Explanatory Document for `tms_3.m`: Battery Cooling Simulation with GUI

MAtriculation Number : 5014405

May 14, 2025

Contents

1 Introduction

The `tms_3.m` script is the most advanced version, building on `tms_2.m` by adding a graphical user interface (GUI) using MATLAB's `uifigure`. It simulates a 48V, 4kWh lithium-ion battery pack (configurable) with a liquid cooling plate, supporting dynamic discharge profiles and multiple coolant types. The script uses an RK4 solver, displays results and plots in a separate dialog, and ensures no extra figure windows. This document explains each code section, emphasizing the GUI and output enhancements.

2 Code Structure

The script is organized into:

- GUI Creation
- Simulation Function
- Local Function (RK4)
- Troubleshooting Notes

3 GUI Creation

```
fig = uifigure('Name', 'Battery Cooling Simulation', ...
    'Position', [100, 100, 600, 500], ...
    'Color', [0.95 0.95 0.95]);

grid = uigridlayout(fig, [1, 2], ...);

panel_left = uipanel(grid, 'Title', 'Battery Configuration', ...);
...
edit_num_series = uieditfield(grid_left, 'numeric', ...);
...
panel_right = uipanel(grid, 'Title', 'Ambient & Coolant', ...);
...
coolant_dropdown = uidropdown(grid_right, ...);
run_button = uibutton(grid_right, 'Text', 'Run Simulation', ...);
```

This section creates a modern GUI using `uifigure` (MATLAB R2016a+), sized at 600x500 pixels with a light gray background. The layout uses a 1x2 `uigridlayout` splitting the window into: - **Left Panel** (Battery Configuration): Inputs for series cells, parallel cells, nominal voltage, capacity, and C-rates (text field for comma-separated values). - **Right Panel** (Ambient Coolant): Inputs for nominal and worst-case ambient temperatures, and a dropdown for coolant type (Water-Glycol 50/50, Water, Ethylene Glycol).

The "Run Simulation" button triggers the `run_simulation` function with input values. Tooltips and bold labels enhance usability, and the aesthetic design uses white panels and a blue button.

4 Simulation Function

```
function run_simulation(num_series, num_parallel, cell_voltage,
   cell_capacity, c_rates_str, ...
        T_ambient_base, T_ambient_worst, coolant_type)
    try
```

```
        if any([num_series, num_parallel, cell_voltage, cell_capacity] <= 0)
            error('Battery configuration, voltage, and capacity must be positive.');
        end
        c_rates = str2num(c_rates_str);
        ...
    catch e
        result_fig = uifigure('Name', 'Simulation Error', ...);
        uitextarea(result_fig, 'Value', {sprintf('Error: %s', e.message)}, ...);
        return;
    end

    cell_resistance_base = 0.007; ...
    ...
    switch coolant_type
        case 'Water-Glycol 50/50'
            Cp_coolant = 3500; h_conv_base = 1000; ...
        ...
    end
    ...
    t = 0:0.01:3600; dt = 0.01;
    ...
    result_fig = uifigure('Name', 'Simulation Results', ...);
    result_text = uitextarea(result_grid, ...);
    result_axes = uiaxes(result_grid, ...);
    ...
    plot(result_axes, time_data, temp_profile, 'b-', ...);
end
```

The `run_simulation` function is the core, handling: - **Input Validation**: Checks for positive battery parameters and valid C-rates, displaying errors in a dialog if invalid. - **Parameters**: Defines battery and cooling parameters, with coolant-specific values (e.g., `Cp_coolant`, `h_conv_base`) adjusted dynamically based on pack size and discharge rate. - **Nominal Simulation**: Uses RK4 to simulate 3600s, interpolating heat from the drive cycle and applying PID-controlled cooling. - **Worst-Case Simulation**: Repeats for 40°C ambient. - **Output**: Creates a 800x500 `uifigure` dialog with a `uitextarea` for results (max/final temperatures, coolant temperatures, stability) and a `uiaxes` for plotting the nominal case temperature profile with reference lines.

The dialog ensures results and plots are self-contained, avoiding extra windows.

5 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
```

Identical to `tms_2.m`, this function computes the temperature derivative for RK4, incorporating PID control, nonlinear cooling, and coolant delay.

6 Troubleshooting Notes

```
% If cell temperature does not stabilize near ambient (35 C     1 C
    or 40 C     1 C ):
% ...
% 9. If errors or extra windows persist, verify file is saved as
    'tms_2.m'...
% 10. Run 'which tms_2.m' to confirm correct file; delete any
    'tms_3.m' files.
```

These notes guide users on stabilizing temperatures and resolving GUI issues, but incorrectly reference `tms_2.m` instead of `tms_3.m`, a minor documentation error.

7 Summary

The `tms_3.m` script is the most user-friendly, adding a GUI and dialog-based output to `tms_2.m`'s robust modeling. It supports configurable battery packs, dynamic discharge, and multiple coolants, with clear visualization and no extra windows. The naming inconsistency in troubleshooting notes should be corrected.