Explanatory Document for `tms_2.m`: Battery Cooling Simulation

Matriculation Number : 5014405

May 14, 2025

Contents

1 Introduction

The `tms_2.m` script enhances `tms_1.m` by introducing a dynamic discharge profile (variable C-rates), PID control for cooling, and support for nominal (35°C) and worst-case (40°C) ambient conditions. It uses an RK4 solver instead of Euler for improved numerical accuracy and includes a Simulink fallback. The script models a 48V, 4kWh lithium-ion battery pack (13s10p) with a liquid cooling plate. This document explains each code section, highlighting improvements over `tms_1.m`.

2 Code Structure

The script is organized into:

- License and Environment Check
- Battery Pack Parameters
- Electrical Load Profile
- Cooling System Parameters
- Simulation (Nominal and Worst-Case)
- Results
- Plotting
- Local Function (RK4)
- Troubleshooting Notes

3 License and Environment Check

```
clear; clc; close all;

matlab_ver = ver;
simscape_licensed = license('test', 'Simscape');
simulink_licensed = license('test', 'Simulink');
fprintf('MATLAB Version: %s\n', matlab_ver(1).Version);
...

use_simulink = simulink_licensed;
if use_simulink
    try
        add_block('simulink/Continuous/Integrator', 'test/Integrator');
        ...
    catch e
        warning('Simulink Integrator block unavailable: %s. Using
            numerical solver.', e.message);
        use_simulink = false;
    end
else
    fprintf('Simulink not licensed. Using numerical ODE solver.\n');
end
```

Similar to `tms_1.m`, this section clears the workspace and checks for MATLAB, Simscape, and Simulink licenses. It tests the Simulink Integrator block (noting a change to `Continuous/Integrator` from `Commonly Used Blocks/Integrator`) and sets `use_simulink` accordingly. The check ensures the script can run with or without Simulink.

4 Battery Pack Parameters

```
cell_voltage = 3.6; % Nominal voltage (V)
cell_capacity = 8.55; % Capacity (Ah, adjusted for ~4kWh)
cell_resistance_base = 0.007; % Base internal resistance (Ohm, 7 mOhm)
...

num_series = 13; num_parallel = 10;
pack_voltage = num_series * cell_voltage; % 46.8V
pack_capacity = num_parallel * cell_capacity; % 85.5Ah
...

rng(42);
cell_resistance = cell_resistance_base * (1 + 0.1 * (2 *
    rand(num_series, num_parallel) - 1));
avg_cell_resistance = mean(cell_resistance(:));

T_ambient_base = 35; T_ambient_worst = 40; T_max_limit = 45; T_runaway
    = 60;
T_coolant_in = 24; T_initial = 25;
```

This section refines `tms_1.m` by: - Reducing `cell_resistance` to 7 mOhm (more realistic) with ±10% variation across cells, seeded for reproducibility. - Introducing a worst-case ambient temperature (40°C) and a thermal runaway threshold (60°C). - Lowering `T_coolant_in` to 24°C for stronger cooling.

The pack remains 13s10p, yielding 4kWh.

5 Electrical Load Profile

```
time_segments = [0 150 300 ... 3600];
c_rates = [0.5 2 1 3 0.5 ... 0.5]; % Aggressive cycle
current_total = c_rates * pack_capacity;
current_cell = current_total / num_parallel;

heat_gen_cell = zeros(1, length(c_rates));
for i = 1:length(c_rates)
    heat_gen_cell(i) = (current_cell(i)^2) * avg_cell_resistance + 0.5;
end
total_heat = heat_gen_cell * num_series * num_parallel;
```

Unlike `tms_1.m`'s constant 2C discharge, this section defines a dynamic drive cycle with 25 segments (0–3600s) and C-rates varying between 0.5C and 3C. Heat generation is computed per segment using $I^2R + 0.5$ W entropic heating, scaled for the entire pack. This models a more realistic, aggressive load.

6 Cooling System Parameters

```
pipe_surface_area = 0.2; h_conv_base = 1000; R_thermal = 0.02;
    tau_delay = 10;
m_dot = 0.02; Cp_coolant = 3500;
```

```

mass_factor = 1.2; ambient_noise_std = 1.0;

k_P = 120; k_I = 1.5; k_D = 25;
```

Improvements over `tms_1.m` include: - Increased `h_conv_base` (1000 W/m²-K) and reduced `R_thermal` (0.02 K/W) for better cooling. - Reduced `tau_delay` (10s) for faster response. - Added coolant flow (`m_dot` = 0.02 kg/s) and specific heat (`Cp_coolant` = 3500 J/kg-K) for outlet temperature calculation. - Replaced proportional control with PID (`k_P`, `k_I`, `k_D`) for tighter temperature tracking. - Increased `ambient_noise_std` to ±1°C.

These changes enhance cooling realism and control.

7 Simulation (Nominal Case)

```
mCp = mass_factor * cell_mass * num_series * num_parallel *
    cell_specific_heat;
T_max = NaN; T_final = NaN; T_coolant_out = NaN;
temp_profile = []; time_data = []; runaway_flag = false;

if use_simulink
    model_name = 'BatteryCoolingModel';
    ...
    set_param(model_name, 'StopTime', '3600', 'Solver', 'ode23t', ...);
    try
        sim(model_name);
        temp_var = evalin('base', 'TempData');
        ...
        T_coolant_out = T_coolant_in + Q_cool_avg / (m_dot *
            Cp_coolant);
    catch e
        ...
    end
else
    t = 0:0.01:3600; dt = 0.01;
    T = zeros(size(t)); T(1) = T_initial;
    Q_cool_delayed = 0; integral_error = 0; prev_error = 0;
    T_coolant_out_sum = 0;
    for j = 2:length(t)
        total_heat_t = interp1(time_segments, total_heat, t(j-1),
            'previous');
        T_ambient = T_ambient_base + ambient_noise_std * randn();
        k1 = thermal_dynamics(...);
        k2 = thermal_dynamics(...);
        k3 = thermal_dynamics(...);
        k4 = thermal_dynamics(...);
        T(j) = T(j-1) + (dt/6) * (k1 + 2*k2 + 2*k3 + k4);
        ...
        T_coolant_out_sum = T_coolant_out_sum + Q_cool;
    end
    T_coolant_out = T_coolant_in + (T_coolant_out_sum / length(t)) /
        (m_dot * Cp_coolant);
end
```

The nominal case (35°C ambient) is simulated using: - **Simulink**: A model with PID control, coolant delay, and nonlinear cooling, similar to `tms_1.m` but with enhanced blocks (e.g., PID

Controller). It calculates coolant outlet temperature. - **RK4 Solver**: Replaces Euler with RK4 for higher accuracy, using four slope estimates per step. Heat is interpolated from the drive cycle, and PID control adjusts cooling dynamically.

The RK4 solver is a significant upgrade, improving stability and precision.

8 Simulation (Worst-Case)

```
T_max_worst = NaN; T_final_worst = NaN; runaway_flag_worst = false;
if use_simulink
    ...
    set_param([model_name '/AmbientBase'], 'Value',
        num2str(T_ambient_worst));
    ...
else
    t = 0:0.01:3600; dt = 0.01;
    T = zeros(size(t)); T(1) = T_initial;
    ...
    T_ambient = T_ambient_worst + ambient_noise_std * randn();
    ...
end
```

The worst-case simulation (40°C ambient) reuses the nominal case logic, updating only the ambient temperature. It ensures the cooling system can handle harsher conditions.

9 Results

```
fprintf('\nNominal Case (Ambient = %.0f C):\n', T_ambient_base);
...
if T_max <= T_max_limit && ~isnan(T_max)
    ...
end
if runaway_flag
    warning('Thermal runaway risk detected: Temperature exceeded
        %.0f C.', T_runaway);
end
```

Results are printed for both cases, including maximum and final temperatures, coolant temperatures, and stability checks (within 45°C and ±1°C of ambient). A runaway flag warns if temperatures exceed 60°C.

10 Plotting

```
figure('Name', 'Temperature Profile');
plot(time_data, temp_profile, 'b-', 'LineWidth', 2, ...);
...
plot([0 3600], [60 60], 'k:', 'LineWidth', 1.5, 'DisplayName',
    'Runaway Threshold (60 C)');
```

The plot is similar to `tms_1.m` but adds a 60°C runaway threshold line and focuses on the nominal case. It uses a traditional `figure` instead of a GUI.

11 Local Function (RK4)

```
function dT = thermal_dynamics(T, t, total_heat_t, T_ambient, T_coolant_in, ...
        h_conv_base, pipe_surface_area, R_thermal, k_P, k_I, k_D, ...
        integral_error, prev_error, Q_cool_delayed, dt, tau_delay, mCp)
    ...
    dT = (total_heat_t - Q_cool_delayed) / mCp;
end
```

The `thermal_dynamics` function computes the temperature derivative for RK4, incorporating PID control, nonlinear cooling, and coolant delay. It's more complex than `tms_1.m`'s Euler-based dynamics.

12 Troubleshooting Notes

```
% If cell temperature does not stabilize near ambient (35 C     1 C or 40 C     1 C ):
% 1. Increase h_conv_base (e.g., 1100 W/m^2-K) or pipe_surface_area (e.g., 0.3 m^2).
% ...
% 12. Ensure file is named correctly (e.g., 'tms_2.m').
```

These notes suggest tuning cooling parameters and checking the environment, with a focus on RK4 and PID adjustments.

13 Summary

The `tms_2.m` script improves `tms_1.m` with a dynamic drive cycle, RK4 solver, PID control, and dual ambient conditions. It lacks a GUI, which is added in `tms_3.m`, but provides robust thermal modeling.