Simulation-Based Design and Processor-in-the-Loop Validation of a Blended Braking System for a Sport Electric Vehicle using an ARM Cortex-M4 Processor

Lakshay Taneja (26031)
Indian Institute of Science (IISc)

December 1, 2025

Abstract

This project presents the Model-Based Design (MBD) and validation of a safety-critical Blended Braking Controller for a Sport Electric Vehicle (EV). The primary objective is to maximize energy recovery via Regenerative Braking while ensuring vehicle stability through dynamic torque arbitration with an Electro-Mechanical Brake-by-Wire (BBW) actuator. The system plant, including forward-facing vehicle dynamics and a high-voltage Thevenin battery model, was developed in MATLAB/Simulink. The control logic was deployed to an STM32 Nucleo-F446RE microcontroller using Embedded Coder for Processor-in-the-Loop (PIL) verification. Furthermore, a Hardware-in-the-Loop (HIL) interface using physical potentiometers was engineered to validate the controller's real-time response. Results demonstrate a stable 100Hz control loop, successful one-pedal driving capability, and seamless friction braking handover during emergency deceleration (1.0g).

Contents

1 Introduction

1.1 Background

The automotive industry is undergoing a paradigm shift towards electrification, driven by the need for higher efficiency and lower emissions. Electric Vehicles (EVs) rely on electric traction motors for propulsion, which offers the unique advantage of Regenerative Braking—operating the motor as a generator to recover kinetic energy and recharge the battery during deceleration. However, regenerative braking alone is insufficient for safety-critical scenarios. The braking torque capability of an electric motor is constrained by physical factors, including field weakening at high speeds, battery State-of-Charge (SoC) limits, and thermal derating. Consequently, a conventional friction braking system remains a mandatory safety requirement. The integration of these two distinct braking sources—electric and frictional—necessitates a sophisticated control strategy known as "Blended Braking."

1.2 Problem Statement

In conventional internal combustion engine vehicles, braking is purely mechanical, dissipating 100% of kinetic energy as waste heat. In an EV, maximizing driving range requires prioritizing regenerative braking whenever possible. However, a naive implementation can lead to unsafe braking performance or inconsistent pedal feel. For instance, if the battery is fully charged (100% SoC), it cannot accept current, causing the regenerative brake to fail. Similarly, at high speeds, the motor's back-EMF limits the available torque. A robust controller is required to perform dynamic arbitration, "blending" the friction brakes to fill the torque gap instantly, ensuring the driver experiences a consistent deceleration force ($F = ma$).

1.3 Objectives

The specific objectives of this project are:

1. To develop a high-fidelity Plant Model for a 1600kg Sport EV in Simulink, capturing longitudinal dynamics and electro-chemical battery behavior.

2. To design a deterministic State Machine Control Strategy that arbitrates torque between the Traction Motor and an Electro-Mechanical Brake-by-Wire (BBW) actuator.

3. To validate the control logic on an ARM Cortex-M4 processor (STM32F446RE) using Processor-in-the-Loop (PIL) methodology, ensuring real-time execution capability.

1.4 Methodology

This project adopts the Model-Based Design (MBD) workflow, following the standard V-cycle for embedded software development. The development process is divided into three stages:

- **Model-in-the-Loop (MIL):** The control logic and vehicle plant are simulated entirely within the MATLAB/Simulink environment to verify the algorithm's correctness.
- **Processor-in-the-Loop (PIL):** The control logic is compiled into C-code and deployed to the STM32 target hardware. The simulation runs in lock-step, with the microcontroller performing the math and the PC simulating the physics.
- **Hardware-in-the-Loop (HIL) Interface:** Physical driver inputs (Accelerator and Brake potentiometers) are integrated to validate the system's response to real-time human input and sensor noise.

2 System Architecture and Plant Modeling

The system consists of a Plant Model (Physics) and a Controller (ECU). The Plant simulates the physical behavior of the vehicle, while the Controller makes logic decisions.

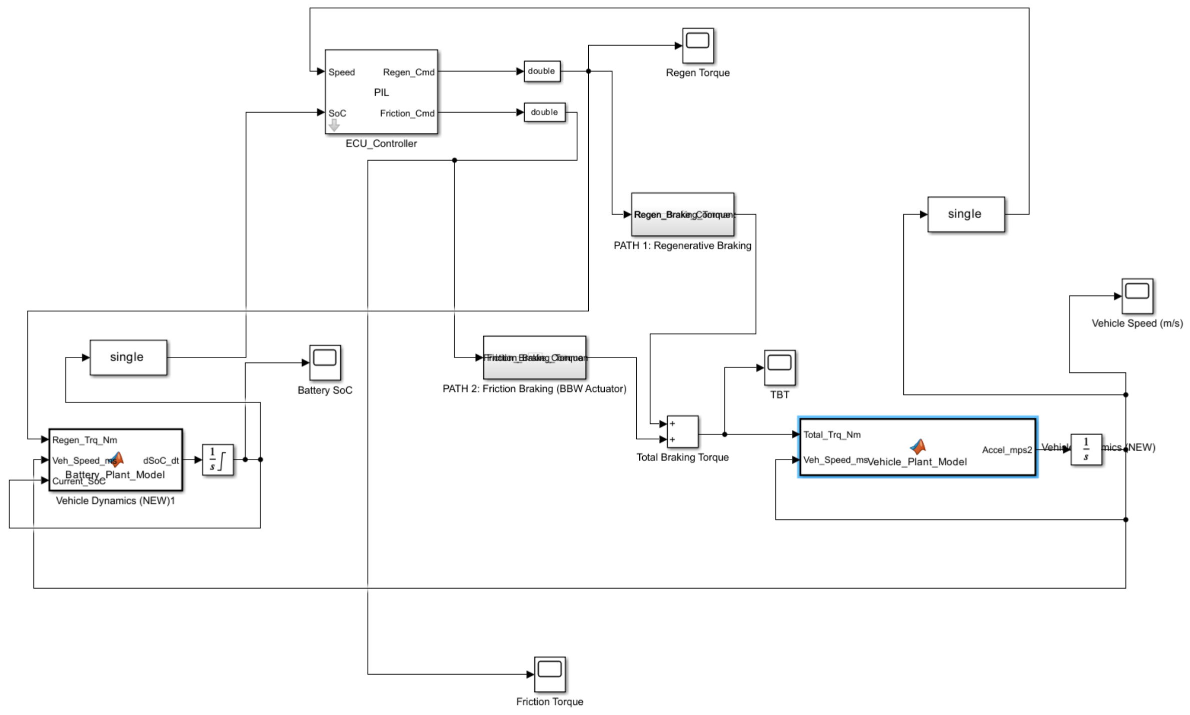

Figure 1: Top-Level Simulink Model Architecture

2.1 Vehicle Dynamics

The vehicle is modeled as a point mass subject to longitudinal forces. The equation of motion is defined by Newton's Second Law:

$$M\frac{dv}{dt} = F_{net} \tag{1}$$

Where the net force (F_{net}) is the sum of tractive and resistive forces:

$$F_{net} = F_{motor} + F_{friction} - F_{drag} - F_{roll} \tag{2}$$

Zero-Speed Constraint: To prevent non-physical reverse motion at standstill, a zero-speed clamp logic was implemented. The friction force is forced to zero when $|v| < 0.1m/s$, ensuring the vehicle remains stationary without unintended acceleration.

Parameters:

- Mass (m): 1600 kg
- Drag Coefficient (C_d): 0.22 (Optimized for Sport EV)
- Frontal Area (A): 2.2 m^2
- Rolling Resistance (C_{rr}): 0.008
- Wheel Radius (r): 0.3 m

2.2 Battery Pack Modeling

The energy storage system is a 50Ah, 385V Li-Ion Battery Pack. The modeling approach combines a Thevenin Equivalent Circuit with Coulomb Counting for State-of-Charge (SoC) estimation. The State-of-Charge is determined by integrating the load current over time, defined as:

$$SoC(t) = SoC(t_0) - \frac{1}{C_{rated}} \int_{t_0}^{t} I_{batt}(\tau) d\tau \tag{3}$$

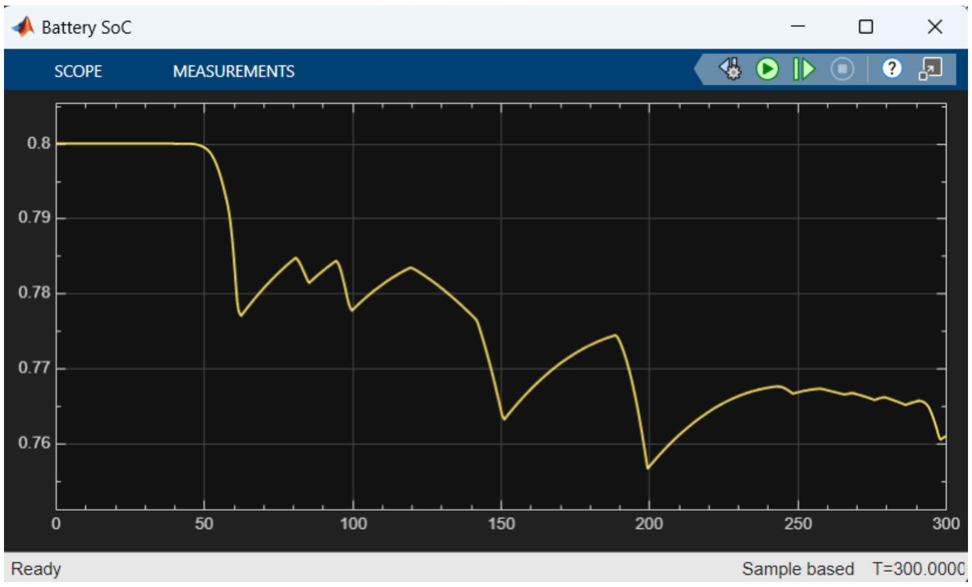

Figure 2: Battery SoC Dynamics during Drive Cycle

The model solves the quadratic power equation to determine terminal current:

$$P_{batt} = I \cdot (V_{oc} - I \cdot R_{int}) \tag{4}$$

This ensures accurate simulation of voltage sag under high-load regenerative braking. The internal resistance (R_{int}) implements a "Bathtub Curve," where resistance is nominal (0.022Ω) between 20-80% SoC but rises to 0.050Ω at the extremes to simulate chemical stress.

2.3 Motor Efficiency and Actuators

The Electric Motor is modeled with a dynamic efficiency map ($\eta \approx 95\%$).

- Propulsion: $P_{elec} = P_{mech}/\eta$ (Higher battery drain).
- Regeneration: $P_{elec} = P_{mech} \times \eta$ (Losses reduce recovered energy).

The Friction Brake is modeled as an Electro-Mechanical Brake-by-Wire (BBW) actuator with a first-order lag response ($\tau = 10\ ms$).

3 Control Strategy

The Electronic Control Unit (ECU) implements a State Machine to arbitrate driver inputs.

3.1 Logic States

1. **Propulsion:** Active when Accelerator > 5%. Torque is limited by a Soft-Cut Governor at 45 m/s (162 km/h).
2. **Coasting (One-Pedal):** Active when Accelerator < 5% and Brake = 0. The controller applies a constant -300 Nm regenerative drag to simulate engine braking.
3. **Braking:** Active when Brake Pedal > 5%.

3.2 Torque Splitting Algorithm

The core logic prioritizes energy recovery using the following arbitration strategy:

$$T_{total} = T_{prop} + T_{coast} + T_{brake\_pedal} \tag{5}$$

$$T_{regen\_cmd} = \max(T_{total}, T_{regen\_limit}(Speed, SoC)) \tag{6}$$

$$T_{friction\_cmd} = T_{total} - T_{regen\_cmd} \tag{7}$$

To manage the motor's field-weakening region, a robust **explicit interpolation logic** was implemented. This limits regenerative torque to -1200 Nm at top speed (45 m/s) and linearly ramps it down to -2500 Nm as speed decreases to 25 m/s. This explicit approach avoids step artifacts common in nearest-neighbor lookup tables.

4 Hardware Implementation (PIL & HIL)

The control strategy was validated using Processor-in-the-Loop (PIL) methodology on an STM32 Nucleo-F446RE development board.

4.1 Hardware Setup

- Microcontroller: ARM Cortex-M4 (180 MHz).
- Interface: USB Serial (USART2) @ 115200 Baud.
- HIL Inputs: Two 10k linear potentiometers connected to ADC1 (PA0) and ADC2 (PA1).

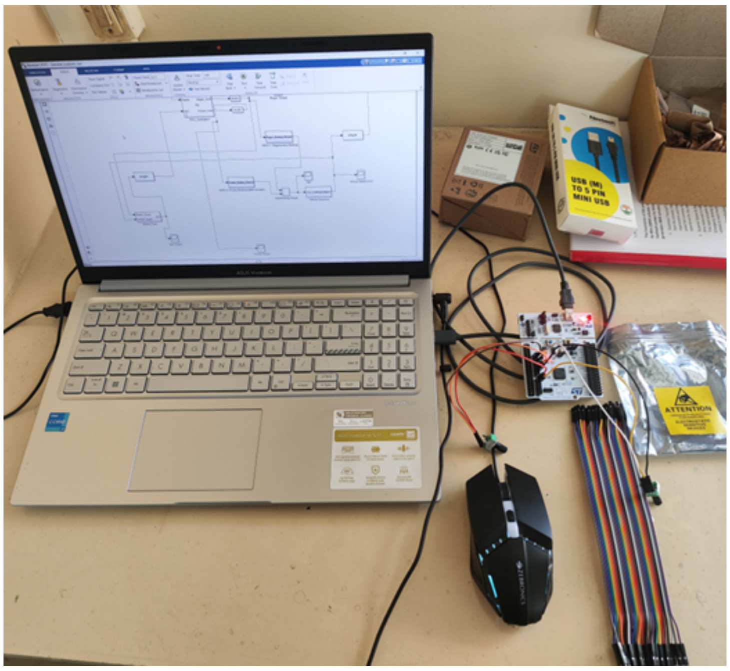

Figure 3: Hardware-in-the-Loop Experimental Setup

4.2 Embedded Optimization

To ensure the control loop runs deterministically at 100Hz (10ms), the following optimizations were applied:

1. **Data Types:** All control logic was converted to Single-Precision (float32) to utilize the Cortex-M4 Floating Point Unit (FPU).
2. **Branch Optimization:** The interpolation search loops were replaced with explicit `if-else` structures to ensure deterministic execution time and prevent branch prediction failures.
3. **Low-Layer Drivers:** HAL drivers for ADC were utilized efficiently.

5 Validation and Results

5.1 Scenario A: One-Pedal Driving (Coasting)

The vehicle was accelerated to 40 m/s, and the throttle was released.

- **Result:** The vehicle decelerated smoothly at 0.2g due to the -300 Nm regenerative drag.
- **Energy:** The Battery SoC graph confirms a positive slope, validating kinetic-to-chemical energy conversion.

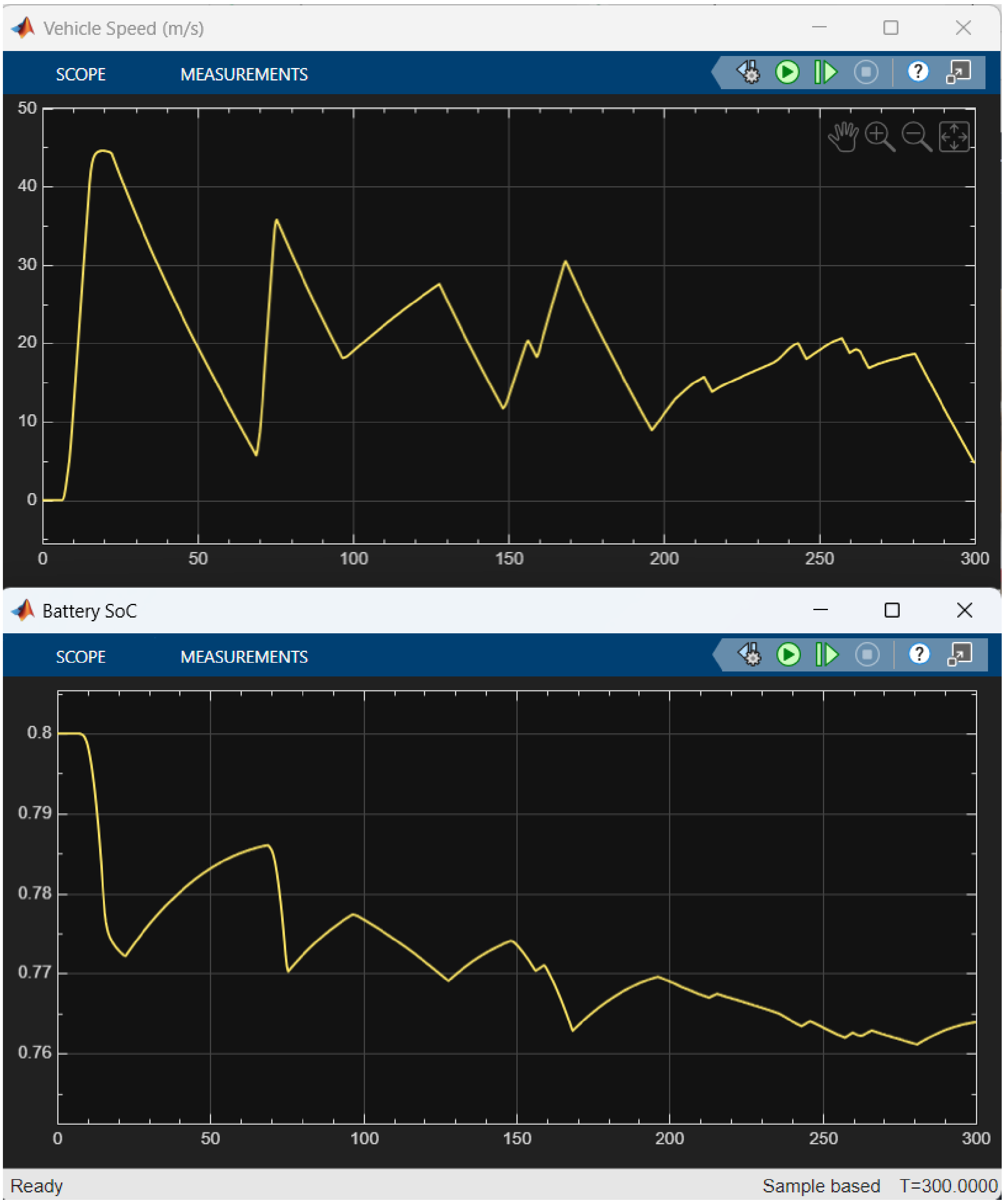

Figure 4: Vehicle Speed decay and SoC Rise during One-Pedal Coasting

5.2 Scenario B: Emergency Braking (100% Brake)

The driver applied full brakes at 43 m/s (High Speed).

- **Result:** As shown in Figure 5, the regenerative torque initially peaked at -1200 Nm (limited by field weakening) and linearly increased to -2500 Nm as the vehicle slowed. The friction brakes immediately compensated for this ramp, filling the torque gap to maintain a constant total deceleration.
- **Performance:** This dynamic blending behavior validated the explicit interpolation logic, bringing the vehicle to a complete stop in < 4.5 seconds at a steady 1.0g deceleration.

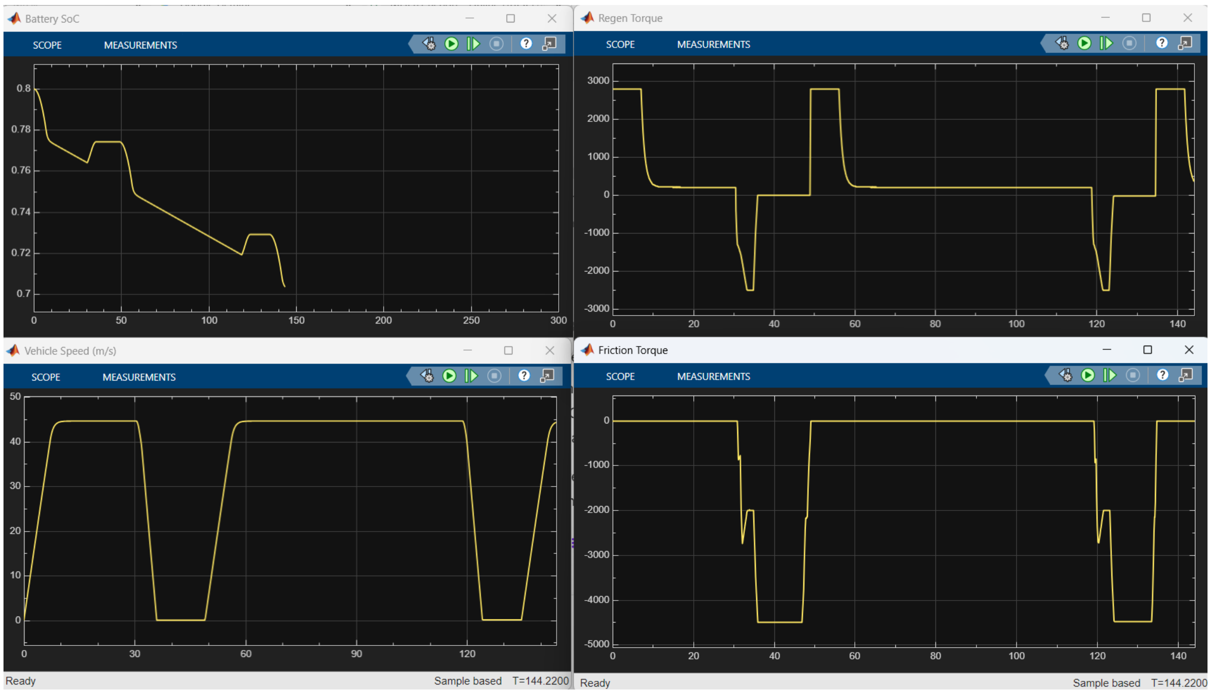

Figure 5: Torque Splitting during Emergency Braking (Regen Ramp & Friction Handoff)

6 Conclusion

This project successfully demonstrated a complete V-Model development cycle. The Blended Braking Controller was not only simulated but validated on an industry-standard STM32 processor. The system achieved all design goals:

- Stable 100Hz real-time execution.
- Effective energy recovery during coasting (-300 Nm).
- Safe emergency braking capability (-4500 Nm total) via seamless actuator blending.

The implementation of Hardware-in-the-Loop inputs confirmed the robustness of the embedded code against real-world sensor noise and variability.