C to 3C, interpolating heat generation for a realistic load.

- **Solver**: Replaces Euler with a Runge-Kutta 4 (RK4) solver for higher numerical accuracy, alongside Simulink support with `ode23t`.
- **Cooling System**: Upgrades to PID control (k_P = 120, k_I = 1.5, k_D = 25) for tighter temperature tracking, increases `h_conv_base` to 1000 W/m^2-K, reduces `R_thermal` to 0.02 K/W, and shortens `tau_delay` to 10s. Adds coolant flow (0.02 kg/s) and specific heat (3500 J/kg-K) to calculate outlet temperature.
- **Ambient Conditions**: Supports nominal (35°C) and worst-case (40°C) ambient temperatures, with increased noise (±1°C) and a 60°C thermal runaway threshold.
- **Battery Parameters**: Reduces `cell_resistance` to 7 mOhm with ±10% variation across cells (seeded for reproducibility) and lowers `T_coolant_in` to 24°C.
- **Output**: Enhanced command-line results with runaway warnings and a `figure` plot including a 60°C runaway threshold line. Still lacks a GUI.
- **Limitations**: No GUI, requiring users to edit code directly for parameter changes.

The document highlights the script's robustness, with RK4 and PID control improving accuracy and stability. Troubleshooting notes focus on tuning PID parameters and verifying the RK4 implementation.

3.3 `tms_3.m`: User-Friendly Interface

The explanatory document for `tms_3.m` (TMS_3.pdf) describes the most advanced script, building on `tms_2.m` by adding a GUI and dialog-based output. Key features include:

- **Discharge Profile**: Retains `tms_2.m`'s dynamic drive cycle, with C-rates input via a GUI text field (comma-separated values).
- **Solver**: Continues using RK4 for numerical simulations, with no Simulink dependency in the provided code (reflecting license constraints).
- **Cooling System**: Inherits `tms_2.m`'s PID control and cooling parameters, but adds a coolant type dropdown (Water-Glycol 50/50, Water, Ethylene Glycol) that adjusts `Cp_coolant` and `h_conv_base` dynamically based on pack size and discharge rate.
- **Ambient Conditions**: Supports nominal and worst-case ambient inputs via GUI fields, maintaining `tms_2.m`'s 60°C runaway threshold.
- **GUI**: Uses `uifigure` (MATLAB R2016a+) with a 600x500 pixel window, split into:
  - **Left Panel**: Inputs for series/parallel cells, voltage, capacity, and C-rates.
  - **Right Panel**: Inputs for ambient temperatures and coolant type, with a "Run Simulation" button.
- **Output**: Displays results in an 800x500 `uifigure` dialog with a `uitextarea` for text (temperatures, stability) and a `uiaxes` for plotting, eliminating extra figure windows. Errors are shown in a separate dialog.
- **Limitations**: Troubleshooting notes incorrectly reference `tms_2.m` instead of `tms_3.m`, a minor documentation error.

The document emphasizes the GUI's usability, making the script accessible to users without coding expertise. Troubleshooting notes address GUI issues and parameter tuning, though the naming error is noted.

3.4 Key Differences

The progression from `tms_1.m` to `tms_3.m` reflects increasing sophistication:

- **Discharge Complexity**: `tms_1.m` uses a constant 2C rate, while `tms_2.m` and `tms_3.m` implement a dynamic 25-segment drive cycle (0.5C–3C), better mimicking real-world loads.
- **Solver Accuracy**: `tms_1.m`'s Euler method is replaced by RK4 in `tms_2.m` and `tms_3.m`, improving numerical stability and precision.
- **Cooling Control**: `tms_1.m` uses proportional control, upgraded to PID in `tms_2.m` and `tms_3.m` for better temperature regulation. `tms_3.m` adds coolant type selection.
- **Ambient Modeling**: `tms_1.m` models only nominal conditions (35°C), while `tms_2.m` and `tms_3.m` include worst-case (40°C) and runaway thresholds (60°C).
- **User Interface**: `tms_1.m` and `tms_2.m` rely on command-line interaction and traditional plots, whereas `tms_3.m` introduces a GUI and dialog-based output for enhanced usability.
- **Parameter Flexibility**: `tms_3.m` allows runtime configuration of battery and cooling parameters via the GUI, unlike the hardcoded values in `tms_1.m` and `tms_2.m`.
- **Documentation Error**: Only `tms_3.m`'s troubleshooting notes contain a naming inconsistency (referencing `tms_2.m`).

4 Rationale for Using MATLAB Numerical Programming

The scripts were implemented using MATLAB's numerical programming approach (Euler in `tms_1.m`, RK4 in `tms_2.m` and `tms_3.m`) instead of Simulink or Simscape models due to academic license constraints:

- **MATLAB License Availability**: The academic license provides full access to core MATLAB functionality, enabling the development of numerical solvers like Euler and RK4. These solvers are implemented directly in MATLAB code, ensuring compatibility across all licensed systems.
- **Simulink License Limitations**: While Simulink is supported in the scripts (checked via `license('test', 'Simulink')`), its availability is not guaranteed in all academic environments. To ensure portability, the scripts include numerical fallbacks, which were prioritized in `tms_3.m`'s implementation (evident from the absence of Simulink-specific code in the simulation function).
- **Simscape Partial License Issue**: The scripts check for Simscape (`license('test', 'Simscape')`), but partial license availability prevents reliable use of Simscape blocks (e.g., thermal or electrical components). Simscape's dependency on specialized toolboxes, which may not be included in standard academic licenses, led to its exclusion. Instead, thermal dynamics are modeled using first-principles equations (e.g., $\frac{dT}{dt} = \frac{Q_{\text{heat}} - Q_{\text{cool}}}{mCp}$).

By focusing on numerical methods, the scripts remain accessible to users with only a core MATLAB license, avoiding reliance on Simulink's graphical modeling or Simscape's physical modeling capabilities. The RK4 solver in `tms_2.m` and `tms_3.m` provides sufficient accuracy for thermal simulations, while the GUI in `tms_3.m` enhances usability without requiring additional toolboxes.

5 Conclusion

The tms_1.m, tms_2.m, and tms_3.m scripts represent a progression from a basic, constant-load thermal simulation to a sophisticated, user-friendly tool with dynamic discharge and GUI support. tms_1.m establishes a simple foundation, tms_2.m enhances realism with RK4 and PID control, and tms_3.m prioritizes accessibility with a GUI. The choice of numerical programming over Simulink and Simscape ensures compatibility with academic license constraints, making the scripts widely usable while maintaining robust thermal modeling capabilities. Future improvements could address the documentation error in tms_3.m and explore additional coolant types or discharge profiles within the numerical framework.