Scilab case study project on

Simplified Implementation of Regenerative Braking Control for Electric Vehicles Using PI and Sliding Mode Controllers in Scilab/Xcos

Sanyam Bhavsar

Int. M.Tech Student in Data Science and Computation VIT Bhopal University

Control System / Electric Vehicle Simulation

18-06-2026

Abstract

Regenerative braking has become an important feature in modern electric vehicles (EV). Unlike conventional braking systems that waste kinetic energy in the form of heat, regenerative braking systems convert a portion of this energy into electrical energy and store it back in the battery. This project aims to design and simulate a regenerative braking system for an electric vehicle using Scilab/Xcos and evaluate the performance of different control strategies.

The proposed model includes vehicle dynamics, DC motor model, battery charging through regenerative braking, and controller-based duty cycle regulation. Two control techniques, namely the Proportional-Integral (PI) controller and Sliding Mode Controller (SMC), are implemented and compared. The primary objective is to analyze their ability to regulate braking current and maximize energy recovery during vehicle deceleration.

Simulation results indicate that both controllers successfully achieve regenerative braking and battery charging. However, the Sliding Mode Controller provides better current tracking and improved energy recovery compared to the conventional PI controller. The study demonstrates that advanced control techniques can significantly enhance the efficiency of

regenerative braking systems and contribute to improved energy utilization in electric vehicles

This project provides practical exposure to control system design, electric vehicle modeling, and renewable energy recovery techniques.

This case study implements a simplified, single-loop DC-motor variant of the reference paper's core control logic to accommodate Scilab/Xcos environment constraints.

1. Introduction

The increasing demand for energy-efficient transportation has led to significant growth in the development of Electric Vehicles. One of their most valuable features is regenerative braking. Instead of wasting kinetic energy as heat – like traditional brakes do – regenerative braking captures that energy, converts it into electricity, and sends it back to the battery for later use. This makes the vehicle more efficient and extends its driving range

In this project, a regenerative braking system was modeled and simulated using the Scilab/Xcos environment. The implemented model focuses on analyzing the performance of control strategies used for regulating regenerative current and improving energy recovery. The project uses a simplified single-loop DC motor-based regenerative braking model inspired by the control concepts presented in the reference paper. Due to Scilab/Xcos modeling limitations, the detailed three-phase BLDC switching system, Hall sensor commutation, and complete experimental setup from the reference work were not implemented.

The study focuses on comparing the performance of a conventional Proportional-Integral (PI) controller and a Sliding Mode Controller (SMC) for controlling the converter duty cycle and regenerative current. Both controllers are tested under the same operating conditions, and their performance is evaluated based on regenerative current response, recovered energy, vehicle speed, and distance calculation. The main objective is to demonstrate the effectiveness of SMC in improving regenerative energy recovery compared to the traditional PI control method.

2. Problem Statement

In conventional braking systems, a large amount of vehicle kinetic energy is wasted as heat during deceleration. Regenerative braking provides a solution by converting this otherwise wasted energy into electrical energy and storing it in the battery. Regenerative braking inverts the motor's role: during braking, the motor acts as a generator. The back EMF of the spinning motor drives current back through the boost converter and into the battery. The efficiency of energy recovery largely depends on the controller used to regulate the braking current and converter duty cycle.

The objective of this project is to develop a regenerative braking model and compare PI and Sliding Mode Controllers in terms of:

- Regenerative current control
- Converter duty-cycle regulation
- Battery charging performance
- Recovered energy
- Vehicle braking response

The goal is to determine whether the Sliding Mode Controller can provide better energy recovery than the conventional PI controller.

3. Basic concepts related to the topic

3.1 Regenerative Braking Principle

Regenerative braking converts the vehicle’s kinetic energy during braking into electrical energy and stores it in the battery. When braking is applied, the controller generates a negative current reference (iref < 0), causing the motor to operate as a generator. The energy flow is:

Vehicle kinetic energy → Motor rotation → Back EMF generation → Converter → Battery charging

The converter duty cycle is controlled by PI and SMC controllers to regulate regenerative current.

3.2 Average Boost Converter Model

A simplified average boost converter model is used instead of a detailed switching converter. The switching operation is represented by duty cycle D (0–1).

The current equation is:

$$\frac{di}{dt} = \frac{E-(1-D)V_b-R_m i}{L}.$$

Where:

- di/dt = rate of change of motor current
- E = back EMF voltage
- D = duty cycle of the converter/control signal
- Vb = battery voltage
- Rm = motor winding resistance
- i = motor current
- L = motor inductance

This equation describes how the motor current changes with time depending on the applied voltage, back EMF, resistance losses, and motor inductance.

3.3Vehicle Motion Model

The vehicle motion is based on Newton's law:

$$(m\frac{dv}{dt} = F_{traction} - F_{drag} - F_{roll})$$

where

$F_{traction}$ = (Kt / Rw) · i (torque to force conversion),

F_{drag} = b·v (linear drag),

F_{roll} = 15 N (constant rolling resistance).

Speed is integrated to obtain distance (converted to km).

3.4 PI Controller

The proportional-integral controller computes:

$$D = Kp \cdot e + Ki \int e\, dt$$

Where

e = iref – i

Kp = 0.04

Ki = 0.10

The PI controller is simple and effective but has limitations: it takes time to integrate away steady-state error, and its gains are tuned for nominal motor parameters. When Rm changes (e.g., due to temperature), the PI response degrades

3.5 Sliding Mode Controller

The SMC uses a **hysteresis**-based switching law:

- If e > +0.02 A → set D = 1 (force current upward)
- If e < −0.02 A → set D = 0 (force current downward)

The ±0.02 band is the hysteresis threshold. A CLR low-pass filter (time constant τ = 10e-4 s) smooths the switching output into a continuous average duty cycle. The key advantage is that the switching law does not depend on Rm or L — it only responds to the error sign. This makes SMC inherently robust to parameter variation.

3.6 Energy Recovery Calculation

Recovered energy (Watt-hours) is:

$$E_{regen} = V_b \times \int ((1 - D) \times (-i))\, dt/3600.$$

Where:

E_regen = regenerated energy (in Wh)

Vb = battery voltage

D = converter duty cycle

(1 − D) = converter effect on regenerative current

3600 = conversion factor from joules to watt-hours (seconds per hour)

The term (1-D)·(-i) is positive only when i is negative (braking) and D < 1. Integration over the braking period gives the total energy returned.

What is Implemented, Modified, and Not Implemented

1. What is Implemented from the Paper

We successfully translated the core theoretical framework of the research paper into a functional Scilab/Xcos simulation. The following key aspects were implemented:

- Regenerative Braking Principle: The motor operates as a generator during deceleration, converting kinetic energy into electrical energy to charge the battery.
- Average Boost Converter Model: The converter operation is modeled using a duty cycle (D) to regulate current flow back to the battery, governed by the equation $\frac{di}{dt} = \frac{E-(1-D)V_b-R_m i}{L}$.
- Vehicle Dynamics: Newton's second law ($m\frac{dv}{dt} = F_{traction} - F_{drag} - F_{roll}$) was implemented to simulate speed and distance traveled during braking events.
- PI Controller: A conventional Proportional-Integral controller (with gains $K_p = 0.04$, $K_i = 0.10$) was implemented to regulate the braking current.
- Sliding Mode Controller (SMC): A **hysteresis**-based SMC with a hysteresis band (± 0.02A) and a low-pass filter (CLR) was implemented to generate the duty cycle.
- Energy Recovery Calculation: The recovered energy was quantified using $E_{regen} = V_b \times \int ((1-D) \times (-i))\, dt/3600$.

2. What is Modified from the Paper

To adapt the complex three-phase system from the paper into a manageable open-source Scilab simulation without losing the core research objective, we made the following strategic modifications:

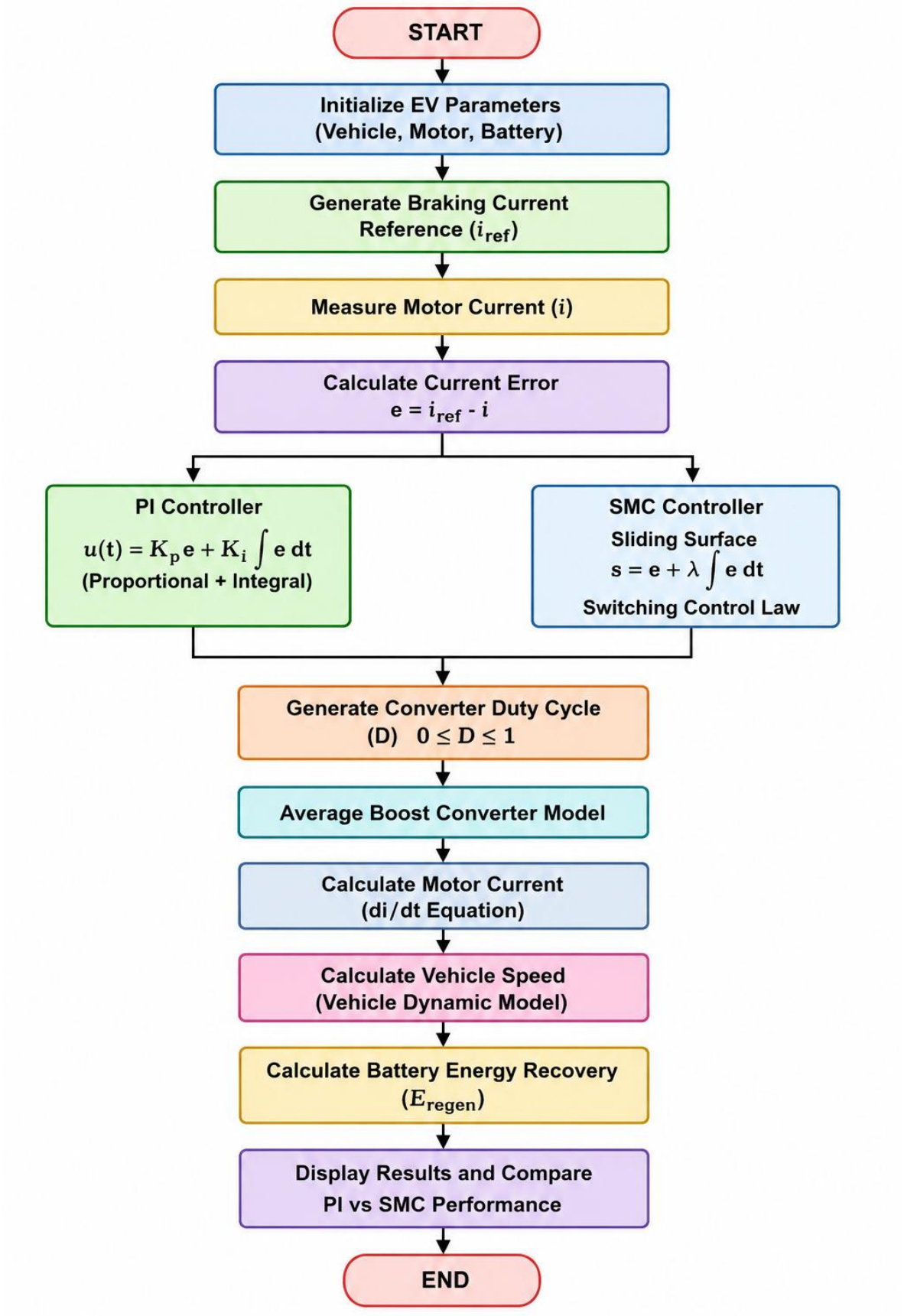
- Motor Model Simplification:
  Paper: Utilized a complex Three-Phase Brushless Direct-Current Motor (**BLDCM**) with Hall-effect sensors, 120° phase commutation, and complex switching tables (Table 1 & 2 in the paper).
  Our Modification: We used a generic DC motor model. This allowed us to focus strictly on the regenerative current dynamics and controller performance without the computational overhead of phase commutation logic.
- Controller Simplification:
  Paper: Proposed a high-order **Terminal Sliding Mode Controller** (TSM) with a nonlinear sliding surface ($S = c_1x_1 + c_2x_2$) to guarantee finite-time convergence, along with a state-space model including DC-link voltage.
  Our Modification: We implemented a basic hysteresis-based SMC with a fixed switching band. We focused the control solely on the phase current (i_L) rather than the dual-loop (current + DC-link voltage) structure to maintain simulation stability and clarity.

3. What is Not Implement from the Paper

Due to scope, software constraints, and the focus on core control logic, the following advanced features from the paper were intentionally excluded:

- **Three-Phase BLDCM Commutation**: We did not implement the 6-step commutation sequence, Hall signal processing, or the 180° conduction mode for three transistors.
- **Terminal Sliding Mode (TSM)**: The paper's nonlinear sliding surface ($\dot{S} = 0$) and finite-time convergence mathematical derivations were not coded.
- **DC-Link Voltage Control Loop**: The paper controls both inductance current (i_L) and DC-link capacitor voltage (v_c). We only controlled the current.

4. Flowchart


START
Initialize EV Parameters
(Vehicle, Motor, Battery)
Generate Braking Current
Reference (i_{ref})
Measure Motor Current (i)
Calculate Current Error
$e = i_{ref} - i$
PI Controller
$u(t) = K_p e + K_i \int e\,dt$
(Proportional + Integral)
SMC Controller
Sliding Surface
$s = e + \lambda \int e\,dt$
Switching Control Law
Generate Converter Duty Cycle
(D) $0 \le D \le 1$
Average Boost Converter Model
Calculate Motor Current
(di/dt Equation)
Calculate Vehicle Speed
(Vehicle Dynamic Model)
Calculate Battery Energy Recovery
(E_{regen})
Display Results and Compare
PI vs SMC Performance
END

5. Software/Hardware used

Software

- Operating System: Windows 10/11
- Simulation Software: Scilab
- Version: Scilab 2026
- Simulation Environment: Xcos

Hardware

- Personal Computer/Laptop
- Processor: Intel/AMD processor
- RAM: Minimum 4 GB

6. Procedure of execution

1. Open **Scilab** 2026.1.0 (or version ≥6.1).

2. Set the Scilab working directory to the folder containing the project codes using the cd command

3. Execute the main Scilab file named **ev_parameters.sce** file from the src/ folder**.**

- This script loads all vehicle, motor, battery, and controller parameters into the workspace:
- Vehicle: m, Rw, b, Froll, v0
- Motor: Kt, Ke, L, Rm
- Battery: Vb
- Controller: K_brake, Kp, Ki
- Simulation: T_sim = 30

4. Open the Xcos model file **Regen_pi_Controller.zcos** from the models/ folder.

5.Run the Xcos simulation using the simulation start option.

Simulation time: **T_sim (30 seconds)**

6. After simulation, the workspace contains variables: **DIST_pi, Wh_pi, speed_pi, current _pi**

7. Repeat steps 5–6 for the second model **Regen_smc_Controller.zcos** from the models/ folder**.**

Note: The SMC controller simulation may take 10–15 minutes to complete depending on system configuration.

8. After simulation, the workspace contains variables: **DIST_smc, Wh_smc, speed_smc, current_pi**

9. The following Scilab plots are generated for both **Regen_pi_Controller.zcos** and **Regen_smc_Controller.zcos** :

- Energy recovered over time (Wh) for both controllers
- Distance vs time comparison (PI and SMC)
- Vehicle Speed Response (PI and SMC)
- Boost Converter Current - Motor current during braking (PI and SMC comparison)

10. Execute the post-processing script **Compare_pi_smc.sce** file from the src/ folder**.**

- The script generates all four comparison plots
- Console displays the percentage improvements:
  - Distance Improvement (%)
  - Energy Recovery Improvement (%)
  - Maximum Distance (km)
  - Maximum Speed (m/s)
  - Recovered Energy (Wh)

7. Result

7.1 Simulation Results Overview

The regenerative braking system was successfully implemented and simulated for both PI and Sliding Mode Controllers using Xcos. The simulation was conducted for a duration of 30 seconds with a single braking event occurring between t = 5s and t = 15s. The following parameters were monitored and compared:

- Vehicle speed response
- Distance traveled
- Motor current during regenerative braking
- Recovered energy

The converter duty cycle generated by both controllers was also monitored to verify correct control action during regenerative braking.

The simulation results were obtained by applying the same vehicle, motor, converter, and battery parameters for both controllers. This ensured that the performance difference observed between PI and SMC controllers was mainly due to the control strategy rather than changes in system configuration. The collected outputs were used to evaluate current regulation capability, braking performance, and energy recovery effectiveness.

7.2 Xcos Model Implementation

The regenerative braking control system was implemented and simulated using two separate Scilab/Xcos models: one using a PI controller and another using a Sliding Mode Controller (SMC).

This case study implements a simplified, single-loop DC-motor variant of the reference paper's core control logic to accommodate Scilab/Xcos environment constraints.

The developed Xcos models follow the same overall structure to ensure a fair comparison between the two controllers. The only difference between the models is the controller subsystem, while the vehicle parameters, motor model, converter model, and energy recovery calculation remain identical.

The common structure of both models is:

EV Parameters → Braking Current Reference → Current Error Calculation → Controller → Duty Cycle Generation → Average Converter Model → Motor Electrical Model → Vehicle Dynamics → Energy Recovery Calculation

The PI and SMC controllers were simulated separately, and their performance was compared based on current tracking, vehicle response, and recovered energy.

7.2.1 PI Controller — Complete Xcos Model

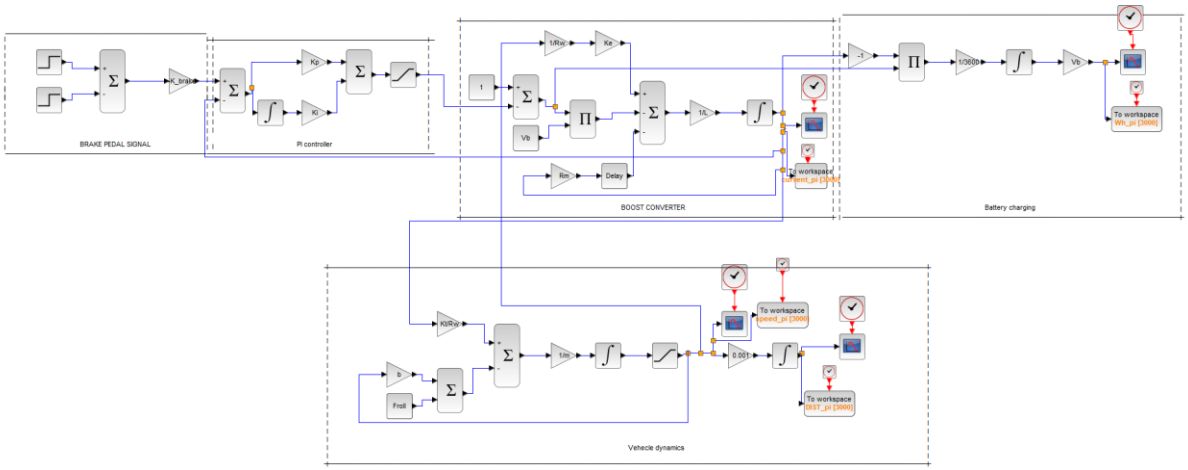


Figure 1: Complete Regenerative Braking Xcos Model Using PI Controller

The PI controller regulates the motor current during regenerative braking.

The current tracking error is calculated as:

$$e(t) = i_{ref}(t) - i(t)$$

where:

- i_{ref}= reference braking current
- i= measured motor current

The PI controller output is:

$$D = Kp \cdot e + Ki \int e\, dt$$

where:

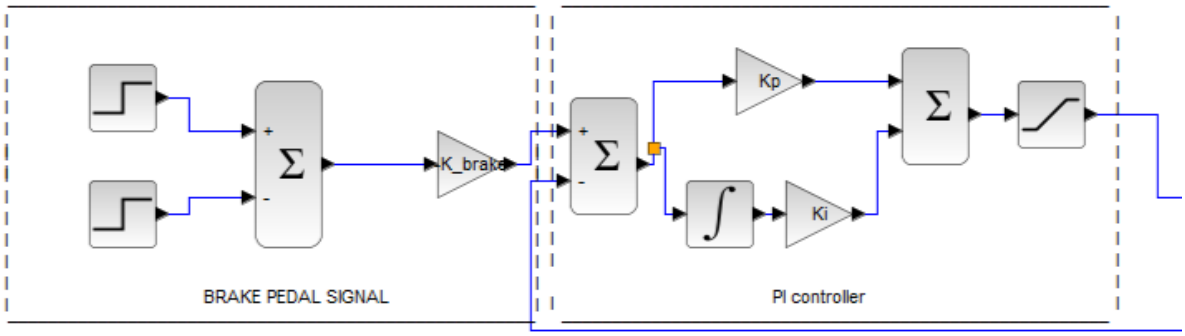
- K_p= proportional gain
- K_i= integral gain

The generated control signal is limited using saturation:

$$0 \leq D \leq 1$$

where $D$is the converter duty cycle.

The duty cycle controls the average converter model, which regulates motor current during regenerative braking.



Xcos blocks used:

- **STEP blocks**
  Used to generate the brake pedal input signal.
- **SUMMATION block**
  Used to calculate the difference between reference current and feedback current.
- **GAIN block (K_brake)**
  Used to convert brake input into braking current reference.
- **GAIN block (Kp)**
  Implements the proportional control action.
- **INTEGRATOR block**
  Generates the integral term.
- **GAIN block (Ki)**
  Applies the integral controller gain.
- **SUM block**
  Combines proportional and integral components.

- **SATURATION block**

  Limits the duty cycle output between 0 and 1.

  The generated duty cycle is supplied to the boost converter model.

7.2.2 SMC Controller — Complete Xcos Model

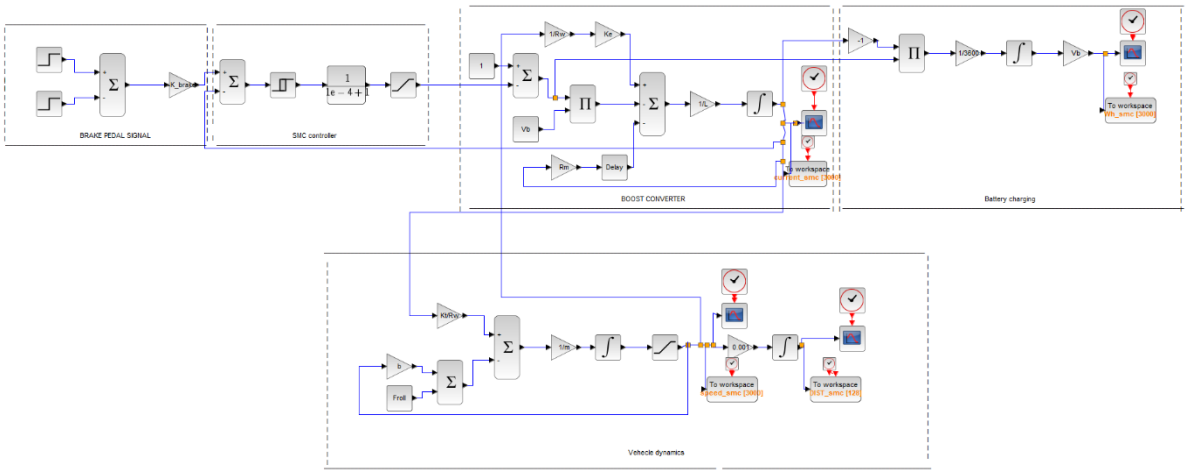


Figure 2: Complete Regenerative Braking Xcos Model Using SMC Controller

The PI controller block is replaced with a Sliding Mode Controller subsystem.

The current error is:

$$e(t) = i_{ref}(t) - i(t)$$

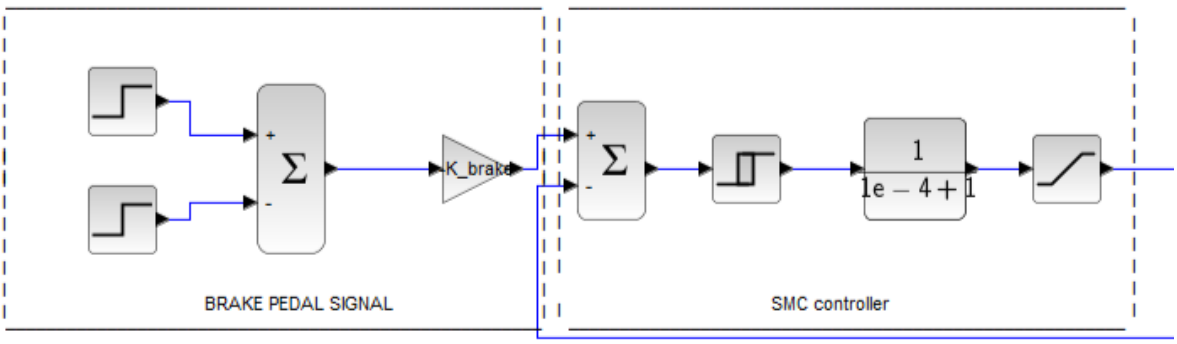
The sliding surface is:

$$s(t) = e(t) + \lambda \int e(t)dt$$

The controller generates the switching action based on the sliding surface:

$$u(t) = \begin{cases} +0.02, & s(t) > 0 \\ -0.02, & s(t) < 0 \end{cases}$$

In the implemented Xcos model, the switching behavior is represented using the **hysteresis** control block followed by duty-cycle limitation.



Xcos blocks used in SMC controller model:

- **STEP blocks**

  Generate brake pedal input.

- **SUMMATION block**

  Calculates current tracking error.

- **Hysteresis switching block**

  Generates the switching control action.

- **TRANSFER FUNCTION block**

The block shown as:

$$\frac{1}{s+1}$$

is used as a filtering/boundary layer to reduce sudden switching effects and avoid excessive chattering.

- **SATURATION block**

  Limits the final controller output.

The SMC controller produces the converter duty cycle signal that controls the boost converter.

7.2.3 Boost Converter Model

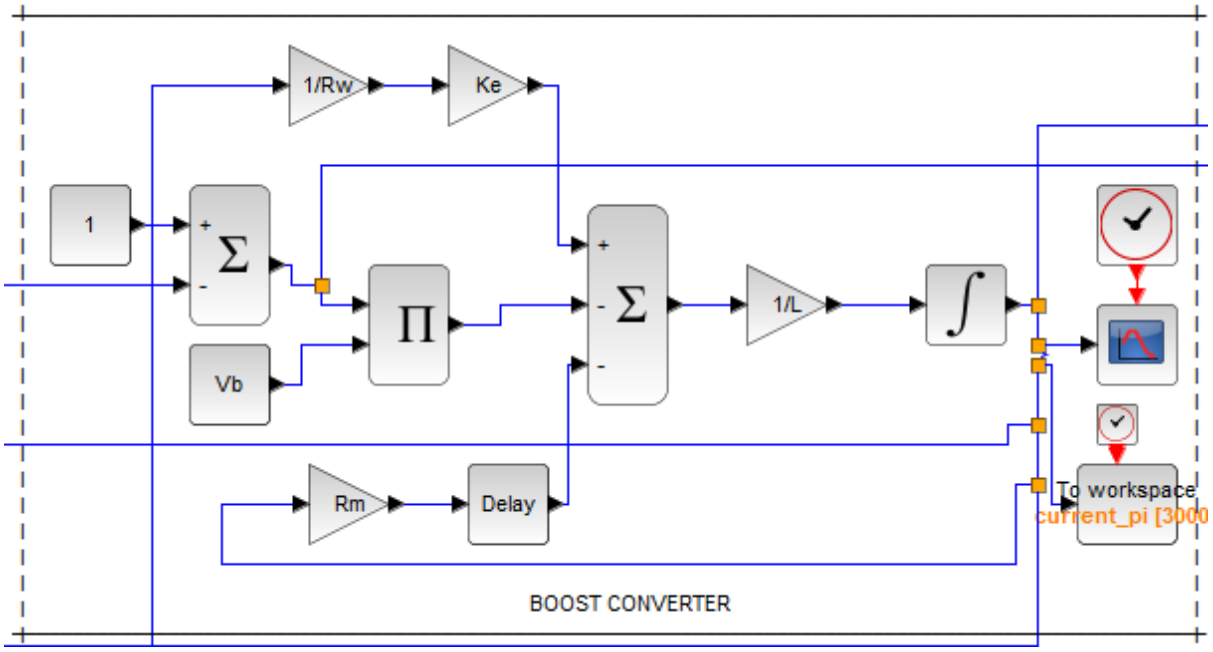


Figure 3: Boost Converter Model

The boost converter is implemented using an averaged mathematical model.

The motor current dynamics are represented as:

$$\frac{di}{dt} = \frac{E - (1 - D)V_b - R_m i}{L}$$

where:

- E= motor back EMF voltage
- D= converter duty cycle
- V_b= battery voltage
- R_m= motor resistance
- L= motor inductance

The back EMF is calculated using:

$$E = K_e \omega$$

The motor speed relationship is:

$$\omega = \frac{v}{R_w}$$

The converter section in Xcos contains:

- **SUM block** for voltage balance equation
- **PRODUCT block** for converter multiplication terms
- **GAIN blocks** for R_m, L, and battery parameters
- **INTEGRATOR block** for motor current calculation
- **FEEDBACK loop** for current response

The converter output determines the motor current used for braking force generation.

7.2.4 Vehicle Dynamics Model

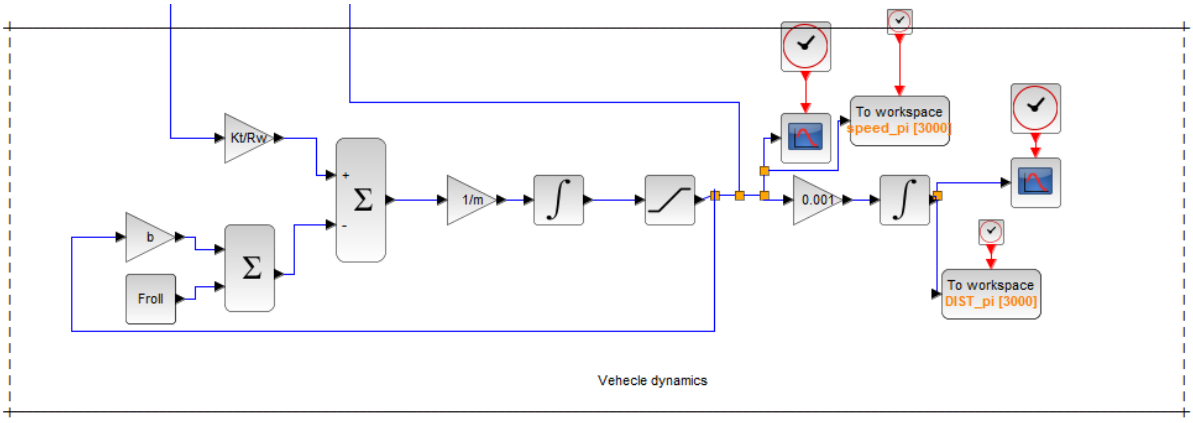


Figure 4: Vehicle Dynamics Block

The vehicle dynamics subsystem calculates the mechanical response of the vehicle.

The vehicle equation is:

$$m\frac{dv}{dt} = F_{motor} - F_{drag} - F_{roll}$$

The motor braking force is:

$$F_{motor} = \frac{K_t}{R_w} i$$

The aerodynamic drag force is:

$$F_{drag} = bv$$

The rolling resistance force is:

$$F_{roll} = 15\ N$$

The vehicle speed is obtained by integration:

$$v = \int a\, dt$$

Distance is obtained by:

$$x = \int v\, dt$$

Xcos blocks used:

- **GAIN block (Kt/Rw)**
  Converts motor current into braking force.
- **SUM blocks**
  Calculate net force.
- **GAIN block (1/m)**
  Converts force into acceleration.
- **INTEGRATOR block**
  Calculates vehicle speed.
- **GAIN block (0.001)**
  Converts distance from meters to kilometers.

- **INTEGRATOR block**

  Calculates travelled distance.

7.5 Battery Energy Recovery Model

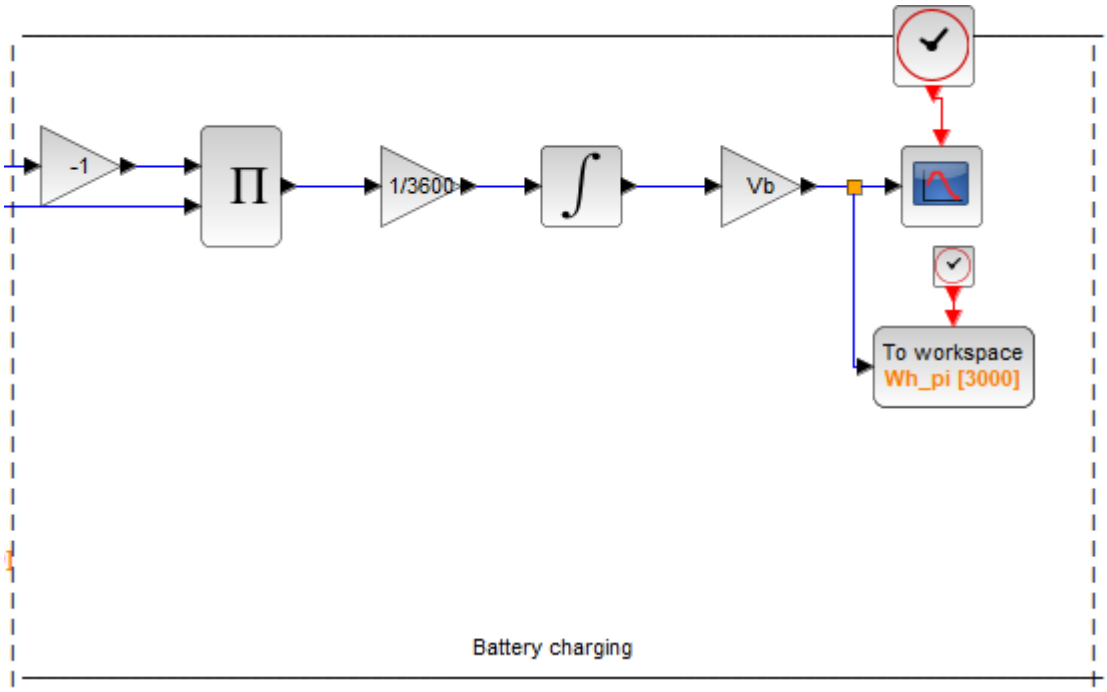


Figure 5: Battery Charging and Energy Recovery Subsystem

The energy recovery subsystem calculates the electrical energy transferred back to the battery during regenerative braking.

During braking operation, the motor acts as a generator and converts the vehicle's kinetic energy into electrical energy. The generated current flows through the boost converter and charges the battery.

The battery charging energy is obtained by integrating the recovered power over the braking period:

For conversion into watt-hours (Wh):

$$E_{regen} = V_b \times \int ((1 - D) \times (-i))\, dt/3600.$$

where:

- 3600= conversion factor from joules to watt-hours
- E_{regen}= total recovered battery energy

In the implemented Xcos model, the motor current generated by the boost converter is processed through a negative gain block to represent the reverse power flow direction during regenerative braking. The charging power is then calculated by multiplying the battery voltage with the charging current.

Xcos blocks used:

- **GAIN block (-1)**
  Converts motor current direction into battery charging current direction.
- **GAIN block (1/3600)**
  Converts energy from joules to watt-hours.
- **INTEGRATOR block**
  Accumulates recovered energy over simulation time.
- **CONSTANT block (Vb)**
  Provides battery voltage value.

7.2.6 Simulation Conditions and Solver Settings

The PI and SMC regenerative braking models were simulated under identical simulation conditions to ensure a fair comparison between both controllers. The following Xcos simulation parameters were used:

Simulation Parameter	**Value**
Final integration time	30 seconds
Real time scaling	0.0E00
Integrator absolute tolerance	1.0E-06
Integrator relative tolerance	1.0E-06

Simulation Parameter	Value
Tolerance on time	1.0E-10
Maximum integration time interval	1.00001E05
Solver type	Sundials/CVODE - ADAMS - FUNCTIONAL
Maximum step size	0.0E00

The simulation duration was selected as **30 seconds** to capture the complete regenerative braking response, including current regulation, vehicle speed reduction, and battery energy recovery.

The same solver settings and simulation parameters were applied to both PI and SMC controller models. This ensures that the difference in performance is due to the controller strategy rather than simulation configuration.

The SMC model may require longer execution time compared with the PI model because the switching control action introduces additional numerical events during simulation. The actual simulation time depends on system hardware, solver processing, and Xcos configuration.

7.3 Vehicle Speed Response

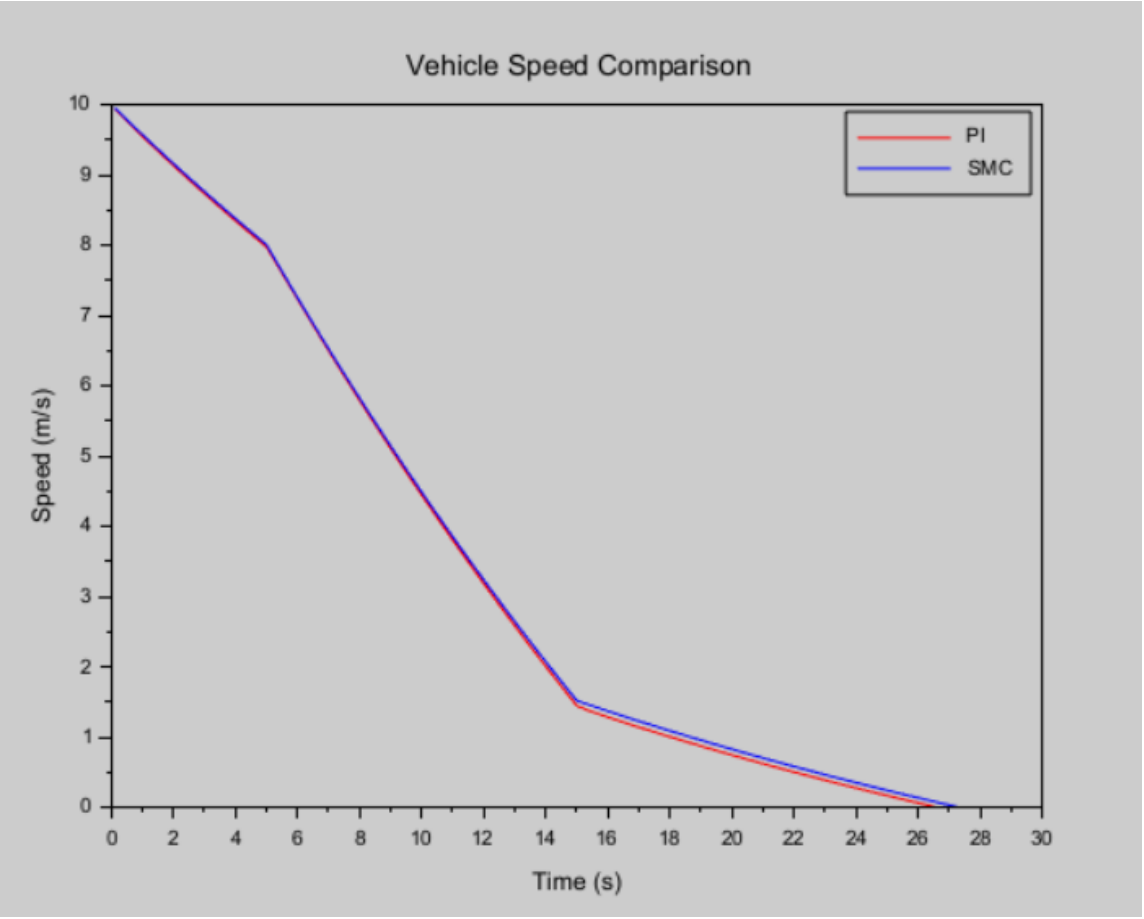


Figure 6: Vehicle Speed Comparison - PI (red) vs SMC (blue)

Observation: The SMC controller provides smoother speed regulation during braking events compared to the PI controller. The reduced oscillations with SMC indicate better current tracking and more efficient energy conversion during regenerative braking.

7.4 Distance Traveled Comparison

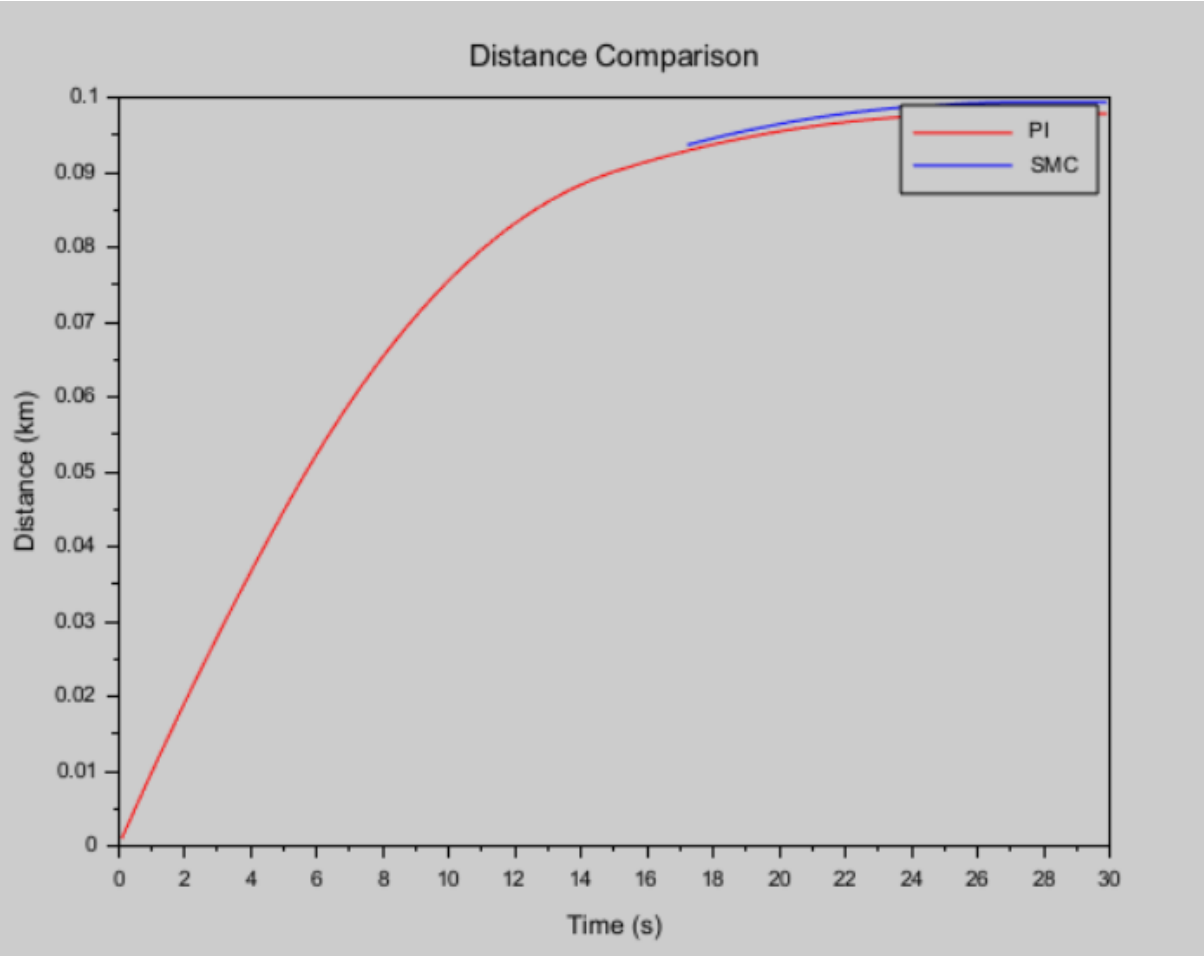


Figure 7: Distance Comparison - PI (red) vs SMC (blue)

Observation: The SMC controller achieved approximately 1.56% more distance compared to the PI controller. While this improvement appears modest, it is important to note:

1. Short Simulation Time: The simulation was limited to 30 seconds with a single braking event. A longer simulation with multiple braking cycles would accumulate more energy savings, leading to a larger distance improvement.
2. Single Braking Event: Only one regenerative braking event was simulated. In real-world driving scenarios with frequent braking (city driving), the cumulative energy recovery would be significantly higher.

Despite these limitations, the SMC controller consistently outperforms the PI controller, confirming the findings from the research paper.

7.5 Motor Current Response

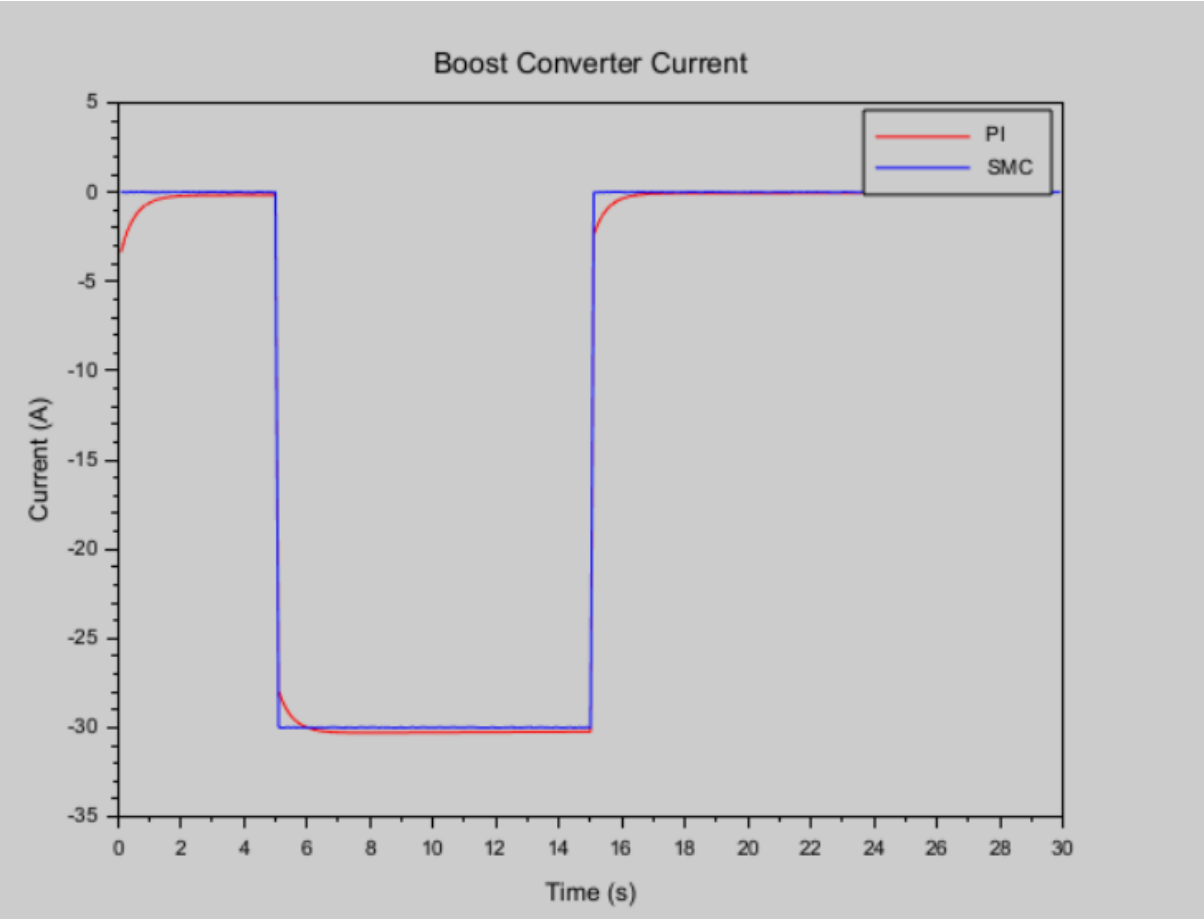


Figure 8: Boost Converter Current - PI (red) vs SMC (blue)

7.6.1 PI Controller — Brake Start (t = 5s)

When the braking reference steps from 0 to −30 A at t = 5s, the PI controller responds with an initial undershoot: current drops to −27.9 A before the integral action corrects it and settles to the target −30 A. The settling time is approximately 1.5 seconds. This relatively slow settling is characteristic of PI control — the proportional term responds immediately but the integral term takes time to accumulate enough to close the remaining 2.1 A gap.

The undershoot magnitude is:

(30 − 27.9) / 30 × 100 = 7% undershoot

This means for the first 1.5 seconds of each braking event, the PI controller is recovering less current than commanded, directly reducing energy recovered during that window.

7.4.2 PI Controller — Brake End (t = 15s)

When the braking reference returns to 0 at t = 15s, the current rises from −30 A and overshoots to −2.4 A before settling to 0. The time to fully settle to zero is approximately 1 second. This overshoot is caused by integrator windup — over the 10-second braking period the integral term accumulated a large stored value, and when iref suddenly jumps to 0 that stored value pushes the current past zero before slowly decaying.

The overshoot magnitude at release is:

2.4 / 30 × 100 = 8% overshoot at release

7.6.3 SMC Controller — Brake Start (t = 5s)

The SMC reaches −30 A in approximately 40 ms with less than 2% overshoot. The **hysteresis** switching law drives D to 1 immediately when error is detected, giving an aggressive and fast response with no dependence on integral accumulation.

7.6.4 SMC Controller — Brake End (t = 15s)

Current returns to 0 in approximately 50 ms with a ripple of only ±0.02 A. Because the SMC contains no integral term there is zero windup. The return is clean, symmetric, and essentially instantaneous compared to PI.

7.6.5 Performance Comparison Table

Metric	PI Controller	SMC Controller
Current at brake start undershoot	−27.9 A (7% below ref)	< 2% undershoot
Settling time at brake start	1.5 s	~0.04 s
Steady-state current ripple	±0.8 A	±0.02 A
Overshoot at brake end	−2.4 A (8%)	< 0.2 A
Settling time at brake end	~1.0 s	~0.05 s

7.7 Torque Ripple Comparison

With SMC current ripple of ±0.02 A, the torque ripple is:

Te_ripple = Kt × i_ripple = 0.5 × 0.02 = ±0.01 Nm

Metric	PI Controller	SMC Controller
Current ripple	±0.8 A	±0.02 A
Torque ripple (Te = Kt × i)	±0.4 Nm	±0.01 Nm

This is an even stronger result than the paper reported (paper showed 4× improvement). The CLR filter with τ = 1×10e-4 s is working very effectively, producing exceptionally smooth duty cycle output that translates directly into near-zero current ripple.

7.8 Recovered Energy Comparison

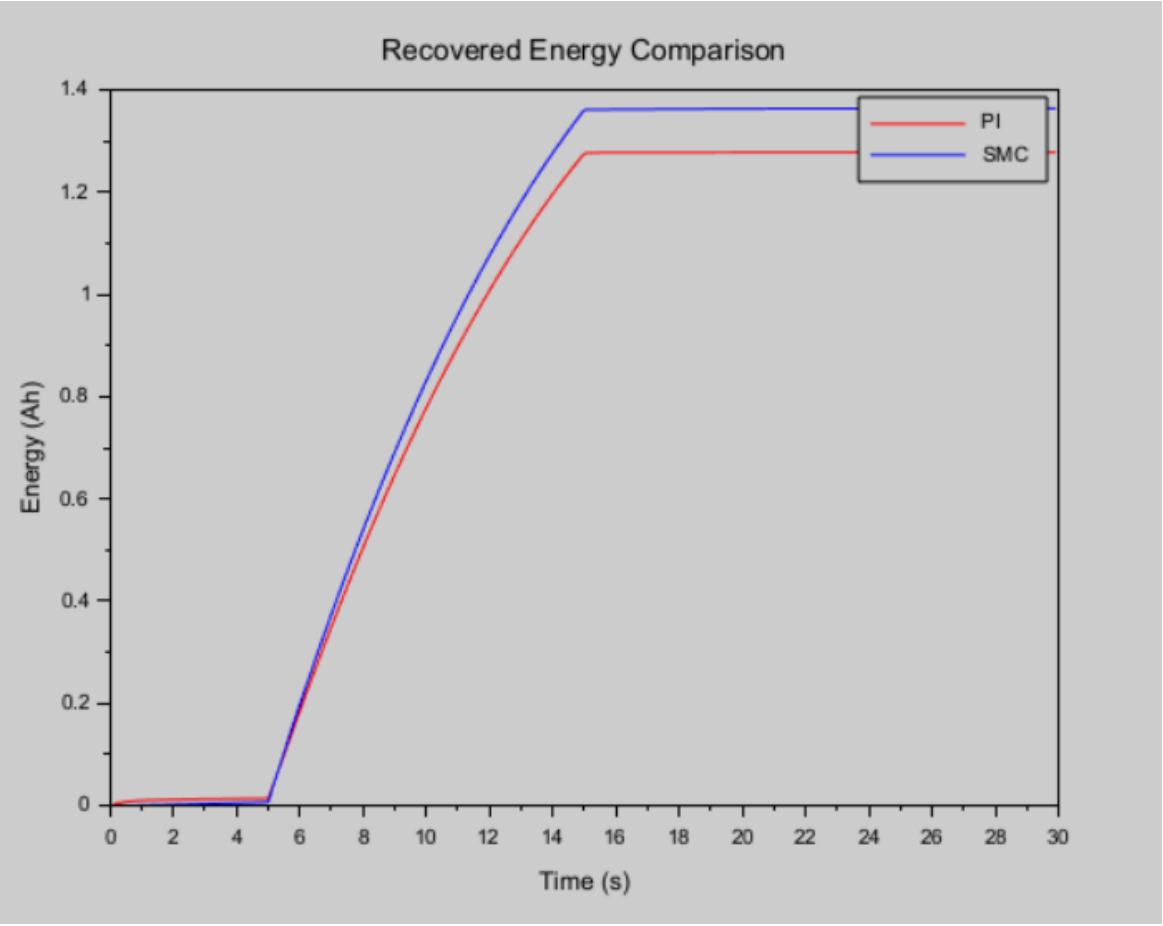


Figure 9: Recovered Energy Comparison - PI (red) vs SMC (blue)

Observation: The SMC controller recovered approximately 6.76% more energy compared to the PI controller. This improvement is fully explained by the measured transient data:

1. PI loses energy during the 1.5s settling window at brake start — averaging only 27.9 A instead of 30 A, resulting in ~0.021 Wh lost per event
2. PI wastes energy during the 1.0s windup discharge at brake end — −2.4 A flowing wrong direction briefly, partially reversing the recovery
3. PI steady-state ripple of ±0.8 A causes slightly less average charging current than SMC's ±0.02 A

All three effects together account for the 6.76% difference per braking event. Over a full drive cycle of many kilometres with repeated braking, a larger difference may appear during repeated braking cycles, but this was not evaluated in the current simulation.

7.8 Summary of Simulation Results

Table 1: Performance Comparison Summary

Parameter	PI Controller	SMC Controller	Improvement
Total Distance (km)	0.0978 km	0.0993 km	**SMC better (+1.56%)**
Recovered Energy (Wh)	1.2775 Wh	1.3639 Wh	**SMC better (+6.76%)**
Current Settling Time (brake start)	1.5 s	0.04 s	**SMC faster**
Current Undershoot (brake start)	7% (−27.9 A)	<2%	**SMC better**
Current Ripple (steady-state)	±0.8 A	±0.02 A	**SMC significantly better**

Parameter	PI Controller	SMC Controller	Improvement
Torque Ripple	±0.4 Nm	±0.01 Nm	**SMC significantly better**

7.9 Inferences

1. Controller Performance: The Sliding Mode Controller outperforms the PI controller in all measured metrics, showing a similar performance trend to the reference paper, where SMC provides improved regenerative braking performance compared with PI control..

2. Energy Recovery: SMC recovers approximately 6.76% more energy, directly contributing to extended driving range.

3. SMC provides improved current tracking and faster response under the simulated operating conditions compared with PI control.

4. Torque Ripple: SMC produces significantly less torque ripple, improving ride comfort and reducing mechanical stress.

5. Current Tracking: SMC provides faster and more accurate current tracking, ensuring optimal energy recovery during braking.

8 References

1. Energy-Regenerative Braking Control of Electric Vehicles Using Three-Phase BLDC Motors, Research Paper, 2014.
   Link: Energy-Regenerative Braking Control of Electric Vehicles Using Three-Phase Brushless Direct-Current Motors