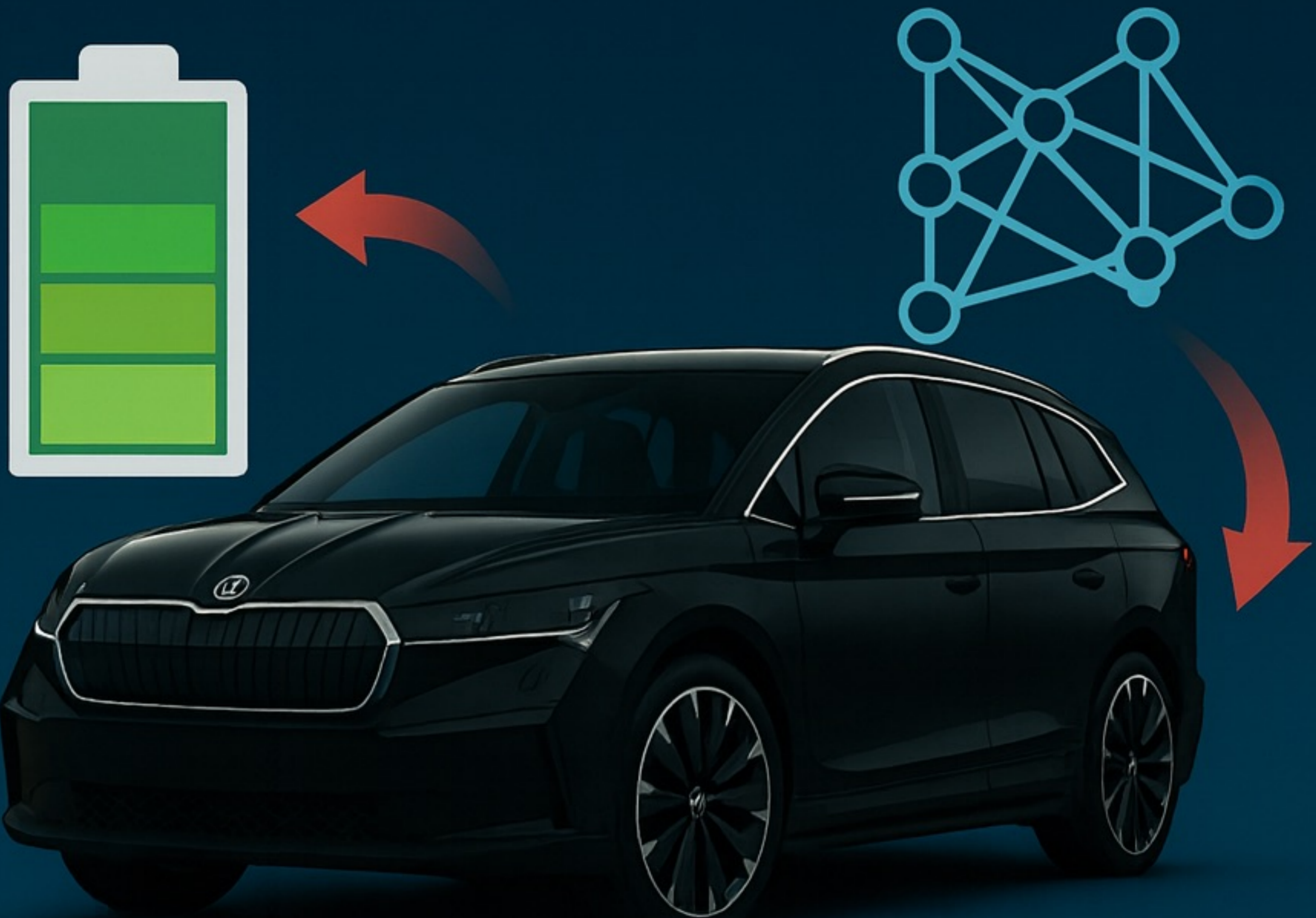
MASTER’S THESIS

REGENERATIVE BRAKING SYSTEM WITH SMART ENERGY RECOVERY



BJARTUR RAGNARSSON Á NORÐI
KONSTANTINOS SPILIOTOPOULOS

Faculty of Engineering and Science
Electro Mechanical System Design
Niels Jernes Vej 10, 9220 Aalborg Øst

Title:
Regenerative Braking System with Smart Energy Recovery

Project:
EMSD Master's Thesis

Project period:
September 2025 - January 2026

Participants:
Bjartur Ragnarsson á Norði
Konstantinos Spiliotopoulos

Supervisor:
Farshid Naseri

Report page count: 75
Appendix page count: 82
Total page count: 177

Abstract

This thesis develops and evaluates a Reinforcement Learning (RL) controller for regenerative braking in a battery-electric vehicle (BEV). The RL agent operates as a torque-split controller, allocating braking demand between regenerative and friction systems under the same physical constraints as a rule-based baseline. A simplified HPPC-inspired State-of-Power (SoP) estimator provides a dynamic battery charging limit, enabling the agent to learn time-varying power constraints.
A Deep Deterministic Policy Gradient (DDPG) agent is trained within a Simulink environment, where an EV drivetrain is modeled. The reward function encourages effective use of the SoP-limited regenerative capability while penalizing constraint violations and abrupt control actions.
Across various drive cycles and state-of-charge levels, the RL controller consistently demonstrates strong performance, despite the highly restricted training data and time. The results highlight RL's potential as a foundation for future adaptive energy-management strategies.

Preface

This project has been conducted as part of the Master's program in Electro-Mechanical System Design at Aalborg University. The work presented in this report concerns the development and evaluation of a reinforcement learning–based regenerative braking controller integrated into an electric vehicle drivetrain model. The study covers literature review, model formulation, simulation-based analysis, and performance assessment within the domain of energy management for electric vehicles.

Throughout the report, abbreviations are introduced by first stating the full term followed by its abbreviated form in parentheses (e.g. State of Power (SoP)). Once introduced, the abbreviated form is used consistently. For completeness and ease of reference, all abbreviations and symbols—regardless of where they are first introduced—are collected in the nomenclature section.

A clear distinction is maintained in how sources are cited throughout the report. When authors are explicitly mentioned or statements are directly attributed to a specific source, the referenced material closely reflects the concepts or approaches presented in that work. Additionally, citations placed at the end of a sentence are used to support or validate claims developed within this report, serving as supporting references rather than direct conceptual sources.

AI tools were utilised for language refinement and spelling corrections. During the making of this project, the following programs have been used:

- **Overleaf LaTeX**: For text processing and formatting of this report.
- **MathWorks MATLAB and Simulink**: For calculations, modelling, simulation, and data processing.

This project has been completed under the supervision of **Farshid Naseri** at Aalborg University.

Bjartur Ragnarsson á Norði   Konstantinos Spiliotopoulos

Aalborg University, 2nd of January 2026

Contents

Nomenclature

	Acronyms
ABS	Anti-lock Braking System
AI	Artificial Intelligence
ANN	Artificial Neural Network
BEV	Battery Electric Vehicle
BMS	Battery Management System
DC	Direct Current
DDPG	Deep Deterministic Policy Gradient
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
ECU	Electronic Control Unit
ESC	Electronic Stability Control
EV	Electric Vehicle
FEA	Finite Element Analysis
FLC	Fuzzy Logic Control
HiL	Hardware-in-the-Loop
HPPC	Hybrid Pulse Power Characterization
HVAC	Heating, Ventilation, and Air Conditioning
ISO 26262	Functional safety standard for road vehicles
LFP	Lithium Iron Phosphate
Li-ion	Lithium-ion
LSTM	Long Short-Term Memory network

LTO	Lithium Titanate
ML	Machine Learning
NCM	Nickel Cobalt Manganese (Li-ion chemistry)
NMC	Nickel–Manganese–Cobalt (Li-ion chemistry)
OCV	Open-Circuit Voltage
OEM	Original Equipment Manufacturer
PI	Proportional–Integral
PM	Permanent Magnet
PMSM	Permanent Magnet Synchronous Motor
PNGV	Partnership for a New Generation of Vehicles
PPO	Proximal Policy Optimization
pu	Per-unit (normalized quantity)
RB	Rule-based (baseline) controller
RB (braking)	Regenerative Braking
RBS	Regenerative Braking System
RC	Resistive–Capacitive (equivalent-circuit element)
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
RWD	Rear-Wheel Drive
SAC	Soft Actor-Critic
SEI	Solid Electrolyte Interphase
SoC	State of Charge
SoH	State of Health
SoP	State of Power
TD error	Temporal-difference error
TD3	Twin Delayed Deep Deterministic Policy Gradient
UNECE R13-H	UN regulation governing passenger vehicle braking systems
WLTP	Worldwide Harmonised Light Vehicles Test Procedure
WLTP Class 3	High-speed WLTP driving class

Control and RL notation

Δt	Sample time
γ	Discount factor
$\mathcal{A}$	Action space
$\mathcal{D}$	Replay buffer distribution
$\mathcal{N}_t$	Exploration noise at time t
$\mathcal{S}$	State space
μ_θ	Deterministic actor policy with parameters θ
$\pi(s)$	Deterministic policy
τ	Soft-update coefficient for target networks
a_t	Action at time step t
$Q(s, a)$	Action-value function
$Q_\phi(s, a)$	Critic function with parameters ϕ
r_t	Reward at time step t
s_t	State at time step t
t	Time
Actor–critic	RL architecture with actor (policy) and critic (value)
Mini-batch	Random sample of stored transitions for training
Replay buffer	Memory of transitions for off-policy learning
Target network	Slow-moving network used to stabilize learning

Driver / PI block signals

e_{out}	Anti-windup error, $e_{\text{out}} = y_{\text{sat}} - y$
e_{ref}	Speed tracking error, $e_{\text{ref}} = v_{\text{ref}} - v$
K_i	Integral gain
K_p	Proportional gain
K_{aw}	Anti-windup gain
K_{ff}	Velocity feedforward gain
v	Vehicle speed

v_{nom}	Nominal speed used for normalization
v_{ref}	Reference vehicle speed
y	PI controller output (pre-saturation)
y_{acc}	Normalized accelerator pedal command
y_{dec}	Normalized brake pedal command
y_{sat}	Saturated controller output
	Braking / torque split / limits
η	Efficiency factor (regen or motor–inverter, context-dependent)
acc_cmd	Normalized accelerator command, $\in [0, 1]$
brk_cmd	Normalized brake command, $\in [0, 1]$
ε	Small constant to avoid division by zero
$a \in [0, 1]$	Agent action: regenerative torque share
$f_{\text{fric,min}}$	Minimum friction share (ABS/safety)
$P_{\text{cap}}(t)$	SoP-based battery charge power cap
P_{regen}	Regenerative electrical power to the battery
T_f	Friction brake torque
T_r	Regenerative brake torque
$T_{\text{br,max}}$	Maximum available total brake torque
T_{req}	Total requested brake torque
x	Normalized ratio, $x = \frac{P_{\text{regen}}}{P_{\text{cap}}+\varepsilon}$
	Motor / inverter / drivetrain
η_{now}	Instantaneous motor–inverter efficiency
ω_m	Motor angular speed [rad/s]
ω_w	Wheel angular speed [rad/s]
$f(\cdot)$	Efficiency-map function, $\eta = f(\omega_m, T_m)$
G	Gear ratio, $G = \omega_m/\omega_w = T_w/T_m$
n	Motor speed [rpm]
P_{DC}	DC bus power

P_{mech}	Mechanical shaft power, $P_{\text{mech}} = T_m\omega_m$
T_m	Motor torque [Nm]
T_w	Wheel torque [Nm]
$T_{\max}(\omega_m)$	Motor torque envelope vs speed

Vehicle body / road loads

grade	Road grade in percent [%]
ρ	Air density
θ	Road grade angle
A	Frontal area
b_{vis}	Viscous damping constant
C_d	Aerodynamic drag coefficient
C_{rr}	Rolling-resistance coefficient
g	Gravitational acceleration
J_{eq}	Equivalent wheel-side rotary inertia
M_{veh}	Vehicle mass
R_{wheel}	Effective wheel radius
T_d	Aerodynamic drag torque
T_g	Grade (slope) torque
T_v	Viscous friction torque
T_{road}	Total resistive road-load torque
T_{rr}	Rolling-resistance torque

Friction brake model

μ_k	Kinetic friction coefficient
d_{bore}	Brake cylinder bore diameter
N_{pads}	Number of brake pads per caliper
P	Hydraulic brake pressure [bar]
r_{pad}	Mean brake pad radius
T_{brk}	Friction brake torque

Battery and SoP (PNGV-style)

i	Battery current (positive discharge, negative charge)
n_p	Number of parallel-connected strings
n_s	Number of series-connected cells
R_0	Ohmic internal resistance
U	Battery terminal voltage
$V_{\max}$	Maximum allowable terminal voltage
$V_{\min}$	Minimum allowable terminal voltage
z	Normalized state of charge, $z = \text{SoC} \in [0, 1]$

Units

m/s^2	Metres per second squared
m/s^3	Metres per second cubed
°C	Degrees Celsius
%	Percentage
A	Ampere
bar	Bar
km/h	Kilometres per hour
kW	Kilowatt
kWh	Kilowatt-hour
m	Metre
min	Minute
Nm	Newton-metre
rpm	Revolutions per minute
s	Second
Wh/km	Watt-hours per kilometre

1 | Introduction

The growing transition toward electric mobility places strong emphasis on improving vehicle efficiency and sustainability. While modern electric vehicles eliminate local emissions, their total energy efficiency still depends heavily on how effectively they recover and manage energy during operation. A key mechanism for this is **regenerative braking**, where the traction motor acts as a generator to convert the vehicle's kinetic energy into electrical energy that recharges the battery during deceleration.

In practice, regenerative braking control must satisfy several competing objectives: ensure safe stopping, comfortable braking, efficient energy recovery, and battery protection. Conventional braking strategies in production vehicles typically use rule-based logic or proportional–integral (PI) controllers to coordinate the torque split between regenerative and hydraulic brakes [1]. These methods are robust and certifiable, but they rely on fixed calibration parameters and therefore cannot adapt effectively when operating conditions change—such as variations in road slope, temperature, vehicle load, or battery state of charge [2]. As a result, regenerative braking is often curtailed conservatively to protect stability and hardware limits, cutting off regeneration early and leaving a non-negligible fraction of recoverable energy unused [3].

Multiple studies report that moving from such conservative rule-based strategies to more optimized or predictive control can increase recuperated braking energy by roughly 5–10 % under comparable driving conditions [4]. Even improvements in this range translate directly into extended effective driving range and reduced charging frequency, both of which are closely linked to range anxiety and user acceptance of battery electric vehicles [5].

This thesis aims to demonstrate that reinforcement learning can enhance regenerative braking control by enabling adaptive, data-driven decision-making within physically constrained environments. By replacing manual calibration with an intelligent learning process, the work seeks to contribute toward more efficient and autonomous energy-management strategies for future electric vehicles.

The remainder of this thesis is organized as follows: Chapter 2 presents a detailed analysis of the regenerative braking problem, covering the fundamentals of electric vehicle braking, a review of current control strategies, and their main limitations. Chapter 3 defines the research objectives and outlines the framework for the proposed solution. Chapter 4 describes the development and validation of the vehicle simulation model, including the modeling of battery dynamics, drivetrain, and braking system constraints. Chapter 5 introduces the reinforcement learning-based control strategy, detailing the agent design, state-action space, reward structure, and the optimization algorithm (DDPG). Chapter 6 presents and analyzes the simulation results, comparing the performance of the RL controller with an idealized rule-based approach. Chapter 7 provides the conclusion, summarizing the key contributions of the work, highlighting the improvements achieved, and suggesting directions for future research. Chapter 8 discusses the implications of the findings, evaluates the strengths and limitations of the proposed strategy, and addresses how the results relate to existing literature and real-world applications. Chapter 9 introduces proposed directions for future work, outlining potential extensions and improvements to the methods and results presented in this thesis.

1.1 Literature Review: AI-Based Regenerative Braking for Electric Vehicles

As mentioned, conventional rule-based regenerative-braking systems are inherently conservative due to fixed curtailing logic and simplified battery constraints, and this motivates a closer look at the existing literature on RB control strategies. This section reviews the scientific evolution of AI-based regenerative braking strategies, from early supervised-learning and fuzzy-logic methods to modern deep reinforcement learning (DRL), highlighting both achievements and remaining gaps.

1.1.1 Early AI Approaches: Supervised Learning and Fuzzy Logic

Supervised learning (ANNs): Artificial neural networks were among the first machine-learning methods applied to RB control. Early ANN-based controllers attempted to map driver demand, State of Charge (SoC), speed, and gradient to optimal braking torque using labelled datasets [6]. These methods demonstrated improved recovery compared to fuzzy baselines and conventional rule-based logic. However, they suffer from two structural limitations: (1) they require large labelled datasets spanning many operating conditions, and (2) they operate in a purely reactive manner, producing instantaneous torque commands without reasoning about future battery constraints or upcoming braking events. As a result, ANNs can approximate expert behaviour but cannot perform sequential decision-making, nor can they optimize over an entire drive cycle.

Fuzzy logic control (FLC): Fuzzy controllers have also been applied to RB due to their ability to encode expert knowledge and handle nonlinear dynamics [7, 8]. Typical improvements of 20–30% in energy recovery have been reported relative to fixed-rule strategies, with hybrid fuzzy–sliding-mode variants providing smoother braking feel and better robustness. However, fuzzy systems rely on manually crafted membership functions and rule sets, and therefore remain limited by designer intuition. They cannot autonomously discover new optimal strategies or adapt to unseen road or battery conditions. As with supervised learning, they lack the capacity to optimize *long-term* objectives such as battery health, comfort, or cumulative energy recovery.

1.1.2 Reinforcement Learning for Regenerative Braking

Reinforcement learning (RL) represents a methodological shift: instead of imitating labelled data or encoding expert rules, RL learns optimal torque-allocation policies through interaction with a simulated environment, optimizing long-term cumulative reward [9]. This is particularly relevant for regenerative braking, where each torque command affects not only instantaneous energy recovery but also future battery temperature, SOC, vehicle dynamics, and overall drive-cycle efficiency.

Discrete-action RL: Q-learning and DQN: Early RL applications discretized the braking command into torque bins. Yin et al. [10] propose a DQN-based RB controller using SOC and power-demand states, achieving **7.4% higher recovery than Q-learning** and **13.1% improvement over rule-based control** on standardized cycles. Their work also highlights the weakness of coarse torque discretization: the agent cannot explore fine-grained torque adjustments, limiting smoothness and optimality. Discrete RL thereby improves upon rule-based design yet remains fundamentally constrained by action discretization, which is incompatible with the continuous nature of braking torque.

Continuous-action RL: DDPG, TD3, and SAC: Continuous-action algorithms overcome this limitation by outputting real-valued torque commands. The most relevant methods are Deep Deterministic Policy Gradient (DDPG) [11], Twin Delayed DDPG (TD3) [12], and Soft Actor-Critic (SAC). DDPG enables efficient learning in deterministic, physics-based simulations, while TD3 improves stability through clipped double Q-learning and delayed policy updates. Recent studies report significant gains in energy recovery when continuous-action RL is used, especially when combined with motor-efficiency maps, SOC-dependent limits, and adaptive reward functions [13]. Crucially, the work of Yin et al. shows that even with discrete actions, RL can surpass rule-based baselines; continuous-action RL is therefore expected to perform even better under high-fidelity modelling conditions.

Role of battery constraints (SoC, SoP, HPPC): Several studies emphasize that regenerative braking performance is fundamentally capped by battery charge-power limits. Accurate online estimation of allowable charging power (*State of Power*, SoP) relies on HPPC-derived internal-resistance models [14]. High SoC and low temperature significantly reduce charge acceptance, and these constraints must be respected by any control strategy. AI-based controllers that incorporate dynamic SoP constraints consistently outperform those using static SoC thresholds.

Critical assessment. Deterministic Reinforcement Learning (DRL) methods outperform supervised and fuzzy approaches in several domains: they handle continuous actions, capture long-term effects, and adapt to nonlinear motor–battery interactions. However, two limitations persist across the literature: (1) most studies evaluate policies only on a narrow set of driving cycles, leaving generalization to unseen scenarios unresolved; and (2) DRL controllers lack formal safety guarantees, making verification challenging for safety-critical automotive applications.

1.1.3 Hybrid and Multi-Method Approaches

Hybrid controllers combine the interpretability of fuzzy logic with the adaptiveness of RL. Examples include fuzzy Q-learning (FQL), ANN-assisted model predictive control

(MPC), and architectures integrating long short-term memory (LSTM) networks for driver intent prediction [8]. These systems show improved adaptability and smoother braking transitions. However, hybrid methods increase architectural complexity and often require careful tuning of interacting subsystems, limiting scalability and interpretability.

1.1.4 Research Gaps and Positioning of This Work

Despite promising results from both pure DRL and hybrid approaches, several challenges remain before AI-based regenerative braking can be deployed in production EVs. Many studies neglect embedded feasibility aspects such as inference latency, memory footprint, and Electronic Control Unit (ECU) resource limitations, even though braking is a safety-critical function [15]. Generalization is also weakly addressed: policies are rarely evaluated on drive cycles or operating conditions that differ significantly from the ones used during training, which limits confidence in real-world robustness. Furthermore, DRL policies are typically treated as black-box neural networks, and systematic safety verification and constraint handling are still largely unexplored. Finally, most works rely on simplified battery models and fixed charging limits, whereas real state-of-power (SoP) constraints depend on temperature, aging, and resistance maps obtained from HPPC tests [16].

The reviewed literature therefore reveals a methodological progression from static rule-based strategies to supervised learning, fuzzy logic, and DRL, but three critical gaps remain for regenerative braking control in realistic EV drivetrains:

1. **Insufficient out-of-distribution testing.** Existing RL-based controllers are usually trained and evaluated on similar or identical drive cycles, with little emphasis on robustness to operating conditions and braking patterns that deviate strongly from the training distribution.

2. **Lack of explicit safety constraint handling.** Safety is often treated indirectly via reward shaping, without hard constraints on deceleration, torque limits, or fallback behavior that would be required in a safety-critical automotive context.

3. **Limited integration of varying SoP.** Most studies assume fixed or overly simplified battery power limits, rather than incorporating a realistic, time-varying SoP model that accounts for SoC, temperature, and internal resistance dynamics.

Contribution of this thesis: This work addresses these gaps by:

1. Testing the RL controller across multiple drive cycles that are significantly different the deliberately limited dataset from the training environment, ensuring robust generalization to diverse real-world scenarios

2. Implementing hard safety constraints within the RL framework, projecting any unsafe actions into a safety region to prevent violations of braking limits

3. Integrating varying SoP constraints into the RL policy, enabling dynamic adaptation to real battery behavior based SoC from a HPPC-derived resistance maps

These contributions provide a more rigorous and practical evaluation of RL-based regenerative braking, assessing its ability to safely and effectively overcome the conservatism of fixed-threshold systems in real-world conditions.

2 | Problem Analysis

This chapter examines electric vehicle braking systems—their principles, limitations, and control strategies—establishing the motivation for investigating Reinforcement Learning as an adaptive alternative to conventional rule-based approaches.

2.1 Electric Vehicle Braking Systems: Fundamentals and Architecture

Electric traction machines used in modern electric vehicles are inherently bidirectional: they operate as motors when delivering torque to the wheels, and as generators during braking, converting mechanical energy back into electrical energy :

$$P = T_m \cdot \omega_m \tag{2.1}$$

where
- P : instantaneous mechanical power at the motor shaft (positive in motoring, negative in generating),
- T_m : motor torque (negative during regenerative braking because it opposes wheel rotation), and
- ω_m : motor angular velocity.

This relation always holds; regenerative braking corresponds to the case where the product $T_m \cdot \omega_m$ becomes negative, indicating power flowing from the wheels back into the electrical system.
Electric vehicles satisfy total braking demand through a combination of regenerative and friction braking:

$$T_{\text{req}} = T_{\text{regen}} + T_{\text{fric}} \tag{2.2}$$

where
- T_{req} : is the total brake torque demand,
- T_{regen} : is the regenerative braking torque contribution, and
- T_{fric} : is the friction brake torque.

The ECU allocates this torque split dynamically based on vehicle state, battery conditions, driver comfort and safety constraints.

This fundamental architecture enables energy recovery during deceleration, but practical implementation requires sophisticated control strategies to manage multiple competing constraints. The following section examines current production approaches to this control problem.

2.2 Current Solution: Rule-Based Control Strategies

Production electric vehicles implement regenerative braking control using fixed if-then rule sets [3]. This section examines their logic, operational constraints, and performance characteristics.

2.2.1 Conventional Rule-Based Logic

Current systems employ threshold-based decision rules, for example:

- If battery SoC $> 90\% \rightarrow$ disable regeneration
- If battery temperature $< 5\,^\circ\mathrm{C} \rightarrow$ limits the charging current by 40%
- If high brake demand detected $\rightarrow$ prioritize friction braking
- If vehicle speed < 12 km/h $\rightarrow$ fade out regeneration

These rules guarantee predictable, safety-oriented behavior but cannot adapt to varying conditions or optimize performance dynamically.

2.2.2 Braking Mode Transitions

Production regenerative braking systems typically implement three distinct operating modes based on braking intensity, often expressed as a normalized deceleration demand Z_N [2]:

1. **Low Braking Intensity ($Z_N < 0.5$):** Regenerative motor braking provides most or all braking torque, maximizing energy recovery while keeping hydraulic pressure low [4].
2. **Moderate Braking Intensity ($0.5 \leq Z_N \leq 0.8$):** As braking intensity increases, hydraulic braking is progressively introduced to supplement regenerative braking, ensuring that deceleration demand and stability requirements are met. [1].
3. **High/Emergency Braking Intensity ($Z_N > 0.8$):** Hydraulic braking dominates to ensure rapid deceleration; regeneration is minimized or disabled so that ABS/ESC performance and stopping distance are not compromised [2].

The braking intensity Z_N determines which mode is active. This piecewise logic is a common industrial pattern to guarantee that emergency braking is never limited by motor or battery constraints, while still exploiting regeneration in low and medium braking regimes [1].

2.2.3 Operating Constraints on Regeneration

The instantaneous maximum regenerative braking torque is constrained by multiple vehicle and environmental factors, including speed, motor capability, and battery limits [17].

Vehicle Speed Limitations. Below approximately 10–15 km/h, the motor back-electromotive force (back-EMF) is insufficient to drive current into the battery, so only hydraulic braking can be used [18]. This low-speed cutoff is a fundamental physical limitation and appears explicitly in many blended braking strategies [17].

Motor Speed and Power Constraints. Maximum motor braking torque depends on motor speed. Below rated speed, the machine operates in a constant-torque region; above rated speed, it operates in a constant-power region, so torque decreases inversely with speed [19]:

$$T_{\text{max}}(n) = \begin{cases} T_{\text{max,rated}} & n \leq n_{\text{rated}} \\ \dfrac{9550\,P_{\text{rated}}}{n} & n > n_{\text{rated}} \end{cases} \tag{2.3}$$

In the context of regenerative braking, this torque–speed characteristic directly limits how much braking torque can be produced electrically at a given vehicle speed, and thus how much of the demanded deceleration can be covered by regeneration rather than friction [20]. This characteristic shapes the regenerative capability across the vehicle's speed range.

Battery SoC Restrictions: When battery SoC exceeds approximately 90−95% [21], the Battery Management System (BMS) significantly restricts or blocks charging current to prevent overcharge damage. Regenerative braking is usually disabled in this range.

Battery Temperature Constraints: Cold temperatures reduce ionic conductivity and slow lithium intercalation kinetics, which significantly limits charge acceptance. As documented in [22], regenerative charging power must be reduced when cell temperature falls below approximately 5−10°C to avoid lithium plating. At the opposite extreme, temperatures above roughly 50°C increase the risk of accelerated degradation and thermal instability, so the BMS similarly restricts charge current in high-temperature conditions.

Vehicle Dynamics and Wheel Slip: The braking force distribution must ensure wheel slip remains within safe bounds (typically 10−20%) to maintain adhesion and stability. The adhesion utilization coefficient must satisfy:

$$\mu_i = \frac{F_{X,i}}{F_{Z,i}} \leq \mu_{\text{road}} \tag{2.4}$$

where
- μ_i : is the adhesion utilization coefficient for wheel i, representing the ratio of longitudinal braking force to normal load,
- $F_{X,i}$: is the longitudinal (braking) force acting on wheel i,
- $F_{Z,i}$: is the normal (vertical) load on wheel i, and
- μ_{road} : is the maximum road–tire friction coefficient available under current surface conditions.

When Anti-lock Braking System (ABS) intervention is required, regenerative braking is overridden to ensure stability [23].

In practice, blended braking architectures rarely operate in a purely regenerative or purely hydraulic mode; instead, a small continuous friction contribution is typically maintained to ensure actuator readiness, robustness against disturbances, and predictable brake feel during transitions. This constant friction floor ensures that the hydraulic braking system remains active and pressurized, so subsystems such as ABS and ESC retain continuous feedback from the friction brakes and can intervene reliably if wheel slip or emergency braking occurs.

Driver Comfort: Driver comfort in regenerative braking is typically assessed through limits on deceleration (approximately 2.5 m/s^2 in normal braking) and jerk (approximately 3–3.5 m/s^3), which help ensure smooth and predictable braking [1]. Transitions between regenerative and friction braking can challenge these limits when motor torque decreases suddenly due to low speed or battery constraints. In this thesis, comfort is discussed only at a conceptual level, as the simplified drivetrain and actuator models do not permit realistic estimation of deceleration or jerk.

2.3 Safety and Regulatory Requirements for Electric Vehicle Braking Systems

Electric vehicle braking systems operate under strict safety and regulatory constraints to ensure passenger protection, vehicle stability, and public road safety.

2.3.1 Functional Safety and Braking as a Critical Function

Braking is the most safety-critical function in any vehicle. Unlike power or comfort systems where performance degradation may be tolerable, brake system failures present immediate and severe hazards.

Safety-Critical Design Principles: Current production braking systems are designed with the principle that braking functionality must never be compromised [24]:

- **Redundancy:** Multiple independent braking pathways ensure that single-point failures do not eliminate braking capability [25].
- **Deterministic Fallback:** The system must always revert to a safe state (mechanical braking) if any active control subsystem malfunctions, as mandated in brake-by-wire safety guidelines [26].
- **Conservative Operation:** When safety and energy efficiency conflict, safety always takes priority. Production regenerative braking strategies therefore prioritize hydraulic braking during uncertainty, ABS operation, or fault conditions [2].
- **Predictable Behavior:** Drivers and surrounding vehicles must be able to reliably predict deceleration behavior. Regulatory frameworks (UNECE Regulation 13-H) prescribe strict limits on variability in braking response [24].

Implications for Regenerative Braking: In current systems, regenerative braking is a secondary enhancement to the primary mechanical braking system—never a replacement [24]. This architectural choice simplifies safety assurance: if regenerative capability becomes unavailable due to SoP limits, low temperature, or subsystem faults, the vehicle still decelerates safely through hydraulic braking. Rule-based controllers exploit this hierarchy, enabling regeneration only when mechanical braking can immediately take over if regeneration is insufficient.

These safety requirements fundamentally shape production control strategies and explain the conservative nature of existing regenerative braking implementations. While advanced control methods may improve energy recovery, they must maintain equivalent or superior safety performance. Any deviation from deterministic behavior requires extensive validation and certification under functional safety standards (ISO 26262) [26].

These safety constraints interact directly with battery operational limits, which impose additional restrictions on regenerative braking. The following section examines battery technology and its specific constraints.

2.4 Battery Technology in Electric Vehicles

Battery systems are the primary energy storage components in electric vehicles and directly constrain how much regenerative braking energy can be safely recovered. High charging power during braking acts as a short, intense stress event on the cells; depending on chemistry and operating conditions, this can accelerate degradation through capacity loss and impedance growth [21]. The purpose of this section is therefore not to provide a full battery primer, but to motivate *why* battery limits (particularly the State-of-Power) must be explicitly considered in braking control.

In automotive applications, lithium-ion batteries are predominantly based on Nickel-Manganese-Cobalt (NMC) or Lithium Iron Phosphate (LFP) chemistries [27]. NMC offers high energy density but is sensitive to high-stress operation (high SoC, high temperature, high C-rate), while LFP is more robust but heavier and has lower energy density [22]. This difference is directly relevant to regenerative braking: aggressive braking at high SoC and elevated temperature is significantly more harmful for NMC cells, whereas LFP can tolerate more stress before aging accelerates [27].In addition to NMC and LFP, some EV and heavy-duty applications employ Lithium-Titanate (LTO) cells. LTO batteries use a lithium-titanate anode, which eliminates SEI-layer formation and enables extremely high charge/discharge rates, exceptional cycle life ($>$10,000 cycles), and excellent low-temperature performance [28]. The main drawback is their low cell-level energy density—typically less than half that of NMC or LFP—resulting in heavier battery packs [29]. As a result, LTO chemistries appear mostly in buses, industrial vehicles, and applications prioritizing power and lifetime over range. Table 2.1 summarizes these chemistries along with their key advantages and limitations.

Chemistry	Main advantages	Main limitations (RB & EV use)
NMC (Ni–Mn–Co)	High energy density (typically 150–250 Wh/kg in automotive cells) and widespread use in passenger EVs; good power capability [27].	Thermally sensitive above $\sim 40°C$; high-stress charging (high SoC, high C-rate, high temperature) accelerates capacity fade and impedance growth, requiring strict control of regenerative charging power [22].
LFP ($LiFePO_4$)	High cycle life (3000+ cycles in typical EV duty), excellent thermal stability and lower cost due to iron-based cathode materials [27].	Lower energy density (around 90–160 Wh/kg) and reduced low-temperature performance; still ages faster under combined high SoC and high C-rate braking, even though it is more tolerant than NMC [22].
LTO (Lithium Titanate)	Extremely long cycle life (>10,000 cycles), very fast charging capability and inherently low risk of lithium plating due to high anode potential [28].	Very low energy density (50–80 Wh/kg) and high cost, which restricts its use mostly to niche high-power or heavy-duty applications rather than mainstream passenger EVs [29].

Table 2.1: Overview of automotive lithium-ion chemistries and their relevance to regenerative braking, based on [27].

2.4.1 Automotive Lithium-Ion Chemistries

The remainder of this thesis focuses on behaviour typical of NMC-based packs, which are representative for contemporary passenger EVs [27].

2.4.2 State of Charge, State of Health and State of Power

State of Charge: SoC represents the remaining charge relative to a nominal capacity:

$$\mathrm{SoC}(t) = \frac{Q(t)}{Q_{\mathrm{nom}}} \times 100\% \tag{2.5}$$

where
- $Q(t)$: is the instantaneous charge stored in the battery at time t,
- Q_{nom} : is the rated (nominal) charge capacity of the battery under standard conditions.

High SoC pushes the cell voltage towards the upper limit and is known to accelerate degradation mechanisms, especially under elevated temperature [22].

State of Health (SoH): Two common definitions are:

Capacity-based:

$$\mathrm{SoH}_Q = \frac{Q_{\mathrm{current}}}{Q_{\mathrm{BOL}}} \times 100\% \tag{2.6}$$

where
- Q_{current} : is the present usable charge capacity of the battery,
- Q_{BOL} : is the battery's beginning-of-life (BOL) charge capacity.

Resistance-based:

$$\mathrm{SoH}_R = \frac{R_{\mathrm{BOL}}}{R_{\mathrm{current}}} \times 100\% \tag{2.7}$$

where
- R_{BOL} : is the internal resistance at beginning-of-life,
- R_{current} : is the current internal resistance of the battery.

Capacity-based SoH is widely used for EV applications and correlates directly with usable range [30]. Resistance-based SoH captures the increase in internal resistance that limits power capability [31].

In this thesis, SoH refers to *capacity-based* SoH, while resistance changes are captured implicitly through SoP.

State of Power (SoP): Describes the maximum charge and discharge power that the battery can sustain over a short time interval without violating its safe operating limits on cell voltage, current, temperature and SoC [16]. In other words, SoP answers how much power the pack can safely provide to, or receive from, the drivetrain at a given instant.

Experimentally, SoP is defined using controlled laboratory tests, most commonly Hybrid Pulse Power Characterization (HPPC) [32]. At a given SoC, temperature and SoH, the cell or pack is brought to equilibrium and then excited with constant-current charge and discharge pulses. From the measured voltage response, an effective internal resistance $R_{\mathrm{int}}(SoC, T, SoH)$ is identified, and the largest currents that keep the terminal voltage within $[V_{\min}, V_{\max}]$ over the pulse duration are computed as

$$I_{\mathrm{dis,max}} = \frac{V_{\mathrm{OC}} - V_{\min}}{R_{\mathrm{int}}}, \tag{2.8}$$

$$I_{\mathrm{chg,max}} = \frac{V_{\mathrm{OC}} - V_{\max}}{R_{\mathrm{int}}}, \tag{2.9}$$

which give the corresponding discharge and charge power limits

$$P_{\mathrm{dis,max}} = V_{\min}\, I_{\mathrm{dis,max}}, \qquad P_{\mathrm{chg,max}} = V_{\max}\, |I_{\mathrm{chg,max}}|. \tag{2.10}$$

Repeating this procedure across SoC, temperature and aging states yields a SoP map that serves as experimental ground truth for modelling and validation [33].

In an actual battery management system this ground truth cannot be re-measured online, so SoP is estimated analytically or numerically. Typical approaches include lookup tables interpolated from HPPC data, model-based estimators built on equivalent-circuit or electro-thermal models, and data-driven or hybrid methods that infer SoP from operating data while enforcing the same voltage, current, temperature and SoC constraints [16].

2.5 Limitations of Current Rule-Based Regenerative Braking

As discussed above, conventional rule-based strategies introduce several inherent limitations. The following subsections describe how they restrict regenerative-braking performance.

2.5.1 Fixed Conservative Limits

A central limitation of rule-based regenerative strategies is their dependence on *pre-calibrated, static limits* for motor torque and charging power. Although production EVs adjust these limits based on operating conditions, the adjustments are typically implemented through *piecewise derating rules* rather than continuous optimisation. Yin et al. [10] demonstrate that such fixed allocation strategies underperform relative to adaptive controllers, even when both use identical hardware constraints. Similarly, the work in [13] shows that fixed SoC-based limits restrict regeneration especially during long braking events.

2.5.1.1 Threshold-Based Derating Structure

The most restrictive bottleneck for regeneration is the battery's allowable charging power. Since neither the battery model used in this thesis nor the reference literature includes thermal dynamics, only SoC-dependent limitations are considered. Yin et al. explicitly neglect battery temperature in their model [10], and the SoP formulation in [13], likewise applies SoC-driven constraints even when thermal behaviour is discussed conceptually.

A simplified SoC-based derating law, consistent with commercial EV behaviour and similar to the approach used in [13], is:

$$P_{\max}(\mathrm{SoC}) = \begin{cases} P_{\max} & \mathrm{SoC} \le 0.80 \\ P_{\max}\left(\dfrac{0.95 - \mathrm{SoC}}{0.15}\right) & 0.80 < \mathrm{SoC} < 0.95 \\ 0 & \mathrm{SoC} \ge 0.95 \end{cases} \tag{2.11}$$

where
- $P_{\max}$: is the maximum regenerative charging power under nominal conditions [13],
- SoC : is the instantaneous battery SoC.

Both benchmark studies [13] observe the same real-world behaviour: full regenerative capability at moderate SoC, smooth tapering above 80%, and complete cutoff near full charge. As shown in [13], this conservative tapering can reduce recoverable braking energy by 10−20% along hilly drive cycles.

2.5.1.2 Calibration Philosophy: Conservative Thresholds

Even when additional derating logic exists, the values used in practice remain conservative. OEMs (Original Equipment Manufacturers) typically select $P_{\max}$ and $T_{\max}$ below true physical capability to allow for:

- Expected battery ageing over vehicle lifetime (SoH dropping from 100% to 80%), matching the behavior described in [13],
- Manufacturing spread in cell resistance and motor efficiency,

- Worst-case load combinations noted in [10] (e.g. high torque demand at high SoC),
- Required certification margins and deterministic behavior for safety validation.

Because these limits remain fixed, the controller cannot exploit favorable conditions such as high SoH, mid-range SoC, or optimal motor speed—conditions under which Yin et al. show that considerably more regenerative energy could be harvested.

2.5.1.3 Adaptation Opportunity

A more advanced controller could replace static thresholds with *continuous, state-dependent* limits adapted from battery and motor operating conditions:

$$P_{\max}(t) = f_P\big(\mathrm{SoC}(t), \mathrm{SoH}(t), \omega_m(t), I_{\text{history}}\big) \tag{2.12}$$

Continuous adaptation, as discussed in [13], can provide:

- smoother limit transitions without discrete jumps,
- exploitation of favourable combined states (mid-SoC, high efficiency, good thermal headroom),
- safe use of short-term over-power capability,
- improved robustness to ageing.

2.6 Artificial Intelligence and Reinforcement Learning Approach

Artificial Intelligence (AI) offers a systematic way to improve control performance based on data rather than hand-crafted calibration. In modern vehicle systems AI is already used for perception, diagnostics, energy management, and driver assistance. For control-oriented problems the most relevant area is **Machine Learning (ML)**, where algorithms learn patterns or decision rules from experience.

Three foundational learning paradigms exist, as described in [34, 9]:

- **Supervised Learning** learns a mapping from inputs to outputs using labelled datasets. It is well suited for estimation tasks (e.g., SoC estimation [30], fault detection [35]), but *fundamentally cannot* address sequential decision making.

  Why it fails for regenerative braking: Supervised learning requires a dataset of optimal control actions, but the optimal torque split at any instant depends on *future states* (upcoming braking events, battery thermal evolution, SoC trajectory). No static input-output mapping exists—the same state (e.g., 30 km/h, 60% SoC, 25°C) requires different actions depending on whether the vehicle is approaching a stop sign or entering highway traffic, as discussed in [9].

- **Unsupervised Learning** discovers structure in data without labels, e.g., clustering or dimensionality reduction. It is useful for driver-profiling or anomaly detection, but *provides no mechanism for action selection.*

  Why it fails for regenerative braking: Unsupervised learning can identify patterns (e.g., "aggressive braking cluster" vs. "gentle braking cluster") but cannot prescribe what torque to apply. The regenerative braking controller must output continuous motor torque commands—unsupervised methods offer no direct path from state observations to control actions, as noted in [9].

- **Reinforcement Learning (RL)** learns a control policy by interacting with an environment, receiving rewards or penalties in response to actions. RL directly addresses sequential decision problems where current actions influence future states and rewards, as formalized in [9] and extended to continuous control in [11].

  Why it fits regenerative braking: The controller must balance immediate energy recovery against future battery degradation, thermal constraints, and comfort. RL naturally handles:

  - *Temporal credit assignment:* A gentle braking action now may enable higher-power regeneration later (by keeping battery temperature optimal)
  - *Multi-objective tradeoffs:* The reward function can encode energy, comfort, and battery health simultaneously
  - *Continuous state and action spaces:* RL methods like DDPG handle real-valued torque commands and continuous state observations (SoC, temperature, speed), as demonstrated in [11, 36]

Key Distinction: Supervised learning asks "given this input, what is the correct output?" Reinforcement learning asks "given this state, which action leads to the best long-term outcome?" Only the second question captures the regenerative braking control problem, consistent with the perspective in [9, 37].

Why RL fits control problems

In RL, an *agent* observes a state s_t, applies an action a_t, and receives a reward r_t from the environment. Over time the agent learns a policy $\pi(a|s)$ that maximizes the expected long-term return:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \tag{2.13}$$

where:

- $s_t \in \mathcal{S}$ is the state at time t (e.g., SOC, battery temperature, vehicle speed)
- $a_t \in \mathcal{A}$ is the action at time t (e.g., regenerative motor torque command)
- $r_t = r(s_t, a_t)$ is the immediate reward received for taking action a_t in state s_t
- $\gamma \in [0, 1)$ is the discount factor determining the importance of future rewards

- $\pi(a|s)$ is the *policy*—a probability distribution (for stochastic policies) or deterministic mapping (for deterministic policies) that prescribes which action to take in each state
- $\mathbb{E}_\pi[\cdot]$ denotes expectation over trajectories generated by following policy π

The policy $\pi(a|s)$ can be interpreted as:

- **Stochastic policy:** $\pi(a|s) = P(a_t = a \mid s_t = s)$, the probability of selecting action a when in state s
- **Deterministic policy:** $a = \pi(s)$, a direct mapping from state to action (used in continuous control methods like DDPG [11])

This formulation matches the braking control problem exactly: efficiency, comfort, and battery protection must be optimized over an entire drive cycle, not instantaneously. The reward function $r(s_t, a_t)$ encodes the multi-objective tradeoff at each timestep, while the summation $\sum_{t=0}^{\infty} \gamma^t r_t$ ensures the controller considers long-term consequences of its actions. RL can learn optimal *trade-offs* directly from experience by encoding all objectives into a single reward function:

$$r(s_t, a_t) = w_1 \cdot r_{\text{energy}}(s_t, a_t) + w_2 \cdot r_{\text{comfort}}(s_t, a_t) + w_3 \cdot r_{\text{battery}}(s_t, a_t) \tag{2.14}$$

where the weights w_i reflect the relative importance of each objective.

Key Potential Advantages Over Rule-Based Control:

- **Adaptivity:** The policy $\pi(a|s)$ can potentially adjust to real-time conditions—battery temperature, SoC, vehicle speed, and even route characteristics (e.g., recognizing recurring downhill sections) [13].
- **Implicit nonlinear modeling:** The agent has the potential to learn the complex, nonlinear coupling between motor torque, vehicle speed, battery SOC, temperature, and efficiency without requiring explicit analytical models. If successful, it could discover that motor efficiency varies from 85% at low speed/high torque to 95% at optimal operating points, and adjust actions accordingly [11].
- **Long-term optimization:** Through the discount factor γ, the agent aims to optimize cumulative recovered energy over the entire drive cycle, not instantaneous torque at each timestep. This could enable strategic decisions such as: *"Apply gentle braking now to keep battery temperature optimal, enabling higher-power regeneration during the upcoming steep descent"* [9].
- **Exploitation of transient margins:** Unlike rule-based thresholds that cannot distinguish between sustained and transient loads, RL could potentially learn to briefly exceed continuous power limits during short braking events without violating safety margins. [13].
- **Safe training environment:** High-fidelity Simulink models provide a reproducible, risk-free environment for exploration and policy learning, avoiding the safety risks and hardware wear associated with on-vehicle trial-and-error learning [11].

These properties make RL a promising candidate for control problems where:

1. Optimal control policy structures are unknown or difficult to derive analytically
2. System dynamics are heavily state-dependent and nonlinear
3. Multiple conflicting objectives must be balanced in real-time

The regenerative braking problem exhibits all four characteristics, motivating the deep reinforcement learning approach explored in this work.

2.7 Deep RL Algorithms Relevant to Vehicle Control

The RL framework presented in Section 2.6 encompasses a broad family of algorithms. Selecting an appropriate algorithm requires understanding the key trade-offs between different approaches, particularly for continuous control problems like regenerative braking.

2.7.1 Algorithm Families for Continuous Control

Modern deep RL algorithms fall into two main families, as discussed in [9, 11]:

Value-Based Methods (e.g., DQN)

Value-based algorithms learn a Q-function $Q(s, a)$ that estimates the expected return for taking action a in state s. The policy is then derived by selecting the action with the highest Q-value:

$$\pi(s) = \arg\max_a Q(s, a)$$

Limitations for regenerative braking:

- Designed for *discrete* action spaces (e.g., $a \in \{0, 1, 2, ..., N\}$)
- Applying them to continuous control requires discretizing the action space into a finite set of bins (e.g., torque share $= \{0.0, 0.1, 0.2, ..., 1.0\}$)
- **Problem:** Coarse discretization (e.g., 10 bins) loses precision; fine discretization (e.g., 100 bins) causes combinatorial explosion in the $\arg\max$ operation, as highlighted in [11]
- The regenerative torque share $a \in [0, 1]$ is inherently continuous—discretization sacrifices control smoothness and introduces artificial jerk

Conclusion: Value-based methods are unsuitable for continuous torque control.

Policy-Gradient and Actor-Critic Methods

These algorithms directly parameterize the policy $\pi_\theta(a|s)$ with neural network weights θ, and optimize it using gradient descent. They naturally handle continuous action spaces by outputting real-valued actions.

Key algorithms for continuous control:

- **DDPG (Deep Deterministic Policy Gradient)**, as introduced in [11]:
  - Outputs a *deterministic* action $a = \pi_\theta(s)$ (no sampling required)
  - Uses an actor network (policy) and critic network (Q-function) trained simultaneously
  - Employs experience replay and target networks for stable training
  - **Best for:** Smooth, deterministic environments with low stochasticity
- **TD3 (Twin Delayed DDPG)**, proposed in [12]:
  - Addresses DDPG's tendency to overestimate Q-values through three improvements:
    1. *Clipped double Q-learning*: Uses two critic networks and takes the minimum to reduce overestimation bias
    2. *Delayed policy updates*: Updates the actor less frequently than critics to reduce policy variance
    3. *Target policy smoothing*: Adds noise to target actions to prevent exploitation of Q-function errors
  - **Best for:** Noisy or stochastic environments where Q-value overestimation is problematic
- **PPO (Proximal Policy Optimization)**, introduced in [38]:
  - Uses a *stochastic* policy that outputs a probability distribution over actions
  - Constrains policy updates to prevent destructive large changes (trust region optimization)
  - **Best for:** Problems requiring exploration or handling multi-modal action distributions

To provide a concise overview of the algorithms discussed above, Table 2.2 summarises their characteristics, advantages, and limitations.

Algorithm	Brief Description	Pros	Cons
DQN	Value-based RL; learns $Q(s, a)$ for discrete actions.	Simple, stable, well-understood.	Not suitable for continuous actions; discretisation causes precision loss and combinatorial growth.
DDPG	Deterministic actor–critic method for continuous actions.	Fast convergence in deterministic systems; continuous outputs; replay buffer + target networks stabilise training.	Sensitive to noise; prone to Q-value overestimation in imperfect environments.
TD3	Extension of DDPG with clipped double Q-learning, delayed updates, and target policy smoothing.	More robust and stable than DDPG; mitigates overestimation issues.	Higher computational cost; slower training due to two critics.
PPO	Stochastic policy-gradient method with clipped objective for stable updates.	Robust under uncertainty; strong exploration; handles complex action distributions.	Less efficient for deterministic control; slower in high-frequency control loops.

Table 2.2: Summary of Deep RL Algorithms for Continuous Control

2.7.2 Algorithm Selection for Regenerative Braking

For this project, the focus is on deterministic, continuous-action methods such as DDPG and TD3, as the regenerative braking torque distribution is naturally represented as a continuous variable $a \in [0, 1]$. The selection between DDPG and TD3 requires careful consideration of their respective strengths in this context.

DDPG vs TD3 Trade-offs

TD3 is superior in noisy or stochastic environments where Q-value overestimation leads to divergence, as demonstrated in [12]. However, for a deterministic physics-based model with smooth state transitions (such as the Simulink EV-drivetrain model), **DDPG converges reliably and is significantly faster**, as originally reported in [11].

The key trade-off is:

- **TD3**: More robust to stochasticity and overestimation, but requires $\sim 2\times$ critic evaluations (due to double Q-learning) and 30–50% longer training time
- **DDPG**: Faster convergence in deterministic settings, but can be unstable if environment dynamics are noisy

Since the Simulink model used in this work is deterministic (no sensor noise, tire slip, or ABS interventions in the baseline implementation), DDPG's speed advantage outweighs TD3's robustness benefits.

2.8 Approach Used in This Thesis

The RL agent is not replacing the entire braking system. Instead, it learns the **decision layer** that determines the regenerative torque share. Safety fallbacks (hydraulic braking, SoP limits, stopping distance constraints) remain active in the environment.

MATLAB/Simulink is selected as the development platform for the EV drivetrain model, due to its native support for continuous-time dynamics, signal routing, and controller implementation. The Reinforcement Learning Toolbox seamlessly integrates with Simulink models, enabling the agent to interact directly with the same environment used for conventional control design. This integration ensures the setup is both reproducible.

The workflow is:

1. Build a physics-based EV model in Simulink.
2. Expose the relevant variables (speed, torque demand, SOC, SoP limits) to the RL agent.
3. Define a reward function combining energy recovery and smoothness.
4. Train the agent over different driving cycles and evaluate performance.

2.9 Required Fidelity of the Drivetrain Model

To evaluate reinforcement learning for regenerative braking, the simulation must capture the key physics of an electric vehicle, but not every subsystem. The goal is a realistic yet computationally efficient drivetrain model that preserves dominant energy flows, actuator limits, and battery constraints.

The model serves two purposes: (1) to provide a realistic environment for RL exploration, and (2) to reproduce core interactions between motor torque, vehicle dynamics, battery power limits, and efficiency.

High-fidelity details like tyre dynamics or ABS logic are omitted to keep simulations fast, as they do not significantly impact the learned braking strategy. Instead, the model focuses on:

- Longitudinal vehicle dynamics,
- Motor/inverter efficiency,
- Battery SoC and real-time charging power limits,
- Blending of regenerative and friction braking,
- Speed-dependent regeneration fade-out.

This abstraction captures the critical nonlinearities affecting braking performance—efficiency variations, SoP constraints, and torque distribution—while keeping the learning loop tractable. The objective is a *representative* EV model, sufficient for developing and evaluating adaptive torque-distribution strategies and demonstrating RL's benefits over rule-based control.

3 | Problem Statement

This chapter defines the problem statement, laying the foundation for the rest of the project.

Through the problem analysis, reinforcement learning was identified as a promising approach to overcome the limitations of conventional braking control and enable adaptive, data-driven decision-making. This leads to the following problem statement:

"How can DDPG-based adaptive braking control strategy intelligently distribute torque between regenerative and friction braking to maximize energy recovery, while ensuring safety and protecting battery health?"

4 | Model development

This chapter develops a simulation-efficient electric drivetrain model for the Škoda Enyaq iV 80, forming the physical backbone for all subsequent regenerative-braking control experiments.

The purpose of this chapter is to construct a complete and simulation-efficient EV drivetrain model that can support the regenerative-braking controllers developed later. Because the study compares a rule-based strategy and an RL agent under identical physical constraints, the model must be both transparent in structure and sufficiently accurate in its dominant energy flows. The chapter therefore proceeds in a layered manner. Section 4.1 defines the modelling requirements; Section 4.2 introduces a generic EV architecture containing the driver, motor controller, motor–inverter block, vehicle body, and battery subsystems. Section 4.3 then specialises this template to the Škoda Enyaq iV 80 by providing manufacturer data, derived parameters, and calibration procedures. Sections 4.4 and 4.5 describe the two data-driven components of the model—the motor efficiency map and the table-based battery with its SoP interface—while Section 4.6 evaluates the resulting drivetrain against key performance figures of the reference vehicle. The chapter concludes with the modelling limitations and a justification of the chosen level of fidelity.

4.1 Requirements

To focus on the fundamental dynamics needed for realistic regenerative braking, a set of minimum requirements is defined for the drivetrain model. The objective is to capture the dominant energy flows and actuator limits that affect braking performance, while avoiding unnecessary complexity. The following components and effects must therefore be represented explicitly in the model:

- Electric traction motor with the capability to operate in both motoring and generating modes.
- Battery system with SoC enabling representation of charging power limitations.
- Friction brake system capable of providing the remaining braking torque when regenerative capacity is insufficient or unavailable.
- Equivalent inertial load representing the vehicle mass at the wheels.
- Road load and environmental resistances acting on the wheels, including

- aerodynamic drag,
- rolling resistance,
- viscous (speed-proportional) friction, and
- slope or road grade.

To satisfy these requirements with minimal structural complexity, the drivetrain is implemented as a traction motor driving a single-speed gearbox, which in turn drives an equivalent rotational inertia representing the vehicle mass. A friction brake acts on the same shaft, and the combined road loads are converted into an equivalent opposing torque at the wheel, an illustration is shown in 4.1. This configuration provides a compact but physically meaningful basis for studying braking torque allocation, energy recovery, and battery loading in the subsequent control design and reinforcement learning experiments.

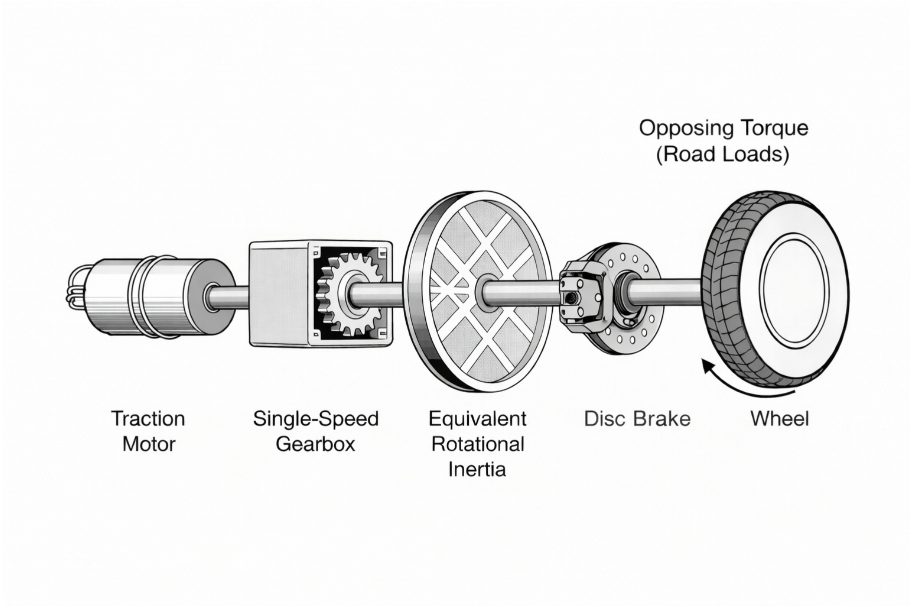


Figure 4.1: Illustration of the simplified model an EV drivetrian.

4.2 EV Modeling

The drivetrain model is organized into the following core subsystems, which are also displayed in Figure 4.2.

- Driver (Driver)
- Controller (Motor_controller)
- Motor and inverter electrical dynamics (Motor_Calculations)
- Vehicle body and motor dynamics (EV_body)
- Battery system (Battery)

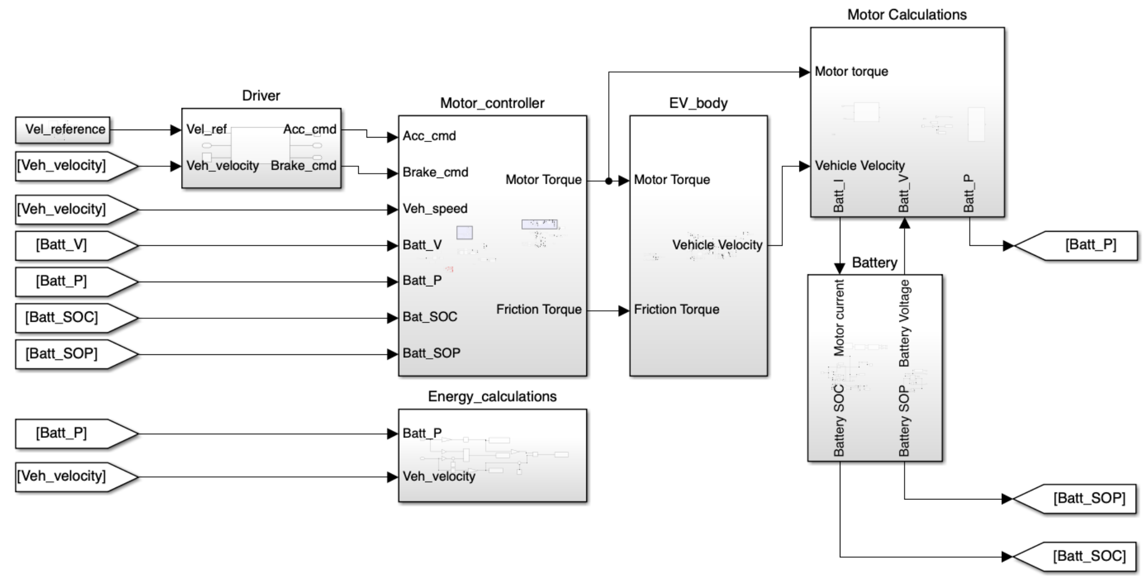


Figure 4.2: Simplified version of the Simulink model. Original version is shown in appendix Figure D.1

Each subsystem is described in detail in the following sections. For all *MATLAB Function* blocks used in the model, as defined in the Simulink documentation [39], the corresponding MATLAB code is provided Appendix chapter C.

4.2.1 Longitudinal Driver

This simulation employs the *Longitudinal Driver* block from Simulink, as described in the official documentation [40], to automate velocity tracking and emulate driver pedal inputs. The block utilizes a Proportional-Integral (PI) control strategy to generate normalized acceleration and braking commands based on the difference between a reference velocity (v_{ref}) and the measured vehicle velocity (v). Two input signals are required: the interpolated reference velocity and real-time vehicle velocity feedback. The resulting outputs are mapped to pedal commands for acceleration and braking.

Figure 4.3 shows the configuration and interconnections of the driver block within the overall Simulink model.

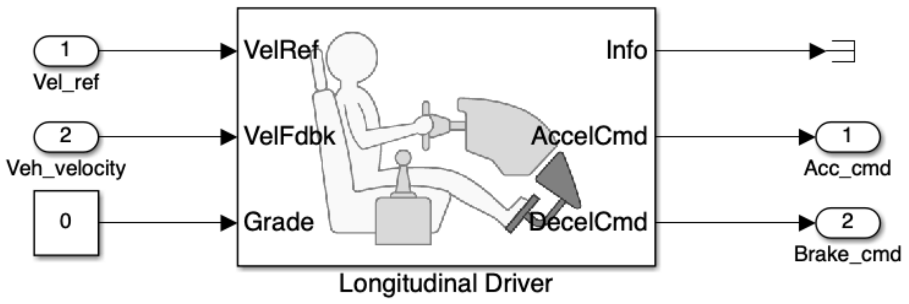


Figure 4.3: Simplified Simulink "Driver" subsystem. Original version is shown in appendix Figure D.2

The PI controller logic implemented in the block is expressed as:

$$y = \frac{K_{ff}}{v_{nom}} v_{\text{ref}} + \frac{K_p e_{ref}}{v_{nom}} + \int \left(\frac{K_i e_{ref}}{v_{nom}} + K_{aw} e_{out} \right) dt + K_g y' \tag{4.1}$$

where:

$$e_{ref} = v_{\text{ref}} - v \tag{4.2}$$

$$e_{out} = y_{sat} - y \tag{4.3}$$

The output y is limited to the admissible range $[-1, 1]$ using:

$$y_{sat} = \begin{cases} -1 & y < -1 \\ y & -1 \leq y \leq 1 \\ 1 & y > 1 \end{cases} \tag{4.4}$$

Pedal command signals are derived as follows:

$$y_{acc} = \begin{cases} 0 & y_{sat} < 0 \\ y_{sat} & 0 \leq y_{sat} \leq 1 \\ 1 & y_{sat} > 1 \end{cases} \qquad y_{dec} = \begin{cases} 0 & y_{sat} > 0 \\ -y_{sat} & -1 \leq y_{sat} \leq 0 \\ 1 & y_{sat} < -1 \end{cases} \tag{4.5}$$

The main variables and gains are defined in the nocmelcemateure

By providing repeatable and deterministic driver commands in response to a given velocity profile, the Longitudinal Driver block establishes a consistent basis for evaluating and comparing energy recovery strategies and battery management approaches throughout the study.

4.2.2 Motor Controller

The motor controller is responsible for translating accelerator and brake pedal inputs into appropriate drive and braking torque commands. For motoring, the controller computes the available torque envelope based on motor speed and power limits, applying a smooth taper to avoid abrupt transitions. For braking, the controller determines the torque split between regenerative and friction braking, taking into account speed-dependent fade, brake intensity, and battery power constraints. The computed torque commands are then fed directly into an efficiency map, which calculates the corresponding electrical power for accurate energy accounting. Additionally, these torque signals are input to an *Ideal Torque Source* block, as defined in the Simulink documentation [41], which acts as the motor in the drivetrain model by translating mechanical torque commands into motion. The implementation, illustrated in Figure 4.4, is described in detail below.

Further details and the construction method for the efficiency map are described in Section 4.2.3.

Motoring: Accelerator Command to Drive Torque

The accelerator input, acc_cmd, is a normalized value in $[0, 1]$ representing throttle de-

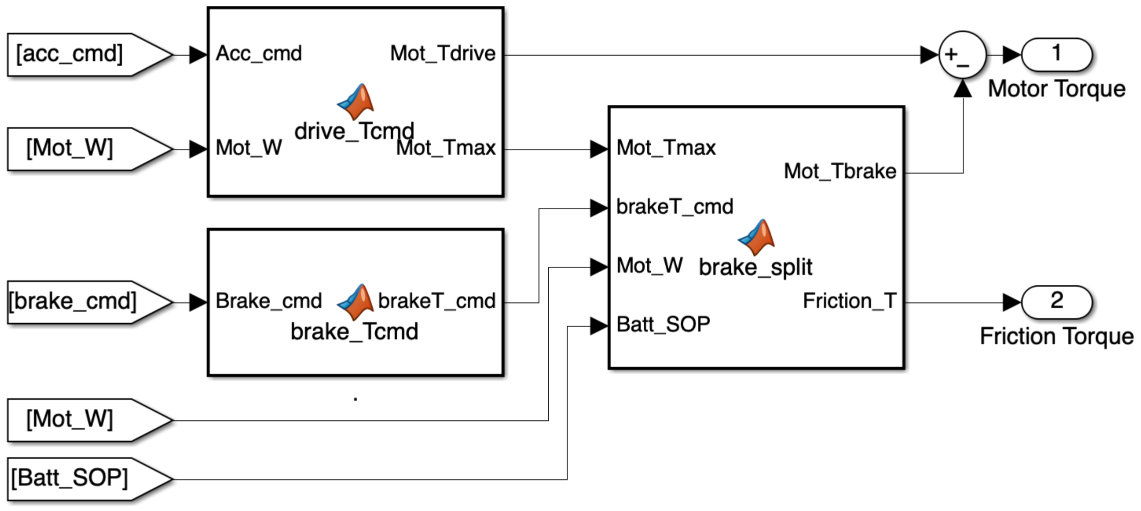


Figure 4.4: Simplified Simulink "Motor controller" subsystem. Original version is shown in appendix Figure D.3 and D.4

mand. This input is scaled to compute the motoring torque envelope $T_{\max}$ as follows:

$$T_{\max} = \min(T_{maxmot}, \frac{P_{maxmot}}{|w_{\text{mot}}|}) \tag{4.6}$$

where T_{maxmot} is the motor's peak allowed torque, P_{maxmot} is the peak power output, and w_{mot} is the current motor speed (rad/s).

To avoid an abrupt torque cutoff at the no-load speed, the maximum motoring torque is multiplied by a smooth taper function. Let w_{taper} denote the speed where the reduction starts and w_{no} the no-load speed. The taper factor is defined as

$$\text{taper}(w_{\text{mot}}) = \begin{cases} 1, & |w_{\text{mot}}| < w_{\text{taper}}, \\ \dfrac{1}{2}\left(1 + \cos\left(\dfrac{\pi\left(|w_{\text{mot}}| - w_{\text{taper}}\right)}{w_{\text{no}} - w_{\text{taper}}}\right)\right), & w_{\text{taper}} \le |w_{\text{mot}}| < w_{\text{no}}, \\ 0, & |w_{\text{mot}}| \ge w_{\text{no}}. \end{cases} \tag{4.7}$$

The middle branch of (4.7) is constructed as a standard raised–cosine (cosine-taper) window [42], chosen to obtain a smooth transition between 1 and 0. First, the motor speed is normalised to

$$z = \frac{|w_{\text{mot}}| - w_{\text{taper}}}{w_{\text{no}} - w_{\text{taper}}},$$

so that $z = 0$ at the onset of tapering $|w_{\text{mot}}| = w_{\text{taper}}$ and $z = 1$ at the no-load speed $|w_{\text{mot}}| = w_{\text{no}}$. Over this interval, the function

$$\frac{1}{2}(1 + \cos(\pi z))$$

decreases smoothly from 1 (at $z = 0$) to 0 (at $z = 1$), with zero slope at both endpoints. Substituting the expression for z into this cosine expression yields the middle case in (4.7).

The effective maximum motoring torque envelope then becomes

$$T_{\max}(w_{\text{mot}}) = T_{\max}(w_{\text{mot}}) \cdot \text{taper}(w_{\text{mot}}). \tag{4.8}$$

so that the available torque is constant up to w_{taper}, smoothly decreases to zero between w_{taper} and w_{no}, and is zero beyond the no-load speed. The actual motoring torque request sent to the drivetrain is then

$$T_{\text{drive\_cmd}} = \text{acc\_cmd} \cdot T_{\max}(w_{\text{mot}}). \tag{4.9}$$

Braking: pedal input and torque split

The rule-based braking in the EV model follows the structure described in Section 2.2. The brake pedal input brk_cmd is a normalised signal in the range $[0, 1]$, where 0 corresponds to no braking and 1 to a full-brake request. The total commanded brake torque is

$$T_{\text{br,cmd}} = brk\_cmd \cdot T_{\text{br,max}}, \tag{4.10}$$

with $T_{\text{br,max}}$ denoting the maximum available braking torque of the vehicle. This commanded torque is distributed between regenerative torque T_r and friction torque T_f with:

Speed-dependent fade of regeneration. For the reasons explained in Section 2.2.3, regenerative braking is progressively reduced as motor speed decreases. To model this effect, a scalar fade factor is defined as

$$\text{fade}(w) = \begin{cases} 0, & w \le w_{\text{off}}, \\ \dfrac{w - w_{\text{off}}}{w_{\text{full}} - w_{\text{off}}}, & w_{\text{off}} < w < w_{\text{full}}, \\ 1, & w \ge w_{\text{full}}, \end{cases} \tag{4.11}$$

so that regeneration is disabled at low speed, increases linearly between w_{off} and w_{full}, and reaches full strength above w_{full}.

Brake-intensity share. The brake command determines the nominal share of the requested torque that may be provided regeneratively:

$$\text{share}(brk\_cmd) = \begin{cases} 1, & brk\_cmd \le 0.5, \\ \dfrac{0.8 - brk\_cmd}{0.3}, & 0.5 < brk\_cmd \le 0.8, \\ 0, & brk\_cmd > 0.8. \end{cases} \tag{4.12}$$

For light braking ($brk\_cmd \le 0.5$) the request is fully assigned to regeneration. Between 0.5 and 0.8 the regenerative share decreases linearly, and for hard braking ($brk\_cmd > 0.8$) the friction brakes are assumed to provide the entire additional torque.

Battery power limit and final split. The maximum admissible regenerative torque is limited by the battery charging power and motor speed.

$$T_{r,\text{cap,pow}} = \frac{P_{\text{cap}}}{\eta\, w}, \tag{4.13}$$

where P_{cap} is the battery-side charging power limit and η is the regenerative efficiency. The overall regenerative torque cap is

$$T_{r,\text{cap}} = \text{fade}(w)\ \min\bigl(T_{\max},\, T_{r,\text{cap,pow}}\bigr), \tag{4.14}$$

and the commanded regenerative and friction torques become

$$T_r = \text{share}(Brk\_cmd)\ \min\bigl(T_{\text{br,cmd}},\, T_{r,\text{cap}}\bigr), \qquad T_f = T_{\text{br,cmd}} - T_r. \tag{4.15}$$

4.2.3 Motor

In this study, the electric motor is represented using an efficiency map and an *Ideal Torque Source* block, as defined in the Simulink documentation [41], rather than a detailed physical model of a specific machine type, such as a permanent magnet synchronous motor (PMSM) or induction motor. This efficiency map directly relates motor operating conditions (such as torque and speed) to efficiency values, enabling calculation of power losses and actual output without modeling the underlying electrical and magnetic processes. The primary motivation for this approach is to simplify the simulation while retaining accuracy in the assessment of energy flows and system performance. This is illustrated in Figure 4.5 and is explained in detail below.

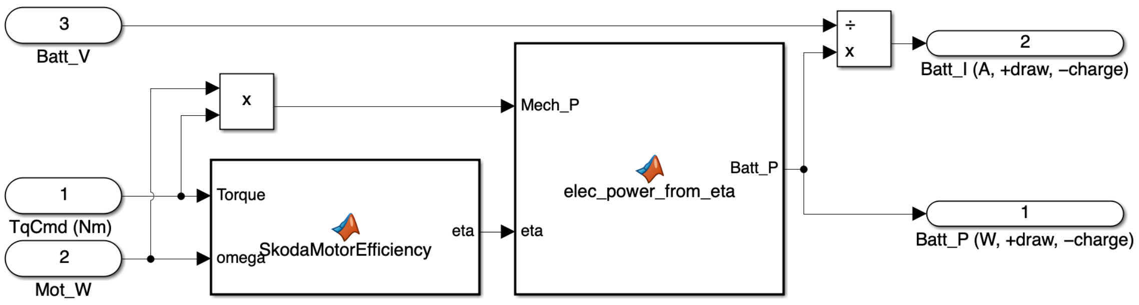


Figure 4.5: Simplified Simulink "Motor calculation" subsystem. Original version is shown in appendix Figure D.5 and D.6

4.2.3.1 Inputs and outputs.

The block receives

- T_{qCmd} [Nm]: motor torque command (positive motoring, negative regeneration),
- ω_{mot} [rad/s]: motor rotational velocity,
- V_{batt} [V]: battery terminal voltage.

and produces

- P_{mech} [kW]: mechanical power at the shaft,
- P_{DC} [W]: power at the DC bus (positive draw, negative charge),
- I_{batt} [A]: battery current,
- η_{now} [–]: instantaneous motor–inverter efficiency.

4.2.3.2 Internal computation.

For a given operating point $(\omega_{\text{mot}}, T_{\text{qCmd}})$ the map implements:

$$P_{\text{mech}} = \omega_{\text{mot}} T_{\text{qCmd}}, \tag{4.16}$$

$$\eta = \eta(\omega_{\text{mot}}, T_{\text{qCmd}}), \tag{4.17}$$

$$P_{\text{DC}} = \begin{cases} \dfrac{P_{\text{mech}}}{\eta}, & P_{\text{mech}} \geq 0 \quad \text{(motoring)}, \\ -\eta P_{\text{mech}}, & P_{\text{mech}} < 0 \quad \text{(regeneration)}, \end{cases} \tag{4.18}$$

$$I_{\text{batt}} = \frac{P_{\text{DC}}}{V_{\text{batt}}}. \tag{4.19}$$

The sign convention is such that $P_{\text{DC}} < 0$ and $I_{\text{batt}} < 0$ during regenerative braking (charging).

4.2.3.3 Map structure.

The efficiency map is implemented as:

$$\eta = f(\omega_{\text{mot}}, T_{\text{qCmd}}), \tag{4.20}$$

Where, at each point $(\omega_{\text{mot}}, T_{\text{qCmd}})$ a scalar efficiency value η is computed. The final implemented efficiency map is described in section 4.4

4.2.4 EV Body

The vehicle body and drivetrain are modeled as a single-speed, two-wheel-drive electric powertrain: an EV motor drives an inertial disc (representing the car mass) through a gearbox. This subsection details the modeling of the inertial load, frictional brakes, gearbox, and resistive torques that together define the dynamics of the vehicle. This implementation, illustrated in Figure 4.6, is described in detail below.

4.2.4.1 Single-Axle Representation

Most electric vehicles distribute braking torque across both front and rear axles, with the front axle typically generating a larger share of braking force due to load transfer during deceleration. In this model, only a single axle is represented and the load transfer is neglected. This simplification is intentional and justified by two considerations.

1. Regenerative braking architecture: In many production EVs, the traction motor is mounted on a single axle. Regenerative braking torque can only be produced on the motor-driven axle, whereas the non-driven axle contributes exclusively through friction brakes. Modeling one driven axle therefore captures the complete physical pathway for energy recovery without omitting any regenerative mechanism.

2. Reduction of model complexity: Including both axles would require modeling dynamic load transfer, brake-force proportioning, separate wheel-speed dynamics, and potentially ABS/ESC coordination. These additions increase state dimensionality without

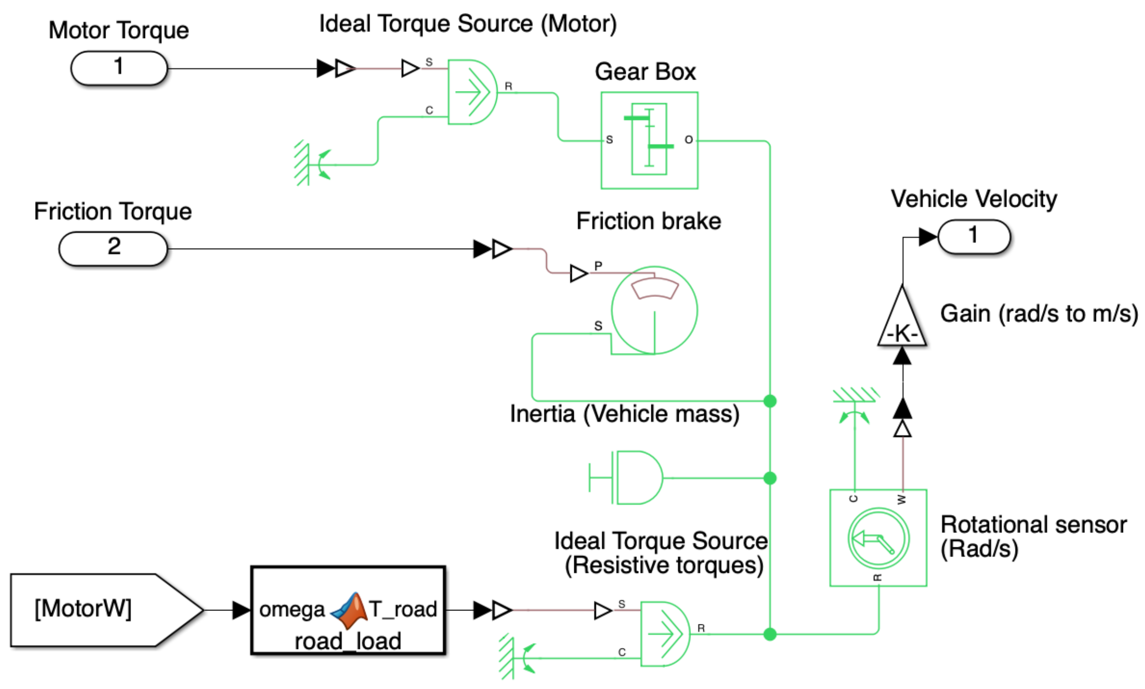


Figure 4.6: Simplified Simulink EV body subsystem. Original version is shown in appendix Figure D.7

improving insight into regenerative braking behaviour, whose primary bottleneck is battery charge-power acceptance rather than multi-axle blending logic.

Accordingly, the full vehicle mass and all longitudinal road loads are mapped to an equivalent single wheel. This preserves the correct longitudinal dynamics while enabling focused analysis of energy flow, torque blending, and control strategies for regenerative braking.

4.2.4.2 Inertial Load

The total vehicle inertia is represented using the *Inertia* block from Simulink, as defined in the documentation [43]. The equivalent rotary inertia applied at the wheel shaft is computed as:

$$J_{\text{Vehicle}} = M_{\text{Vehicle}} \cdot R^2_{\text{Wheel}} \tag{4.21}$$

where M_{Vehicle} is the vehicle mass and R_{Wheel} is the effective wheel radius.

4.2.4.3 Frictional Brake

The brake system is modeled using the *Disc Brake* block from Simulink, as defined in the documentation [44]. A normalised brake command $brk\_cmd \in [0, 1]$ is scaled to a hydraulic pressure P (in Bars), calibrated such that $brk\_cmd = 1$ corresponds to a severe braking or full-stop event.

Internally, the block computes the friction torque based on the applied pressure P, the actuator bore diameter D_b, the mean pad radius R_m, the number of pads N, and the kinetic friction coefficient μ_k. For a rotating wheel ($\omega \neq 0$), the brake torque is

$$T = \frac{\mu_k \, P \, \pi \, D_b^2 \, R_m \, N}{4}. \tag{4.22}$$

4.2.4.4 Gearbox

A fixed-ratio single-speed gearbox is represented using the *Gearbox* block from Simulink, as defined in the documentation [45]. The gearbox connects the motor to the wheel and transmits speed and torque between them. Gearbox efficiency is neglected for simplicity. The core mechanical relationship is:

$$G = \frac{\omega_m}{\omega_w} = \frac{T_w}{T_m} \tag{4.23}$$

where ω_m (T_m) and ω_w (T_w) are the angular velocity (torque) of the motor and wheel, respectively.

4.2.4.5 Resistive Torques

All external road loads— aerodynamic drag, rolling resistance, viscous friction, and grade resistance—are combined into a single resistive torque. These are computed within a *MATLAB Function* block, as defined in the Simulink documentation [39], and summed into the drive shaft via an *Ideal Torque Source* block, also defined in the Simulink documentation [41].

This approach ensures all external effects are applied as a single reaction to wheel motion, simplifying the dynamic model.

The vehicle velocity V and wheel angular velocity ω_w are related by:

$$V = \omega_w \cdot R_{\text{Wheel}} \tag{4.24}$$

The resistive components are calculated as:

Aerodynamic Drag

$$T_d = \frac{1}{2} \rho \, C_d \, A \, R_{\text{Wheel}} \, V^2 \tag{4.25}$$

where ρ is air density [kg/m^3], C_d is the drag coefficient, A is frontal area [m^2], R_{Wheel} is wheel radius, and V is vehicle velocity.

Rolling Resistance

$$T_{rr} = C_{rr} \, M_{\text{Vehicle}} \, g \, R_{\text{Wheel}} \cos(\theta) \cdot \text{sgn}(\omega_w) \tag{4.26}$$

where C_{rr} is rolling resistance coefficient, g is gravitational acceleration, and
$\theta = \tan^{-1}(\text{grade}\,[\%]/100)$ converts percentage grade to radians. A positive θ models uphill, negative θ models downhill.

Viscous Friction

$$T_v = b_{\text{vis}} \, V \tag{4.27}$$

where b_{vis} is a viscous damping constant.

Slope (Grade)

$$T_g = M_{\text{Vehicle}} \, g \, R_{\text{Wheel}} \, \sin(\theta) \tag{4.28}$$

Total Reactive Torque

$$T_{\text{road}} = -(T_{rr} + T_g + T_d + T_v) \tag{4.29}$$

The negative sign indicates opposition to vehicle movement.

For numerical stability near zero speed, a smoothed sign function based on the hyperbolic tangent is used:

$$\text{sgn}(\omega_w) = \tanh\left(\frac{\omega_w}{\omega_0}\right) \tag{4.30}$$

Where $\omega_0 = 0.5$. This suppresses chattering and ensures a smooth, continuous transition from standstill to motion.

4.2.5 Battery

The battery in this study is modeled using the *Battery (Table-Based)* block from Simscape Electrical, as defined in the documentation [46]. This block implements an equivalent-circuit battery model whose parameters are obtained from tabulated data rather than from a fixed analytic formula. In particular, the block uses lookup tables to describe the open-circuit voltage (OCV), internal resistance, and optional dynamic resistive–capacitance (RC) elements as functions of SoC and, if provided, cell temperature and current direction. A more detailed description, including the particular tables and operating strategy, is provided in Section 4.5. Figure 4.7 shows the battery subsystem implemented in Simulink.

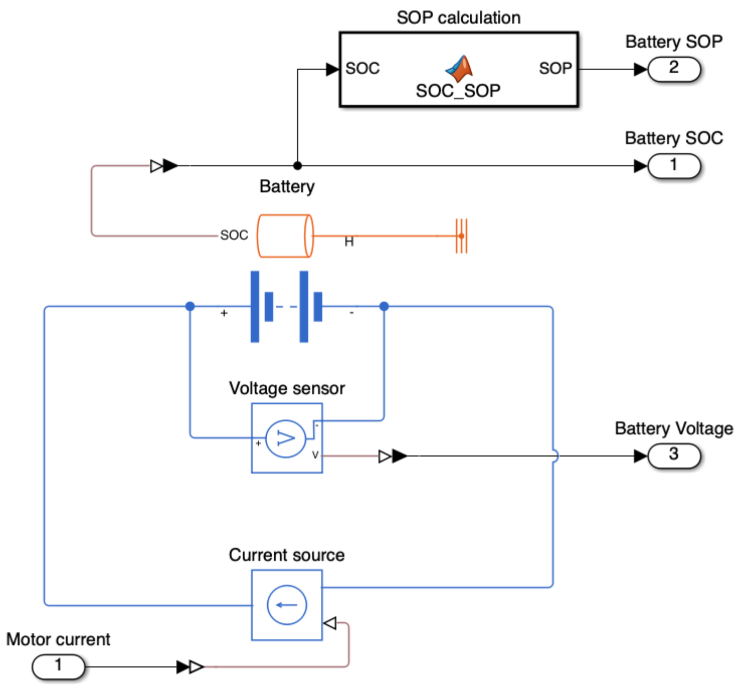


Figure 4.7: Simplified Simulink "Battery" subsystem. Original version is shown in appendix Figure D.8

4.3 Reference Vehicle

Figure 4.8: Skoda Enyaq iV 80

In this work, the reference vehicle is chosen as a Škoda Enyaq iV 80 with a 150 kW drivetrain. This model is one of the most widely sold battery-electric vehicles in Denmark in 2024–2025 and is therefore representative of a typical mid-size family EV in the current market [47]. A Škoda Enyaq iV 80 is shown in Figure 4.8. The main vehicle and powertrain parameters used in the simulation model are summarised in Table 4.1.

Category / Parameter	**Value / Unit**
Powertrain	
Motor type	PMSM (Permanent Magnet Synchronous Motor)
Maximum power	150 kW
Maximum torque	310 N·m
No-load speed	14.000 rpm (estimation see section 4.3.1)
Gearbox	Single-speed reduction, ≈10.5:1 ratio
Battery System	
Battery type	Li-ion (NCM high-voltage DC)
Configuration	96s3p
Nominal voltage	352 V
Usable capacity (net)	77 kWh (82 kWh gross)
Performance	
Top speed	160 km/h
Acceleration (0–100 km/h)	8.7 s
Energy consumption (WLTP combined)	159 – 180 Wh/km
Stopping distance (130-0 km/h)	80 m
Vehicle Body and Aerodynamics	
Drag coefficient (C_d)	0.255–0.277
Frontal area (A)	≈ 2.54 m^2 (estimated from width $\times$ height)
Wheel radius	0.358 m (235/55 R19 tire)
Kerb weight (incl. driver)	2107–2148 kg

Table 4.1: Key technical parameters of the Škoda Enyaq iV 80 (RWD). [48], [49], [50],

4.3.1 Škoda Enyaq Simulink model

The general EV model structure is described in Section 4.2. For the Škoda Enyaq iV 80, some of the required model parameters (such as vehicle mass, battery capacity, and rated motor power) are taken directly from manufacturer and database specifications, while others (such as drivetrain efficiency and motor no-load speed) are estimated to match typical values reported in the literature and public data sources [51, 19]. These parameter choices are described first, and the resulting Škoda Enyaq iV 80 parameter set is then applied to the general model structure to obtain the specific Simulink implementation used for all subsequent simulations.

Motor no-load speed:
The motor no-load speed represents the maximum rotational speed of the traction machine under negligible mechanical load. As detailed, manufacturer data for the Enyaq iV 80 traction motor are not publicly available, this quantity is approximated from the specified vehicle top speed and the fixed gear ratio. For a top speed of approximately $v_{\max} = 160$ km/h, a wheel radius of $r = 0.358$ m, and a single-speed reduction ratio of $G = 10.5{:}1$, the corresponding top motor speed is

$$\omega_{m,160} = \frac{v_{\max}}{r} G = \frac{160/3.6}{0.358} \cdot 10.5 \approx 13{,}000 \text{ rpm.} \tag{4.31}$$

In practice, the no-load speed must exceed the operating speed at the aerodynamic top-speed point, since aerodynamic drag and drivetrain losses limit the achievable vehicle speed before the motor reaches its true unloaded speed. In this work, the motor no-load speed is therefore set to 14,000 rpm, corresponding to an overspeed margin of approximately 8–10 % relative to the estimated top-speed operating speed.

4.3.1.1 Frictional brake modelling

As described in Section 4.2.4.3, the friction brakes are implemented using the Simscape *Frictional Brake* block, as defined in the official documentation. The block is parameterised by the mean pad radius, cylinder bore, and number of brake pads. In the absence of detailed caliper data for the reference vehicle, these parameters are selected to represent a typical front-axle disc brake for a mid-size EV, with mean pad radius $r_{\text{pad}} = 0.14$ m, cylinder bore diameter $d_{\text{bore}} = 0.057$ m, and two brake pads per caliper ($N_{\text{pads}} = 2$).

The input to the *Frictional Brake* block is hydraulic pressure, whereas the high-level brake command in the vehicle model is a normalised signal in the range 0–1. To obtain a realistic mapping between this normalised command and the physical brake torque, a simple calibration procedure is applied. First, a target stopping distance from 130 km/h to 0 km/h is defined, consistent with representative braking data reported in [52] (≈ 80 m). The vehicle model is then driven to a steady speed of 130 km/h, and a full brake command (*brk_cmd* = 1) is applied. A scalar gain is inserted between the normalised brake signal and the brake pressure input, and this gain is tuned such that the simulated stopping distance matches the 80 m target. In the final model, a gain of 180 yields a stopping distance of ≈ 80 m, and the resulting brake torque is ≈ 4400 Nm.

This calibration effectively defines the maximum friction brake torque available in the simulation. It is further assumed that any higher equivalent braking torque demand would lead to wheel slip and thus lie outside the admissible operating region of the tyre–road interface.

4.4 Electric Motor Efficiency Map

An exact analytical efficiency map for the Škoda Enyaq iV 80 drive motor is not available from manufacturer data. To enable realistic performance and energy simulations, a general polynomial efficiency mapping approach is adopted following the method proposed in

[19]. This method is physically grounded and widely used for practical electric machine modelling when detailed test data is unavailable.

4.4.1 Theoretical Basis of the Electric Motor Efficiency Map

The loss function model is based on the observation that the power loss P_{loss} in an electrical machine can be described as a sum of terms, each involving integer powers of torque (T) and speed (ω), as originally proposed in [19]:

$$P_{loss}(T, \omega) = \sum_{m,n} k_{mn} T^m \omega^n \tag{4.32}$$

where k_{mn} are fitting coefficients for each term, and m, n are integers.

This polynomial approach has a solid physical basis. In a typical surface-mounted permanent magnet (PM) machine, as analysed in [53]:

- No-load iron (eddy-current) losses are proportional to ω^2.
- Copper losses are proportional to T^2 (torque is proportional to current).
- Cross-terms and higher-order effects (windage, field-weakening) may be included for accuracy.

Thus, for first-order loss estimation, one writes:

$$P_{loss}(T, \omega) = k_{20} T^2 + k_{02} \omega^2 \tag{4.33}$$

as a simple approximation. For more accurate modelling, higher-order polynomial and cross-terms are added as justified by finite element calculations.

4.4.2 Polynomial Loss Surface and Normalisation

The efficiency map in this work is constructed by synthesizing a polynomial loss model in per-unit (pu) coordinates, scaled to the Skoda motor's rated values:

$$T_{\text{pu}} = \frac{T}{T_b}, \quad \omega_{\text{pu}} = \frac{\omega}{\omega_b} \tag{4.34}$$

where $T_b = 310\,\text{Nm}$ and $\omega_b = 2\pi \times 14000/60\,\text{rad/s}$ (motor peak specifications).

Using the polynomial form proposed in [19], a custom set of coefficients is synthesised in this work to emulate the loss behaviour of a high-power surface PM traction machine:

$$\begin{aligned} P_{\text{loss,pu}} = {} & -0.002 + 0.175\,\omega_{\text{pu}} - 0.065\,T_{\text{pu}} + 0.181\,\omega_{\text{pu}}^2 + 0.577\,T_{\text{pu}}\,\omega_{\text{pu}} + 0.697\,T_{\text{pu}}^2 \\ & +0.443\,\omega_{\text{pu}}^3 - 0.542\,T_{\text{pu}}^2\,\omega_{\text{pu}} - 1.043\,T_{\text{pu}}\,\omega_{\text{pu}}^2 + 0.942\,T_{\text{pu}}^3 \end{aligned} \tag{4.35}$$

The actual machine loss in watts is obtained by scaling the per-unit loss with a loss base power:

$$P_{\text{loss}} = P_{\text{loss,pu}} \cdot P_{\text{loss,base}}. \tag{4.36}$$

In this work, $P_{\text{loss,base}}$ is set to 9 kW. This value is chosen to be consistent with typical loss levels reported for high-power EV traction machines, where rated efficiencies in the range of 95–97 % imply loss fractions of approximately 3–5 % of the mechanical output power at high load [19, 54]. For a 150 kW drive, this corresponds to a loss range of roughly 4.5–7.5 kW at high torque and speed. Choosing $P_{\text{loss,base}} = 9$ kW therefore provides a slightly conservative margin while remaining consistent with published efficiency data for automotive traction motors.

4.4.3 Efficiency Calculation and Plot

Mechanical power is calculated as in equation 2.1, repeated here for convenience:

$$P = T_m \cdot \omega_m$$

The instantaneous efficiency is then:

$$\eta(T_m, \omega_m) = \frac{P}{P + P_{\text{loss}}} \tag{4.37}$$

Figure 4.9 illustrates the resulting efficiency map for the Skoda motor across its operating range.

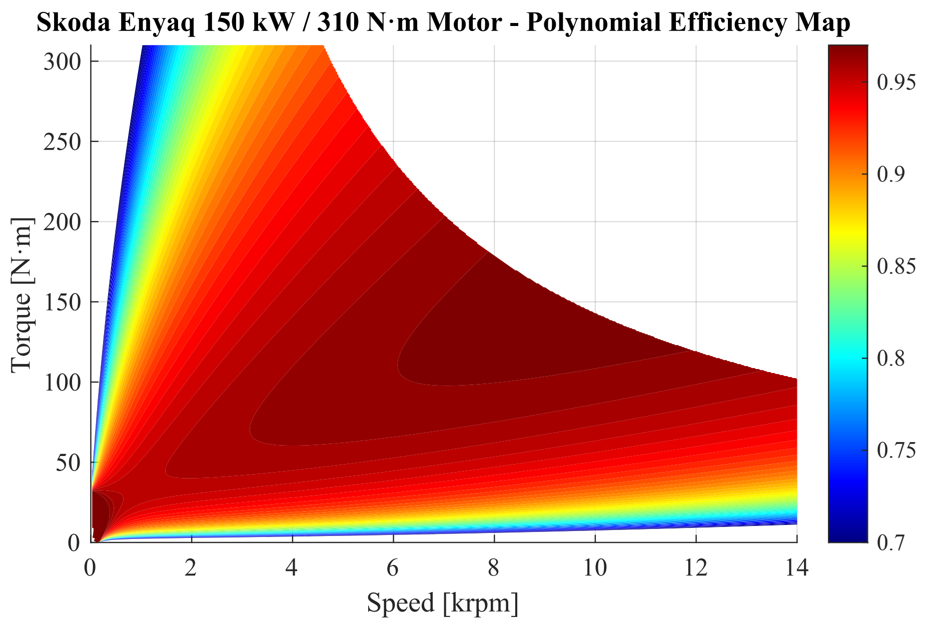


Figure 4.9: Skoda Enyaq 150 kW Motor – Polynomial Model Efficiency Map.

4.5 Battery Model Implementation

The battery block used in the Simulink model implements a static equivalent-circuit model in which the terminal voltage is computed from

$$U = \text{OCV}(z, T) - R_0(z, T, i)\, i, \tag{4.38}$$

where z is the SoC, T is the temperature and i is the current, With the open-circuit voltage (OCV) and internal resistance R_0 obtained from user-supplied lookup tables. No dynamic voltage behaviour (e.g. RC pairs) or electrothermal coupling is included, matching the simplified modelling strategy adopted for the rest of the drivetrain.

The model is parameterised over a 1D SOC grid

$$z \in [0.0,\ 0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5,\ 0.6,\ 0.7,\ 0.8,\ 0.9,\ 1.0],$$

4.5.0.1 Open-Circuit Voltage Table

The OCV curve is based on a representative NMC cell discharge characteristic typical of high-energy automotive cells [55, 30]. The per-cell open-circuit voltage (OCV) values used in this work are

$$\mathrm{OCV}_{\mathrm{cell}}(z) = [3.34,\ 3.53,\ 3.59,\ 3.63,\ 3.66,\ 3.70,\ 3.75,\ 3.83,\ 3.92,\ 4.03,\ 4.13]\,\mathrm{V},$$

as illustrated in Figure 4.10.

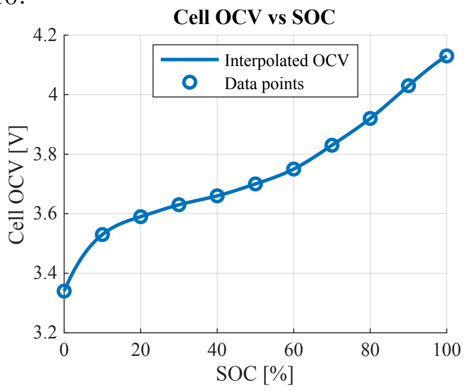


Figure 4.10: OCV(SOC) data used in the EV drivetrain model

To obtain pack-level voltages for the 96s3p configuration, the OCV table is scaled by the series cell count:

$$\mathrm{OCV}_{\mathrm{pack}}(z) = 96 \cdot \mathrm{OCV}_{\mathrm{cell}}(z).$$

This yields terminal voltages in the range $[240, 403]$ V, consistent with public data for the Enyaq iV 80 battery pack.

4.5.0.2 Internal Resistance Table

The internal resistance model follows the same SoC grid as the OCV table and reflects the typical U-shaped dependence of cell resistance on SoC. The per-cell discharge resistance values are

$$R_{0,\mathrm{cell}}(z) = [1.50,\ 1.20,\ 1.00,\ 0.90,\ 0.85,\ 0.80,\ 0.80,\ 0.85,\ 1.00,\ 1.20,\ 1.50]\,\mathrm{m\Omega}$$

as illustrated in Figure 4.11.

Because the pack consists of 96 series cells and three parallel strings, the pack-level resis-

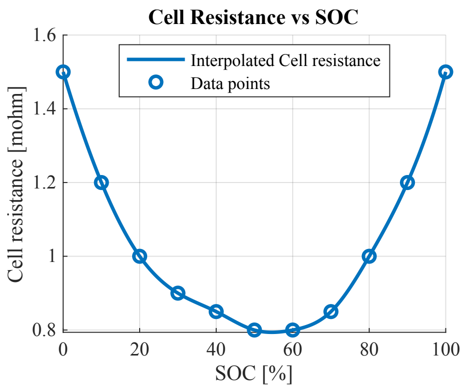


Figure 4.11: R0(SOC) data used in the EV drivetrain model

tance during discharge is

$$R_{0,\text{pack}}(z) = \frac{96}{3}\,R_{0,\text{cell}}(z).$$

To reflect the higher resistance observed during charging, as reported in [27], the charge table is scaled by an additional factor of 1.05:

$$R_{0,\text{charge}}(z) = 1.05\,R_{0,\text{pack}}(z).$$

Battery Electrical Behaviour and Role in Control

The battery subsystem is represented by a static equivalent-circuit model,

$$U = \text{OCV}(z) - R_0(z)\,i, \tag{4.39}$$

$$P_{\text{batt}} = U\,i, \tag{4.40}$$

with $i > 0$ during discharge and $i < 0$ during regenerative operation. This formulation captures SoC-dependent voltage and resistance behaviour while remaining numerically efficient for drivetrain-level simulations.

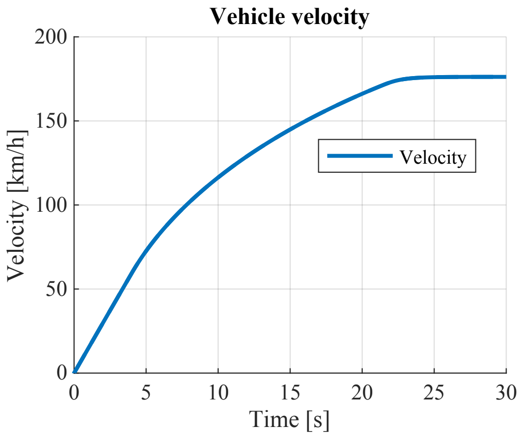
The battery block provides the SOC to the control architecture but does not compute regenerative power limits. These are obtained from an external PNGV-based State-of-Power estimator using the battery SoC and OCV, as described in Section 5.1.1, and are used to constrain regenerative braking.

4.6 Simulation results

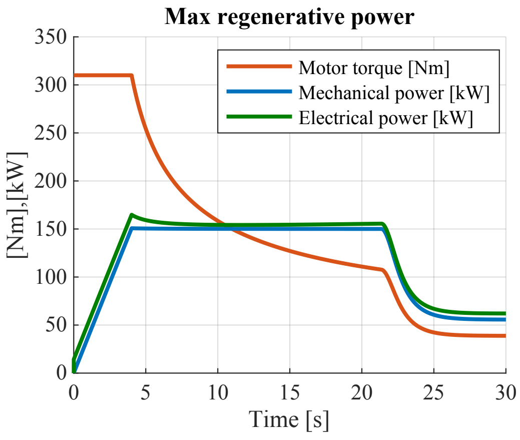
The Simulink drivetrain model is evaluated against key performance figures of the reference vehicle to assess whether its behaviour is sufficiently realistic for subsequent regenerative braking studies. The comparison focuses on 0–100 km/h acceleration time, achievable top speed, maximum motor torque, and energy consumption over a WLTP Class 3 drive cycle.

4.6.1 Acceleration, top speed and maximum torque.

A full-acceleration test is carried out by applying a constant accelerator command of 1.0, corresponding to a fully pressed pedal. The resulting vehicle speed and motor torque trajectories are shown in Figures 4.12a and 4.12b. The simulated 0–100 km/h time is approximately 7.5 s, the steady-state top speed is about 176 km/h, and the peak motor torque is 310 Nm. Relative to the nominal Škoda Enyaq iV 80 specifications, this corresponds to deviations of roughly 11 % in acceleration time, 9 % in top speed, and negligible deviation in maximum torque. These differences are considered acceptable for the system-level analysis carried out in this work.



(a) Simulated 0–100 km/h acceleration and top-speed behaviour of the EV model.



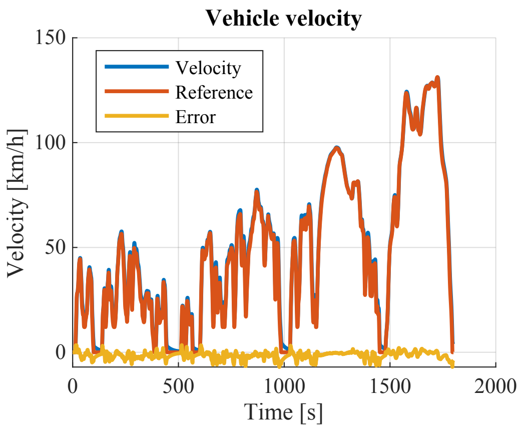
(b) Simulated motor torque during full-acceleration test.

Figure 4.12: Results from acceleration test

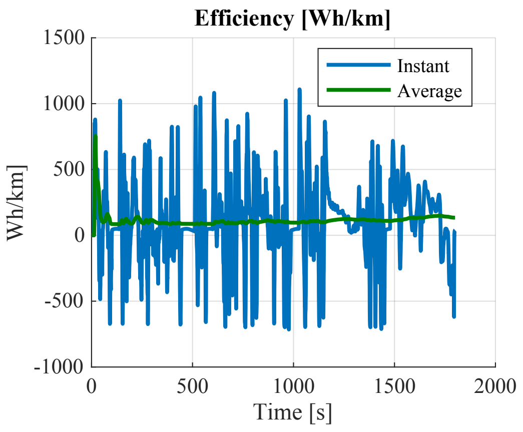
4.6.2 Energy consumption (WLTP).

To assess the overall energy efficiency, the model is driven over the WLTP Class 3 drive cycle for its full duration of approximately 30 min. The corresponding vehicle speed and specific energy consumption are shown in Figures 4.13a and 4.13b. The simulated net energy consumption is 137 Wh/km, which lies somewhat below the 159–180 Wh/km range reported for the real vehicle under WLTP conditions. Part of this deviation can be attributed to modelling assumptions such as standard air density (ambient temperature) and the omission of auxiliary loads (cabin heating, lighting, infotainment, etc.), which increase the measured consumption in a real car. Given these simplifications, the simulated value is regarded as satisfactory for the purposes of drivetrain and braking-control evaluation.

Additional drivetrain dynamics. For completeness, additional internal variables (such as motor electrical quantities and resistive shaft torques) are plotted to illustrate the dynamic behaviour of the drivetrain. This is shown in appendix chapter A.



(a) Simulated velocity profile for the WLTP Class 3 drive cycle.



(b) Simulated specific energy consumption over the WLTP Class 3 drive cycle.

Figure 4.13: Results from WLTP evaluation test

4.7 Limitations of the EV Modeling Approach

Despite achieving good physical realism and simulation speed, the electric vehicle (EV) modeling methodology is subject to several important limitations. These stem both from practical constraints (data availability, computational cost) and from deliberate modeling choices made to balance fidelity with tractability and efficiency.

4.7.1 Surrogate Efficiency Maps vs. Measured Data

The motor efficiency maps employed in this work are surrogate, polynomial-based loss models derived from literature trends (see Subsection 4.4.2), rather than manufacturer-calibrated or experimentally measured datasets. While they reproduce the dominant efficiency characteristics—such as the efficiency island and increasing losses at high torque or speed—they do not capture machine-specific design effects and are not calibrated against dynamometer data. As a result, absolute loss and energy estimates may deviate from real hardware, particularly near the operational boundaries where analytical polynomials may under- or over-predict losses [19].

4.7.2 Omission of Secondary Effects

The drivetrain model captures the dominant loss mechanisms but omits several second-order effects that are commonly neglected in longitudinal energy studies, as discussed in [27, 33]. In particular, thermal dynamics and temperature-based derating, long-term aging and degradation, auxiliary electrical loads (e.g., HVAC and infotainment), and detailed mechanical nonlinearities such as tyre slip and speed-dependent gearbox losses are not explicitly modeled. The impact of auxiliary electrical loads on overall energy management has been widely studied in the literature [51]. These effects are either represented through simple aggregate terms or neglected where their influence on regenerative-braking energy recovery is expected to be limited.

4.7.3 Simulation Environment vs. Real-World Operation

The proposed control strategies are evaluated in a simulation environment, but several real-world aspects remain unrepresented. In particular, the controllers are not subjected to embedded hardware constraints such as sensor noise, sampling jitter, or limited computational resources. Moreover, standardized drive cycles, while suitable for repeatable benchmarking, do not capture the full variability of real driving, nor are rare events such as component faults, communication delays, or extreme transients explicitly modeled. These factors may influence performance and robustness in real-vehicle deployment.

5 | Solution

This chapter describes the development of the reinforcement learning controller and the idealised rule-based controller used as its benchmark.

5.1 Rule-Based Baseline and Constraint Structure

The rule-based braking controller follows the same deterministic structure and region-based logic described in Section 4.2.2. In particular, the braking-region thresholds and the prioritisation of regenerative versus friction braking are implemented exactly as previously defined and are not repeated here. The available regenerative power is constrained by a time-varying power cap P_{cap}, which is estimated via the State-of-Power (SoP) formulation.

The only implementation-specific detail introduced in this chapter concerns the speed-dependent regeneration fade and the constant small friction brake share. Regenerative torque is linearly reduced between 100 rpm and 600 rpm, and a constant small friction torque is set to 0.5 Nm.

Battery charging power cap (SoP-based):
The rule-based controller is designed to operate under the same physical charging constraints as the learning-based agent. To ensure a fair comparison, both controllers use a time-varying SoP (SoP) limit that depends on the instantaneous battery state of charge. At each time step, the allowable regenerative charging power is given by

$$P_{\text{cap}}(t) = P_{\text{SoP}}(\text{SoC}(t)).$$

This dynamic cap reflects the real battery's ability to accept charge and is consistent with State-of-Power–based constraint modelling approaches reported in the literature [55, 56, 16, 57]. By applying the same envelope to both controllers, differences in performance can be attributed to their respective control strategies rather than differences in allowable limits.

Since the charging power is physically limited to 150 kW, the controller power cap P_{cap} is set to 120 kW,

$$P_{\text{cap}}(t) = \min\left(P_{\text{SoP}}(t),\, 120 \text{ kW}\right),$$

ensuring that the effective SoP limit is determined by the controller logic rather than by the hardware constraint.

With this modification, the rule-based controller operates under the same dynamic constraints as the RL agent. This establishes a fair and controlled baseline: any observed differences in energy recovery or torque distribution arise from the controllers' decision-making strategies rather than from discrepancies in available regenerative power. The following section describes how the RL agent is formulated within this shared physical envelope.

5.1.1 HPPC PNGV-Based State-of-Power Limit Used in This Work

As introduced in the Problem Analysis chapter, the SoP defines the maximum admissible battery charge and discharge power at a given operating point. In this work, only the charging capability is considered. The SoP is implemented using the conventional PNGV HPPC method, as summarised by Plett [55].

This estimation assumes a static equivalent-circuit relation at cell level,

$$U = \mathrm{OCV}(z) - R\,i, \tag{5.1}$$

with a constant internal resistance R and a SoC–dependent open-circuit voltage $\mathrm{OCV}(z)$, where $z = \mathrm{SoC} \in [0,1]$ denotes the normalised battery state of charge. Given fixed terminal-voltage bounds $V_{\min}$ and $V_{\max}$, the admissible charge current is computed as

$$i_{\mathrm{chg}}^{\min} = \frac{\mathrm{OCV}(z) - V_{\max}}{R}. \tag{5.2}$$

These current limits are scaled to pack level using the series–parallel configuration yielding

$$P_{\mathrm{chg}}^{\min} = n_s n_p\, \eta\, V_{\max}\, i_{\mathrm{chg}}^{\min}, \tag{5.3}$$

Where n_s is the number of cells in series, n_p is the number of cells in parlallel, and η is an optional efficiency factor. The resulting SoP as a function SoC is displayd in Figure 5.1

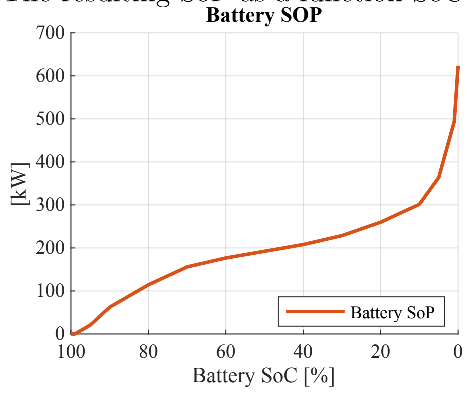


Figure 5.1: PNGV HPPC SoP method

The available regenerative charging capability is then defined as

$$P_{\mathrm{cap}} = -P_{\mathrm{chg}}^{\min}\ (>0),$$

and is supplied as a time-varying constraint to both the rule-based controller and the RL agent.

This formulation deliberately neglects dynamic voltage effects, SoC-dependent resistance, and thermal constraints. The resulting SoP signal is smooth, and computationally lightweight, making it well suited for large-scale RL training.

5.2 RL Control Formulation

This section details the DDPG agent formulation, including the actor–critic network architectures, the action definition and control hierarchy, and the observation vector that enables the RL agent to learn energy-optimal braking allocation while operating within the same SoP-constrained envelope as the rule-based baseline.

5.2.1 Overall Architecture Choice

The agent is implemented as a Deep Deterministic Policy Gradient (DDPG) controller, as introduced in Section 2.7. DDPG combines an actor network (policy) and a critic network (state–action value function) for continuous-action control tasks, as originally proposed in [11] and implemented in the MATLAB Reinforcement Learning Toolbox [58]. Both the actor and critic are realised as deep feedforward neural networks (fully connected multilayer perceptrons). Training uses soft-updated target networks to stabilise temporal-difference learning, as specified by the DDPG algorithm.

In physical terms, the actor output represents the regenerative torque fraction used to split braking demand between motor and friction brakes.

5.2.2 Actor Network Design

The actor network maps the observation vector to a single continuous control action a_t. Its architecture follows the standard DDPG actor design commonly adopted in the literature, as introduced in [11], and is defined as follows:

- **Input layer:** Feature input layer with dimensionality equal to the state vector.
- **Hidden layers:** Two fully connected layers with 256 neurons each and ReLU activation functions.
- **Output layer:** A fully connected (FC) layer with one neuron, followed by a `tanh` activation and a scaling layer that maps the bounded output to the admissible regenerative torque range in Simulink.

This architecture represents a conventional DDPG actor. ReLU hidden layers provide sufficient representational capacity and stable optimisation, while the `tanh`-bounded output ensures continuous actions remain within predefined limits. The 256–256 layer configuration was selected as a moderate-capacity setting that trained reliably without additional tuning, consistent with architectural choices commonly reported in the DDPG literature [11]. During training, exploration noise is added to the actor output, whereas at inference the deterministic policy is used.

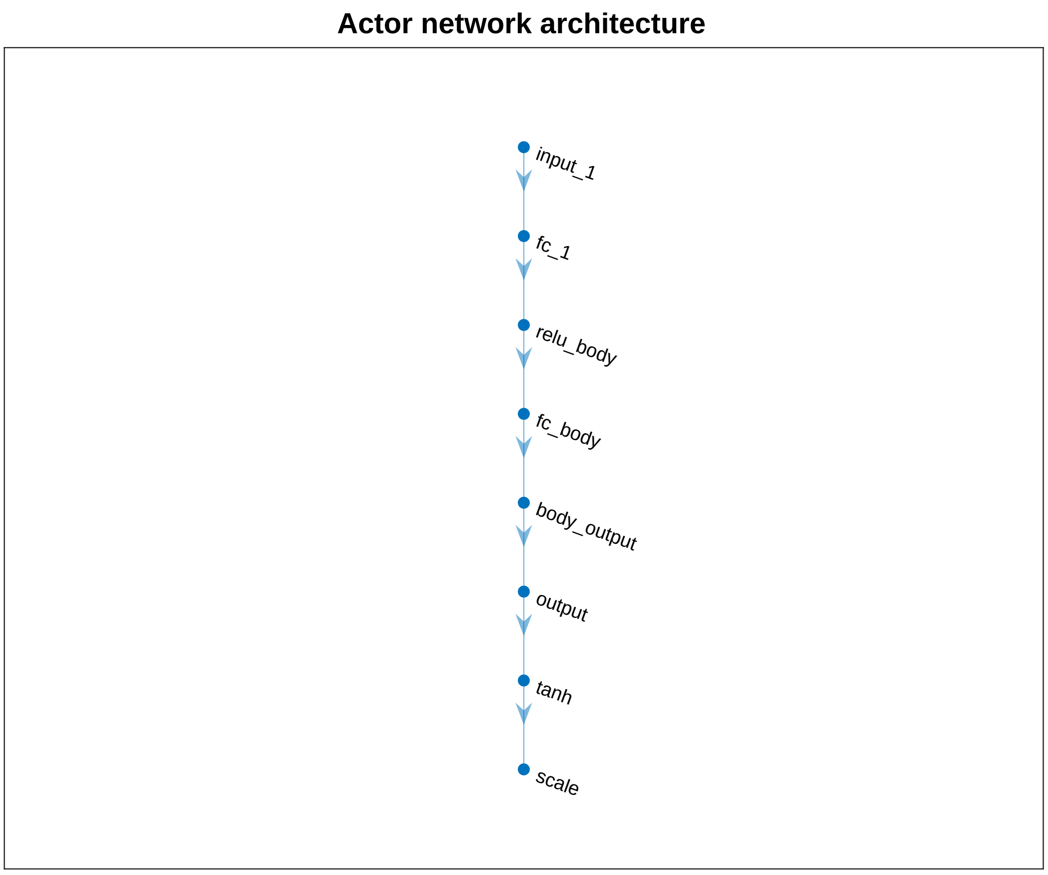


Figure 5.2: Actor network topology exported from MATLAB. The network is a feedforward MLP with two 256-neuron ReLU layers and a `tanh`-bounded output followed by action scaling.

5.2.3 Critic Network Design

The critic network estimates the state–action value $Q(s_t, a_t)$, i.e. how good it is to take action a_t in state s_t in terms of expected future reward. To do this, it processes the state and the action in two separate paths before combining them:

- **State stream:** state input $\rightarrow$ fully connected layer (512 neurons) $\rightarrow$ ReLU.
- **Action stream:** action input $\rightarrow$ fully connected layer (32 neurons) $\rightarrow$ ReLU.
- **Merged pathway:** concatenation of the two streams $\rightarrow$ fully connected (512) $\rightarrow$ ReLU $\rightarrow$ fully connected (256) $\rightarrow$ ReLU $\rightarrow$ fully connected (1).

Intuitively, the critic first builds an internal representation of the state (e.g. speed, SoC, braking demand) and a separate representation of the action (requested torque split). After merging these, the larger hidden layers allow the network to learn how state and action interact, including nonlinear effects such as the sharp change in value when the action approaches the SoP limit.

The final layer is linear and outputs a single scalar Q-value without any bounding. This is required in temporal-difference learning, where the critic must be able to represent both large positive and large negative returns.

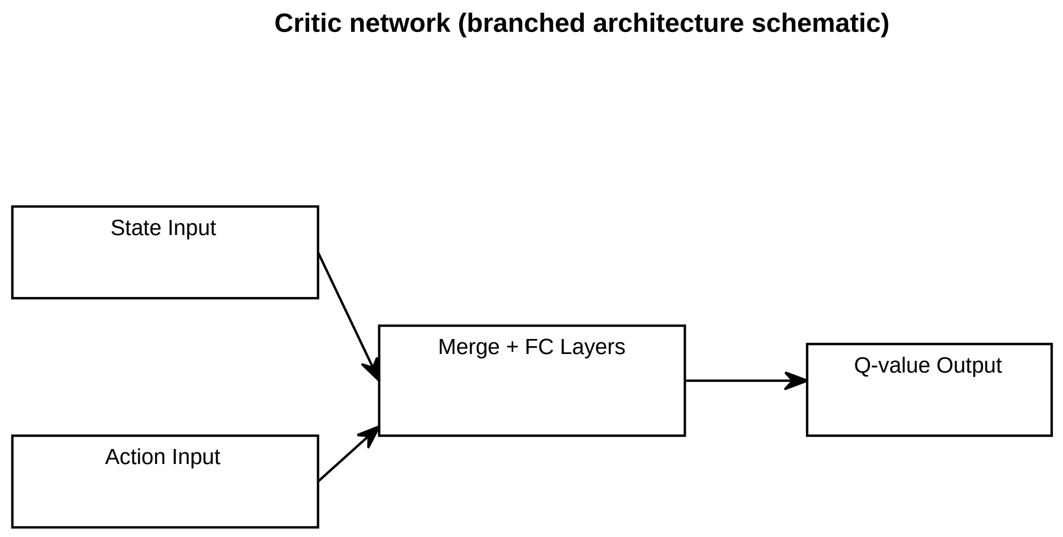


Figure 5.3: Critic network topology exported from MATLAB. The merged block consists of two fully connected layers with 512 and 256 neurons, respectively.

The architecture follows the DDPG critic structure recommended in the Reinforcement Learning Toolbox documentation, with separate state and action streams that merge into a common network [58]. The layer sizes are slightly increased to provide sufficient capacity, while still remaining close to the tested default configuration rather than being the result of ad-hoc trial-and-error.

5.2.4 Action Definition and Control Hierarchy

The RL action a_t represents a regenerative torque fraction:

$$T_{\text{regen}} = a_t \, T_{\text{req}}, \qquad T_{\text{fric}} = T_{\text{req}} - T_{\text{regen}}.$$

The RL controller operates under the same externally imposed physical constraints as the rule-based strategy. The **braking torque demand** T_{req} is fixed from outside the agent and is not influenced by the RL policy. These constraints are not learned; the RL agent only decides how to shape the regenerative braking command inside this constrained action space.

5.3 Simulink Integration and Reward Function

This section describes how the DDPG agent is integrated into the nonlinear EV drivetrain model in Simulink and how the reward function is constructed. The reward is designed to balance two objectives: (i) maximising regenerative energy recovery under a time-varying SoP-based charging limit, and (ii) producing smooth, realistic control actions.

5.3.1 Integration in Simulink

The RL agent interacts with the full nonlinear EV drivetrain model through a dedicated *RL Agent* block in Simulink [59].

The observations and reward are generated within the MATLAB function `makeObsReward()`. The actor outputs a single continuous action a_t, representing the fraction of the braking request allocated to regenerative torque. This is passed to a torque-splitting block that converts a_t into regenerative and friction torque commands applied to the plant model. Transitions $(\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1})$ are stored in replay memory for off-policy DDPG learning.

This integration ensures that the RL agent is evaluated under the same physical constraints and drivetrain dynamics as the rule-based controller, making comparisons between the two meaningful and unbiased.

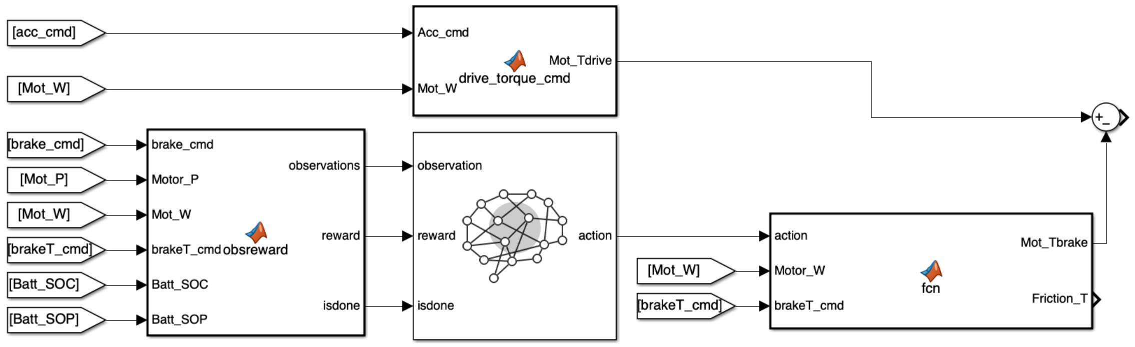


Figure 5.4: Simplification of Integration of observation, reward computation, and actor output in Simulink. The original is shown in appendix figure D.4

5.3.2 Reward Function Design

The reward function guides the RL agent to use as much safe regenerative braking power as possible, while avoiding sudden changes in the control signal. This section explains how the observations and reward are constructed from the physical signals.

Inputs and Preprocessing

The function `makeObsReward` receives the following signals from the Simulink environment:

- brake command $Brake\_cmd \in [0, 1]$,
- electrical motor power P (positive during regeneration),
- motor speed n_{rpm},
- total braking torque request T_{req},
- battery state of charge $SoC \in [0, 1]$,
- battery charging power limit P_{cap} (SoP),
- current action a_{act} and previous action a_{prev} (both $\in [0, 1]$).

All inputs are first clamped to physically meaningful ranges (for example, SoC is limited to $[0, 1]$, torques and powers are forced to be non-negative). This avoids numerical issues and prevents the agent from seeing impossible values.

Observation Vector

The agent observes a six-dimensional state vector $\mathbf{o} \in \mathbb{R}^6$:

$$\begin{aligned}
o_1 &= Brake\_cmd, \\
o_2 &= \min\left(\frac{P_{\text{ee}}}{P_{\text{cap,max}}}, 1\right), \\
o_3 &= \min\left(\frac{n_{\text{rpm}}}{n_{\text{base}}}, 1\right), \\
o_4 &= \min\left(\frac{T_{\text{req}}}{T_{\text{rated}}}, 1\right), \\
o_5 &= SoC, \\
o_6 &= \min\left(\frac{P_{\text{cap}}}{P_{\text{cap,max}}}, 1\right).
\end{aligned}$$

Here, $n_{\text{base}} = 4000$ rpm and $T_{\text{rated}} = 300$ Nm are normalisation constants, and $P_{\text{cap,max}}$ is the maximum observed SoP value. All components are scaled to approximately $[0, 1]$, which improves learning stability and keeps the different physical quantities comparable for the neural networks.

Energy-Use Term

The main reward component, r_{energy}, encourages the agent to use the available regenerative power without greatly exceeding the SoP limit. To this end, the ratio

$$x = \frac{P_{\text{ee}}}{P_{\text{cap}} + \varepsilon}$$

is computed, where ε is a small number to avoid division by zero. The interpretation is:

- $x = 0$: no regenerative power is used,
- $x = 1$: the agent uses power equal to the current cap P_{cap},
- $x > 1$: the requested electrical power exceeds the cap.

The reward is then shaped in three regions:

1. **Safe region** ($x \leq 0.8$): The agent is far from the limit, so using more regen is always good:

   $$r_{\text{energy}} = x.$$

   This grows linearly from 0 to 0.8 and pushes the agent to increase regenerative power when it is clearly safe.

2. **Near the limit** ($0.8 < x \leq 1$): The agent is close to the cap, so the reward rises more slowly:
$$r_{\text{energy}} = 0.8 + 0.15\,\frac{x - 0.8}{0.2},$$
which increases from 0.8 at $x = 0.8$ to 0.95 at $x = 1$. This tells the agent that using almost all of the allowed power is good, but there is little extra gain in pushing right up to the exact limit.

3. **Over the cap** ($x > 1$): Exceeding the cap is discouraged by a penalty:
$$r_{\text{energy}} = -2\,(x - 1).$$
The reward becomes smaller as the over-use increases, but it is not instantly catastrophic for small violations. This penalty gives the agent a clear learning signal to back off slightly on the next episode, instead of making the episode useless due to a huge negative reward.

Overall, this shaping teaches the agent to:

- avoid wasting regenerative potential (large reward when x is close to 1),
- respect the SoP limit (increasing penalty as x exceeds 1).

Smoothness Term

To discourage very abrupt changes in the control action, a small smoothness penalty is added:
$$r_{\text{smooth}} = -0.05\,|a_{\text{act}} - a_{\text{prev}}|.$$
Large step changes in the torque split reduce the reward slightly, encouraging smoother braking behaviour. The weight -0.05 is chosen to be small so that energy recovery dominates the learning objective, while still nudging the agent towards less jerky commands.

Total Reward and Clamping

The total reward is the sum of both components:
$$r = r_{\text{energy}} + r_{\text{smooth}}.$$
For numerical robustness, the reward is finally clipped to the interval $[-2, 1]$. This prevents extremely large values from destabilising training and keeps the scale of the critic targets bounded.

The function always returns `isdone = false`, since episode termination is governed by the drive cycle rather than by the reward itself.

5.4 Training data

To keep both runtime and storage usage manageable, the training data was designed to be as short and informative as possible. Since the task is purely braking control, the drive cycle starts at 130 km/h and consists of three representative braking scenarios that exercise the relevant limits:

- **Soft braking:** Pedal input below the blending threshold, so braking is limited only by the requested deceleration and the motor's regenerative capability.
- **Medium braking:** Braking strong enough that regeneration is capped by the 150 kW motor power limit, while the pedal input is still below the friction-blending threshold.
- **Hard braking to standstill:** Pedal input above the blending threshold (Section 4.2.2), so friction braking is added even if the power limit has not yet been reached.

Three constant-deceleration phases are tuned to realise these cases, with target decelerations of

$$a_{\mathrm{soft}} \approx -2.2\ \mathrm{m/s}^2, \quad a_{\mathrm{med}} \approx -2.8\ \mathrm{m/s}^2, \quad a_{\mathrm{hard}} \approx -4.0\ \mathrm{m/s}^2.$$

Medium braking is placed first because it reaches the regenerative power limit before the safety-related friction blending becomes active. Between each braking phase the vehicle cruises for 2 s at constant speed. The resulting training drive cycle, with a total duration of approximately 19 s, is shown in Figure 5.5. The plotted speed profile is obtained by running the rule-based controller without a battery power cap, and is used only to illustrate the structure of the designed braking sequence.

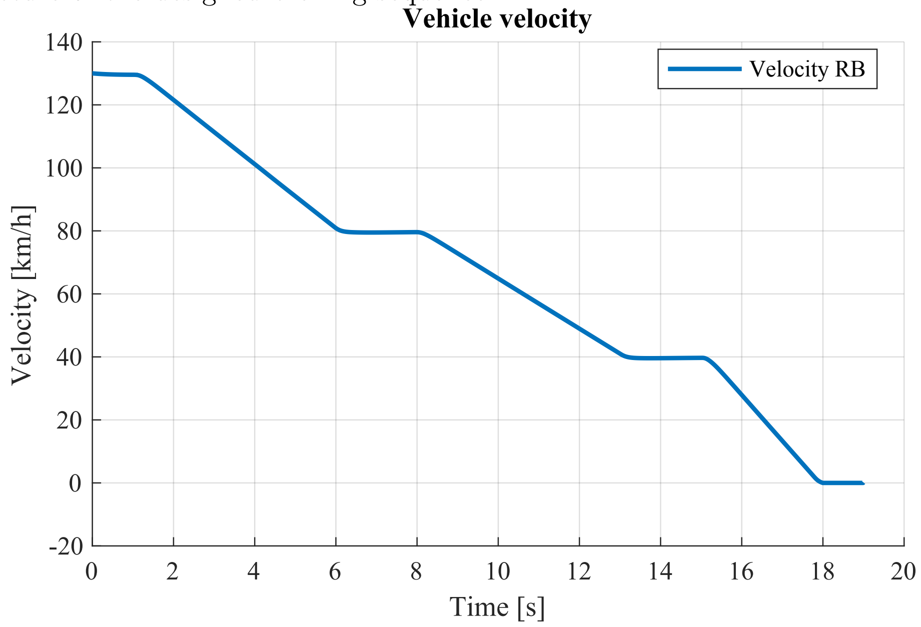
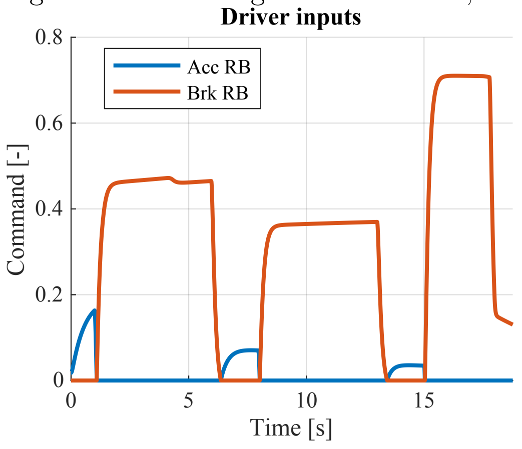
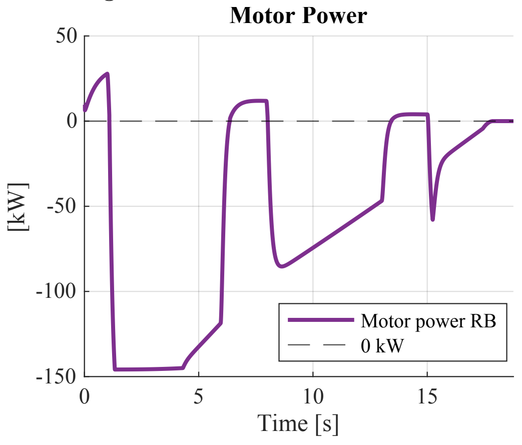


Figure 5.5: Drive cycle used for RL training.

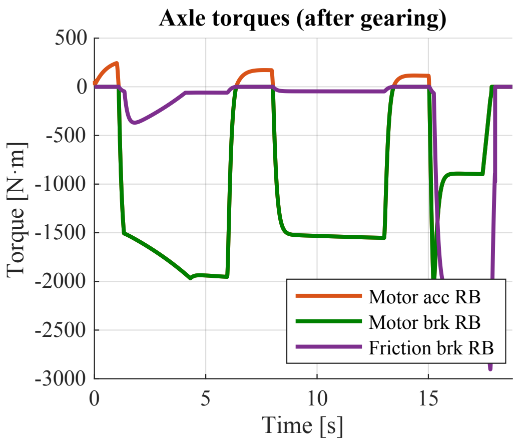
Figure 5.6 illustrates the corresponding driver inputs, axle torques, brake bias and motor power. In Figure 5.6a the first braking event is just below the *brk_cmd* = 0.5 blending threshold; Figures 5.6b and 5.6d show that this still drives the motor up to the 150 kW regenerative limit, after which additional torque is supplied by the friction brakes. In the last braking event the pedal demand is well above the blending threshold, so the safety logic allocates a larger friction share, clearly visible in Figures 5.6c and 5.6d.



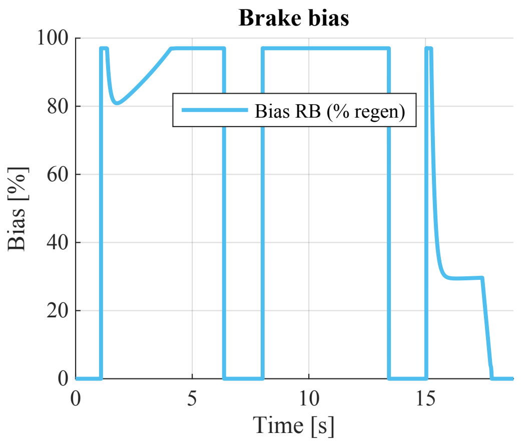
(a) Driver input.



(b) Motor power.



(c) Axle torques.



(d) Brake bias.

Figure 5.6: Driver input, torque split and motor power for the training drive cycle.

For each training episode, the initial battery state of charge is sampled uniformly in the range 30–100 %. The RL agent then adjusts the brake bias over the three braking events to maximise the reward, i.e. to recover as much energy as possible while avoiding violations of the instantaneous SoP limit.

This short, structured drive cycle is considered adequate because it: (i) exercises all relevant braking regimes (power-limited, blending-limited, unlimited, and standstill), (ii) allows rapid exploration over many SoC values, and (iii) keeps training data compact enough to fit within the available storage and computation budget.

5.5 Training Procedure

Training is organized as a sequence of episodes, each corresponding to one complete simulation of a drive cycle. Within each episode, the following steps are repeated at every simulation time step:

1. The environment computes the current observations from the plant and driver signals and assembles the state vector $\mathbf{s}_t$.
2. The instantaneous reward r_t is evaluated using the piecewise formulation defined in Section 5.3.2, reflecting how effectively regenerative braking power is utilised relative to the SoP-based power cap, while penalising abrupt changes in the control action.
3. Given the current state $\mathbf{s}_t$, the actor network produces an action a_t, representing the regenerative braking fraction, with exploration noise added during training to encourage policy exploration.
4. The environment transitions to the next state $\mathbf{s}_{t+1}$, and the tuple $(\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1})$ is stored in a replay buffer.
5. During training updates, mini-batches sampled from the replay buffer are used to train the critic network by minimising the temporal-difference error between the predicted action-value $Q(\mathbf{s}_t, a_t)$ and a target constructed from the observed reward and the critic's estimate of future returns at the next state.
6. The actor network is then updated via policy gradient ascent, adjusting its parameters to maximise the critic's estimated action-value, thereby increasing the likelihood of selecting actions that yield higher long-term cumulative reward.

5.5.1 Training Convergence and Interpretation of Learning Behaviour

Figure 5.7 illustrates the episodic reward over the full training procedure. The episode value Q_0 denotes the expected cumulative discounted return starting from the initial state of an episode, i.e. the value of the policy before any control action is applied. It is commonly used as a scalar performance indicator to assess overall episode-level performance rather than step-wise behaviour. Although a smooth, monotonic increase might be expected in simpler reinforcement-learning settings, such behaviour is uncommon for long-horizon continuous-control tasks with strong penalties and state-dependent constraints, as discussed in the reinforcement-learning literature [9]. The SoP-limited regenerative braking problem introduces exactly these difficulties: each episode spans thousands of time steps, violations of the SoP limit result in sharp penalties, and exploratory noise perturbs the agent's actions throughout training. As a result, the episodic reward naturally exhibits noticeable variability, including occasional negative spikes.

Despite this variability, the training behaviour is fully consistent with what is expected from DDPG and related off-policy actor–critic methods, as documented in [9, 11, 60]. Two key trends indicate successful learning: (i) the running-average reward increases over time, and (ii) the frequency and severity of negative outliers gradually decrease as exploration noise decays. These features reflect a stabilising policy, even though individual episodes remain noisy. The following factors explain why meaningful convergence occurs despite the irregular episodic returns.

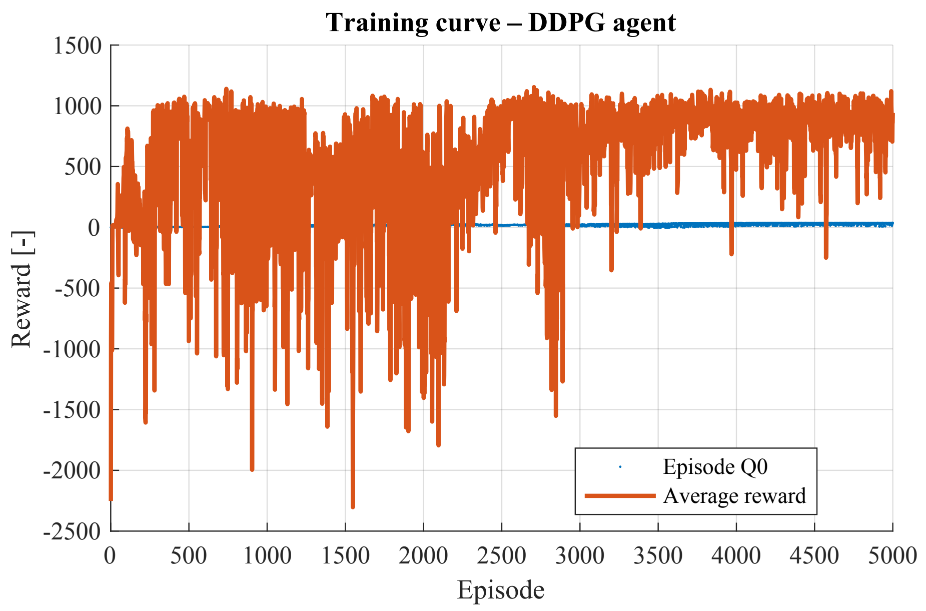


Figure 5.7: Episodic reward versus training episode. The curve shows the high variance characteristic of off-policy actor–critic methods under exploration noise and long-horizon constraint penalties.

1) Episodic reward is highly sensitive to isolated errors. Each episode covers a complete drive cycle and aggregates thousands of reward contributions. A single violation of the SoP limit or an unusually sharp exploratory action early in the episode can dominate the total reward, generating a large negative spike. These spikes therefore reflect isolated exploratory transitions rather than the quality of the underlying deterministic policy.

2) Off-policy learning decouples episodic return from policy quality. DDPG updates both actor and critic using random mini-batches drawn from replay memory, not the most recent episode, as is standard for off-policy actor–critic algorithms [9]. Consequently:

- critic values fluctuate as they incorporate transitions from different stages of training,
- episodic reward does not directly track policy improvements,
- the actor can become stable even while episodic returns remain noisy.

3) Physical constraints stabilise exploration. The Simulink drivetrain model enforces torque, current, and motor-speed limits, preventing the agent from taking unsafe or unrealistic actions. These constraints provide a natural boundary for exploration, allowing the agent to learn a robust brake-splitting strategy even when exploratory noise temporarily drives it toward poor operating points.

Summary. The episodic reward curve exhibits the high variance typical of off-policy deep RL with long-horizon penalties, but its upward trend and improved stability over time are clear signs of convergence.

5.5.2 Hyperparameter Choices and Justification

A careful selection of hyperparameters is required for the RL agent to learn efficiently and stably in the Simulink environment. The main settings, together with their rationale, are summarised below.

- **Sample time** (Δt): 0.01 s.
  This sampling interval allows the agent to react to changes in brake demand and power limits with sufficient temporal resolution, while keeping the overall simulation cost manageable.

- **Number of episodes**: 5000.
  Each episode corresponds to a full drive cycle. This duration was found sufficient for the agent to explore the state space, learn the SoP-based constraint boundary, and converge to a stable policy without excessive simulation time.

- **Actor and critic learning rates** ($\alpha_{\text{actor}}, \alpha_{\text{critic}}$): both set to 3×10^{-4}.
  These moderately low values are consistent with commonly recommended settings for DDPG in continuous-control tasks, as reported in the literature and software documentation [11, 58]. Higher learning rates led to unstable value-function estimates and policy oscillations in preliminary runs, whereas significantly lower rates slowed convergence without improving final performance.

- **Replay buffer size**: 2×10^5 transitions.
  This capacity allows experience from many episodes to be retained simultaneously, providing a diverse distribution of states, actions, and SoP conditions for off-policy updates. A smaller buffer reduced diversity and increased the risk of overfitting to recent episodes, while substantially larger buffers increased memory usage without noticeable gains in stability.

- **Mini-batch size**: 256.
  Mini-batches of 256 transitions provide a good trade-off between gradient variance and computational efficiency. Smaller batches (below 128) produced noticeably noisier updates, whereas larger ones (512 and above) increased memory usage and computation time with limited improvement in convergence behaviour.

- **Discount factor** (γ): 0.995.
  This discount factor emphasises long-term returns over a horizon of approximately $1/(1-\gamma) \approx 200$ time steps (about 2 s at $\Delta t = 0.01$ s), which aligns with typical braking event durations. The agent therefore considers not only instantaneous energy recovery but also the cumulative impact of its decisions over the course of each braking episode.

- **Exploration noise (variance)**: initial variance 0.10 with gradual decay.
  During training, zero-mean Gaussian noise with initial variance 0.10 is added to the actor output to encourage exploration of different torque splits. The variance is reduced over episodes, so that early training emphasises exploration, while later training focuses on refining the policy around promising behaviours.

- **Target network update rate** (τ): 0.001.
  The target actor and critic parameters θ' are updated by soft averaging

  $$\theta' \leftarrow \tau\theta + (1-\tau)\theta',$$

  after each training step, where θ are the current network parameters. The value $\tau = 0.001$ is a standard choice in the DDPG algorithm, as introduced in [11], and was found to stabilise learning by preventing abrupt changes in the target Q-values used for temporal-difference error computation.

5.5.3 Sufficiency and Rationale

The hyperparameter configuration described above follows established DDPG best practices, as reported in the literature and software documentation [11, 58], and has been adapted to the specific constraints of Simulink-based vehicle simulation. The chosen sample time and episode length yield a sufficiently dense and extensive data set for learning, the learning rates and target-network update rate provide stable convergence, and the exploration schedule allows the agent to discover useful behaviours before gradually exploiting them.

Computational setup. Training was executed on the following hardware configuration:

- CPU: **Apple Silicon M3**
- RAM: **18 GB**
- STORAGE: **512 GB**
- MATLAB version: **R2024a**
- Operating system: **macOS**

A single episode required approximately **5 seconds** of wall-clock simulation time, resulting in a total training duration of approximately **6.94 hours** for 5000 episodes.
Overall, this configuration provides a stable and reproducible training procedure. The next chapter describes the drive cycles and evaluation metrics used to assess the trained policy and quantify its performance relative to the idealized rule-based strategy.

All MATLAB functions implementing the proposed drivetrain model and control strategy are analytically documented and provided in full in the Appendix.

6 | Results

This chapter presents the quantitative results of the study, comparing the reinforcement learning controller with the idealised rule-based baseline across different drive cycles, operating conditions, and battery states. At the end of the chapter, a summarising table with the key numerical results is presented.

6.1 Evaluation Metrics

The evaluation focuses on how closely the RL controller reproduces the behaviour of the idealised rule-based controller under the same SoP-based power limits. The rule-based controller is constructed as a "perfect" baseline that always uses as much regenerative power as the effective constraint allows. Each evaluation is carried out for three different initial SoC values $[100\%, 95\%, 75\%]$ in order to probe different operating regimes of the SoP constraint: at 100% SoC the effective $P_{\max}$ is zero over at the beginning of the cycle, at 95% SoC it varies with the SoP estimate, and at 75% SOC it is constant. The following metrics are used:

- **Regenerative power and proximity to the limit:**
  The regenerative power P_{regen} is the instantaneous electrical power returned to the battery. The admissible limit is given by the SoP estimator and clipped by a constant cap $P_{\text{cap}} = 120$ kW, so that

  $$P_{\max} = \min(\text{SoP}, \ P_{\text{cap}}).$$

  The rule-based controller tracks $P_{\max}$ by design. By plotting P_{regen} from the RL controller together with $P_{\max}$ and the rule-based P_{regen}, it is possible to see, for each braking event, how much of the braking is handled regeneratively, how closely the RL policy follows the ideal reference, and whether it ever exceeds the allowed limit.

- **SoC trajectory :**
  The SoC curve over a drive cycle shows the cumulative effect of all traction and braking events. If the RL and rule-based controllers produce similar SoC trajectories, then the RL policy achieves a comparable overall energy balance. Differences in final SoC directly indicate differences in net recovered energy.

- **Average energy consumption (Wh/km):**
  The specific energy consumption summarises the total electrical energy used per

kilometre over the entire cycle. This metric reflects the combined impact of regeneration, traction losses, and any suboptimal decisions. Since the rule-based controller is a best-case benchmark under the given constraints, the key question is whether the RL controller reaches the same Wh/km, or only slightly higher, when operating under the same $P_{\max}$.

Additional signals are provided in appendix chapter B for detailed inspection of the drivetrain behaviour. These include torque bias and braking energies, driver inputs, instantaneous and average efficiency, motor torque and electrical/mechanical power, motor current and voltage, resistive torques (rolling resistance, aerodynamic drag), wheel torques (motoring, regenerative and friction braking), as well as total energy spent and distance travelled. These plots are used to check that the conclusions from the core metrics are consistent with the underlying physical quantities.

6.1.1 Drive Cycles Used for Evaluation

Two drive cycles are used to evaluate the RL controller: a standardised WLTP cycle and an additional synthetic cycle. Each serves a distinct purpose.

WLTP (Worldwide Harmonised Light Vehicles Test Procedure). WLTP is the industry standard used by vehicle manufacturers to characterise energy consumption, range, and emissions. It contains realistic urban, suburban, and high-speed segments but only a limited number of strong braking events. Using WLTP ensures that the evaluation covers representative real-world operating conditions and enables comparison against established automotive benchmarks.

Synthetic random-rich cycle. Alongside WLTP, a custom MATLAB-generated drive cycle is introduced. This cycle is deliberately non-realistic and highly irregular, containing numerous accelerations, soft and hard brakings, emergency stops, and extended coasting intervals. Its goal is not to emulate typical driving behaviour but to expose the controllers to a broad range of braking intensities that WLTP seldom provides. This is essential for stress-testing regenerative braking performance, particularly in situations where friction braking dominates or where the SoP constraint varies rapidly.

Using both cycles provides complementary coverage: WLTP for realism and comparability, and the synthetic cycle for diverse and demanding braking scenarios. Together, they enable a more complete assessment of whether the RL controller generalises across the full range of regenerative braking conditions.

6.1.2 Custom random-rich drive cycle

As mentioned previously, the aim is to create a long, non-repeating speed profile with numerous accelerations, decelerations, full stops, and coasting periods, while always respecting the physical capabilities of the vehicle.

The cycle is built step by step. At each step the script draws a manoeuvre type at random from a set that includes

- small, medium, and large accelerations,

- soft, medium, and near-emergency brakings,
- short cruising holds at constant speed,
- full-stop events followed by a standstill period.

When the vehicle is close to standstill, the probability is biased towards acceleration; at higher speeds it is biased towards braking so that more high-intensity brake events are produced.

For each manoeuvre, a target speed v_1 and a constant longitudinal acceleration a are selected within predefined bounds,

$$\begin{aligned} v_1 &= \operatorname{clip}\big(v_0 + \Delta v_k,\ 0,\ V_{\max}\big), && && (6.1)\\ a &\in [A_{\min}, A_{\max}] && \text{(accelerations)}, && (6.2)\\ b &\in [-B_{\max}, -B_{\min}] && \text{(brakings)}, && (6.3) \end{aligned}$$

where v_0 is the current speed, $V_{\max}$ is the imposed top speed (160 km/h), and clip$(\cdot)$ limits the speed to the feasible range. The parameters $[A_{\min}, A_{\max}]$ and $[B_{\min}, B_{\max}]$ define soft-to-hard acceleration and soft-to-emergency braking levels, respectively.

If $a \neq 0$, the duration of the ramp is chosen such that the speed changes from v_0 to v_1 with constant acceleration,

$$T_{\text{ramp}} = \frac{|v_1 - v_0|}{|a|}, \tag{6.4}$$

and the speed is updated linearly over this interval,

$$v(t) = v_0 + a\,t, \qquad 0 \leq t \leq T_{\text{ramp}}. \tag{6.5}$$

For hold segments ($a = 0$) the speed is simply kept constant for a random dwell time. Special manoeuvres choose $v_1 = 0$ with large negative a and then keep $v(t) = 0$ for a short period, producing full stops and emergency-braking events.

By repeating this procedure until the desired total duration is reached, the script generates a statistically rich time–speed trace that (i) spans the full operating range from standstill to $V_{\max}$, (ii) contains many soft and hard braking events, including emergency stops, and (iii) remains kinematically feasible since all speeds and accelerations are bounded by the vehicle limits. An example of the drive cycle is shown in Figure 6.1.

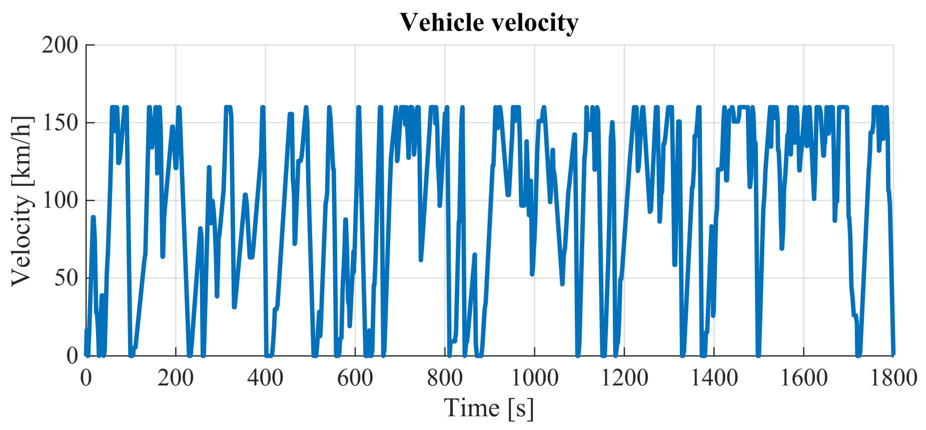
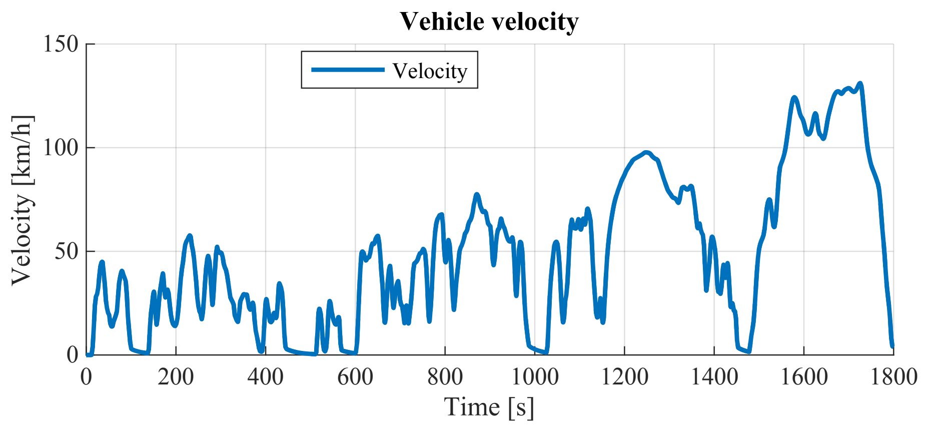


Figure 6.1: Speed profile of the custom drive cycle used for evaluation of the RL and rule-based controllers.

6.2 Results from the WLTP drive cycle



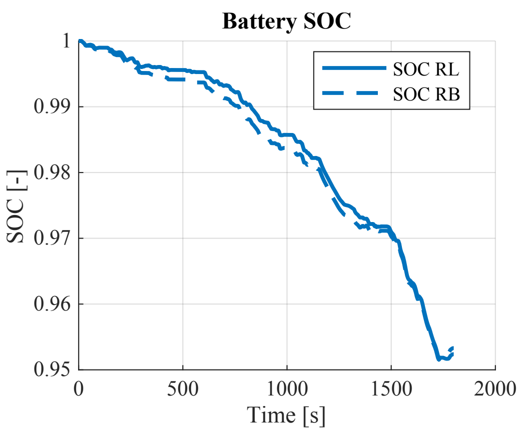
The WLTP drive cycle, described in Section 6.1.1, is evaluated in two configurations:

- **Standard:** Flat road, with the same speed–tracking "driver" dynamics that were used during training.
- **Slope, constant downhill:** The same WLTP speed profile driven on a constant -5% road gradient.

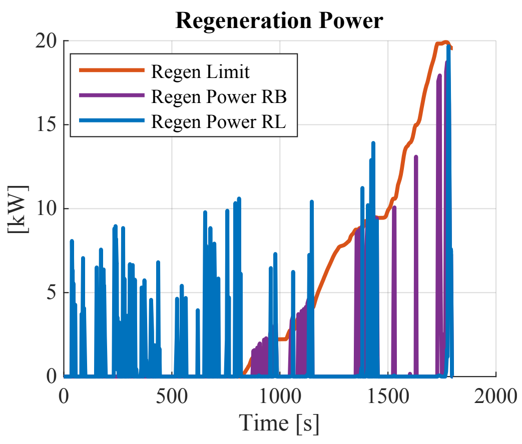
The *standard* case is used to assess how well the RL policy reproduces the behaviour of the perfect rule-based controller under nominal conditions. The *constant downhill* case introduces a sustained opportunity for regeneration and tests whether the RL controller can still follow the ideal reference when the braking demand is dominated by gravitational torque rather than discrete pedal inputs. Together, these two situations provide a clear picture of RL performance in both typical mixed driving and a regeneration-intensive scenario.

6.2.1 Standard

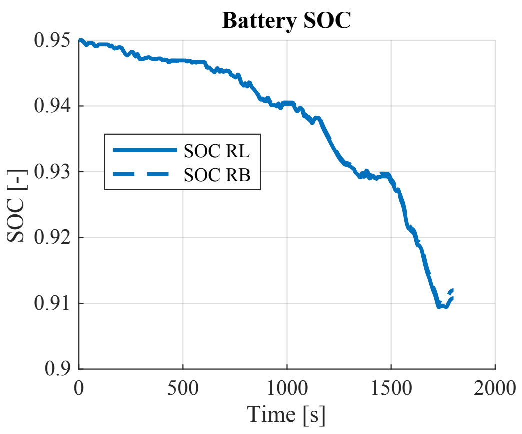
For the standard WLTP case, the behaviour of the RL controller is compared to the ideal rule-based baseline at three different initial SOC levels. Figure 6.2 shows SoC and regenerative power side by side for start SoC of 100%, 95%, and 75%.



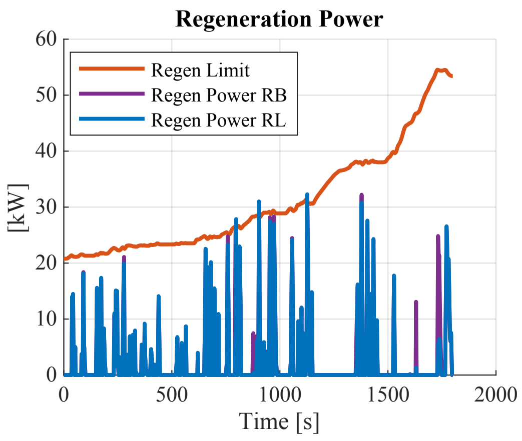
(a) SoC, start SoC = 100%



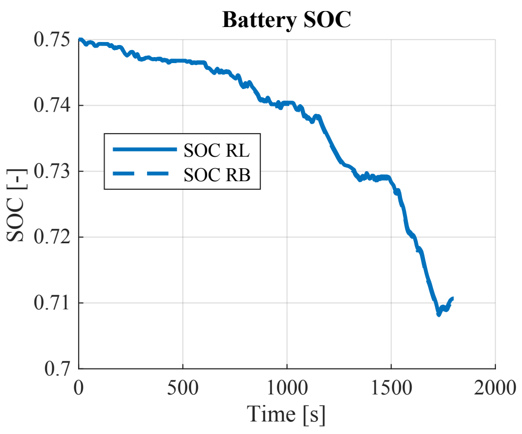
(b) P_{regen}, start SoC = 100%



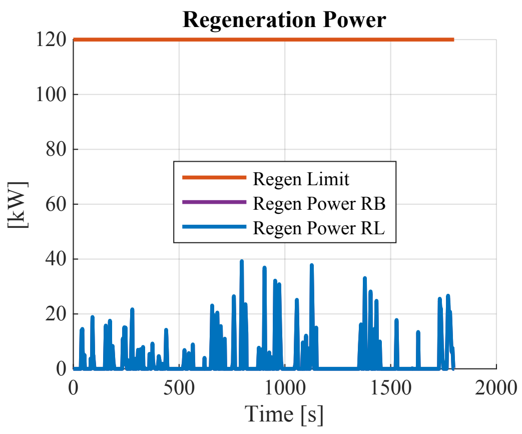
(c) SoC, start SoC = 95%



(d) P_{regen}, start SoC = 95%



(e) SOC, start SOC = 75%



(f) P_{regen}, start SoC = 75%

Figure 6.2: SoC (left) and regenerative power (right) for the standard WLTP drive cycle at three different initial SoC levels, for the RL controller and the ideal rule-based baseline.

At a start SoC of 100%, the effective power limit is zero for an initial part of the cycle. In this region, Figure 6.2b shows that the RL controller still produces a small regenerative power of roughly 8 kW, while the rule-based reference remains at zero. This indicates that the RL policy has difficulty fully respecting a $P_{\max} = 0$ constraint, and the resulting extra charging is visible as a slight deviation in SoC in Figure 6.2a.

For a start SoC of 95%, the effective power limit becomes time-varying rather than zero. Figure 6.2d shows that the RL regenerative power closely follows the rule-based profile, with only a very small overshoot of the limit around $t \approx 1000$ s. The SoC trajectories in Figure 6.2c are almost identical, which means that the RL controller regenerates essentially the same total energy as the ideal baseline under these conditions.

For a start SoC of 75%, the effective power limit is essentially constant at the cap value and does not become active for this drive cycle, because the WLTP braking events are not intense enough to demand the full 120 kW. In this regime, all braking demand that can physically be handled by the motor is expected to be covered by regenerative braking. This is reflected in Figures 6.2e and 6.2f, where the SoC reduction and P_{regen} closely match the rule-based reference.

6.2.2 Slope, constant downhill

To investigate the behaviour of the controllers in a scenario with sustained gravitational loading, the WLTP velocity profile is repeated on a constant downhill slope of 5%. Figure 6.3 illustrates how the slope appears as an additional torque on the wheels throughout the cycle.

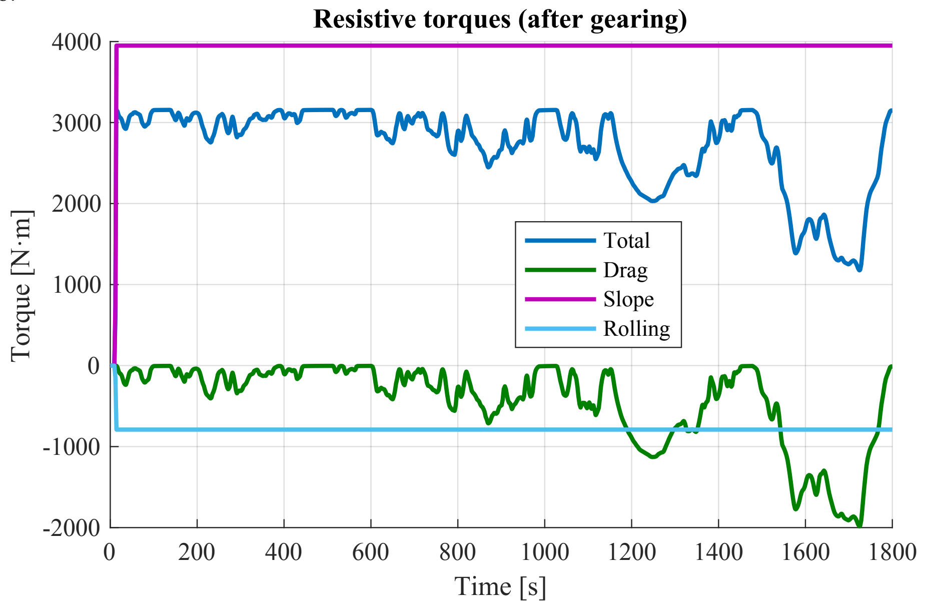
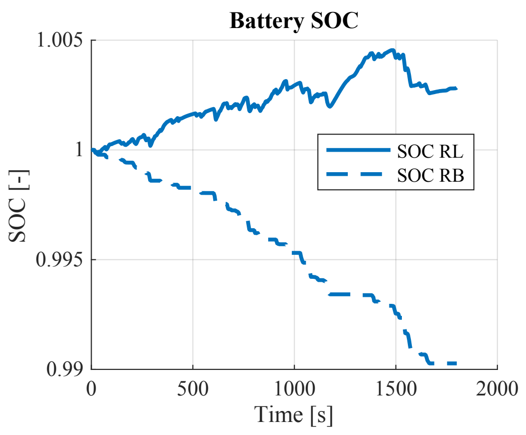
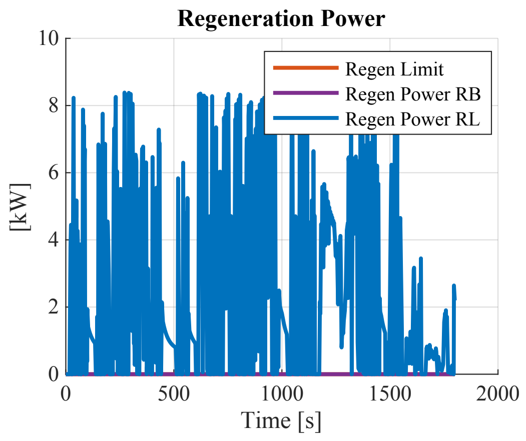


Figure 6.3: Resistive torques acting on the wheels for the WLTP Class 3 cycle on a constant 5% downhill slope. The gravitational component appears as an almost constant positive torque in the motoring direction of approximately 4000 Nm.

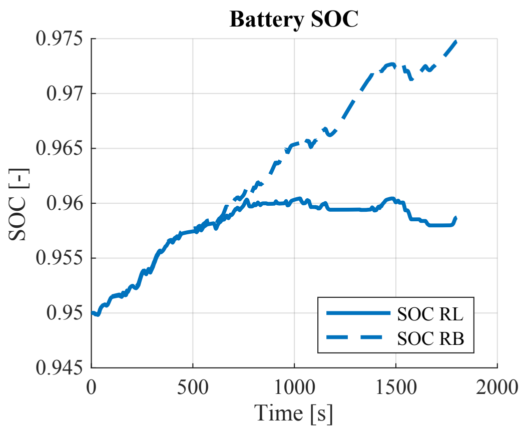
In the same manner as for the standard case, Figure 6.4 compares the SoC trajectories and regenerative power for the RL controller and the ideal rule-based baseline at three different initial SoC levels.



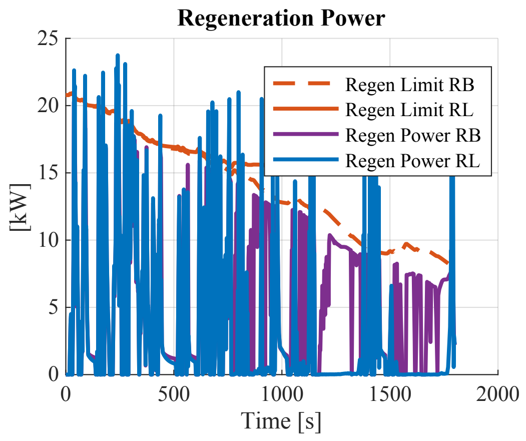
(a) SoC, start SoC = 100%



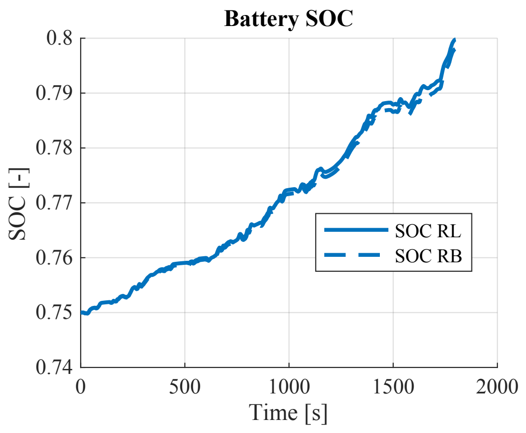
(b) P_{regen}, start SoC = 100%



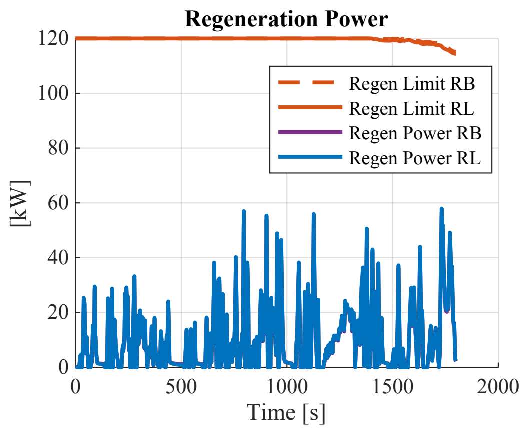
(c) SoC, start SoC = 95%



(d) P_{regen}, start SoC = 95%



(e) SoC, start SoC = 75%



(f) P_{regen}, start SoC = 75%

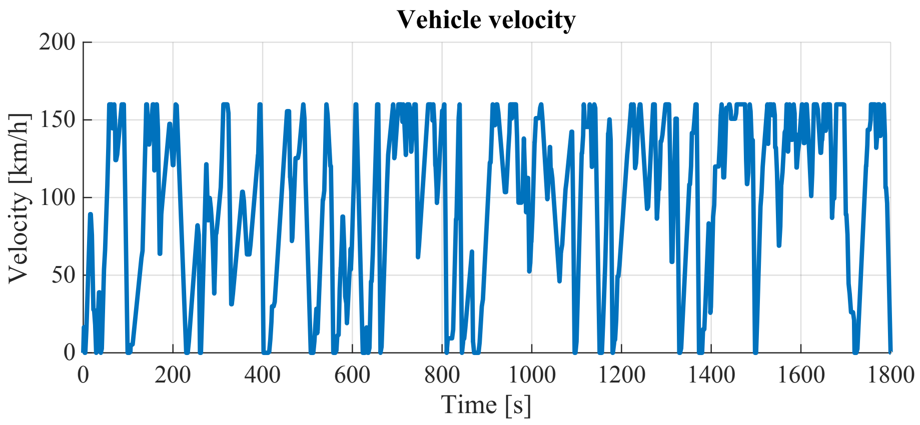
Figure 6.4: SoC (left) and regenerative power (right) for the WLTP Class 3 cycle on a constant 5% downhill slope at three different initial SoC levels, for the RL controller and the ideal rule-based baseline.

At a start SoC of 100%, the downhill case shows the same behaviour as on flat road, but more clearly. The effective power limit is zero in the first part of the cycle, yet Figure 6.4b shows that the RL controller still applies about 8 kW of regenerative power instead of staying at $P_{\max} = 0$. This extra charging causes the SoC to increase slightly above its initial value of 100% resulting in overcharging the battery, as seen in Figure 6.4a.

For a start SoC of 95%, the power limit varies with time and is active more often than in the flat case because of the constant slope. Figure 6.4c shows that the RL controller ends with a slightly lower SoC than the rule-based baseline, even though Figure 6.4d reveals several brief overshoots of the power cap. A reasonable interpretation is that the RL agent sometimes chooses less efficient braking distributions in order to satisfy its reward trade-off, so the extra energy gained during the overshoots is more than compensated by intervals with reduced or poorly timed regeneration.

With a start SoC of 75%, the SoP limit is high and effectively never binds for this drive cycle. In this situation, the constant downhill torque does not change the qualitative control problem, and Figures 6.4e and 6.4f show that RL and the rule-based baseline produce almost identical SoC and regenerative power trajectories. This indicates that, when the constraint is inactive, the RL policy recovers essentially the same amount of energy as the ideal baseline, even on a sustained downhill drive cycle.

6.3 Results from the Random Drive Cycle

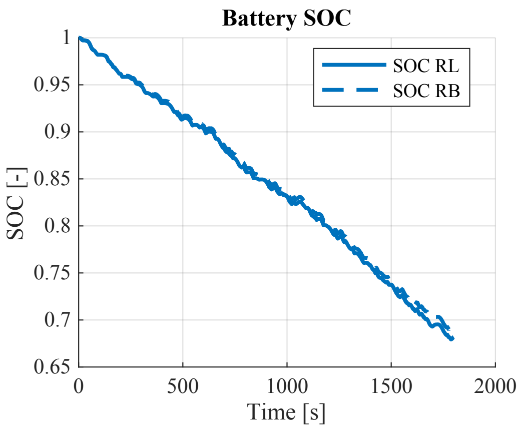


The random drive cycle, described in Section 6.1.2, is evaluated in two configurations designed to assess the robustness of the RL controller under non-structured, highly variable driving conditions:

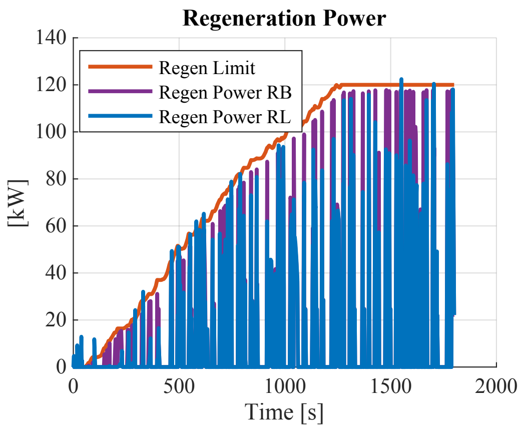
- **Standard:** The baseline configuration, using the same driver dynamics as in training. This case assesses how well the controllers generalise to a stochastic, non-repeatable speed profile without altering the underlying driver behaviour.
- **Different driver dynamics:** A more aggressive driver profile is simulated by increasing the proportional and integral gains of the driver block. This creates sharper accelerations and braking inputs, testing the controllers' resilience to higher-frequency torque fluctuations and less predictable braking behaviour.

6.3.1 Standard

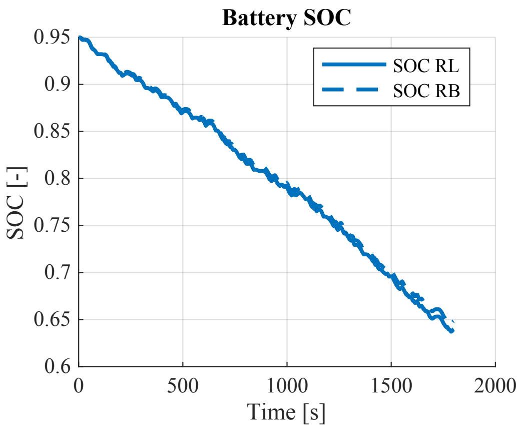
Figure 6.5 reports the SoC evolution (left) and regenerative power P_{regen} (right) for three different initial SoC levels, together with the SoP-based power cap P_{cap} and the ideal rule-based baseline.



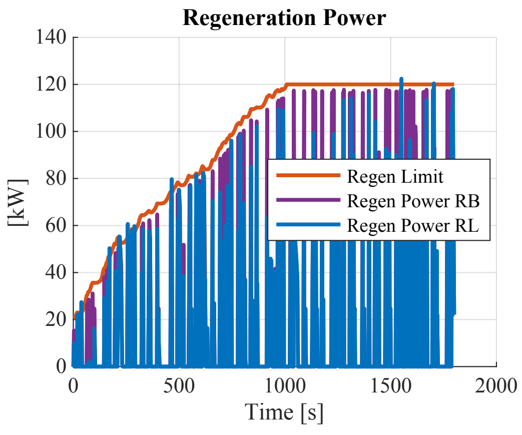
(a) SoC, start SoC = 100%.



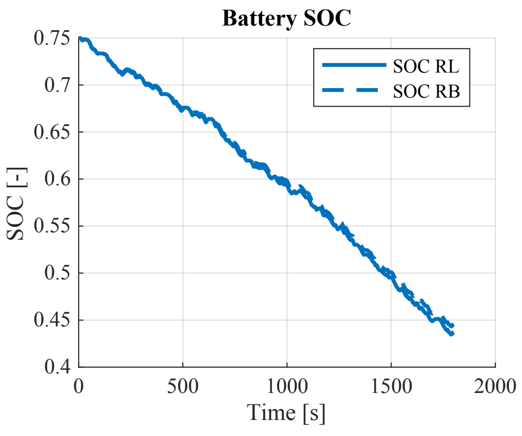
(b) P_{regen}, start SoC = 100%.



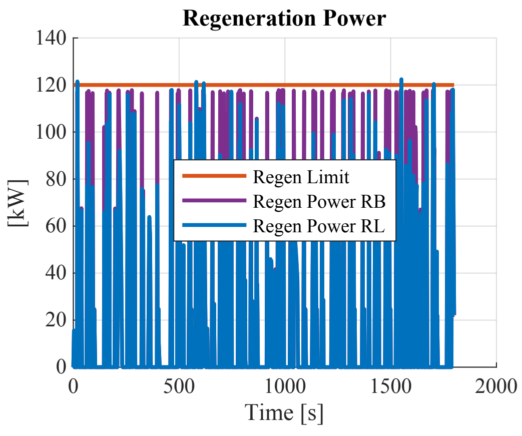
(c) SoC, start SoC = 95%.



(d) P_{regen}, start SoC = 95%.



(e) SoC, start SoC = 75%.



(f) P_{regen}, start SoC = 75%.

Figure 6.5: SoC (left) and regenerative power (right) for the random drive cycle at three different initial SoC levels, for the RL controller and the ideal rule-based baseline.

At an initial SoC of 100%, the RL controller again struggles when P_{cap} is close to zero: distinct overshoots of the power limit are visible at the start of the drive in Figure 6.5b. Once the SoC moves away from the near-full region, the agent largely respects the cap and only introduces a few short intervals of missed regenerative opportunity, seen as purple segments where the ideal controller applies more P_{regen}.

For an initial SoC of 95%, the qualitative behaviour is similar, but the initial overshoot almost disappears and the tracking of P_{cap} in Figure 6.5d is visibly tighter. Remaining differences relative to the ideal reference mainly correspond to locally reduced P_{regen} and thus small, isolated losses in recoverable energy rather than systematic violations of the constraint.

With an initial SoC of 75%, the controller clearly stays within the power cap throughout the cycle in Figure 6.5f, but at the expense of more conservative behaviour. Several braking events show that the RL agent selects less regenerative power than the ideal baseline even when P_{cap} is not binding, which indicates additional lost opportunities but no aggressive use of the battery.

Across all three initial SoC levels, the SoC trajectories of the RL controller and the ideal rule-based strategy are nearly identical in Figures 6.5a–6.5e. This close agreement suggests that, despite local overshoots and occasional conservatism, the RL controller achieves mostly the same net energy recovery as the idealised reference on the random drive cycle.

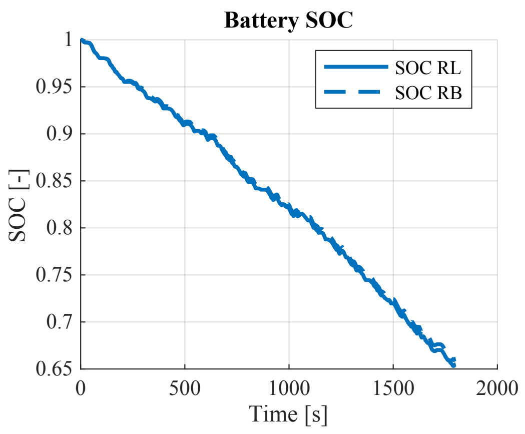
6.3.2 Different Driver Dynamics

Figure 6.6 shows the SoC and regenerative power trajectories for the alternative, more aggressive driver profile at the three initial SoC levels.

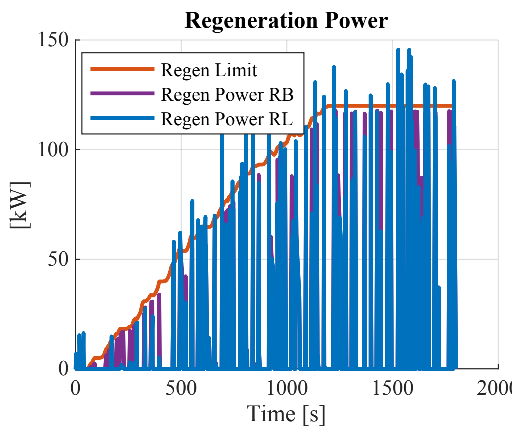
At an initial SoC of 100%, the RL controller still struggles when P_{cap} is close to zero: pronounced overshoots of the power limit are visible in Figure 6.6b. In contrast to the standard-driver case, this overshoot behaviour is no longer confined to the very beginning of the cycle, but recurs more frequently as a consequence of the more aggressive braking inputs.

For initial SoC values of 95% and 75%, the qualitative behaviour in Figures 6.6d and 6.6f is similar to the 100% case, with comparable levels of overshoot across all three SoC starts. The main effect of the alternative driver dynamics is therefore an overall amplification of the overshoot pattern, rather than a fundamental change in how the RL controller responds to the SoP-based power limit.

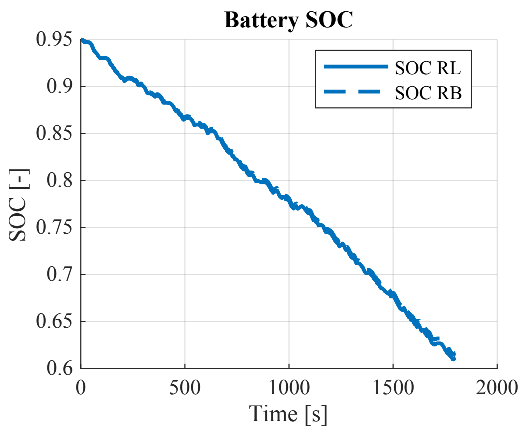
Figure 6.6 shows that, despite these more frequent and pronounced overshoots, the SoC trajectories of the RL controller and the ideal rule-based baseline remain almost indistinguishable in Figures 6.6a–6.6e. This close agreement indicates that the net energy balance over the drive cycle is essentially preserved, and that the RL controller maintains a comparable overall efficiency even under more aggressive driver behaviour.



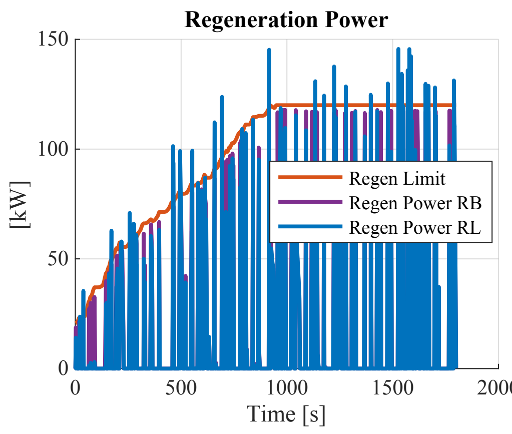
(a) SoC, start SoC = 100%.



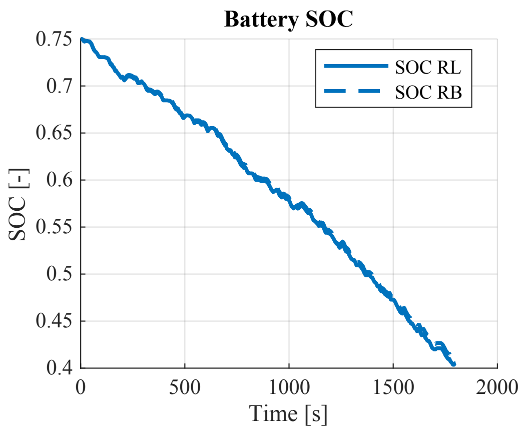
(b) P_{regen}, start SoC = 100%.



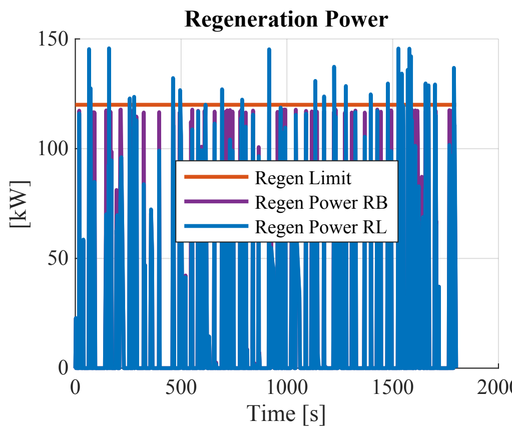
(c) SoC, start SoC = 95%.



(d) P_{regen}, start SoC = 95%.



(e) SoC, start SoC = 75%.



(f) P_{regen}, start SoC = 75%.

Figure 6.6: SoC (left) and regenerative power (right) for the random drive cycle with different driver dynamics at three initial SoC levels, for the RL controller and the ideal rule-based baseline.

6.4 Summary and conclusion

Table 6.1 summarises the net energy usage of the RL controller relative to the ideal RB baseline across all evaluated drive cycles and initial SoCs. A positive value in the "Diff" column means that the RL controller consumed more energy per kilometre than the RB strategy, while a negative value would imply that the RL controller achieved lower net energy usage than the baseline.

Drive cycle	**SoC start [-]**	**RL [Wh/km]**	**RB [Wh/km]**	**Diff [%]**
WLTP3 flat	1.00	169.951	166.654	+1.98
WLTP3 flat	0.95	138.079	133.854	+3.15
WLTP3 flat	0.75	131.674	131.656	+0.01
WLTP3, slope 5%	1.00	-9.752	32.618	-129.89
WLTP3, slope 5%	0.95	-29.367	-83.175	+64.69
WLTP3, slope 5%	0.75	-159.121	-159.385	+0.16
Std. random	1.00	496.994	483.182	+2.86
Std. random	0.95	479.262	466.025	+2.84
Std. random	0.75	463.654	450.632	+2.90
Std. random, alt. drv	1.00	565.204	555.217	+1.80
Std. random, alt. drv	0.95	548.733	540.330	+1.56
Std. random, alt. drv	0.75	536.691	527.923	+1.66

Table 6.1: Net energy usage of RL and rule-based controllers for different drive cycles and initial SoCs. The percentage difference is computed as $(\mathrm{RL} - \mathrm{RB})/\mathrm{RB} \cdot 100$.

6.5 Energy Consumption Comparison

For the flat WLTP cycle, the RL controller consumes slightly more energy than the rule-based controller at 100 % and 95 % SoC (approximately 2–3 %). At 75 % SoC, the RL controller is only 0.01 % less efficient than the rule-based controller, demonstrating that it is able to recover almost all of the regenerative energy available when the Pcap is not the limiting factor.

On the WLTP cycle with a 5 % downhill slope, the relative differences in energy consumption are notable. At 100 % SoC, the RL controller demonstrates a large improvement in efficiency, but the underlying simulations reveal that this is due to systematic violations of the power cap, where the RL agent fails to respect $P_{\mathrm{cap}} = 0$ and effectively overcharges the battery. At 95 % SoC, the RL controller instead becomes significantly less efficient than the rule-based baseline (about 65 % higher net energy usage), even though both controllers have comparable access to regeneration; this indicates that the learned policy does not generalize well to this combination of high SoC and sustained downhill braking. At 75 % SoC, the RL controller is only 0.16 % less efficient than the rule-based baseline, indicating that it can regenerate nearly all available energy when Pcap is not the active constraint—even in downhill scenarios

For the standard random drive cycle, the RL controller consistently consumes about 2.8–2.9% more energy than the RB strategy across all initial SoC levels. In the random cycle with alternative driver dynamics, the RL controller still tracks the power cap reasonably well but shows more frequent and pronounced overshoots of P_{cap}, which is reflected in a smaller yet systematic increase in energy usage of around 1.5–1.8% compared to the RB baseline. When both controllers respect the SoP constraint, the RL policy therefore tends to incur a small additional energy cost in the tested scenarios.

Finally, it should be noted that the RB controller here is an ideal "perfect" reference without explicit safety margins on the power cap. In a realistic production implementation, P_{cap} would typically be scaled by a Factor of Safety to account for SoP uncertainty, ageing, and thermal effects, thereby reducing the available regenerative power for the rule-based strategy. Under such conservative derating, the RB controller would move further away from the SoP-limited optimum, whereas a well-trained RL policy operating closer to the true P_{cap} could realise a meaningful efficiency advantage while still respecting the underlying physical limits.

7 | Conclusion

This chapter summarizes the key points of the report and provides some final concluding thoughts.

This thesis has explored the application of reinforcement learning (RL) for regenerative braking control in battery-electric vehicles (BEVs), specifically focusing on the torque-split between regenerative and friction braking systems. The proposed RL controller, based on the Deep Deterministic Policy Gradient (DDPG) algorithm, was designed to operate within realistic physical and safety constraints, including dynamic State-of-Power (SoP) limitations estimated via a simplified HPPC-inspired model. By comparing the RL policy to an idealized rule-based benchmark under identical SoP constraints, the study demonstrates that an RL-based approach can closely approximate optimal energy recovery across a variety of drive cycles and battery states, despite a limited and carefully constrained training dataset.

The results show that energy recovery performance is particularly robust at lower SoC levels, where the controller closely matches the rule-based baseline while respecting the SoP limits. However, at high SoC, under sustained downhill operation and under aggressive driver dynamics, some limitations in generalization become apparent, including occasional violations of the SoP cap and slightly more conservative behavior. These findings highlight the importance of broad training coverage and robust constraint handling for real-world deployment.

The work advances the state of the art by integrating a dynamic SoP estimator into the RL framework, enabling the agent to adapt to real battery behavior. Furthermore, the study addresses three critical research gaps: out-of-distribution testing, explicit safety constraint handling, and the integration of varying SoP limits. These contributions make the proposed approach more rigorous and practical, bridging the gap between theoretical RL applications and industrial implementation requirements.

In summary, this thesis demonstrates that RL-based regenerative braking control is a promising foundation for future adaptive energy-management strategies in electric vehicles. With continued development, such controllers have the potential to significantly improve energy efficiency and user acceptance of battery-electric mobility.

8 | Discussion

This chapter interprets the obtained results, compares them to existing work, and discusses what they imply for the practical use of RL-based regenerative braking.

8.1 Overall interpretation

The results show that the DDPG-based regenerative braking controller can closely reproduce the behaviour of an idealised rule-based torque-split strategy under shared physical constraints, particularly at medium and low State of Charge (SoC) levels where the State-of-Power (SoP) limit is not strongly binding. Across WLTP and random drive cycles, the RL policy generally recovers nearly the same amount of energy as the ideal baseline, with net energy differences mostly within a few percent except in deliberately challenging high-SoC downhill scenarios.

8.2 Comparison with existing literature

In contrast to many published studies, this thesis does not benchmark the RL controller against a conservative, production-style rule-based strategy. Instead, the reference controller is constructed as an idealised, SoP-aware rule-based benchmark that always applies the maximum physically admissible regenerative torque under the same constraints as the RL agent. A review of the literature showed that "rule-based" controllers are defined very differently across papers, often with undocumented calibration choices, which makes direct efficiency comparisons unreliable and can exaggerate the apparent advantage of RL. By elevating the rule-based baseline to a near-*perfect* reference rather than an arbitrary industrial tuning, the evaluation in this thesis focuses on how close the learned policy can come to physically optimal torque splitting, instead of how much it can improve over a potentially weak or overly conservative rule set.

8.2.1 Generalisation and robustness

Previous RL-based regenerative braking studies typically train and test on a small set of similar drive cycles, and often do not report performance under strongly different operating patterns or driver behaviours. This thesis deliberately evaluates the RL controller on cycles that differ significantly from the training data, including randomised velocity profiles and more aggressive driver dynamics, revealing that the policy remains broadly stable and energy-efficient but exhibits recurring constraint violations when SoP is small and braking inputs are sharp. These findings support the view that generalisation to out-of-distribution scenarios and explicit safety handling remain open challenges for practical deployment.

8.3 Interpretation of the benchmark design

A central design choice in this thesis is that the RL agent is evaluated against a deliberately idealised rule-based benchmark rather than against a realistic production controller. The benchmark is constructed under the assumption of perfect knowledge of the instantaneous SoP, braking demand, and drivetrain limits, and it deterministically applies the maximum admissible regenerative torque at every time step, with friction braking only supplying the residual torque needed to satisfy safety and drivability constraints. In practice, production controllers cannot rely on such perfect information, and must instead incorporate margins for estimation error, sensor noise, actuator delays, and certification requirements, which inevitably makes them more conservative.

This difference in assumptions explains why the rule-based controller designed in this work is unrealistically strong when compared to typical industrial strategies and to the "rule-based" baselines found in the literature. Many published studies define their rule-based controllers with simplified or ad-hoc thresholds for SoC, temperature, and braking intensity, often without formal justification or calibration to physical optima, which can significantly under-utilise the available regenerative capability. By contrast, the benchmark in this thesis effectively represents an upper bound on what a perfectly informed rule-based strategy could achieve within the same SoP and torque envelopes, so matching its performance is already a non-trivial target for any learning-based policy.

The RL controller, on the other hand, does not see the underlying analytical structure of the benchmark policy and does not have explicit access to an optimisation routine; it must discover a feasible torque-split strategy purely through interaction with the environment and a scalar reward signal. The fact that the learned policy can approach the behaviour of this idealised controller in many operating regimes, while respecting the same hard constraints, indicates that RL can internalise complex constraint interactions that would otherwise have to be hand-crafted in a rule-based design. At the same time, the remaining performance gap and the observed SoP violations in difficult scenarios highlight that, without additional safety mechanisms and richer training data, RL alone cannot yet guarantee the conservative robustness that is mandatory for deployment in real braking systems.

8.4 Implications for real vehicles and safety

From a safety perspective, the results reinforce the automotive practice of treating regenerative braking as an enhancement to, rather than a replacement for, friction braking, since the friction system can always take over when SoP-limited regeneration or RL decisions are insufficient. The observed SoP-cap overshoots at high SoC highlight that RL reward shaping alone is not sufficient as a safety mechanism; hard projections of actions into a certified safe set or supervisory safety layers would be required before deployment in a safety-critical brake-by-wire system. At the same time, the close match to the ideal baseline at moderate SoC suggests that RL could be valuable as an adaptive layer that tracks a conservative reference and opportunistically improves regeneration when conditions allow, rather than as an unconstrained replacement controller.

8.5 Relation to research gaps and future directions

The literature review identified three main gaps: limited out-of-distribution testing, lack of explicit safety constraint handling, and simplified SoP integration. This thesis partially addresses these gaps by (i) testing the RL controller on multiple, deliberately different drive cycles, (ii) embedding safety-related structure in the environment through enforced friction braking at high braking intensities, a speed-dependent fade-out of regeneration, and a constant minimum friction share, and (iii) using a SoP signal derived from established HPPC-style models, but the results also show that these measures are not yet sufficient to guarantee safe performance in all scenarios. Building on this, the future work chapter can focus on richer training datasets, more robust RL algorithms such as TD3 or SAC, higher-fidelity SoP and thermal models, and explicit safety filters or shielded RL formulations that enforce constraints by design rather than only through rewards

9 | Future Work

This thesis has demonstrated that a reinforcement learning (RL) controller can closely approximate a strong rule-based regenerative braking benchmark under realistic constraints. Several opportunities remain to extend the approach toward greater realism, robustness, and practical deployment.

Broader Training Scenarios

The RL agent was trained on a highly limited dataset consisting of a single short drive cycle. Future work should expand training to include a wider range of operating conditions, such as low-SOC operation, extended downhill braking, and varying ambient conditions. Broader coverage would improve robustness and reduce sensitivity to scenario-specific behaviour, as discussed in recent studies on SOC- and condition-dependent battery behaviour [61].

Alternative RL Algorithms

This work employed DDPG as a baseline continuous-control algorithm. More advanced actor–critic methods such as TD3, SAC, or PPO could improve training stability, reduce value overestimation, and enhance exploration efficiency, as demonstrated for continuous-control problems in the literature [12]. A systematic comparison of these algorithms under identical constraints would provide clearer insight into their suitability for regenerative braking control, following recent comparative studies [13].

Improved State-of-Power Modelling

The simplified PNGV-based SoP estimator provided a simple and computationally efficient constraint signal. Future work should integrate more realistic SoP estimation approaches that account for dynamic battery behaviour and temperature effects, enabling the RL agent to operate within a more accurate representation of battery limits, as explored in recent battery modelling studies [62].

Higher-Fidelity Drivetrain Modelling

The drivetrain model used static efficiency maps and simplified dynamics. Incorporating inverter saturation, electromagnetic transients, drivetrain compliance, and nonlinear tyre effects would allow future controllers to optimise not only energy recovery but also braking smoothness, comfort, and transient behaviour, consistent with higher-fidelity drivetrain modelling approaches reported in the literature [20].

Extended Control Formulations

The present controller outputs a single scalar representing the regenerative torque share. Future formulations could extend the action space to independently command regenerative and friction braking torques, or to distribute braking forces across axles. Such extensions would enable investigation of coordinated braking, axle-wise force distribution, and stability-oriented control strategies consistent with regulatory and control-design frameworks [63, 24].

Thermal, Ageing, and Long-Term Effects

Battery temperature and ageing significantly influence available charging power and internal resistance. Integrating thermal dynamics and ageing-aware constraints would allow the RL controller to balance short-term energy recovery against long-term battery health, as highlighted in battery degradation and thermal management studies [21].

Robustness and Sim-to-Real Transfer

Bridging the gap between simulation and real-world deployment remains a key challenge. Techniques such as domain randomisation and uncertainty-aware training have been shown to improve robustness to modelling errors, road conditions, and driver variability in learning-based control systems [64].

Embedded Deployment and HiL Validation

Finally, real-time feasibility and validation on hardware-in-the-loop (HiL) platforms represent essential steps toward industrial adoption. Future work should address computational constraints through model compression and verify controller behaviour using real motors, inverters, and braking hardware, in line with prior embedded deployment and validation studies [65, 15].

References

[1] D.A. Crolla and D. Cao. “The impact of hybrid and electric powertrains on vehicle dynamics, control systems and energy regeneration”. In: *Vehicle System Dynamics* 50.1 (2012), pp. 95–109. DOI: 10.1080/00423114.2011.586430.

[2] S.A. Oleksowicz et al. “Regenerative braking strategies, vehicle safety and stability control systems: Critical use-case proposals”. In: *Vehicle System Dynamics* 51.5 (2013), pp. 684–699. DOI: 10.1080/00423114.2012.754307.

[3] Farshid Naseri et al. “An efficient regenerative braking system based on battery/-supercapacitor for electric, hybrid, and plug-in hybrid electric vehicles with BLDC motor”. In: *IEEE Transactions on Vehicular Technology* 66.5 (2017), pp. 3724–3738.

[4] J. Ruan et al. “The dynamic performance and economic benefit of a blended braking system in a multi-speed battery electric vehicle”. In: *Applied Energy* 183 (2016), pp. 1240–1258. DOI: 10.1016/j.apenergy.2016.09.028.

[5] Emilia M. Szumska. “Regenerative braking systems in electric vehicles: A comprehensive review of design, control strategies, and efficiency challenges”. In: *Energies* 18.10 (2025), p. 2422. DOI: 10.3390/en18102422.

[6] Hongwen He, Rui Xiong, and Jing Fan. “Evaluation of ANN-Based Regenerative Braking Control for Electric Vehicles”. In: *Applied Energy* 88.11 (2011), pp. 4215–4226.

[7] M. Ali and J. Wang. “Optimal Fuzzy Logic Controller for Regenerative Braking of Electric Vehicles”. In: *Journal of Advanced Transportation* 2025 (2025), pp. 1–10. DOI: 10.1155/2025/123456.

[8] P. Anders and L. Smith. “Fuzzy Q-Learning for Real-Time Energy Recovery in EVs”. In: *Engineering Applications of Artificial Intelligence* 120 (2024), p. 105678. DOI: 10.1016/j.engappai.2024.105678.

[9] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* 2nd. MIT Press, 2018. ISBN: 978-0262039246.

[10] Jie Yin et al. “Regenerative Braking Control Strategy Based on Adaptive Weight Coefficients Using Deep Q-Learning”. In: *Energy* 275 (2023), p. 127483. DOI: 10.1016/j.energy.2023.127483.

[11] T. P. Lillicrap et al. “Continuous Control with Deep Reinforcement Learning”. In: *Proceedings of the 4th International Conference on Learning Representations (ICLR).* Introduces the DDPG algorithm with deep feedforward actor and critic networks using ReLU hidden layers and a tanh-bounded action output for continuous control tasks. 2016.

[12] Scott Fujimoto, Herke van Hoof, and David Meger. “Addressing function approximation error in actor-critic methods”. In: *Proceedings of the 35th International Conference on Machine Learning (ICML).* 2018, pp. 1587–1596.

[13] Ziran Peng and Zhenyu He. “Optimization of regenerative braking control strategy for dual-motor electric vehicles based on deep reinforcement learning”. In: *IEEE Transactions on Intelligent Transportation Systems* 26.7 (2025), pp. 10954–10967. DOI: 10.1109/TITS.2025.3553875.

[14] Zheng Chen, Jia Qin, and Ronald G. Harley. “Battery State-of-Power Estimation Based on a Second-Order HPPC Model”. In: *IEEE Transactions on Vehicular Technology* 61.7 (2012), pp. 2787–2796. DOI: 10.1109/TVT.2012.2204053.

[15] M. White and V. Sharma. “Challenges and Opportunities in Deploying AI Controllers for EVs”. In: *IEEE Transactions on Industrial Informatics* 21.5 (2025), pp. 5890–5906. DOI: 10.1109/TII.2025.123456.

[16] Wentao Ma, Shizhuo Ren, and Peng Guo. “Review of State of Power Estimation for Li-Ion Batteries: Methods, Issues, and Prospects”. In: *Journal of Electrochemical Science and Technology* 13.1 (2025), pp. 1–22. DOI: 10.33961/jecst.2024.00804. URL: https://doi.org/10.33961/jecst.2024.00804.

[17] Guangyan Xu et al. “Fully electrified regenerative braking control for deep energy recovery and maintaining safety of electric vehicles”. In: *IEEE Transactions on Vehicular Technology* 65.3 (2016), pp. 1186–1198. DOI: 10.1109/TVT.2015.2410694.

[18] Y. Gao and M. Ehsani. “Electronic braking system of EV and HEV: Integration of regenerative braking, automatic braking force control and ABS”. In: *SAE Transactions* 2001-01-2478 (2001).

[19] Amin Mahmoudi et al. “Efficiency Maps of Electrical Machines”. In: *2015 IEEE Energy Conversion Congress and Exposition (ECCE).* IEEE, 2015, pp. 2791–2799. DOI: 10.1109/ECCE.2015.7310051.

[20] Seth R. Sanders et al. “Modeling and Control of Electric Drive Vehicles”. In: *Proceedings of the IEEE* 101.2 (2013), pp. 294–308. DOI: 10.1109/JPROC.2012.2216016.

[21] Xuebing Han et al. “A review on the key issues of the lithium ion battery degradation among the whole life cycle”. In: *eTransportation* 1 (2019), p. 100005. DOI: 10.1016/j.etran.2019.100005.

[22] Wei Li et al. “Degradation mechanisms of Li-ion batteries: A state-of-the-art review”. In: *Physical Chemistry Chemical Physics* 23.14 (2021), pp. 8200–8221.

[23] R. Martin et al. “Combining regenerative braking and anti-lock braking for enhanced braking performance and efficiency”. In: *IEEE Transactions on Vehicular Technology* 61.7 (2012), pp. 3063–3077.

[24] United Nations Economic Commission for Europe. *Uniform provisions concerning the approval of passenger cars with regard to braking.* Tech. rep. Regulation 13-H. UN, 2014.

[25] Society of Automotive Engineers. *Brake by wire and steer by wire safety considerations.* Tech. rep. J3016. SAE International, 2019.

[26] *Functional safety of electrical/electronic/programmable electronic safety-related systems.* International Organization for Standardization, 2018.

[27] Tuhibur Rahman and Talal Alharbi. “Exploring lithium-ion battery degradation: A concise review of critical factors, impacts, data-driven degradation estimation techniques, and sustainable directions for energy storage systems”. In: *Batteries* 10.7 (2024), p. 220. DOI: `10.3390/batteries10070220`.

[28] Ufine Battery. *NMC vs LFP vs LTO battery: EV & energy storage guide.* Accessed 18 Nov 2025. 2025. URL: `https://www.ufinebattery.com/nmc-vs-lfp-vs-lto-battery/`.

[29] EVLithium. *NMC vs LFP vs LTO batteries: Complete comparison guide.* Accessed 18 Nov 2025. 2025. URL: `https://www.evlithium.com/nmc-vs-lfp-vs-lto-batteries/`.

[30] M. S. Hossain Lipu et al. “A review of state of charge and state of health estimation methods for lithium ion battery in electric vehicles: Challenges and recommendations”. In: *Journal of Cleaner Production* 205 (2018), pp. 115–133. DOI: `10.1016/j.jclepro.2018.09.065`.

[31] Christoph R. Birkl et al. “Degradation diagnostics for lithium ion cells”. In: *Journal of Power Sources* 341 (2017), pp. 373–386.

[32] US Department of Energy. *PNGV Battery Test Manual.* Tech. rep. U.S. Department of Energy, 2001.

[33] K. Smith, C. Rahn, and C.-Y. Wang. “Power and thermal characterization of a lithium-ion battery pack for hybrid-electric vehicles”. In: *Journal of Power Sources* 160.1 (2006), pp. 662–673.

[34] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* 4th. Pearson, 2020. ISBN: 978-0134610993.

[35] Rui Zhao, Peng Liu, and Zonghai Chen. “A review on battery fault diagnosis methods for electric vehicles”. In: *Energy Procedia* 158 (2019), pp. 4615–4620. DOI: `10.1016/j.egypro.2019.01.733`.

[36] David Silver et al. “Deterministic Policy Gradient Algorithms”. In: *ICML* (2014).

[37] R. K. Gupta and Y. Lee. “Neural Network-Based Prediction for Regenerative Braking in Electric Vehicles”. In: *International Journal of Vehicle Design* 83.2 (2023), pp. 159–176. DOI: `10.1504/IJVD.2023.100501`.

[38] John Schulman et al. *Proximal policy optimization algorithms.* arXiv:1707.06347. 2017.

[39] MathWorks. *Mathlab Function.* Accessed 18 Nov 2025. 2024. URL: `https://se.mathworks.com/help/simulink/slref/matlabfunction.html`.

[40] MathWorks. *Longitudinal driver.* Accessed 18 Nov 2025. 2024. URL: https://se.mathworks.com/help/vdynblks/ref/longitudinaldriver.html.

[41] MathWorks. *Ideal torque source.* Accessed 18 Nov 2025. 2024. URL: https://se.mathworks.com/help/simscape/ref/idealtorquesource.html.

[42] Wikipedia. *Window function.* Seen 16/12/25. URL: https://en.wikipedia.org/wiki/Window_function.

[43] MathWorks. *Inertia.* Accessed 18 Nov 2025. 2024. URL: https://se.mathworks.com/help/simscape/ref/inertia.html.

[44] MathWorks. *Disc brake.* Accessed 18 Nov 2025. 2024. URL: https://se.mathworks.com/help/sdl/ref/discbrake.html.

[45] MathWorks. *Gearbox.* Accessed 18 Nov 2025. 2024. URL: https://se.mathworks.com/help/simscape/ref/gearbox.html.

[46] MathWorks. *Battery table based.* Accessed 18 Nov 2025. 2024. URL: https://se.mathworks.com/help/sps/ref/batterytablebased.html.

[47] FDM. *Elbiler slaar flere rekorder danmark.* Accessed 18 Nov 2025. 2025. URL: https://fdm.dk/nyheder/nyt-om-biler/2025-10-elbiler-slaar-flere-rekorder-danmark.

[48] Skoda-storyboard. $ENYAQ_iV - 80$. Seen 17/10/25. URL: https://cdn.skoda-storyboard.com/2020/08/TD-ENYAQ-iV-en.pdf.

[49] Volksmagen-group. *in-brief-the-all-rounder-the-1-speed-gearbox.* Seen 17/10/25. URL: https://www.volkswagen-group.com/en/press-releases/in-brief-the-all-rounder-the-1-speed-gearbox-17030.

[50] ev-database. *Skoda-Enyaq-iV-80.* Seen 17/10/25. URL: https://ev-database.org/car/1280/Skoda-Enyaq-iV-80.

[51] Fengchun Sun, Rui Xiong, and Hongwen He. "Energy Management Strategies for Electric Vehicles: Review and Outlook". In: *IEEE Transactions on Transportation Electrification* 1.1 (2015), pp. 4–17.

[52] fastestlaps.net. *Skoda Enyaq iV 60/80 (204 PS) specs.* Seen 17/10/25. URL: https://fastestlaps.com/models/skoda-enyaq-iv-80.

[53] Paul Elbert et al. "Analytical method for calculating and mapping the efficiency of high-power traction motors in electric vehicles". In: *IEEE Transactions on Transportation Electrification* 3.1 (2017), pp. 70–85. DOI: 10.1109/TTE.2016.2622702.

[54] Gianmario Pellegrino et al. "Comparison of Induction and PM Synchronous Motor Drives for EV Application Including a Recent Design for Efficiency Improvement". In: *IEEE Transactions on Industrial Electronics* 59.2 (2012), pp. 803–811. DOI: 10.1109/TIE.2011.2164770.

[55] Gregory L. Plett. "High-Performance Battery-Pack Power Estimation Using a Dynamic Cell Model". In: *IEEE Transactions on Vehicular Technology* 53.5 (2004), pp. 1586–1593. DOI: 10.1109/TVT.2004.832408.

[56] M. Wu et al. “State of power estimation of power lithium-ion battery based on an equivalent circuit model”. In: *Journal of Energy Storage* 52 (2022), p. 104618. DOI: 10.1016/j.est.2022.104618.

[57] L. Pei et al. “State of power estimation of power lithium-ion battery based on an improved method”. In: *Journal of Energy Storage* 51 (2022).

[58] MathWorks. *Deep Deterministic Policy Gradient (DDPG) Agent.* https://www.mathworks.com/help/reinforcement-learning/ug/ddpg-agents.html. Documentation for the DDPG agent implementation in Reinforcement Learning Toolbox, including recommended actor and critic network structures for continuous control. 2024.

[59] MathWorks. *RL Agent.* Seen 16/12/25. URL: https://se.mathworks.com/help/reinforcement-learning/ref/rlagent.html?s_tid=srchtitle_support_results_1_rl+agent.

[60] V. Mnih et al. “Human-level Control Through Deep Reinforcement Learning”. In: *Nature* 518.7540 (2015). Demonstrates the use of deep feedforward networks as function approximators in reinforcement learning and motivates deep neural architectures for control problems., pp. 529–533.

[61] E. Santos and R. Green. “Impact of State of Charge and Temperature on Regen Effectiveness in EVs”. In: *Journal of Power Sources* 548 (2025), p. 232211. DOI: 10.1016/j.jpowsour.2025.232211.

[62] R. Guo, W. Shen, and Y. Zhang. “Recent advancements in battery state of power estimation: A review”. In: *Journal of Power Sources* 589 (2024), p. 233743. DOI: 10.1016/j.jpowsour.2023.233743.

[63] Qiao Tang et al. “A novel electro-hydraulic compound braking system coordinated control strategy for a four-wheel-drive pure electric vehicle driven by dual motors”. In: *Energy* 241 (2022), p. 122750. DOI: 10.1016/j.energy.2021.122750.

[64] Josh Tobin et al. *Domain randomization for transferring deep neural networks from simulation to the real world.* arXiv:1703.06907. 2017.

[65] Yunqiang Yuan et al. “A novel regenerative electro-hydraulic brake system: Development and hardware-in-loop tests”. In: *IEEE Transactions on Vehicular Technology* 67.12 (2018), pp. 11440–11452. DOI: 10.1109/TVT.2018.2872030.

Appendix

A | EV drivetrain

This appendix chapter further elaborates on the modelled EV drivetrain

A.1 Plots from original WLTP test

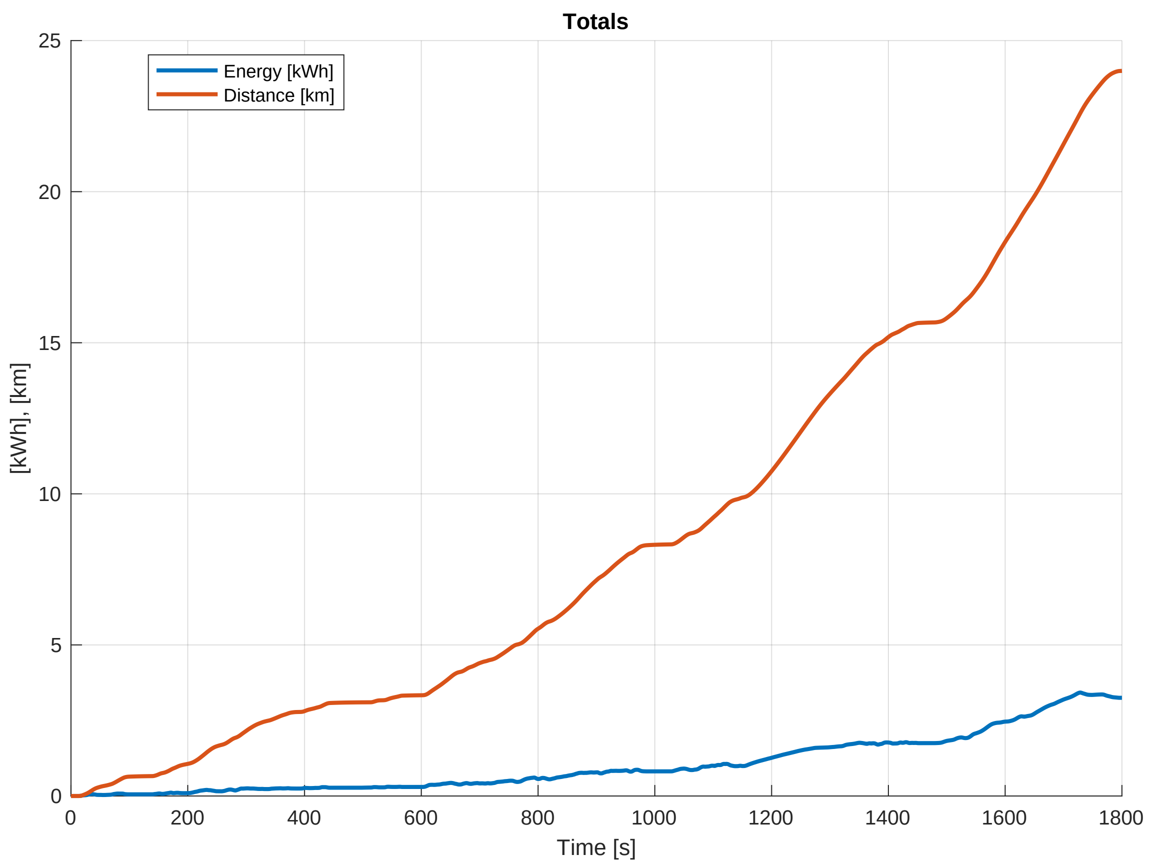


Figure A.1: Total energy consumption in original WLTP test

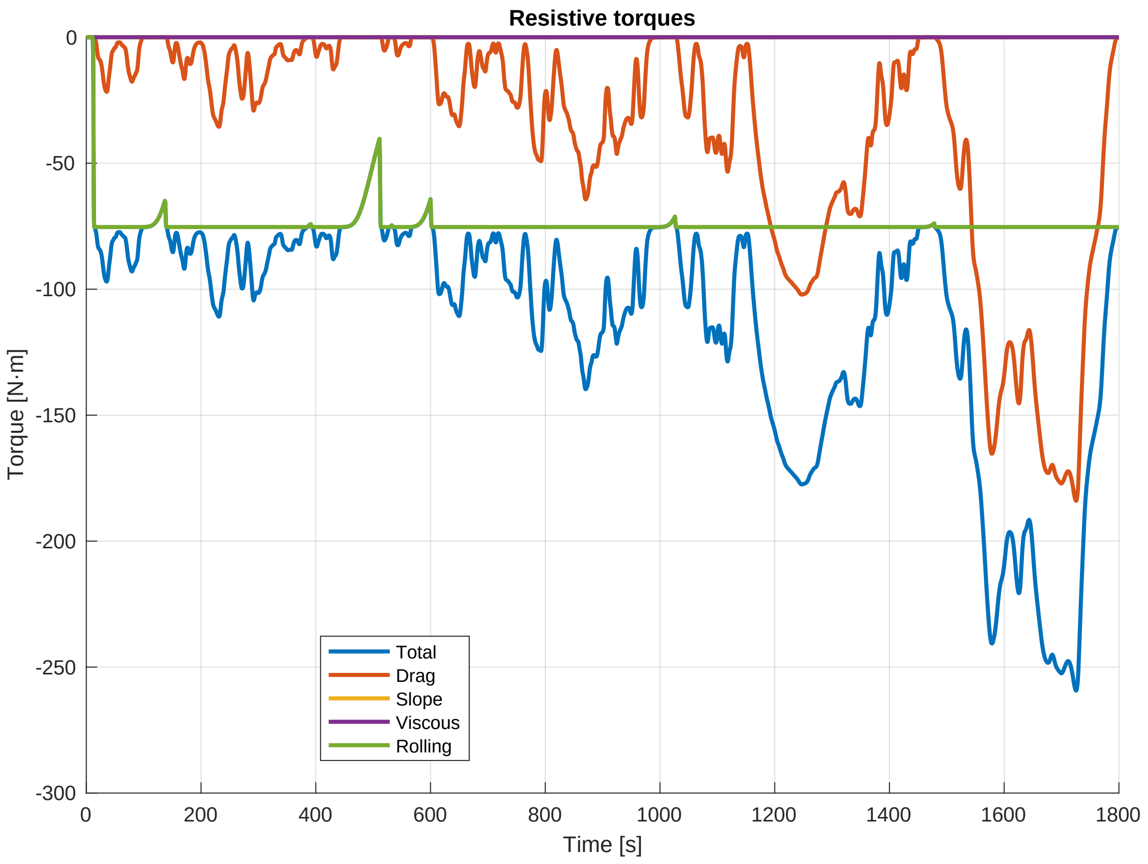


Figure A.2: The resistive torques in original WLTP test

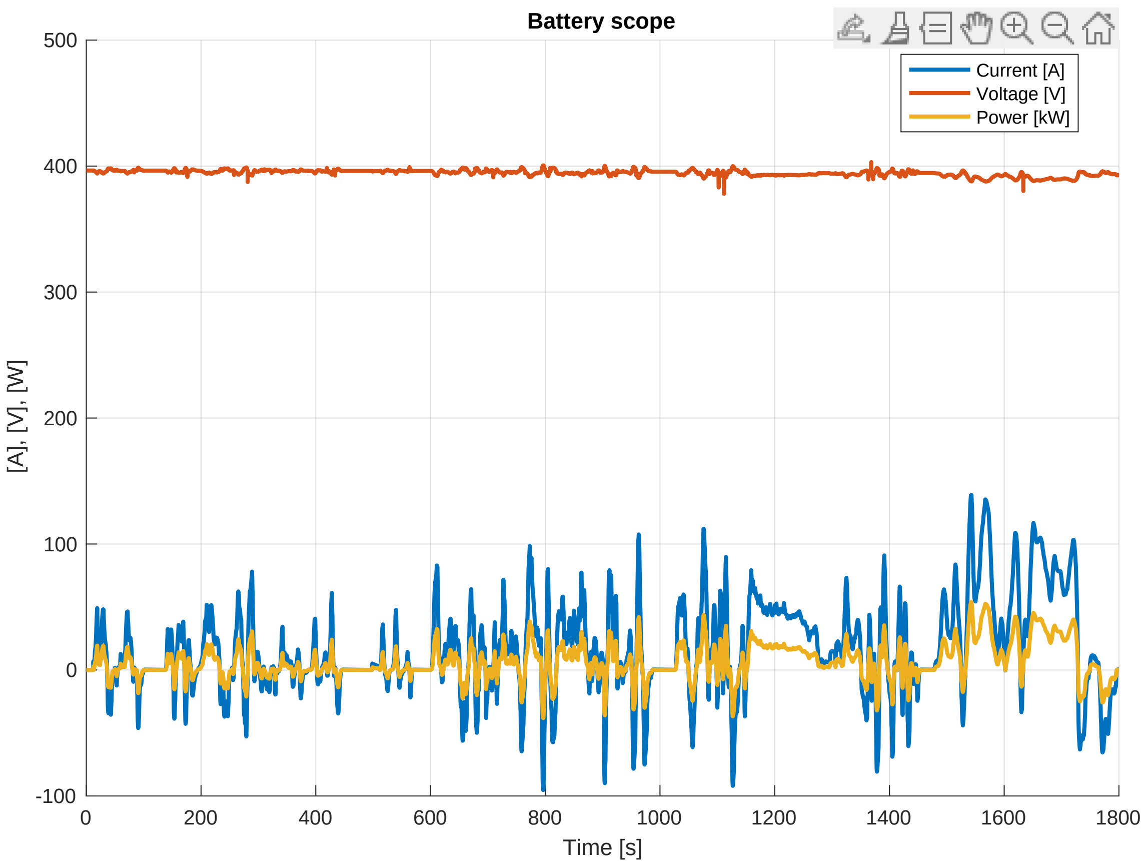


Figure A.3: Motor parameters in original WLTP test

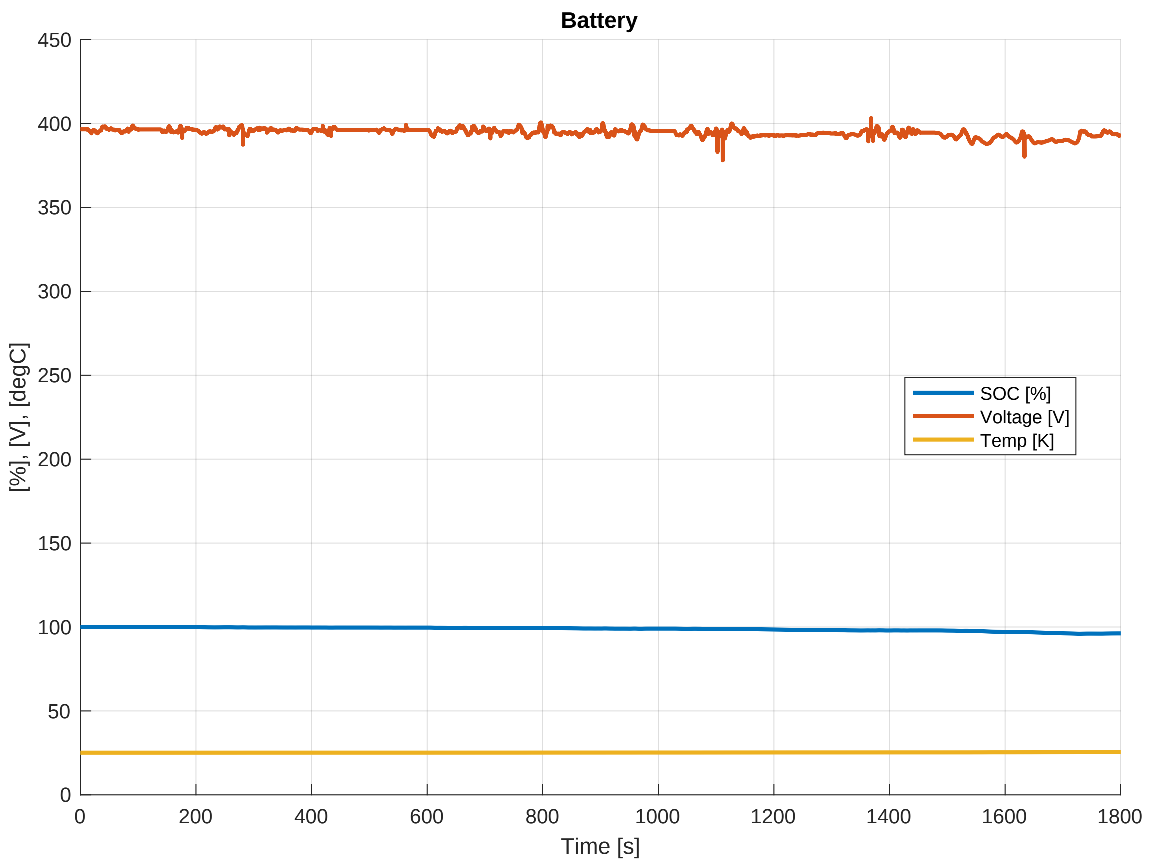


Figure A.4: Battery parameters in original WLTP test

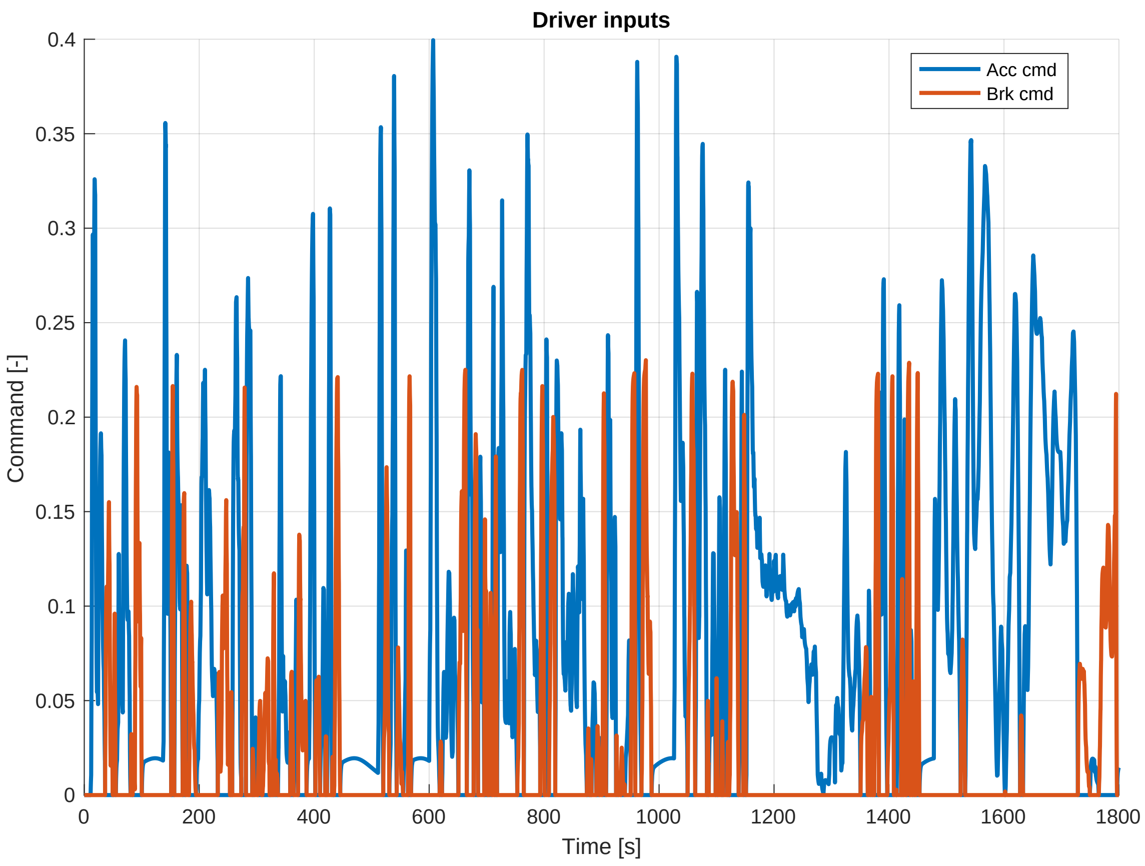


Figure A.5: Wheel torque in original WLTP test

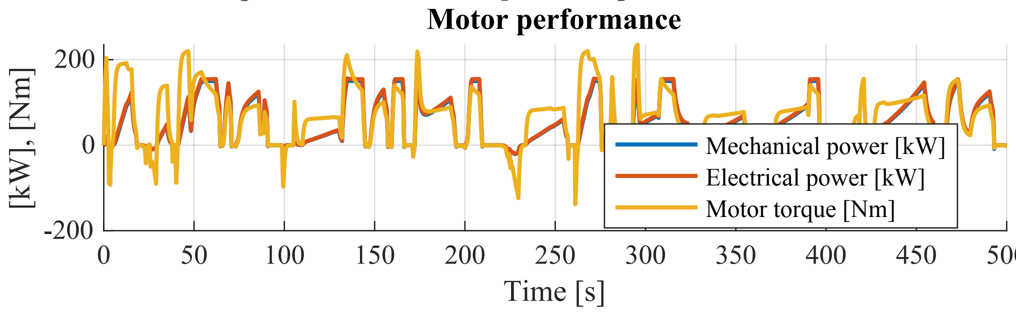


Figure A.6: Wheel torque in original WLTP test

B | Elaborated results

Further graphs, plots, and explanations, from the "Results" chapter are presented in this chapter

B.1 WLPT Standard

B.1.1 SOC start at 100%

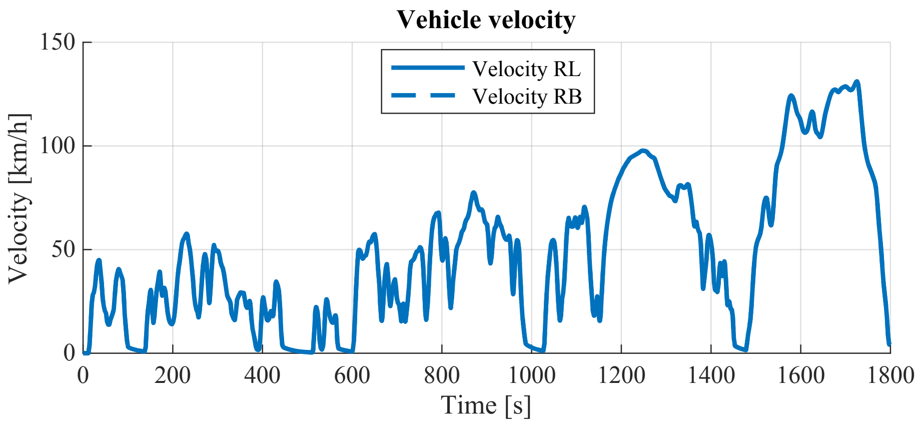


Figure B.1: SoC start = 100%

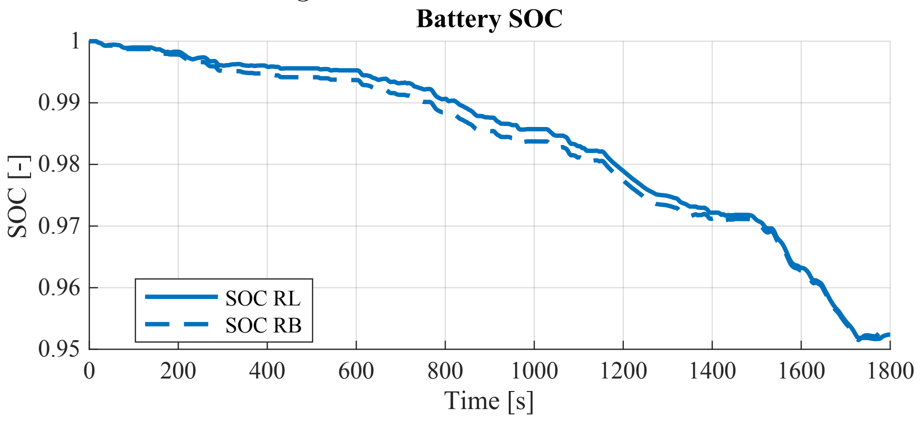


Figure B.2: SoC start = 100%

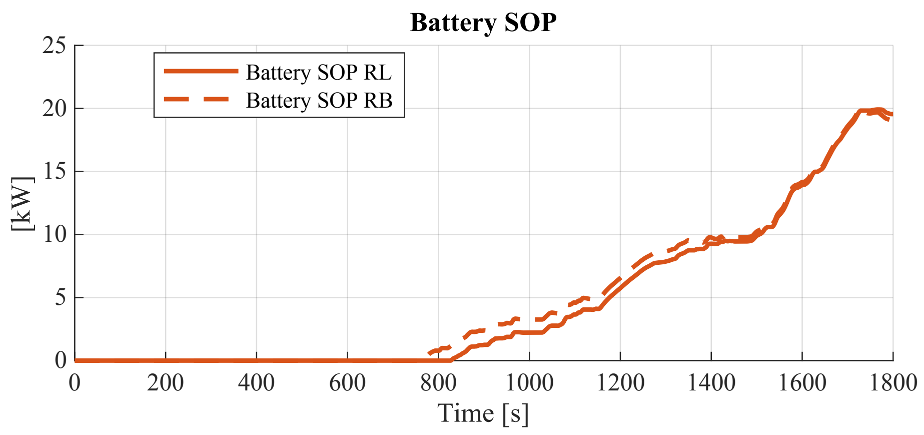


Figure B.3: SoC start = 100%

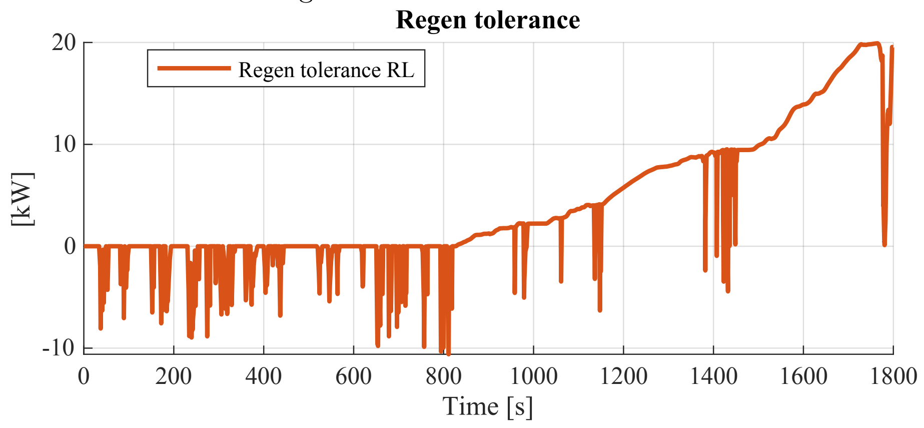


Figure B.4: SoC start = 100%

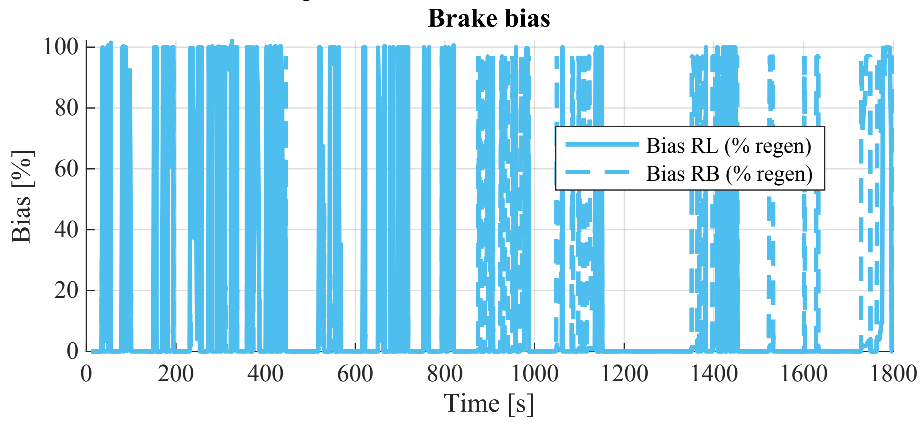


Figure B.5: SoC start = 100%

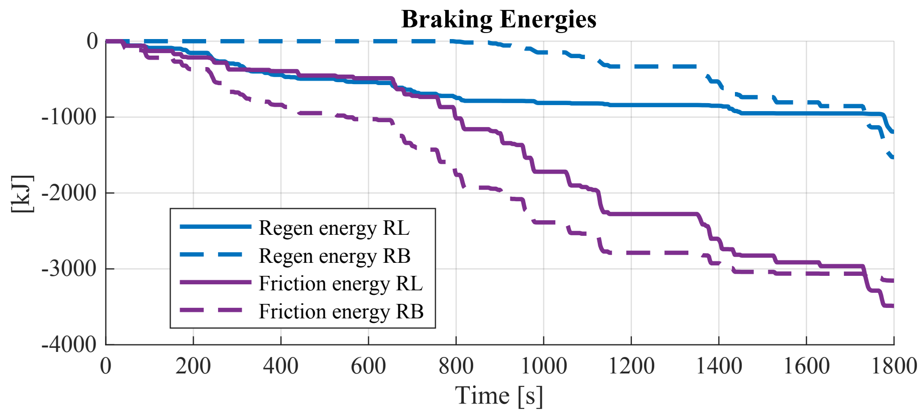


Figure B.6: SoC start = 100%

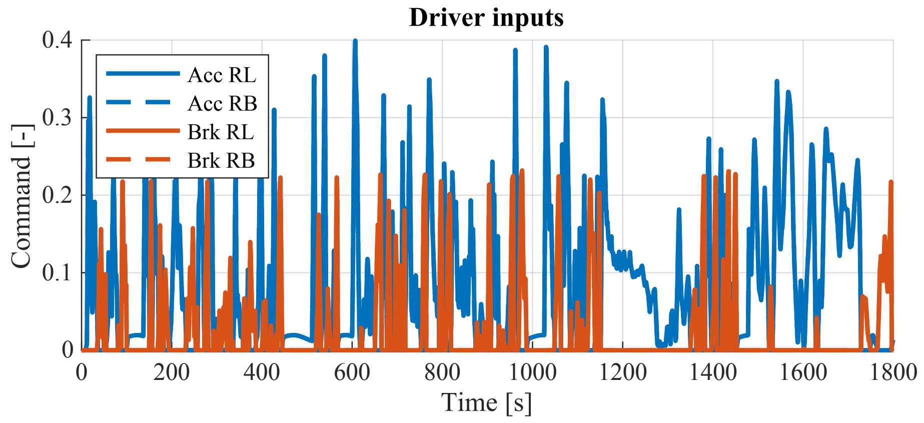


Figure B.7: SoC start = 100%

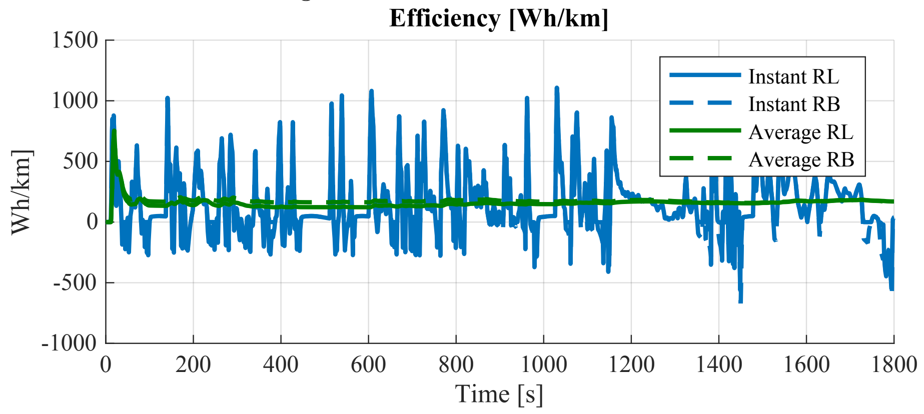


Figure B.8: SoC start = 100%

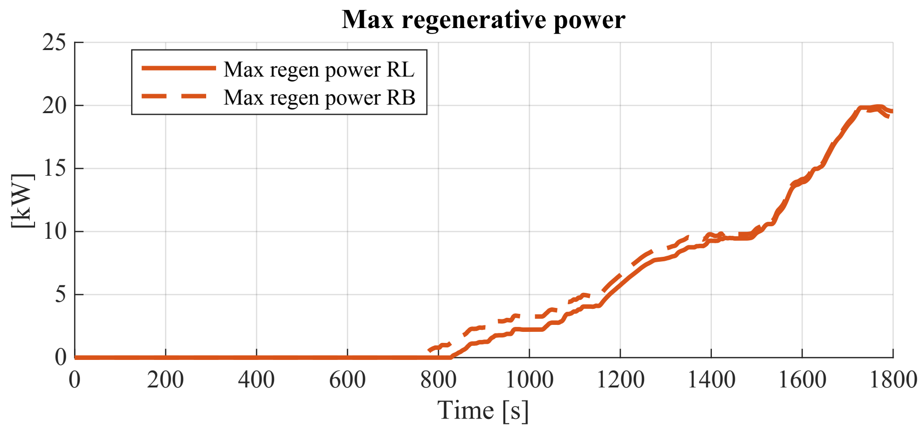


Figure B.9: SoC start = 100%

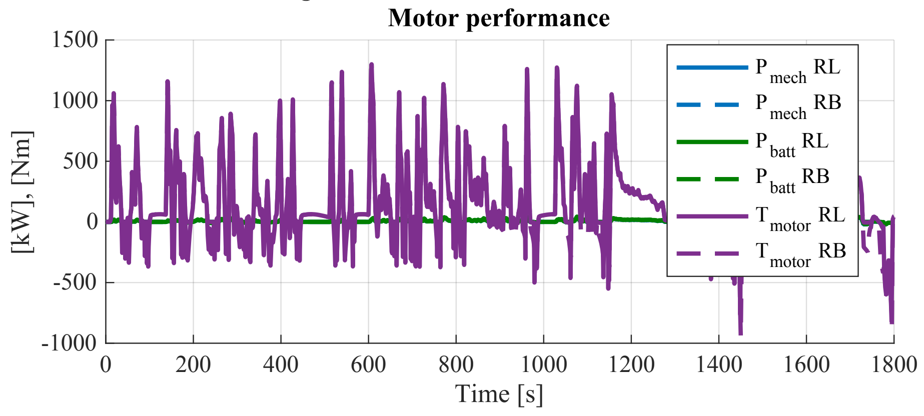


Figure B.10: SoC start = 100%

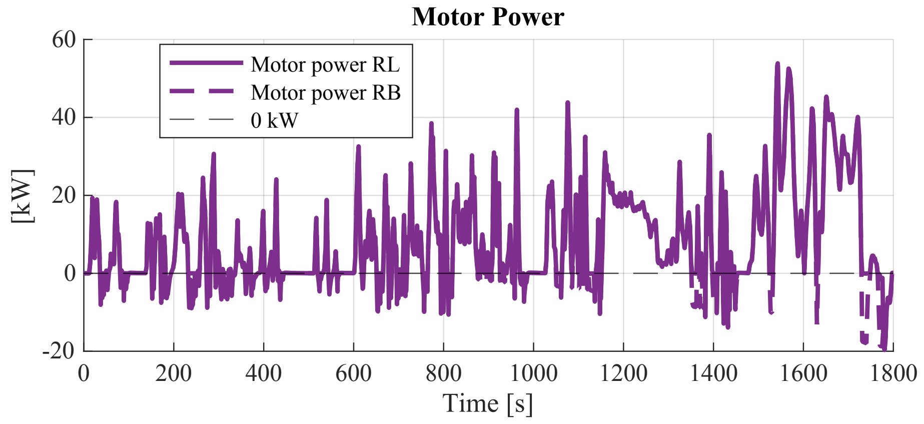


Figure B.11: SoC start = 100%

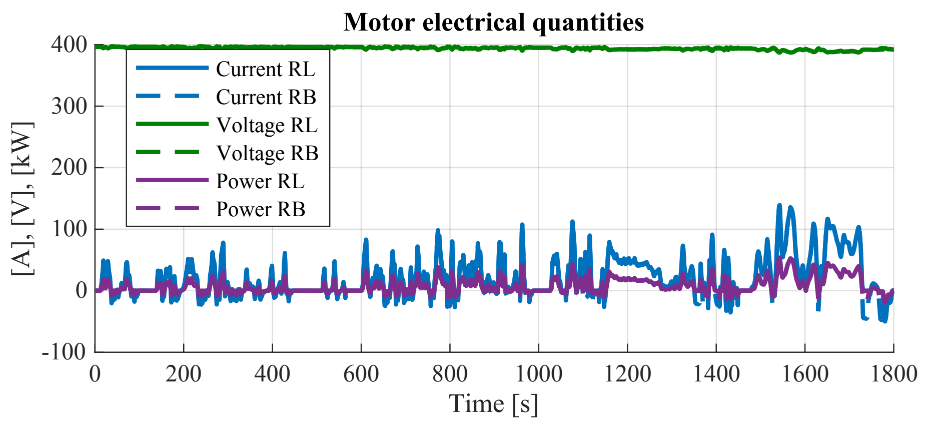


Figure B.12: SoC start = 100%

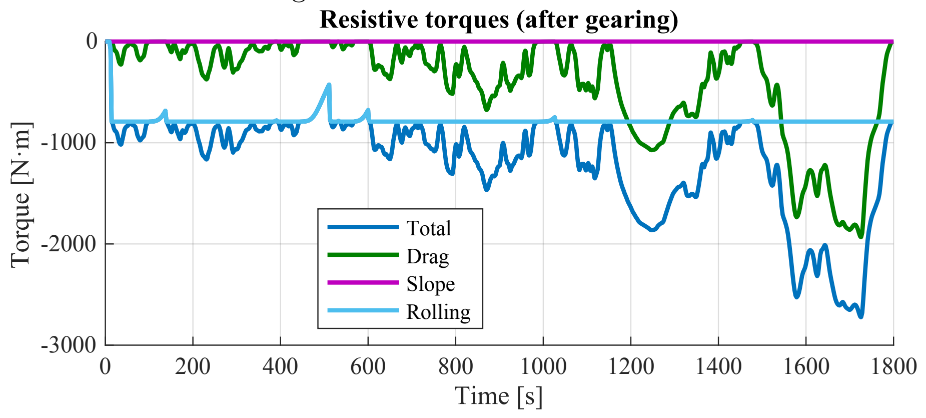


Figure B.13: SoC start = 100%

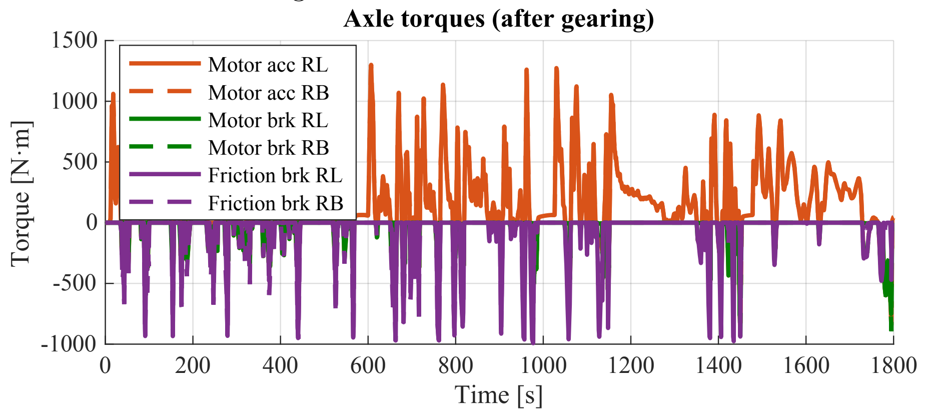


Figure B.14: SoC start = 100%

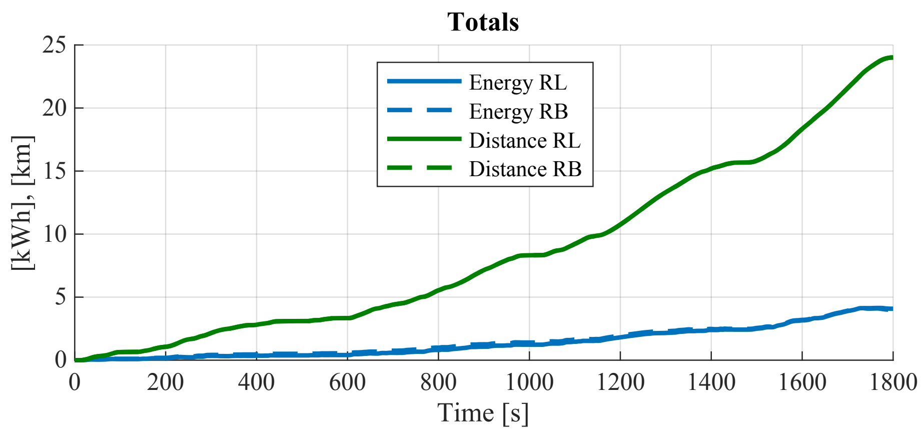


Figure B.15: SoC start = 100%

B.1.2 SOC start at 95%

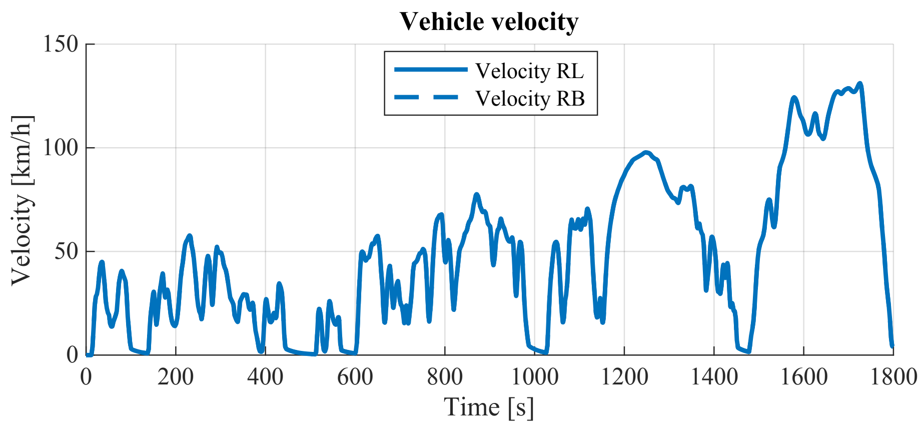


Figure B.16: SoC start = 95%

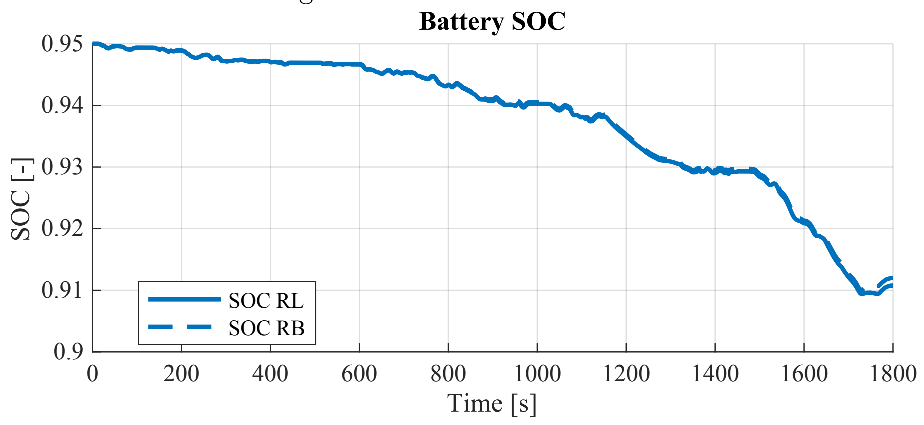


Figure B.17: SoC start = 95%

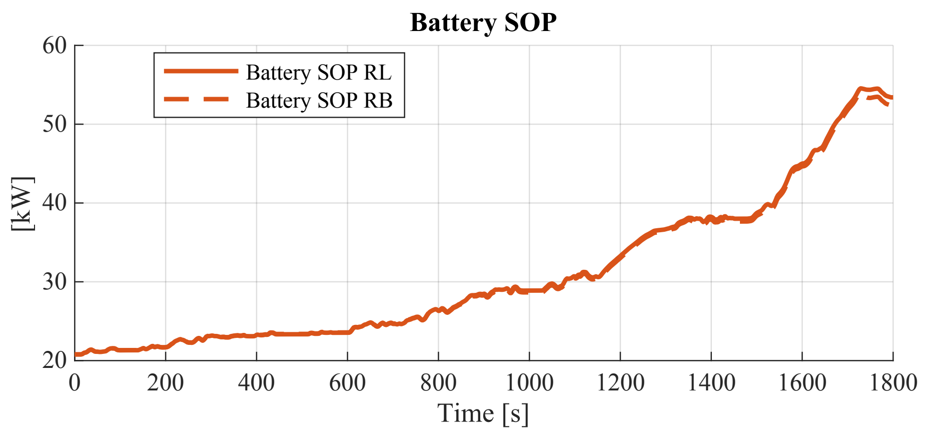


Figure B.18: SoC start = 95%

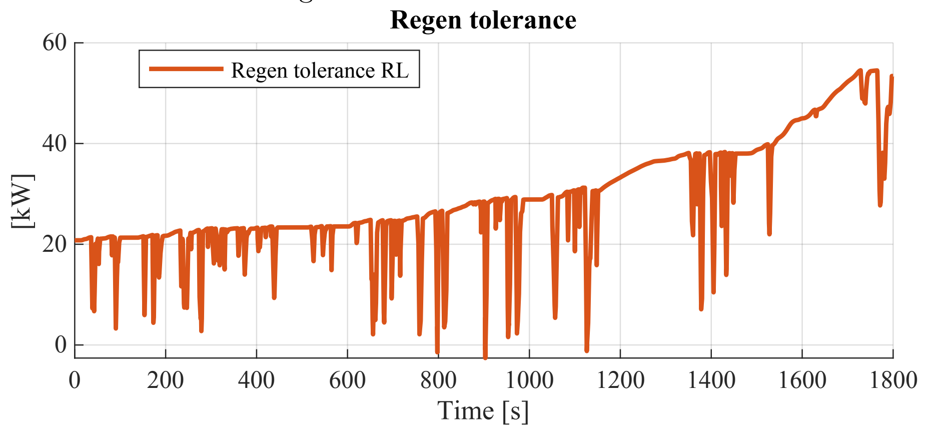


Figure B.19: SoC start = 95%

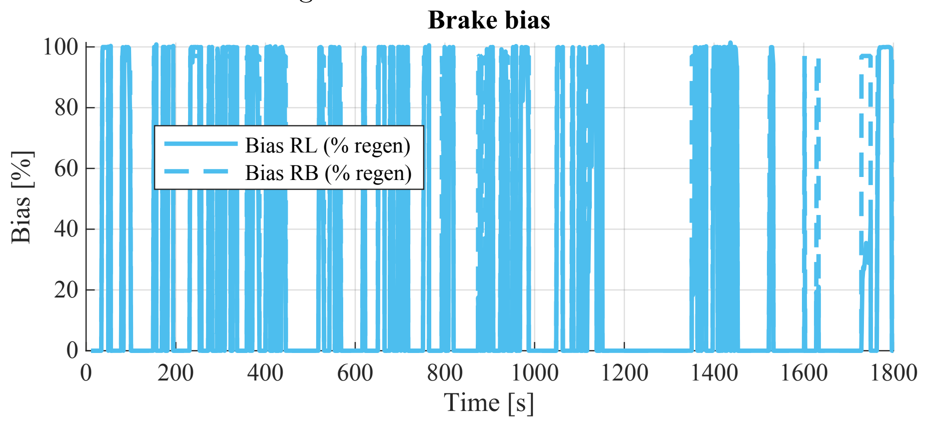


Figure B.20: SoC start = 95%

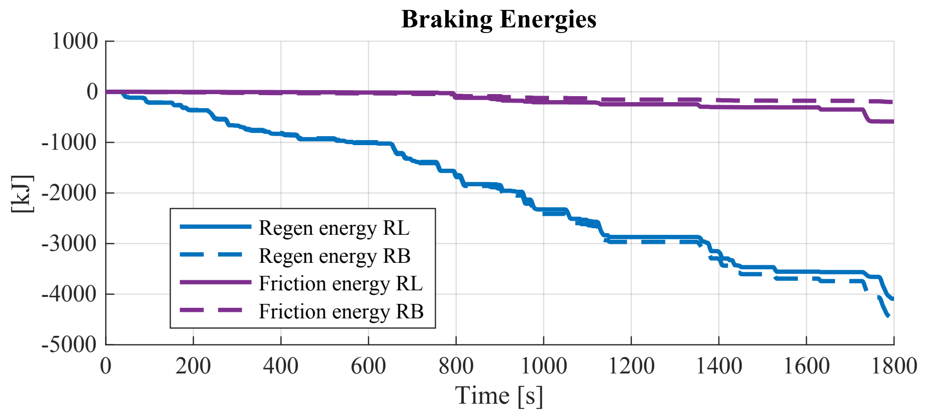


Figure B.21: SoC start = 95%

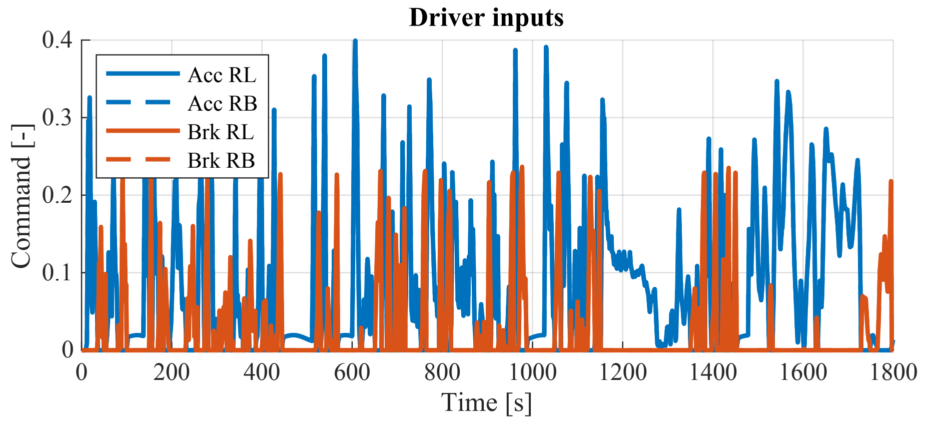


Figure B.22: SoC start = 95%

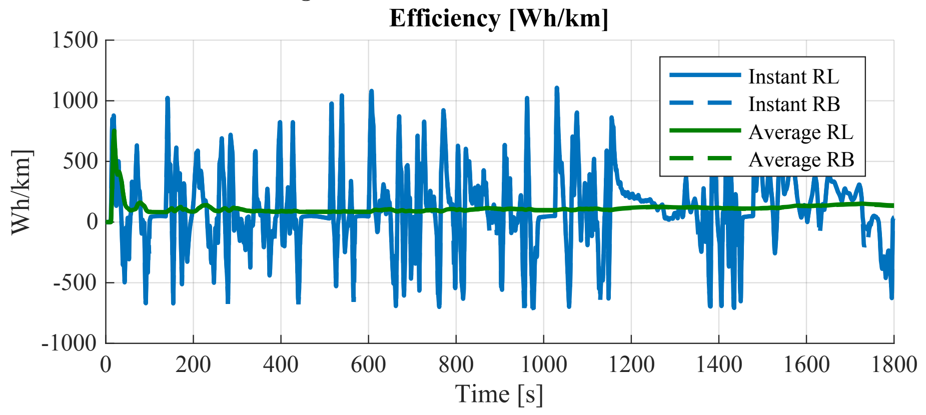


Figure B.23: SoC start = 95%

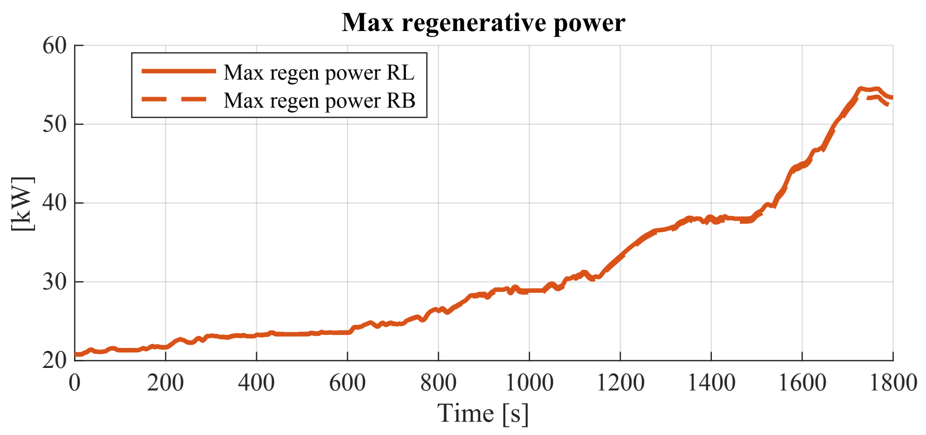


Figure B.24: SoC start = 95%

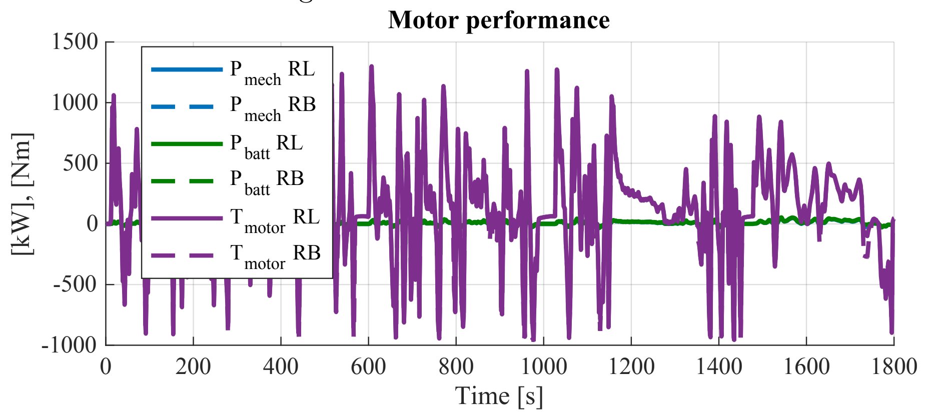


Figure B.25: SoC start = 95%

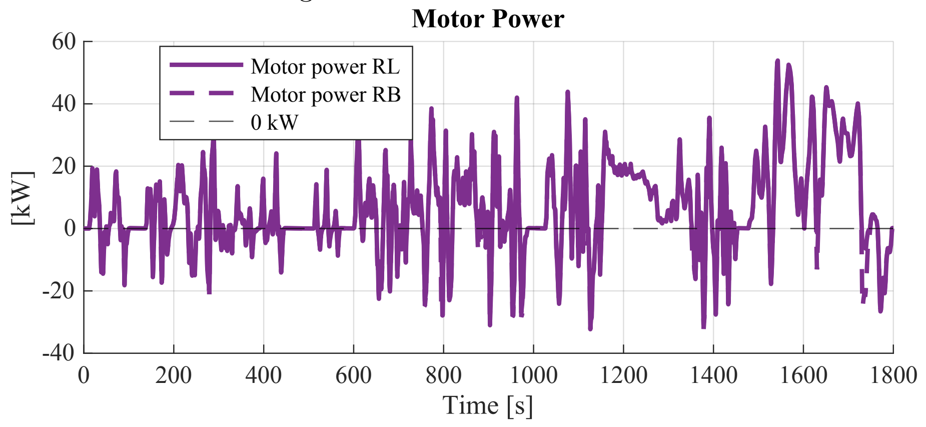


Figure B.26: SoC start = 95%

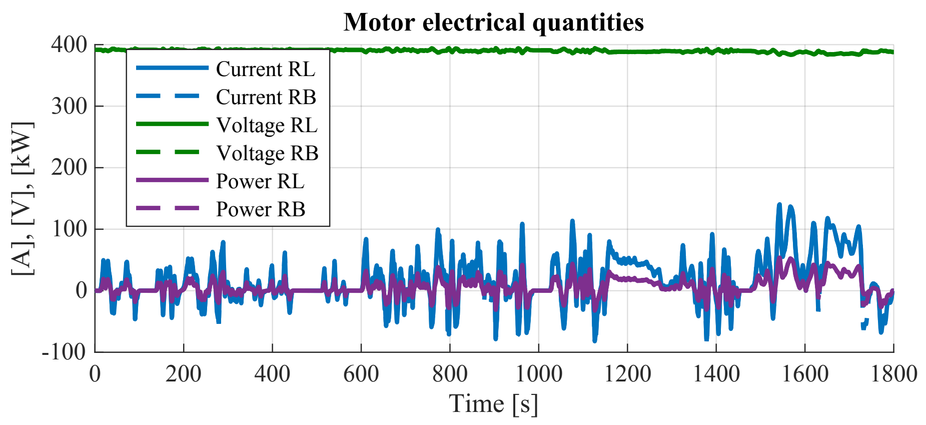


Figure B.27: SoC start = 95%

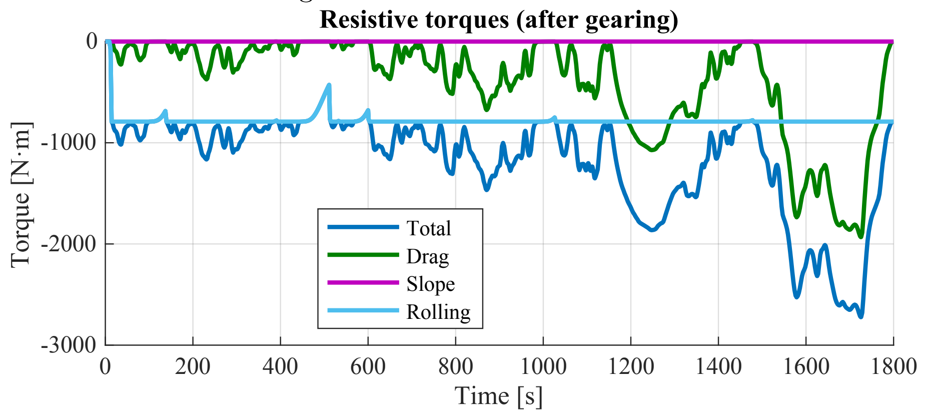


Figure B.28: SoC start = 95%

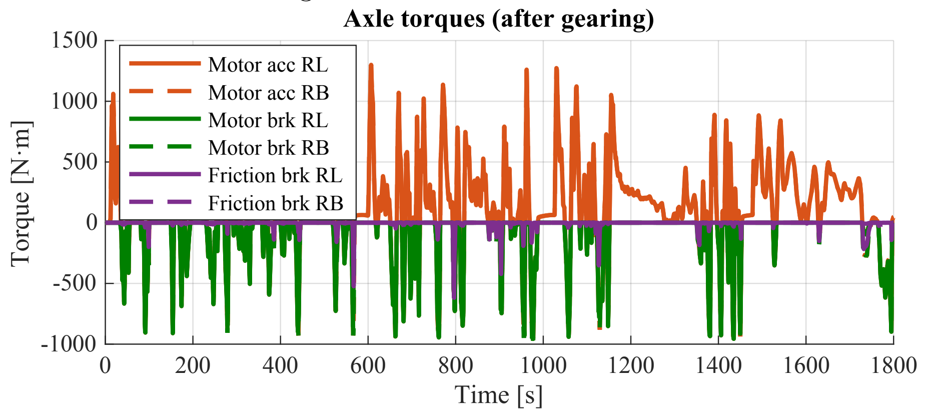


Figure B.29: SoC start = 95%

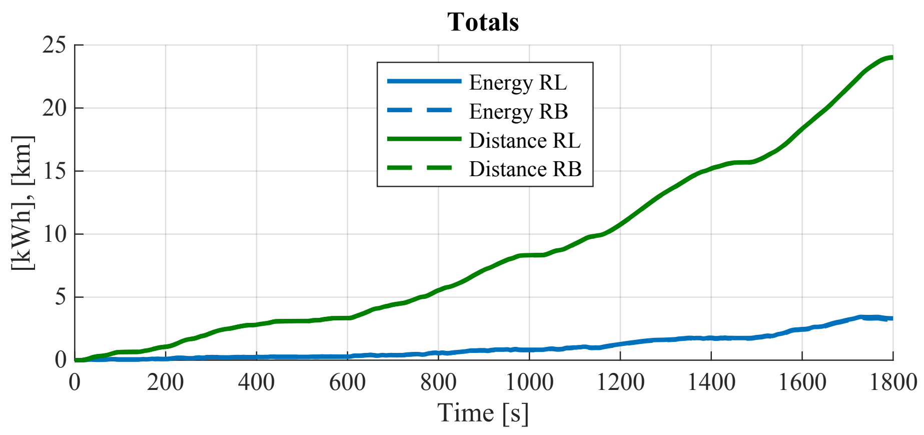


Figure B.30: SoC start = 95%

B.1.3 SOC start at 75%

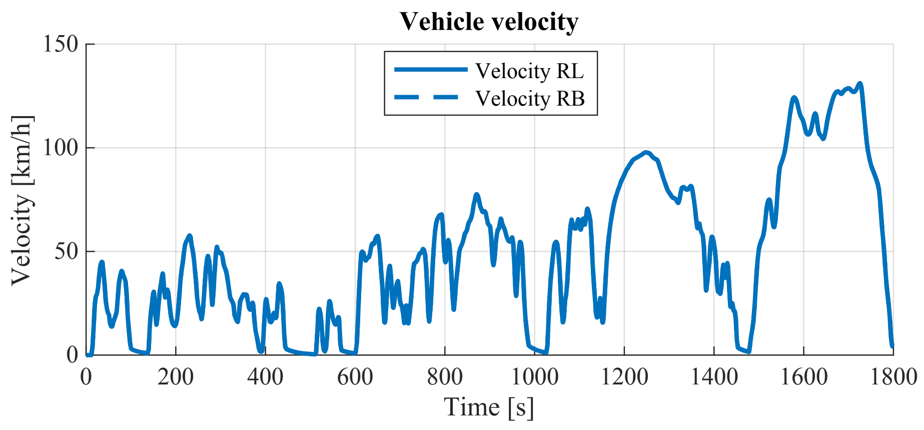


Figure B.31: SoC start = 75%

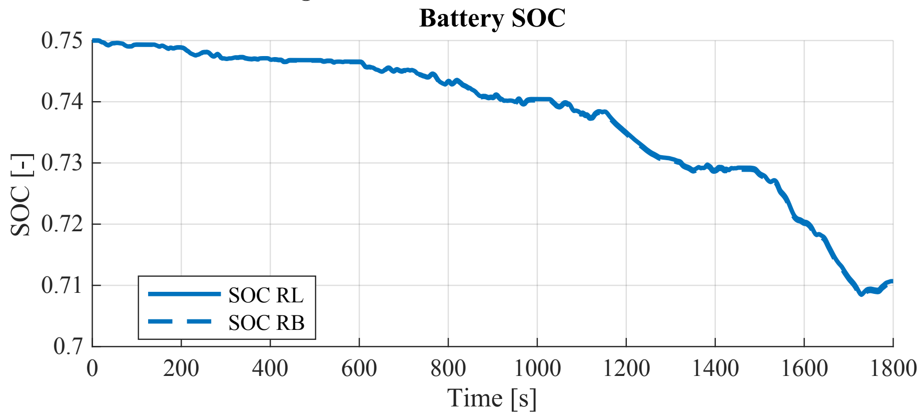


Figure B.32: SoC start = 75%

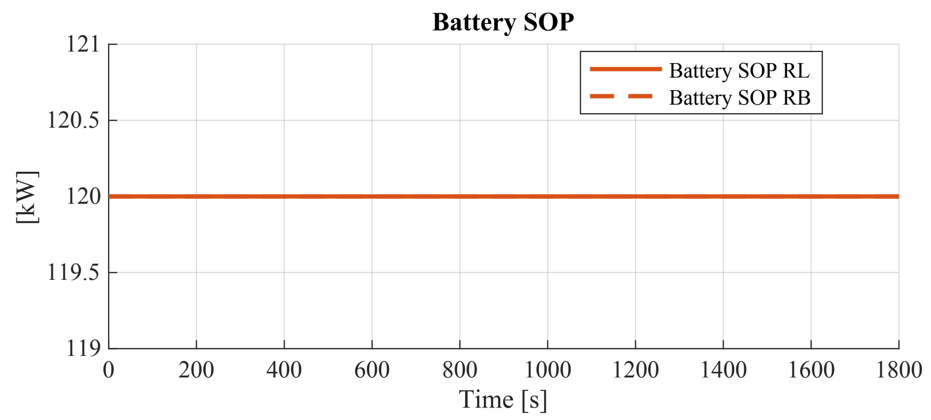


Figure B.33: SoC start = 75%

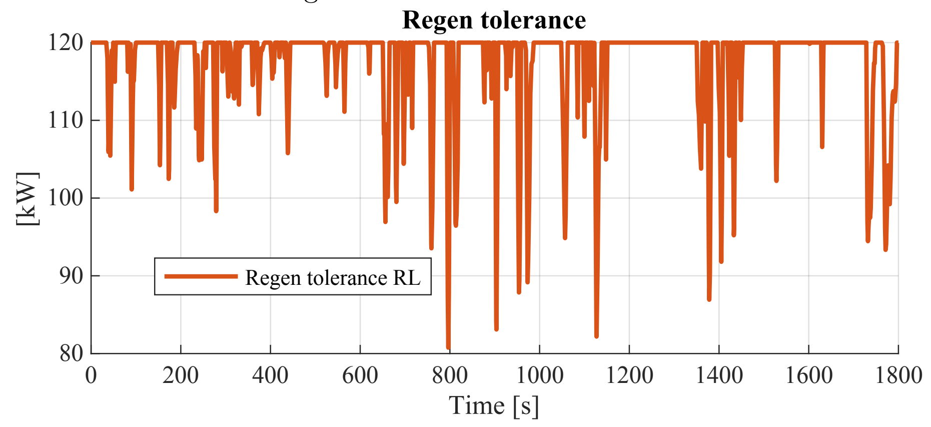


Figure B.34: SoC start = 75%

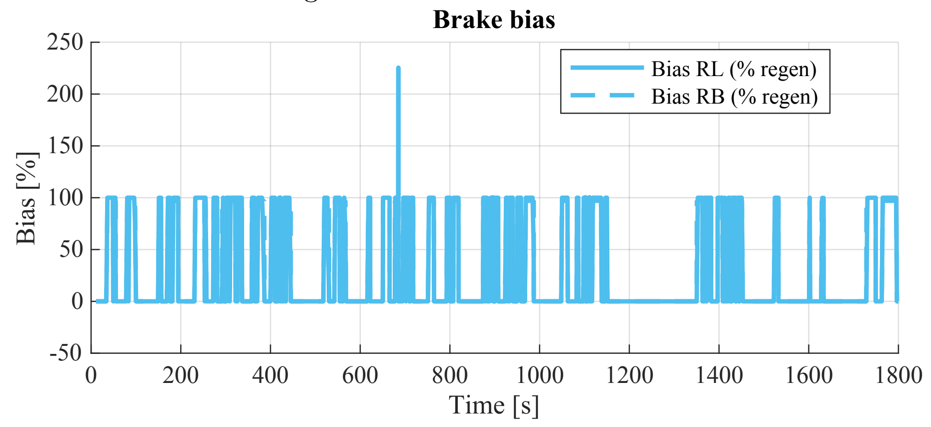


Figure B.35: SoC start = 75%

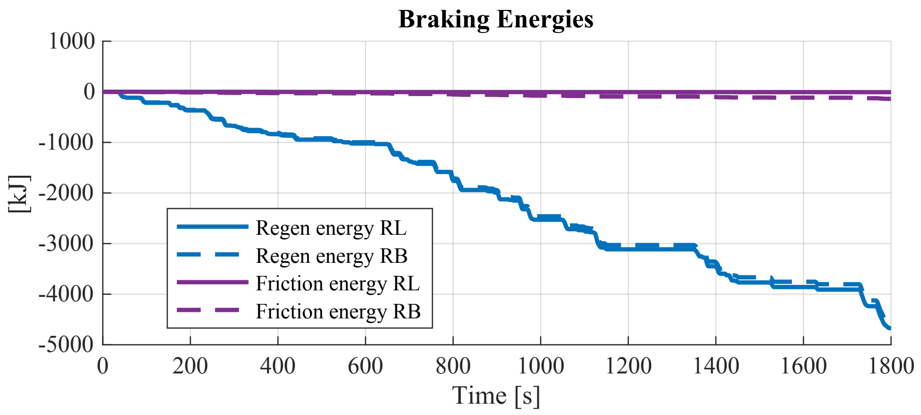


Figure B.36: SoC start = 75%

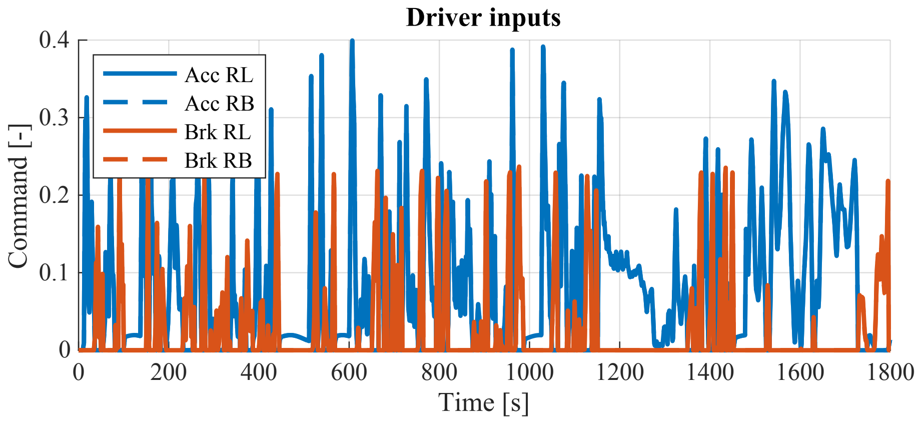


Figure B.37: SoC start = 75%

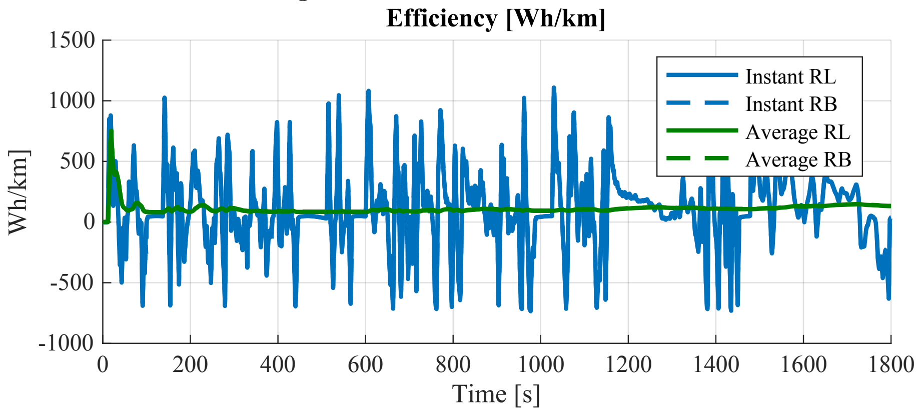


Figure B.38: SoC start = 75%

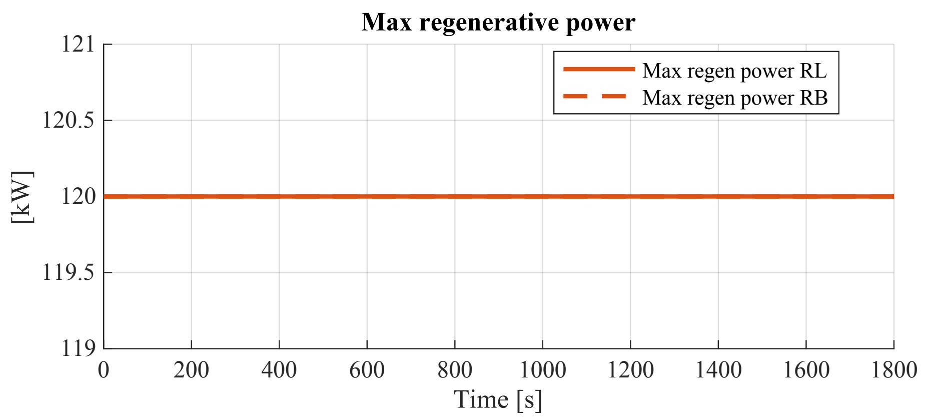


Figure B.39: SoC start = 75%

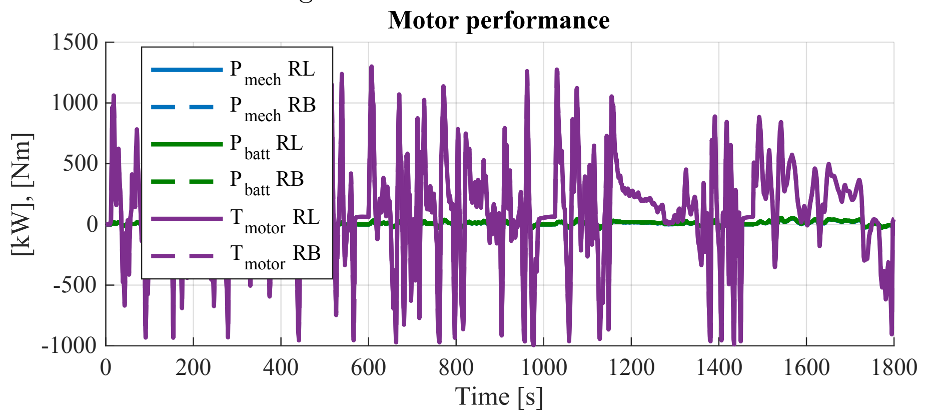


Figure B.40: SoC start = 75%

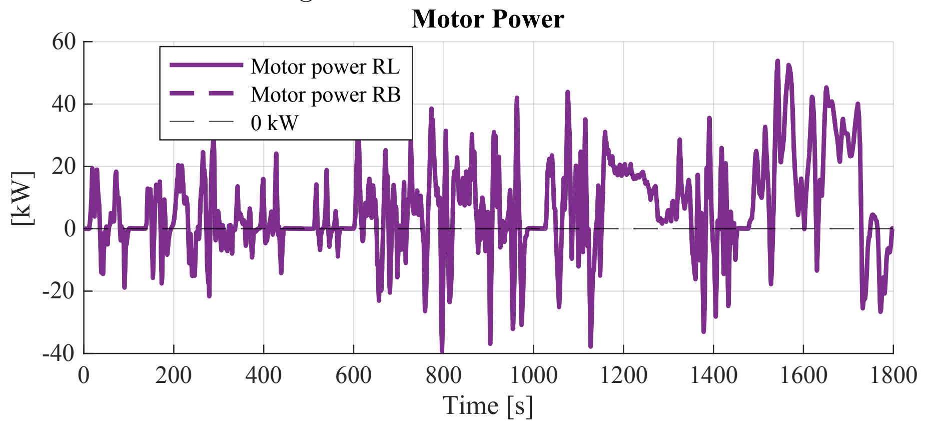


Figure B.41: SoC start = 75%

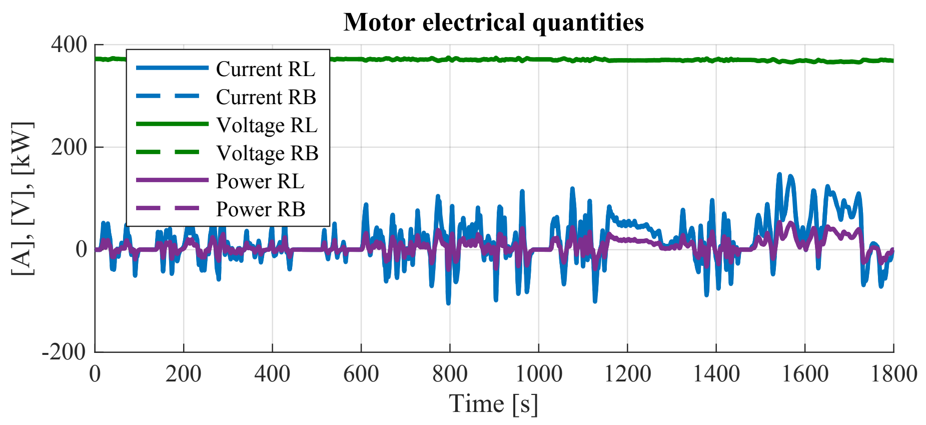


Figure B.42: SoC start = 75%

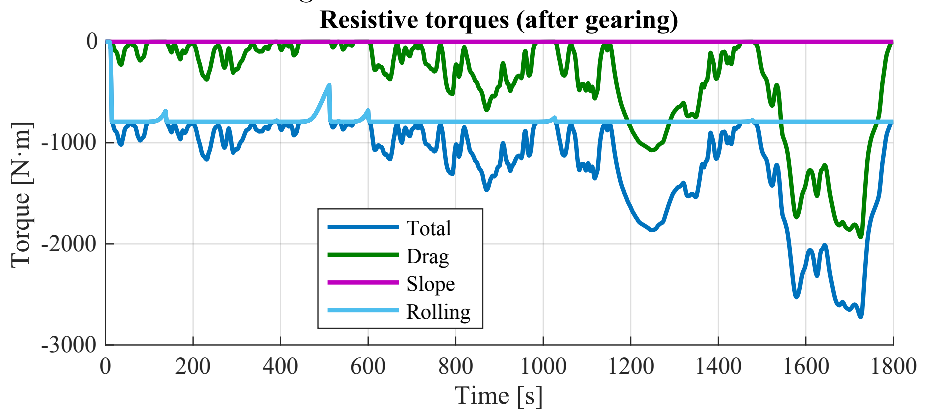


Figure B.43: SoC start = 75%

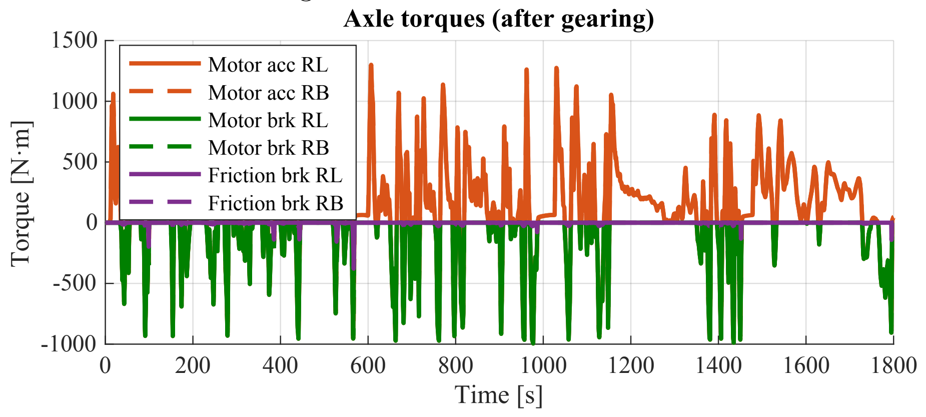


Figure B.44: SoC start = 75%

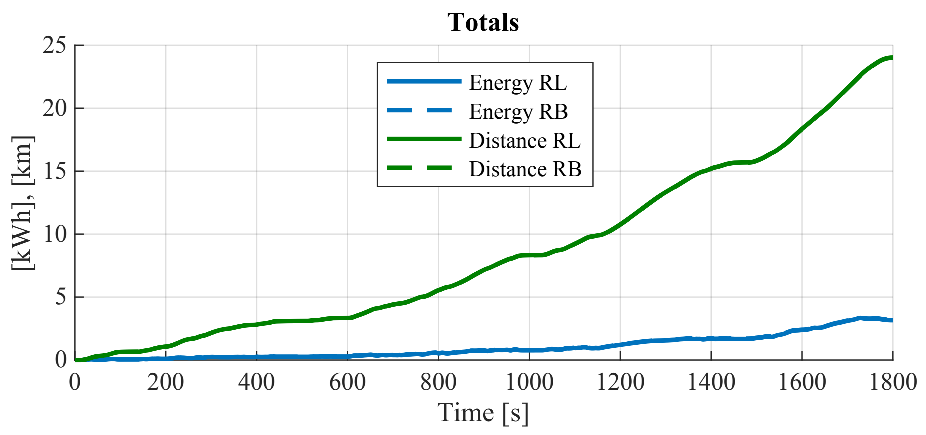


Figure B.45: SoC start = 75%

B.2 WLTP slope

B.2.1 SOC start at 100%

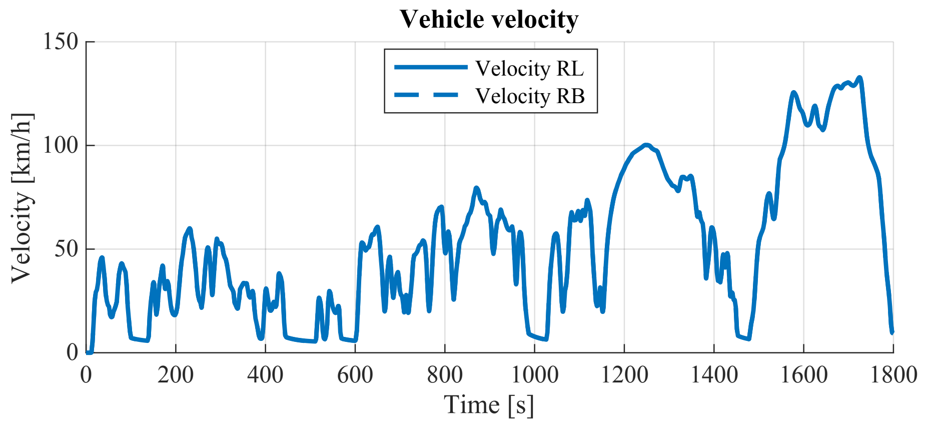


Figure B.46: SoC start = 100%

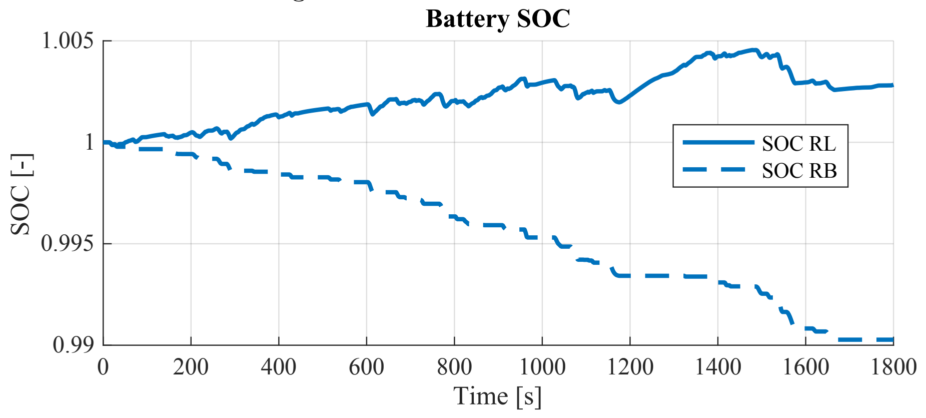


Figure B.47: SoC start = 100%

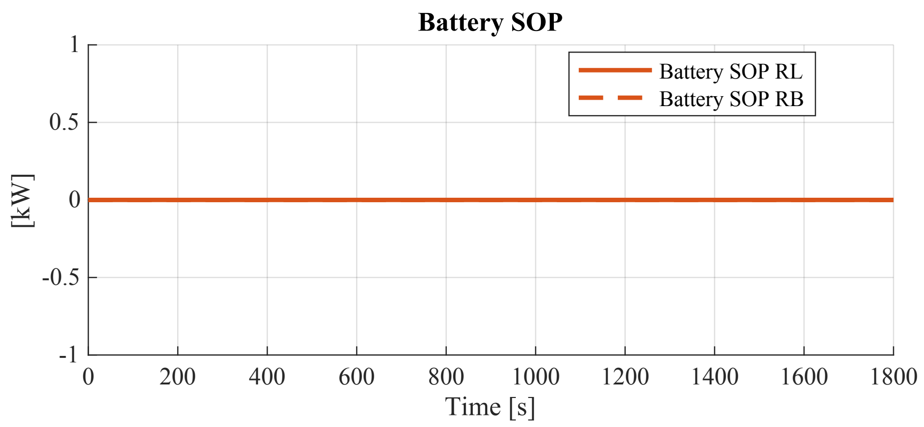


Figure B.48: SoC start = 100%

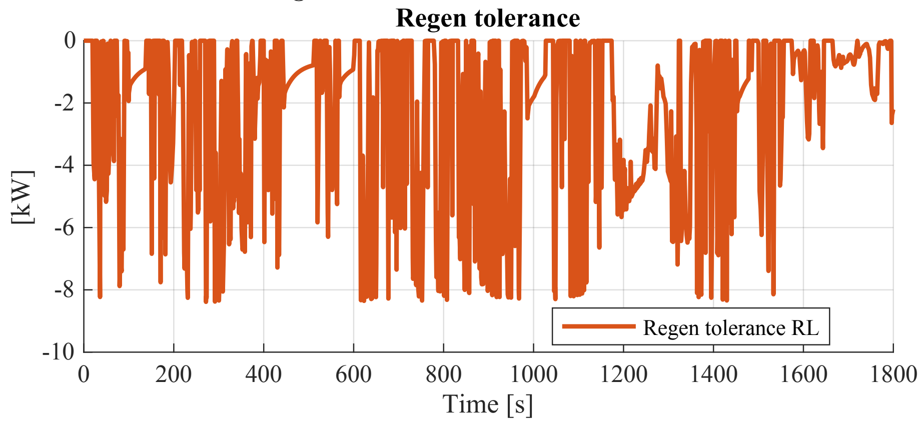


Figure B.49: SoC start = 100%

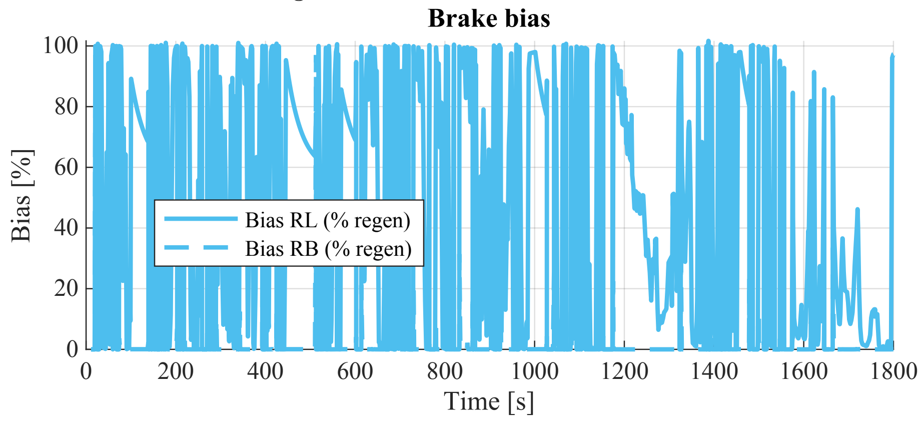


Figure B.50: SoC start = 100%

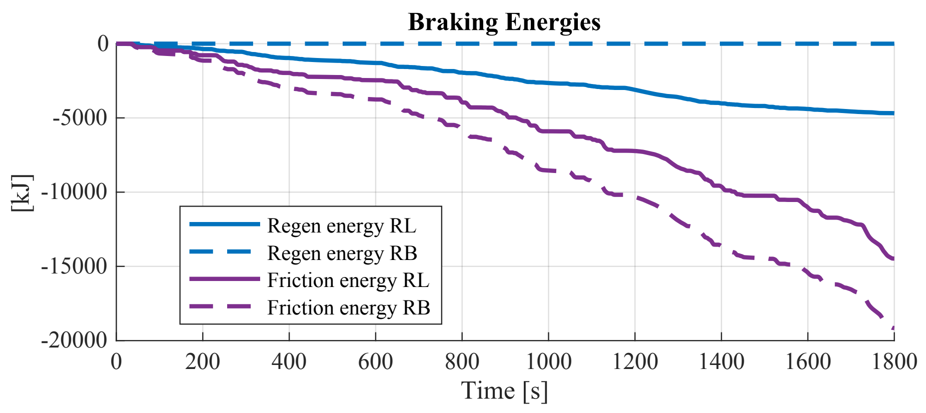


Figure B.51: SoC start = 100%

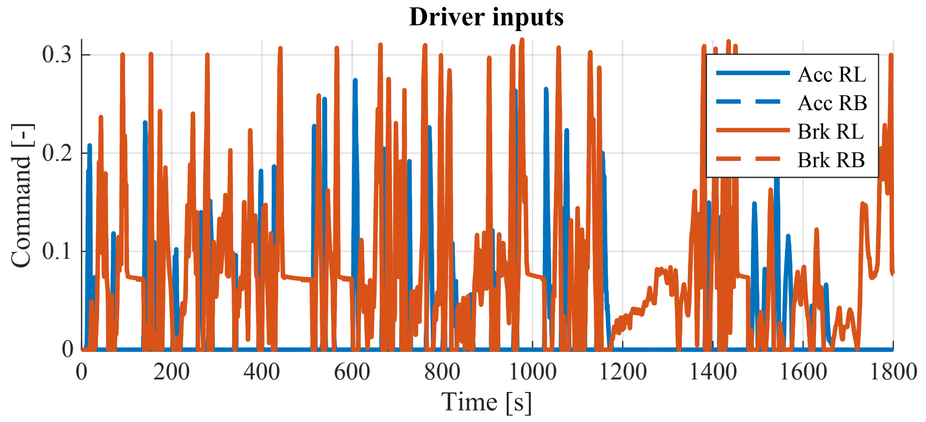


Figure B.52: SoC start = 100%

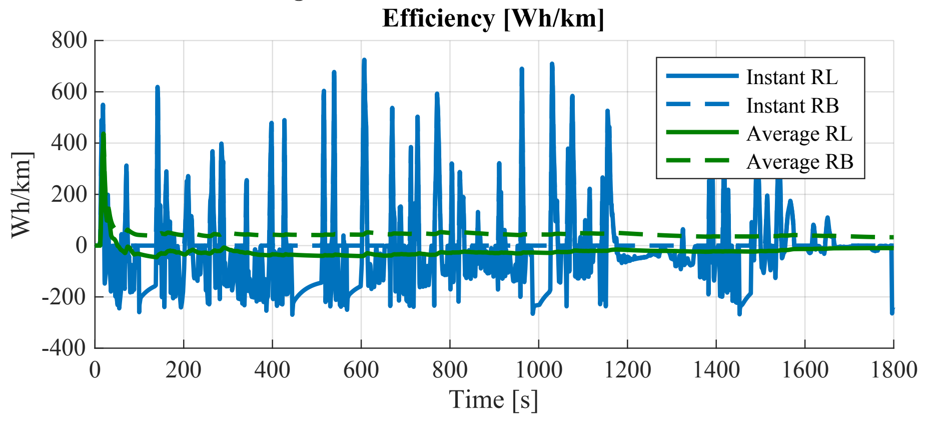


Figure B.53: SoC start = 100%

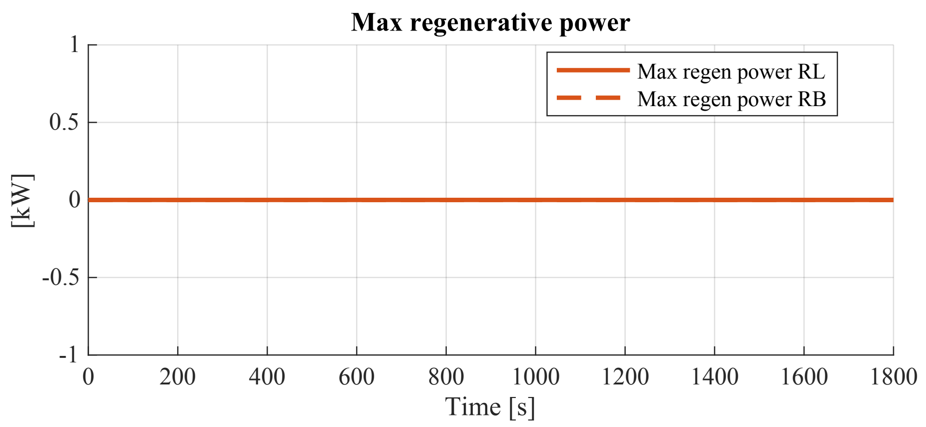


Figure B.54: SoC start = 100%

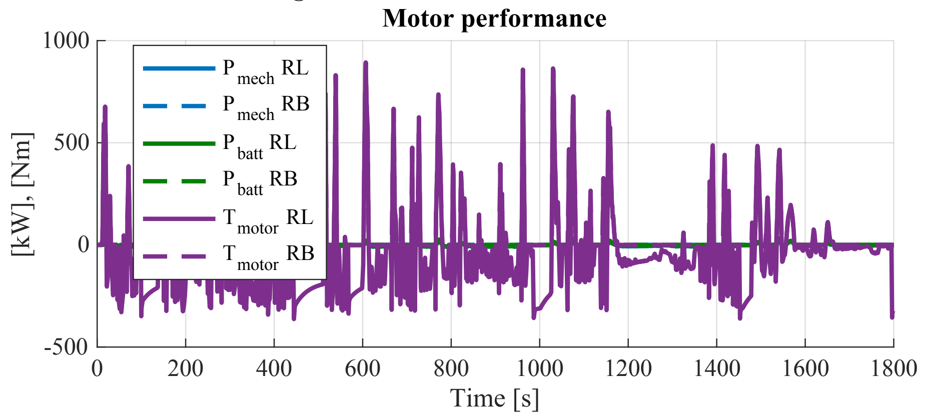


Figure B.55: SoC start = 100%

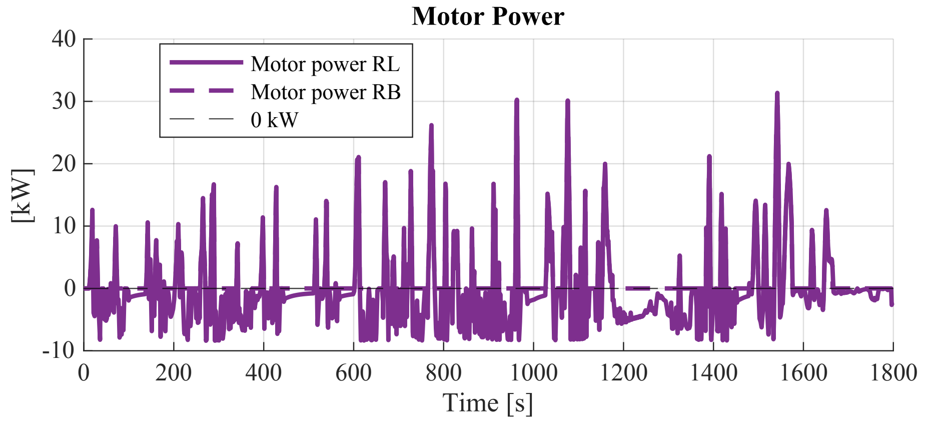


Figure B.56: SoC start = 100%

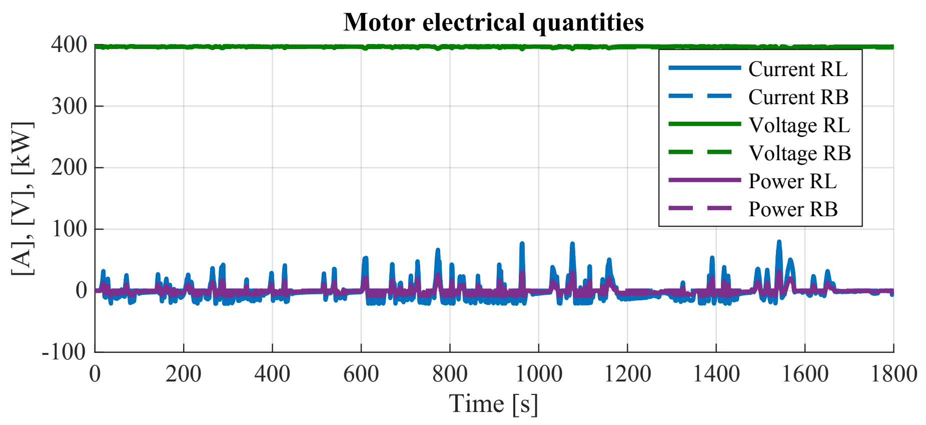


Figure B.57: SoC start = 100%

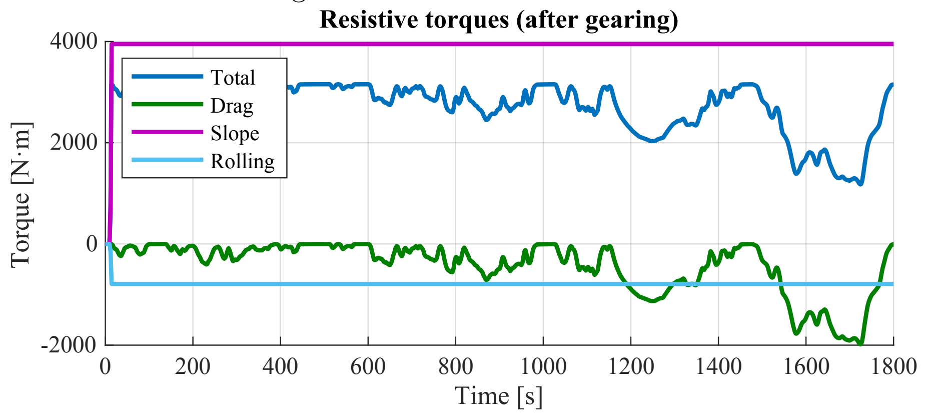


Figure B.58: SoC start = 100%

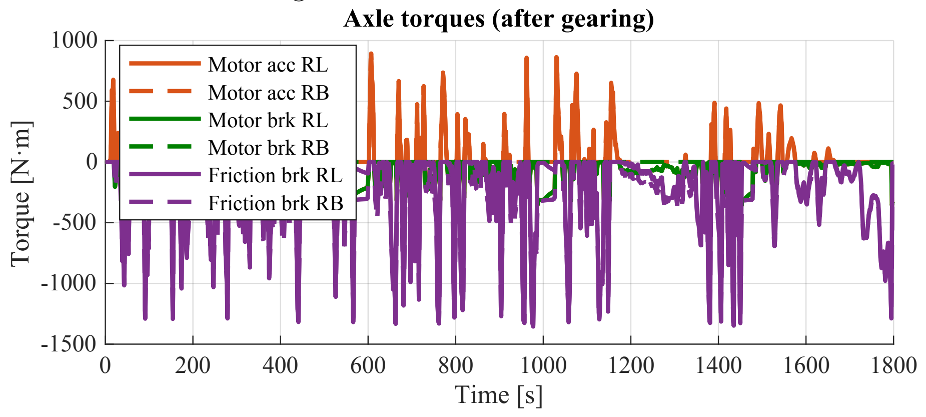


Figure B.59: SoC start = 100%

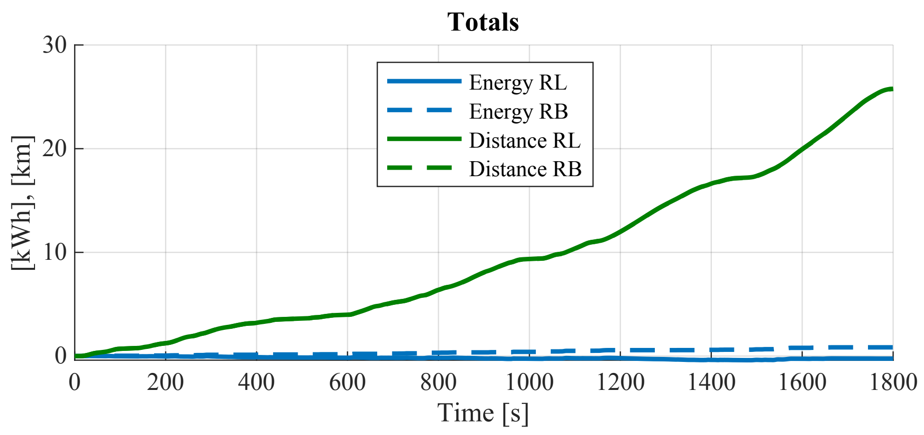


Figure B.60: SoC start = 100%

B.2.2 SOC start at 95%

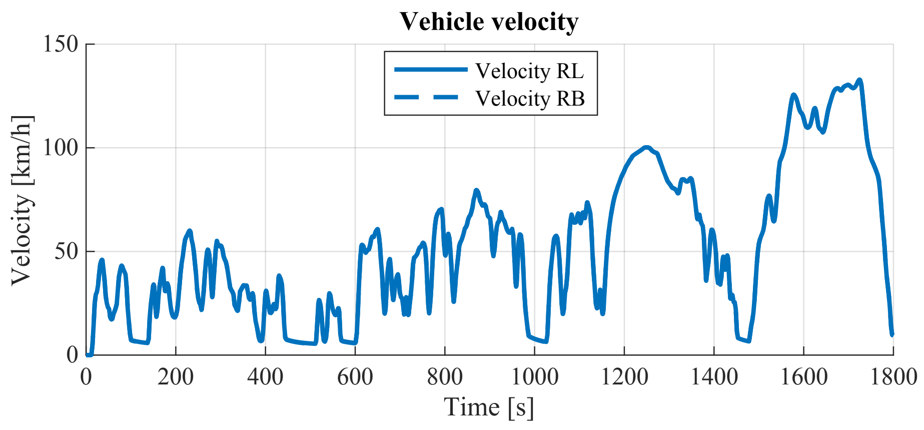


Figure B.61: SoC start = 95%

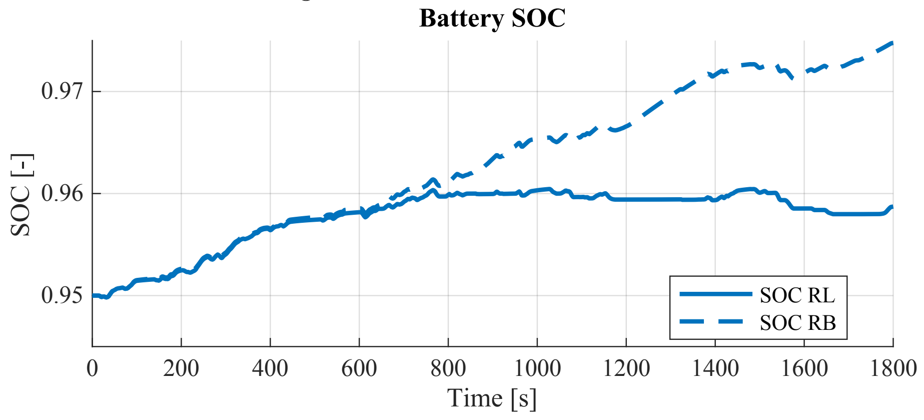


Figure B.62: SoC start = 95%

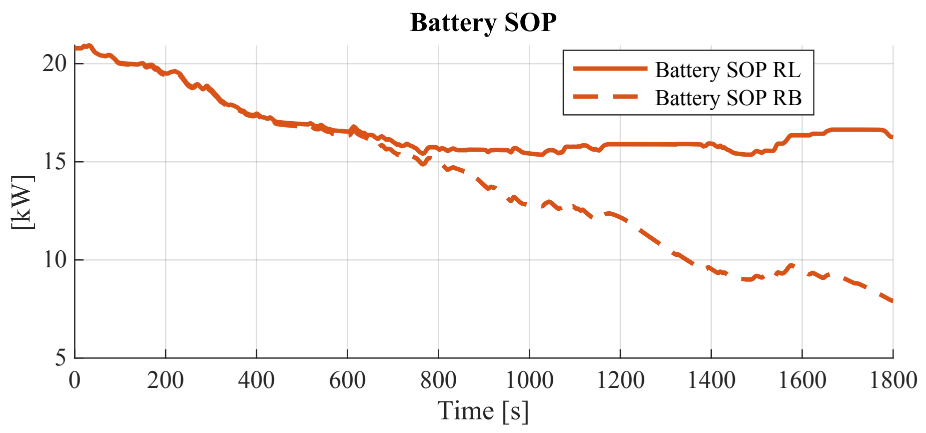


Figure B.63: SoC start = 95%

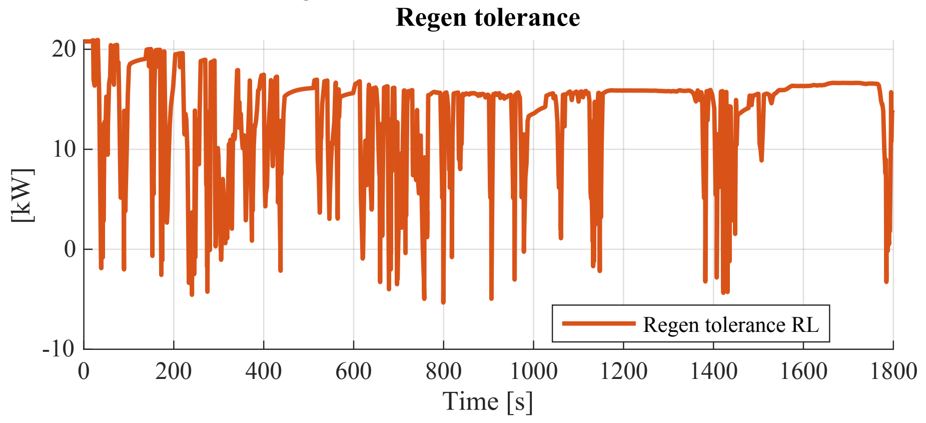


Figure B.64: SoC start = 95%

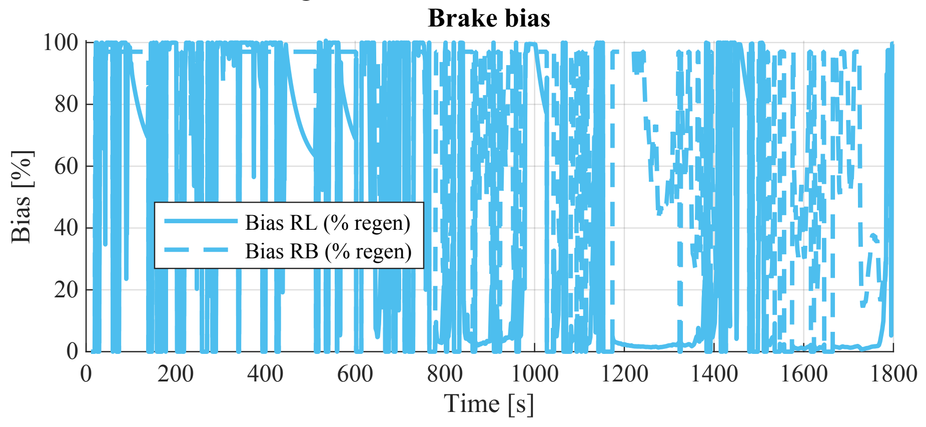


Figure B.65: SoC start = 95%

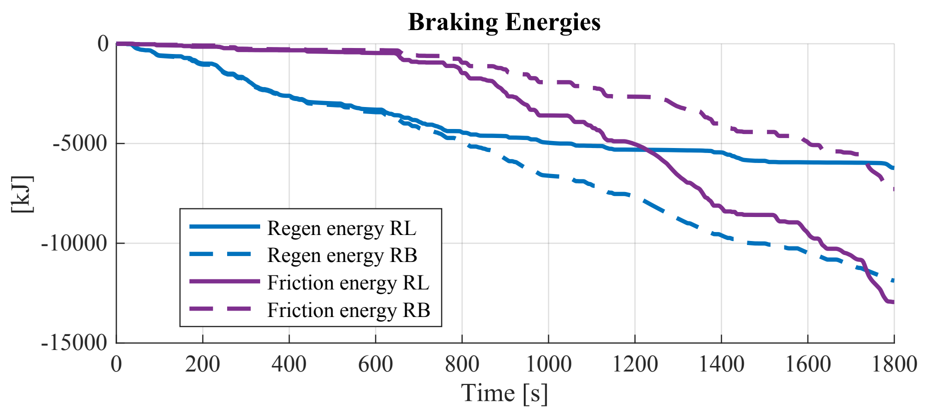


Figure B.66: SoC start = 95%

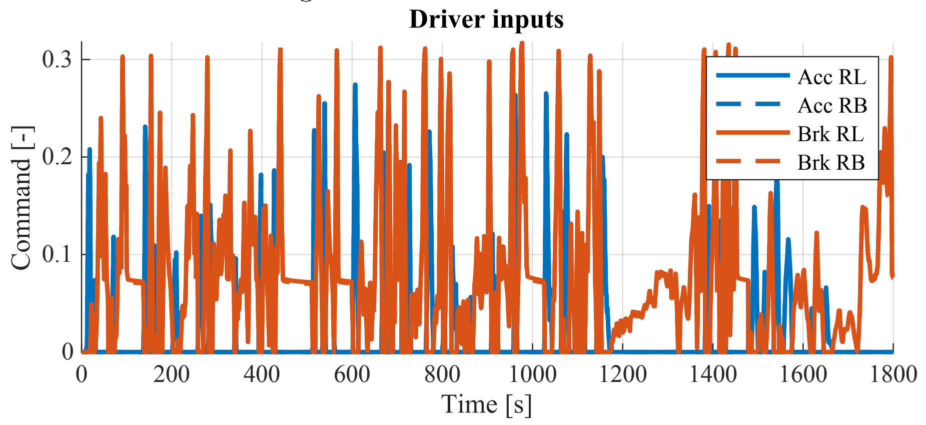


Figure B.67: SoC start = 95%

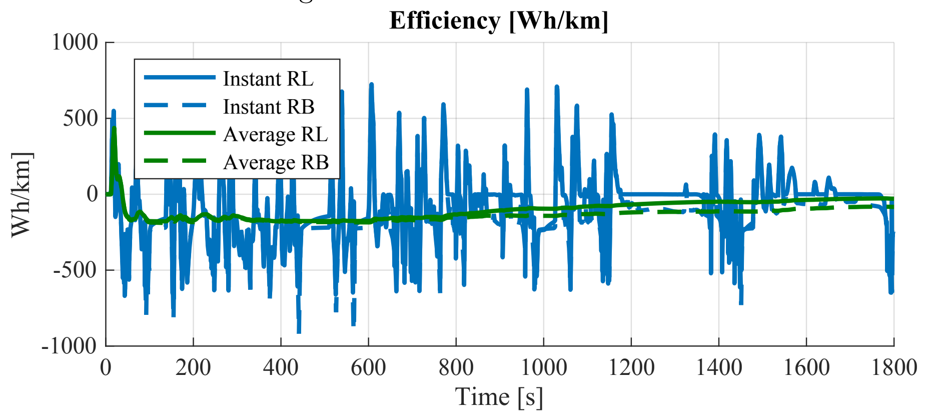


Figure B.68: SoC start = 95%

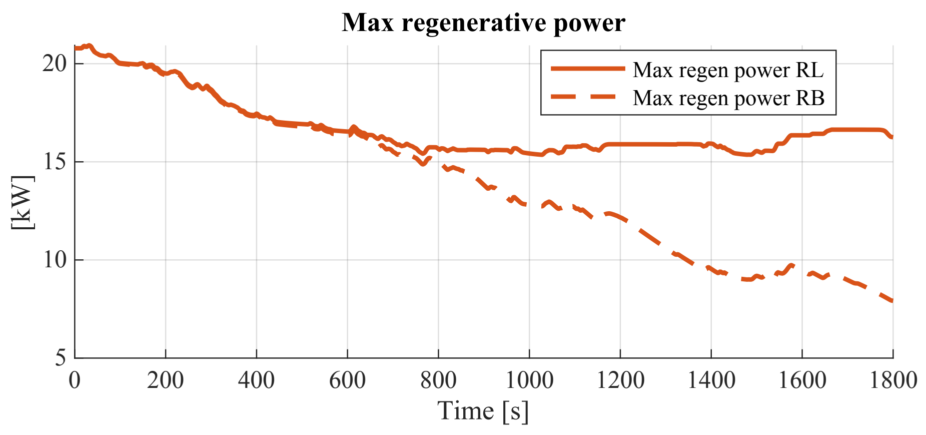


Figure B.69: SoC start = 95%

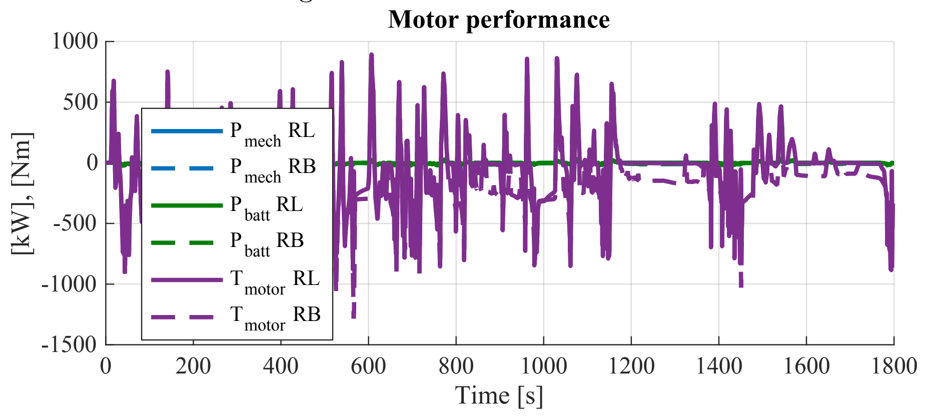


Figure B.70: SoC start = 95%

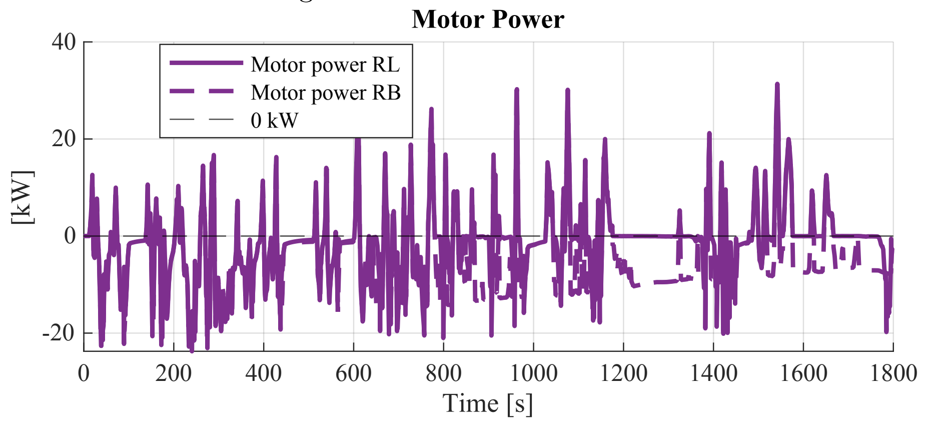


Figure B.71: SoC start = 95%

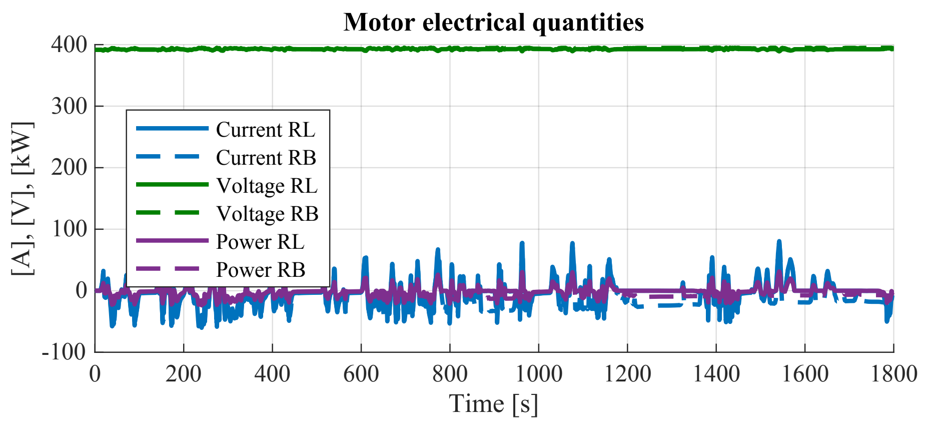


Figure B.72: SoC start = 95%

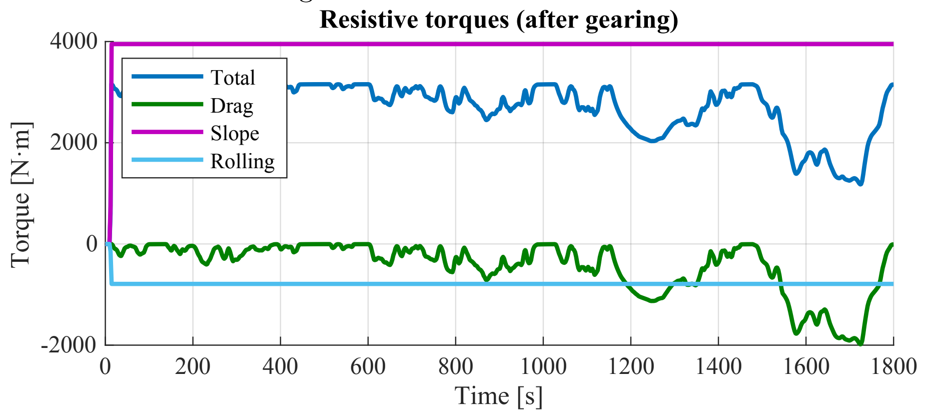


Figure B.73: SoC start = 95%

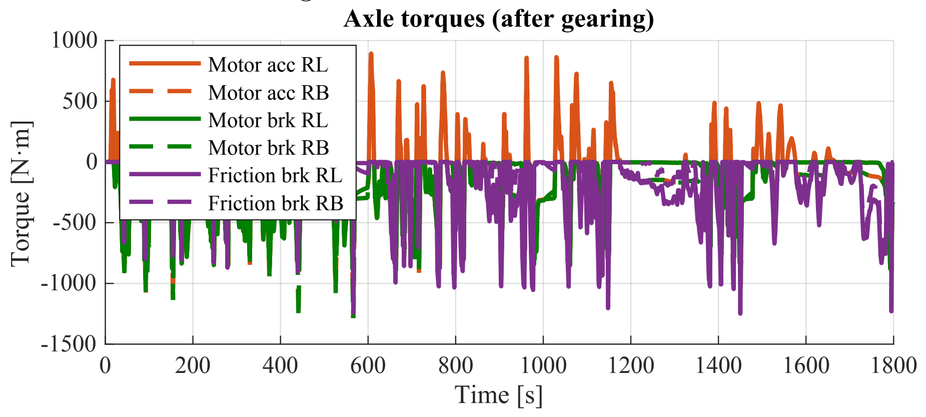


Figure B.74: SoC start = 95%

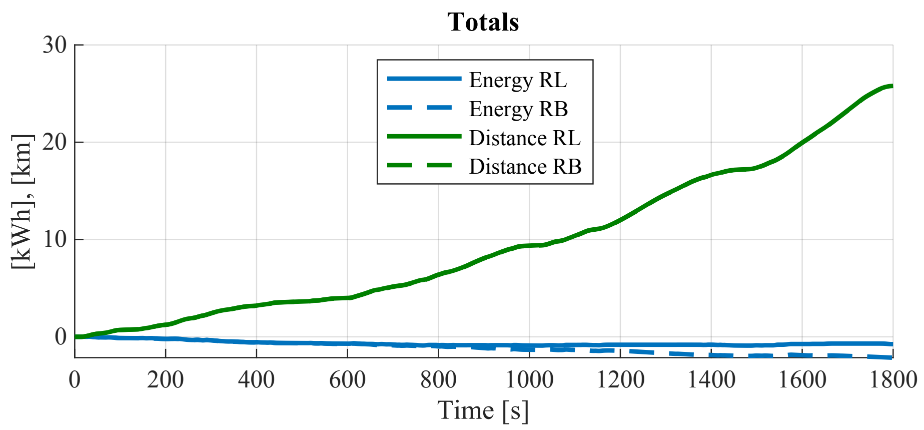


Figure B.75: SoC start = 95%

B.2.3 SOC start at 75%

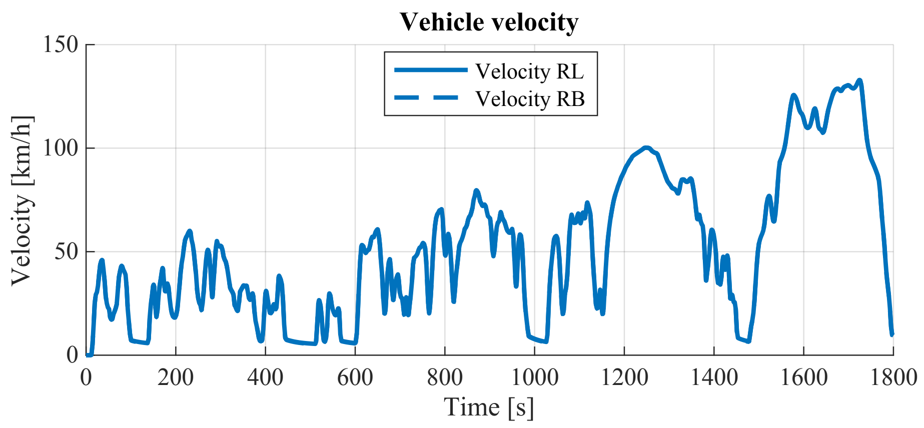


Figure B.76: SoC start = 75%

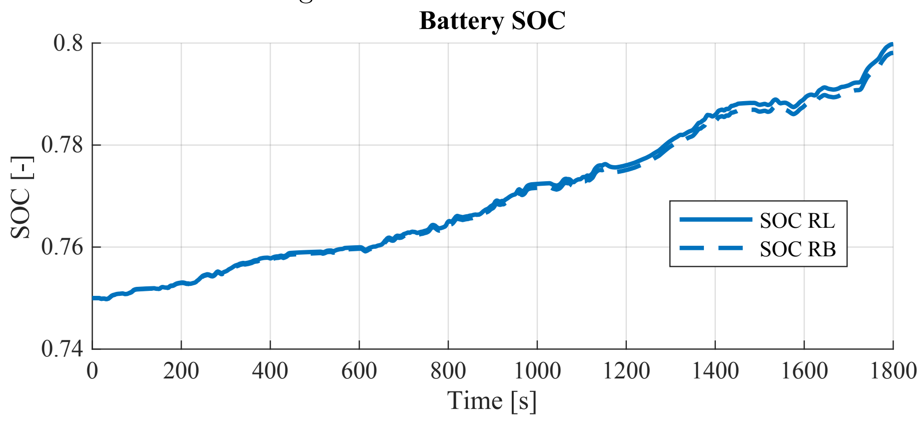


Figure B.77: SoC start = 75%

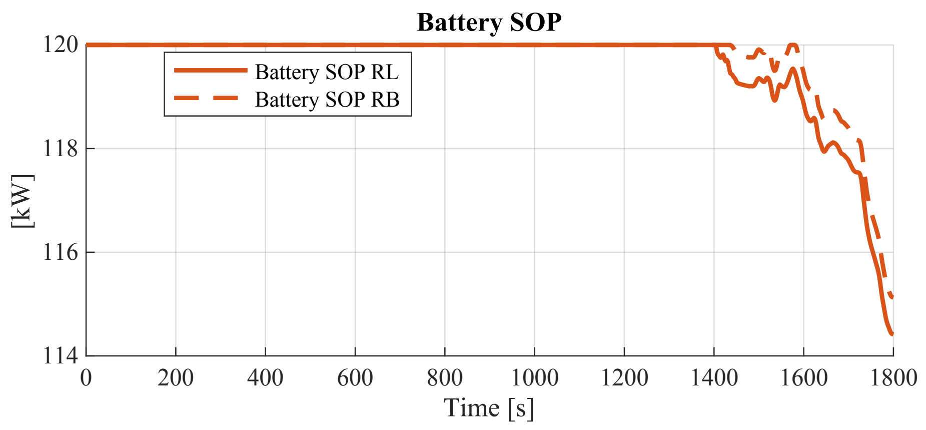


Figure B.78: SoC start = 75%

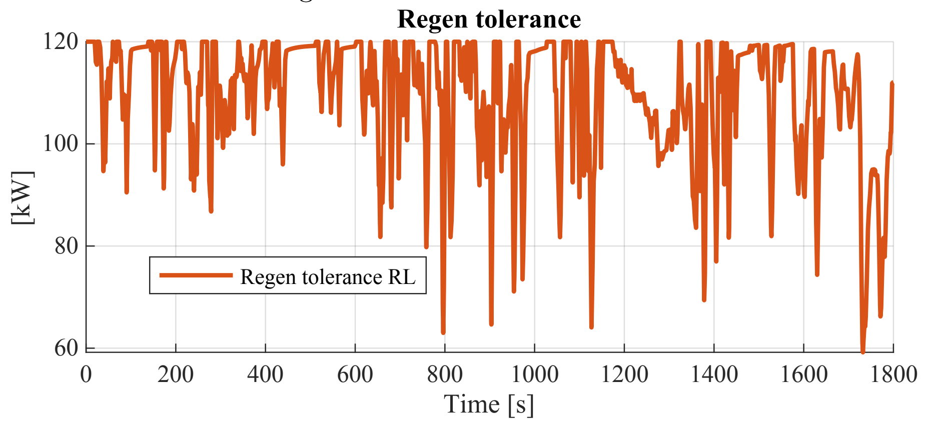


Figure B.79: SoC start = 75%

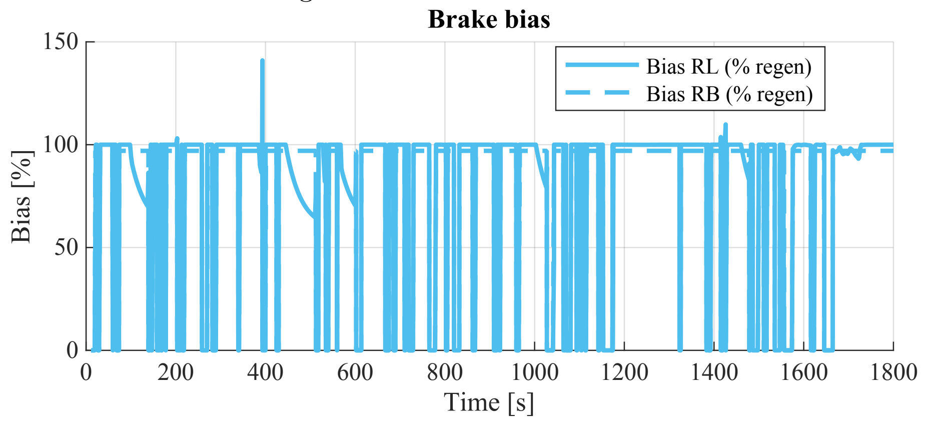


Figure B.80: SoC start = 75%

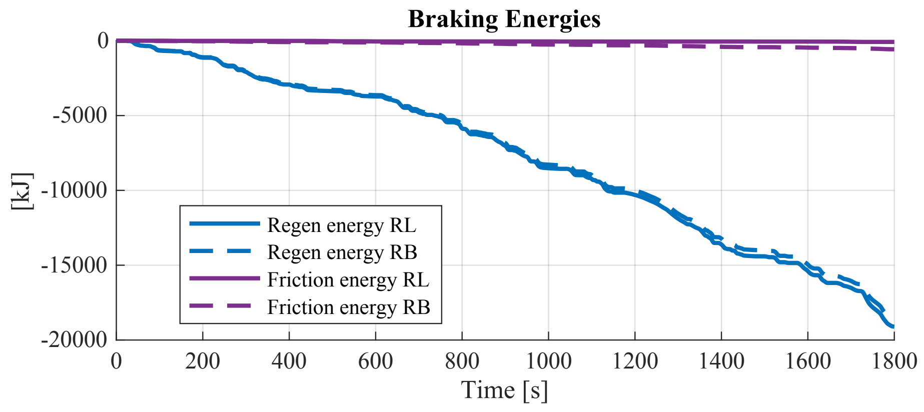


Figure B.81: SoC start = 75%

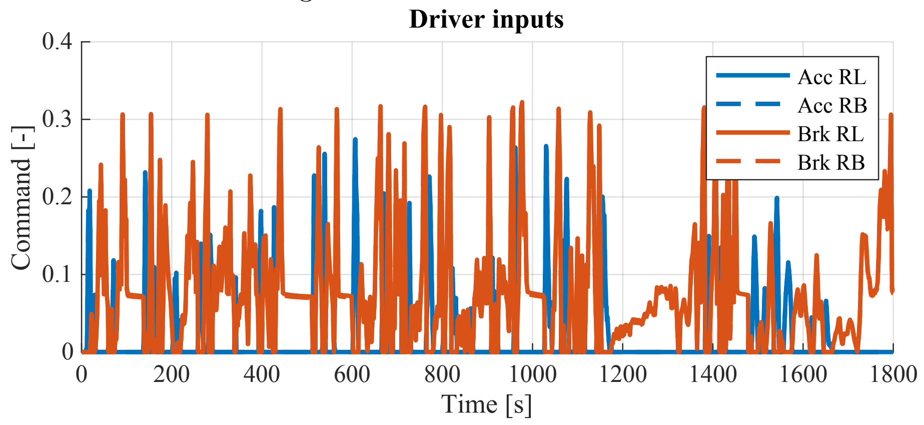


Figure B.82: SoC start = 75%

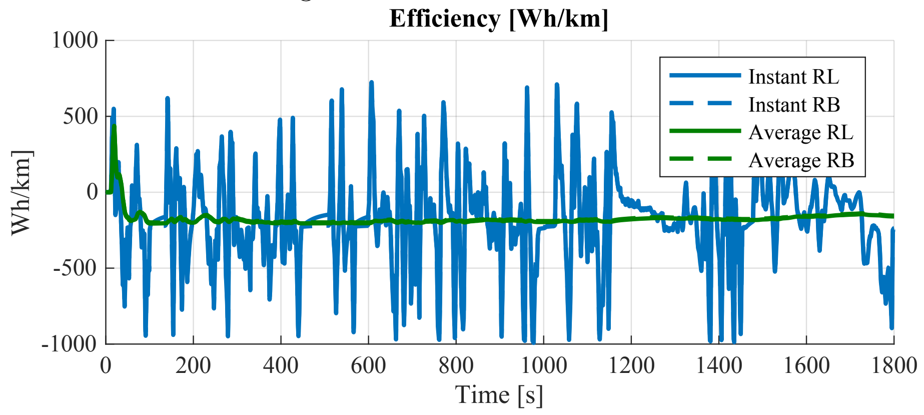


Figure B.83: SoC start = 75%

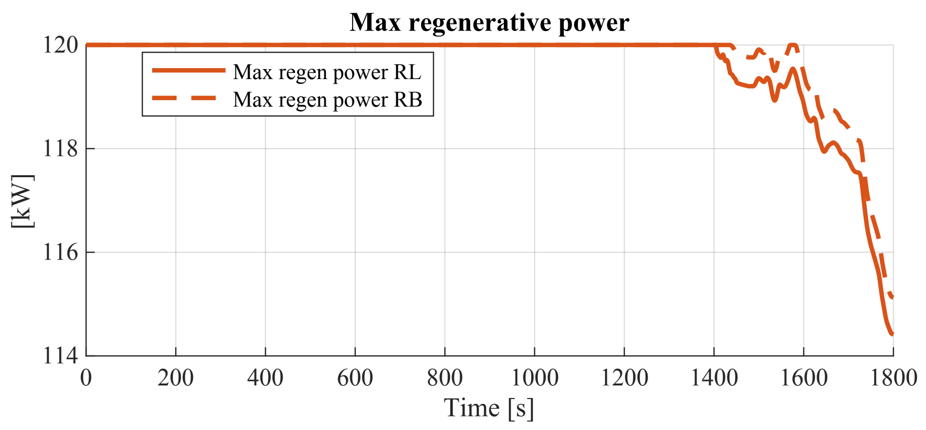


Figure B.84: SoC start = 75%

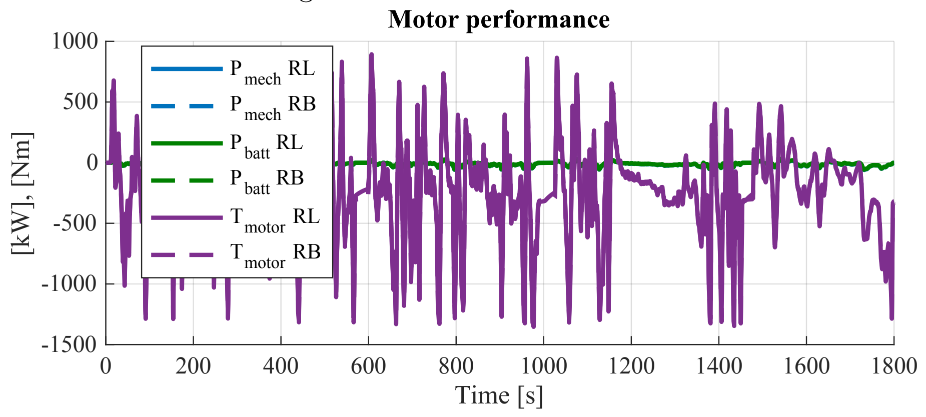


Figure B.85: SoC start = 75%

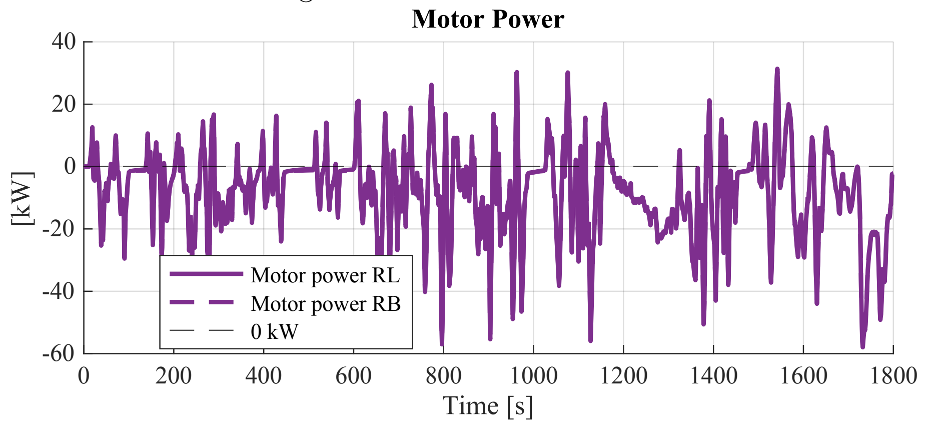


Figure B.86: SoC start = 75%

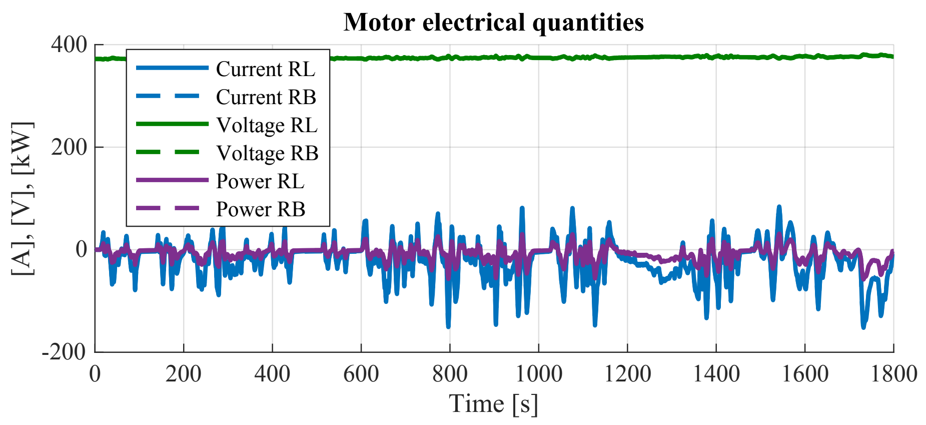


Figure B.87: SoC start = 75%

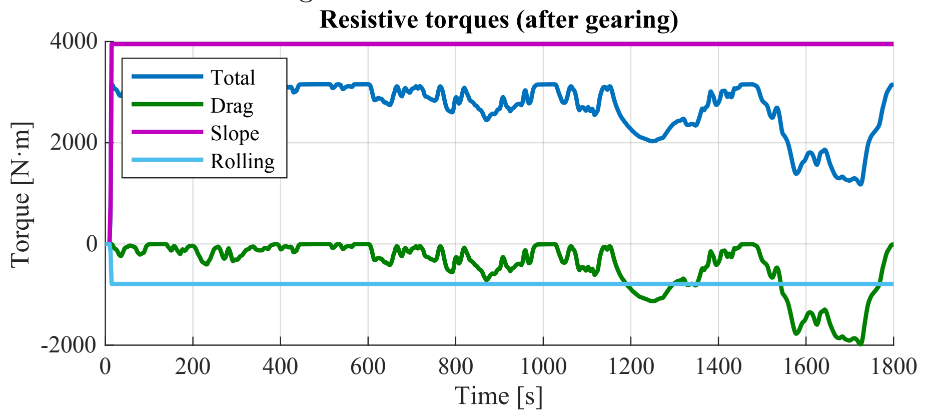


Figure B.88: SoC start = 75%

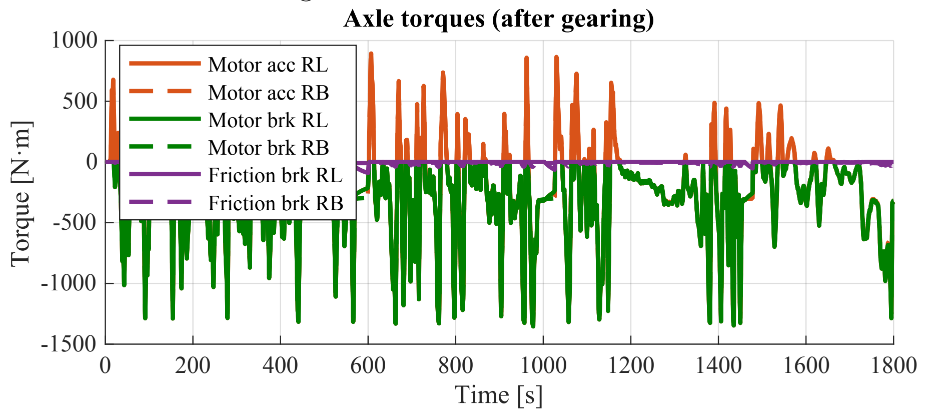


Figure B.89: SoC start = 75%

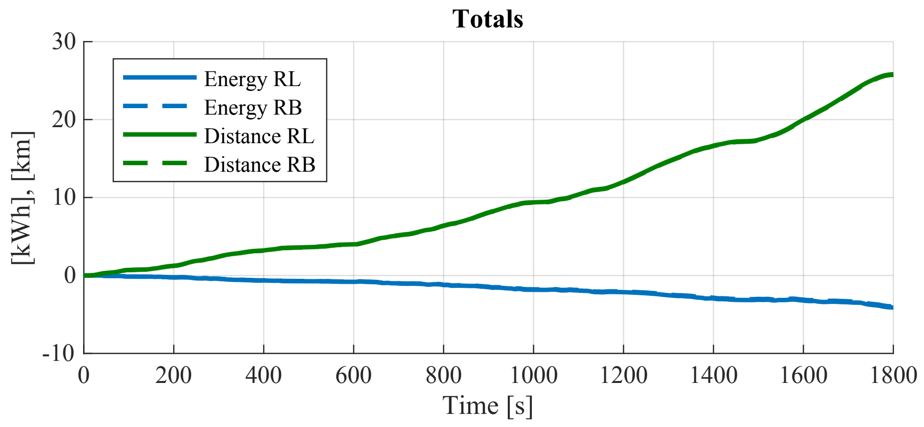


Figure B.90: SoC start = 75%

B.3 Random drivecycle standard

B.3.1 SOC start at 100%

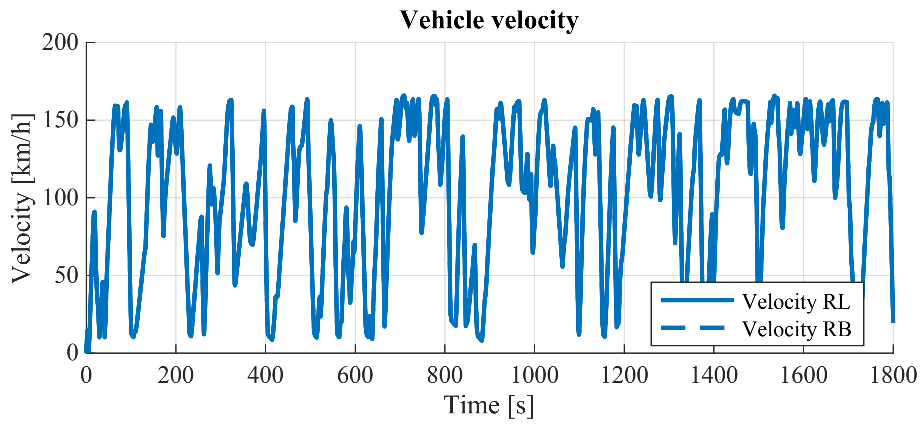


Figure B.91: SoC start = 100%

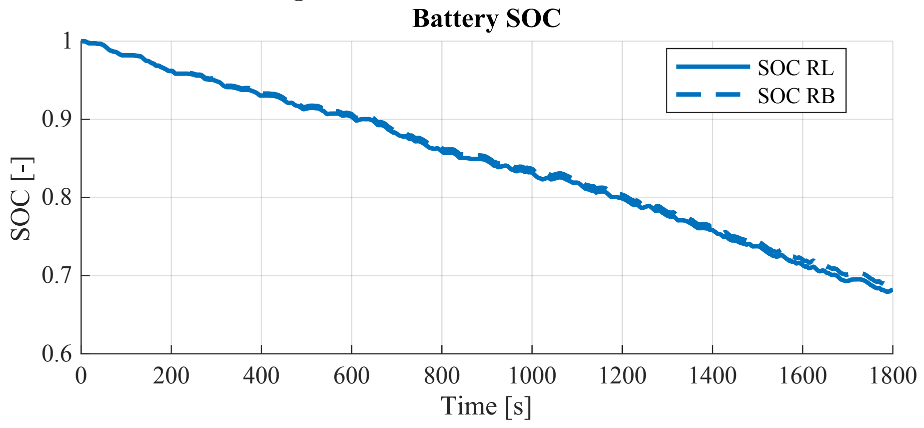


Figure B.92: SoC start = 100%

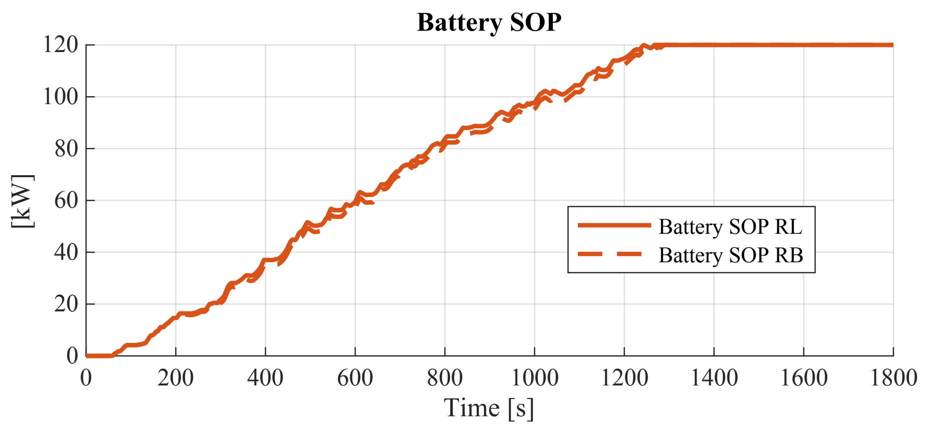


Figure B.93: SoC start = 100%

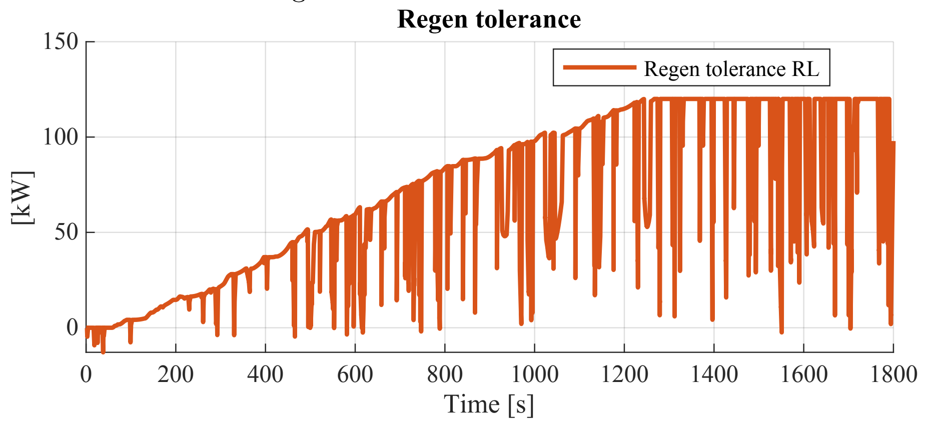


Figure B.94: SoC start = 100%

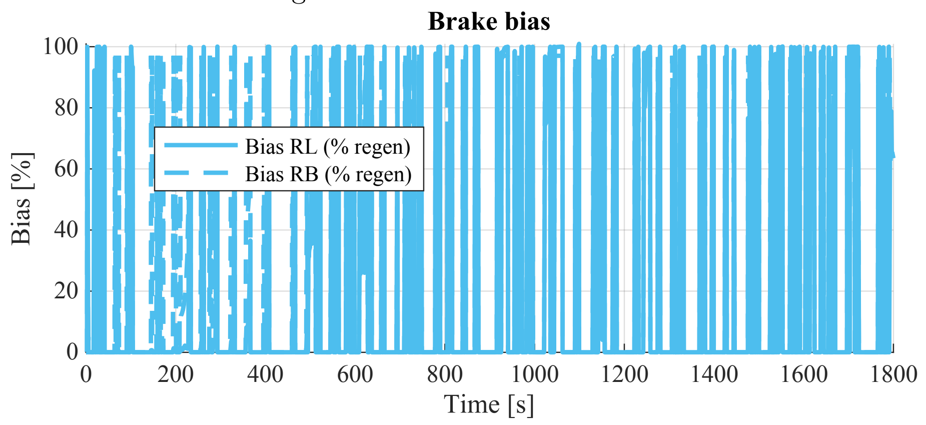


Figure B.95: SoC start = 100%

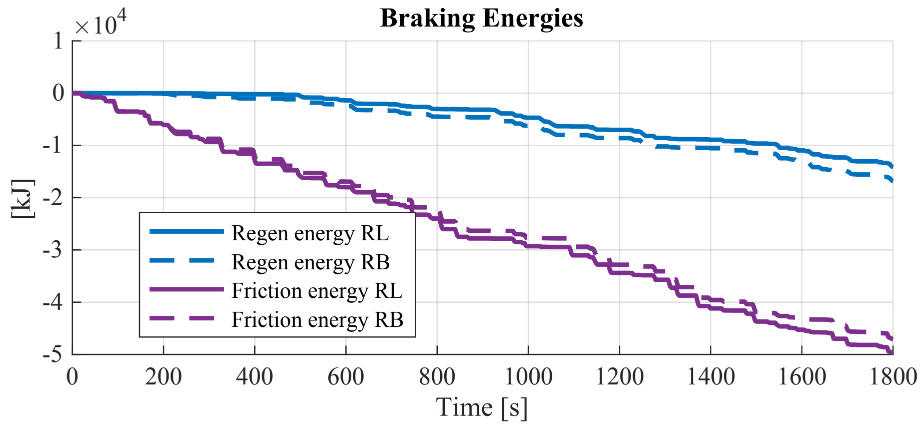


Figure B.96: SoC start = 100%

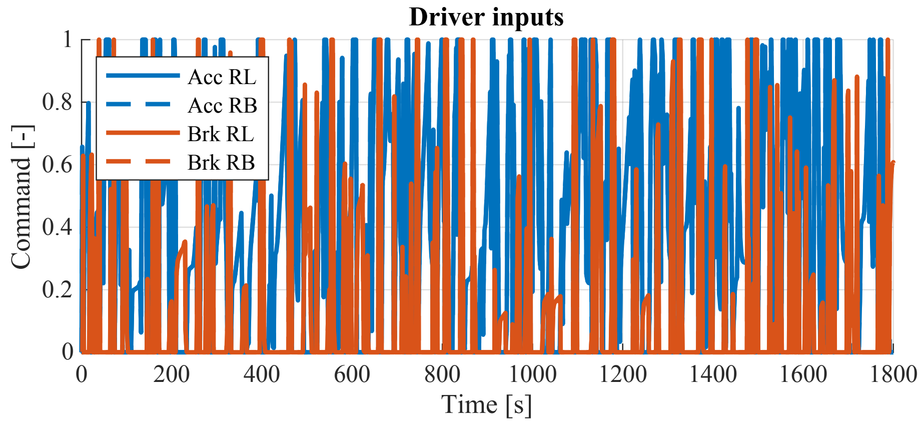


Figure B.97: SoC start = 100%

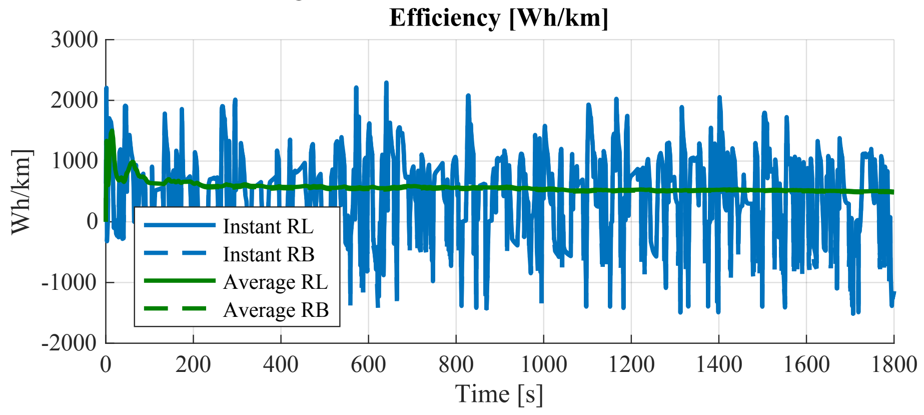


Figure B.98: SoC start = 100%

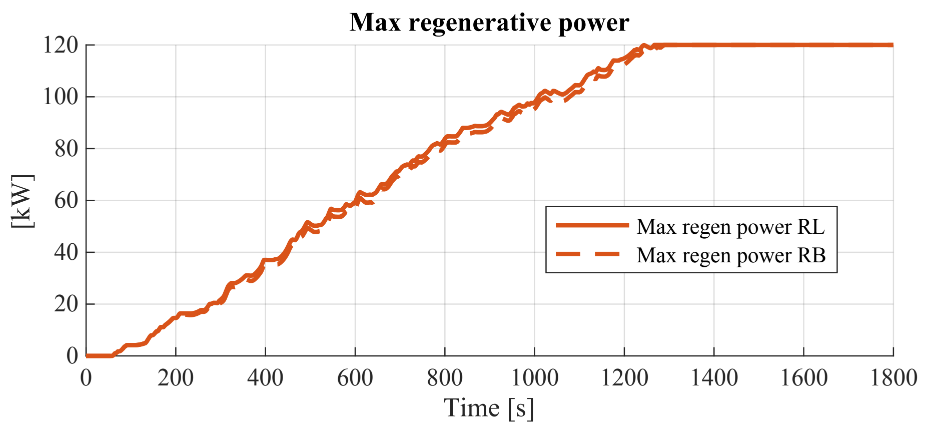


Figure B.99: SoC start = 100%

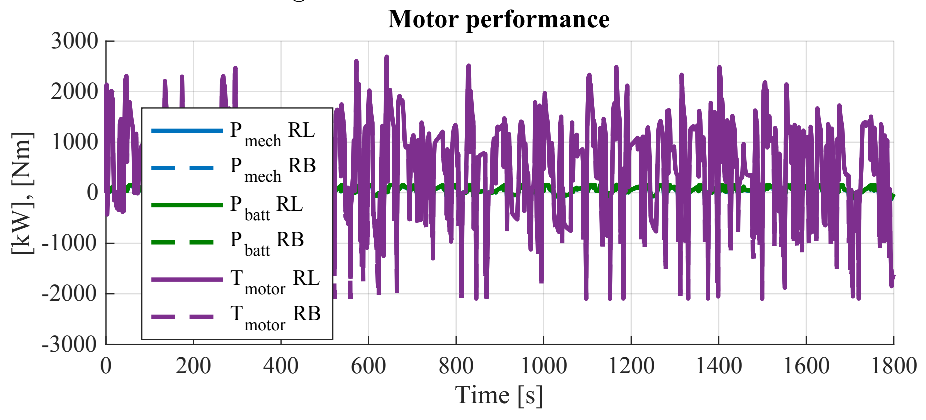


Figure B.100: SoC start = 100%

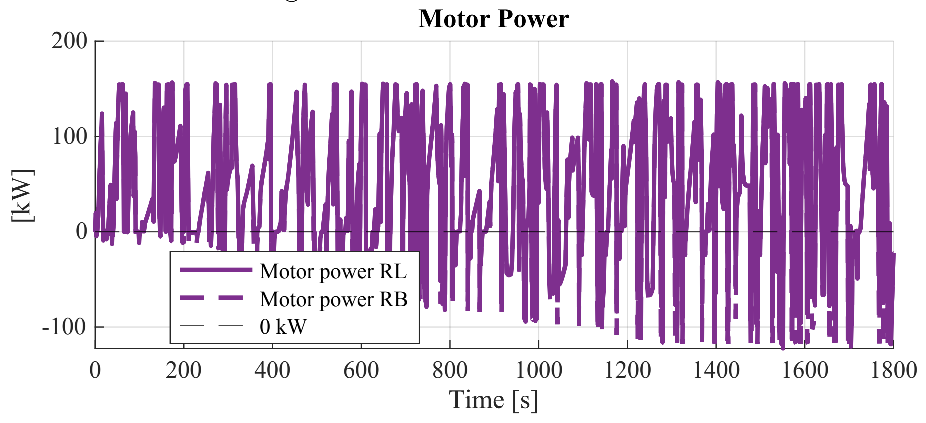


Figure B.101: SoC start = 100%

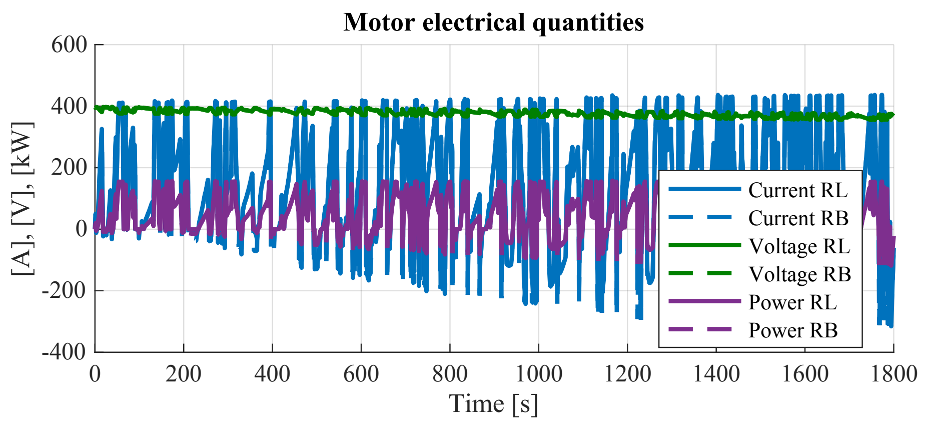


Figure B.102: SoC start = 100%

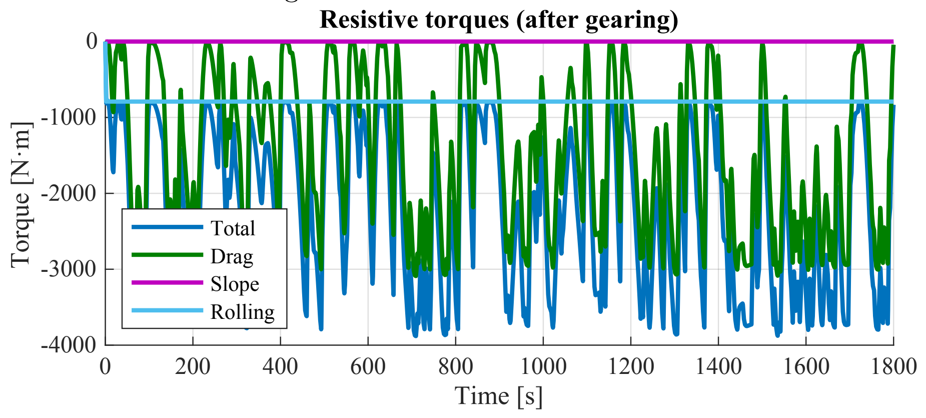


Figure B.103: SoC start = 100%

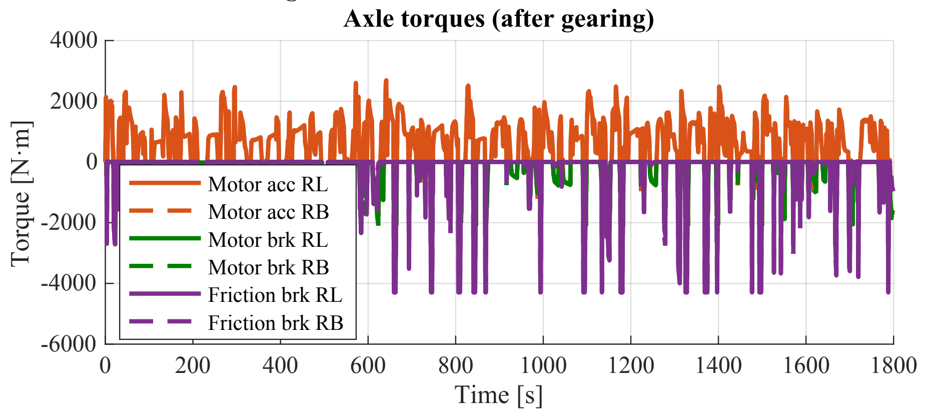


Figure B.104: SoC start = 100%

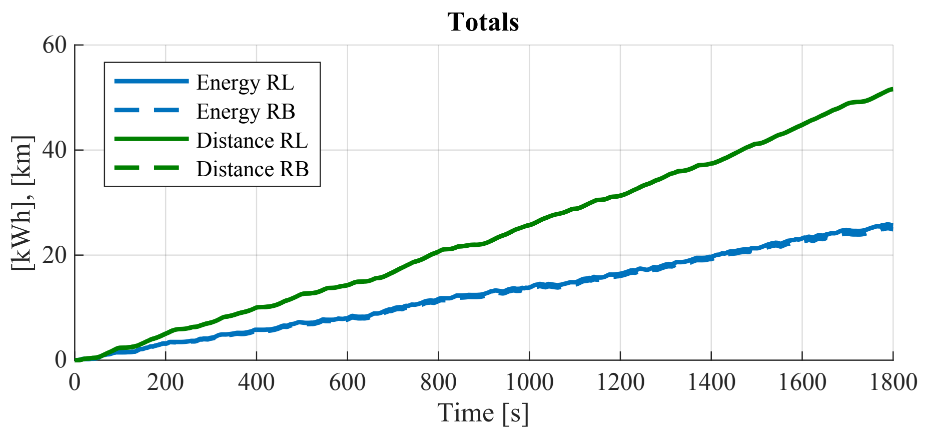


Figure B.105: SoC start = 100%

B.3.2 SOC start at 95%

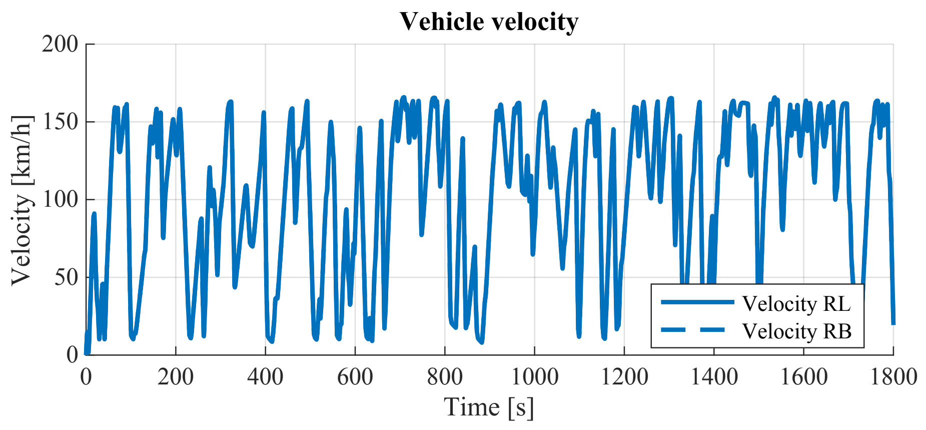


Figure B.106: SoC start = 95%

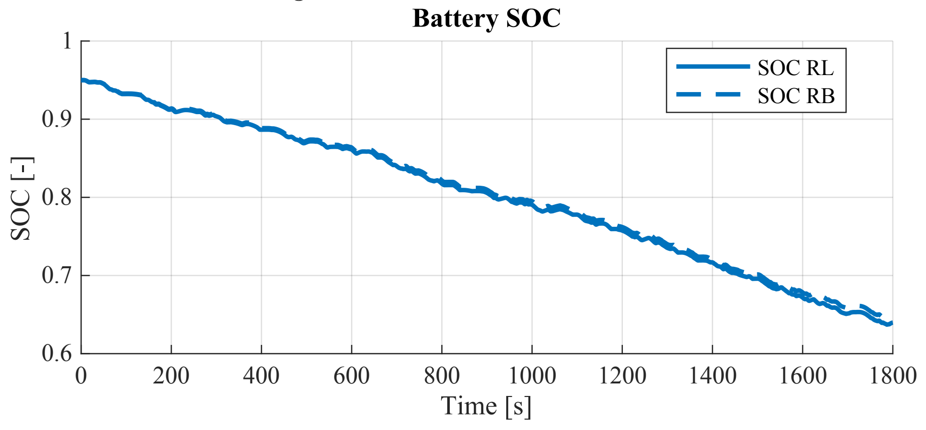


Figure B.107: SoC start = 95%

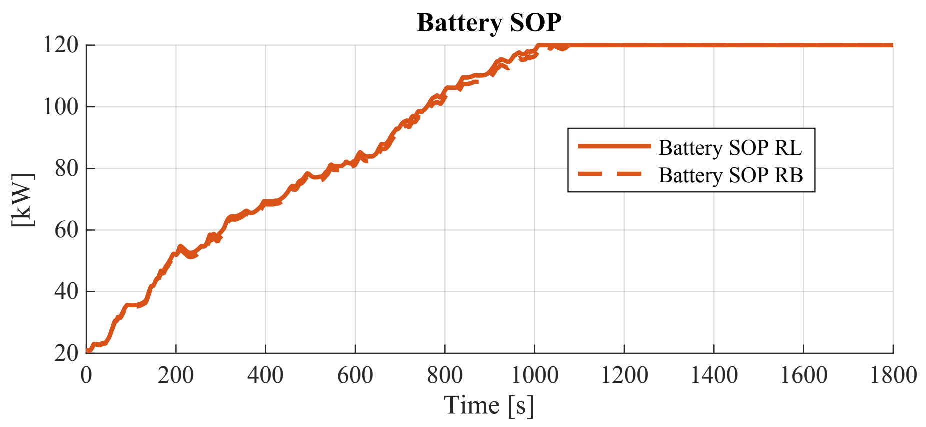


Figure B.108: SoC start = 95%

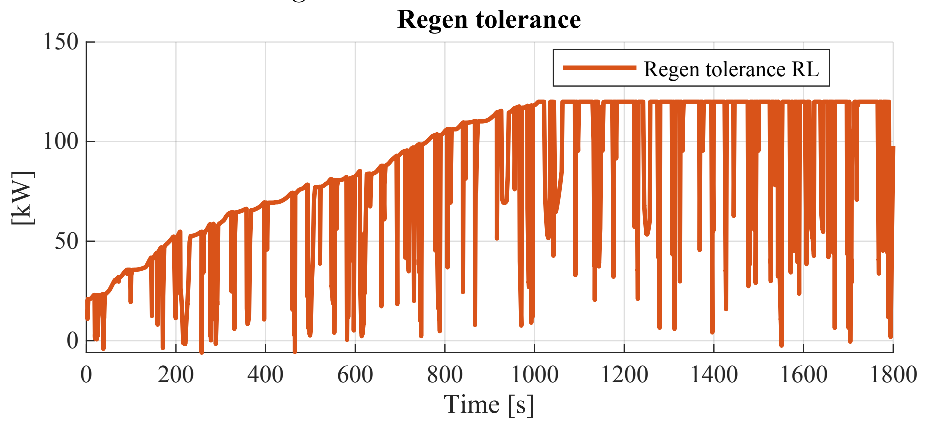


Figure B.109: SoC start = 95%

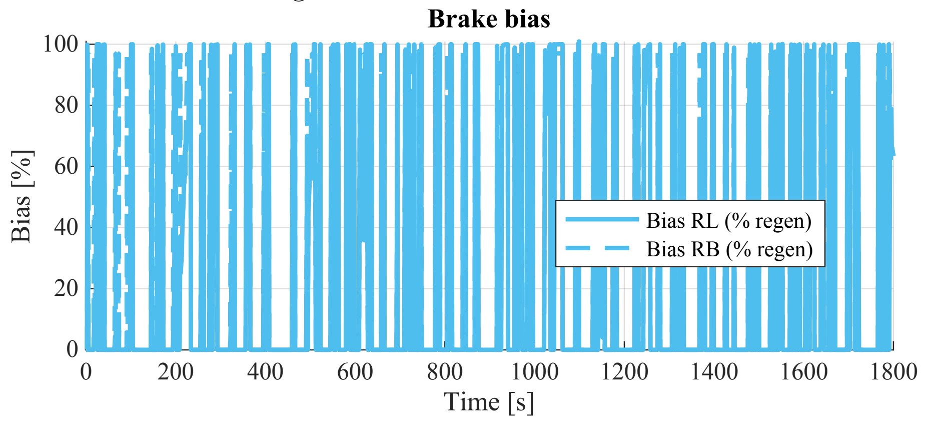


Figure B.110: SoC start = 95%

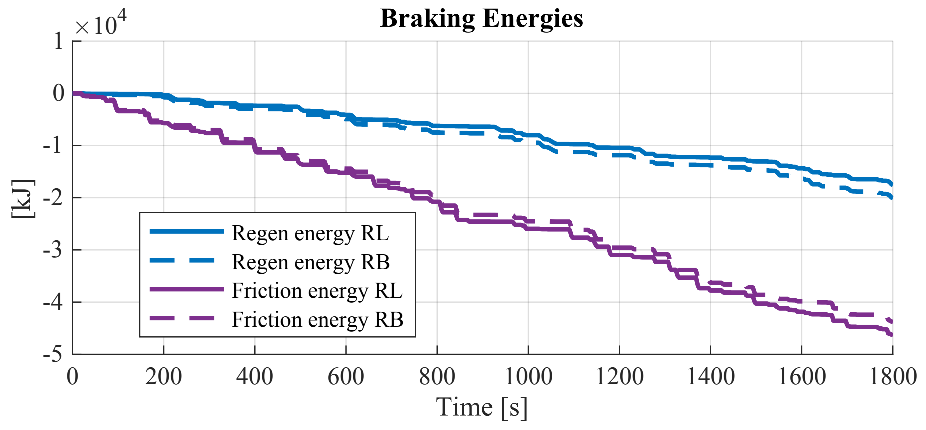


Figure B.111: SoC start = 95%

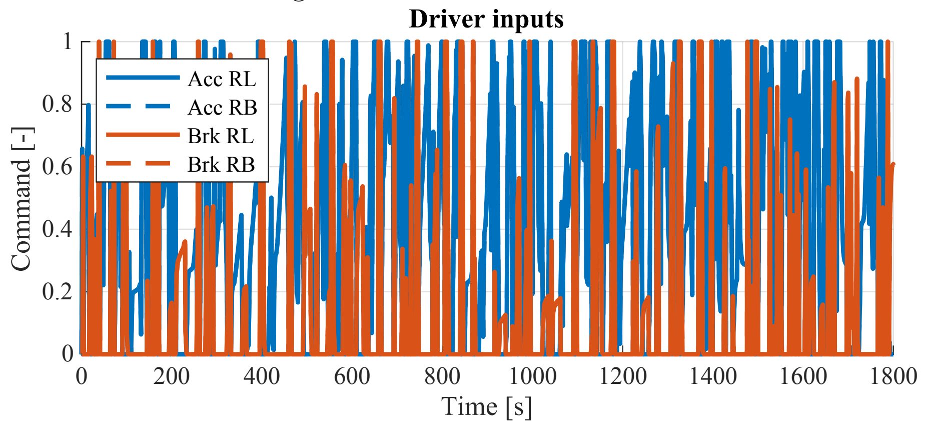


Figure B.112: SoC start = 95%

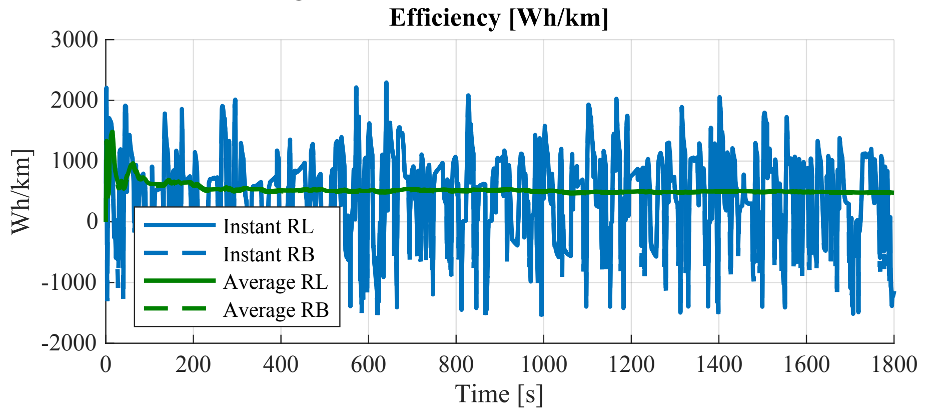


Figure B.113: SoC start = 95%

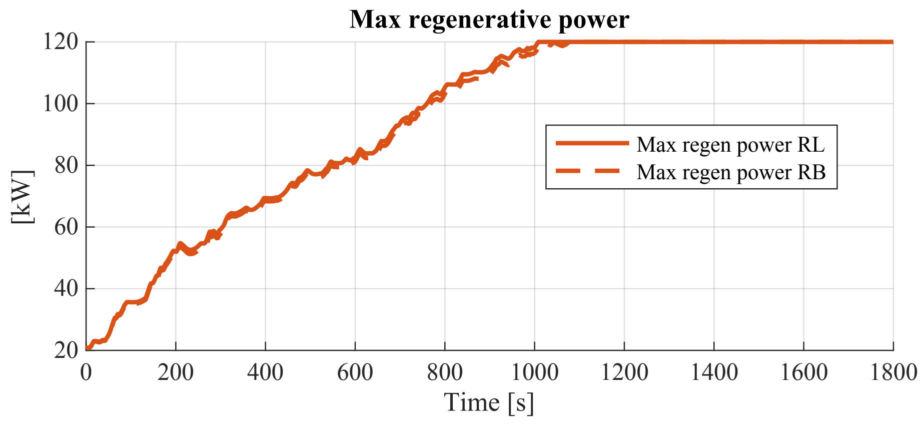


Figure B.114: SoC start = 95%

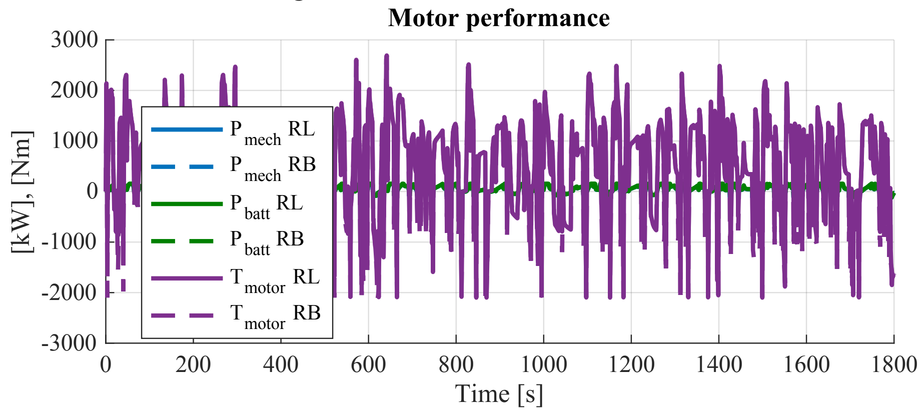


Figure B.115: SoC start = 95%

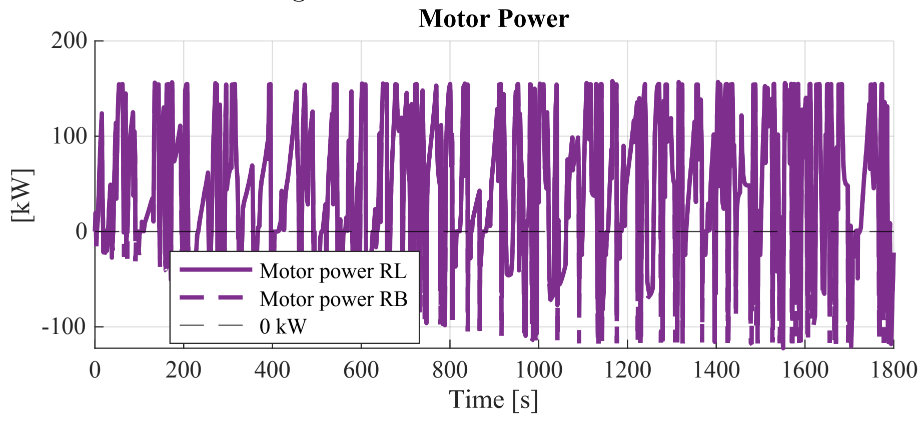


Figure B.116: SoC start = 95%

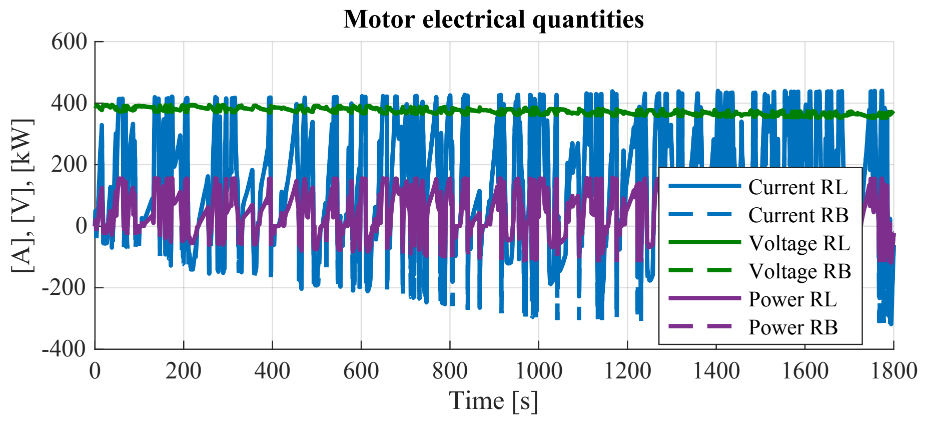


Figure B.117: SoC start = 95%

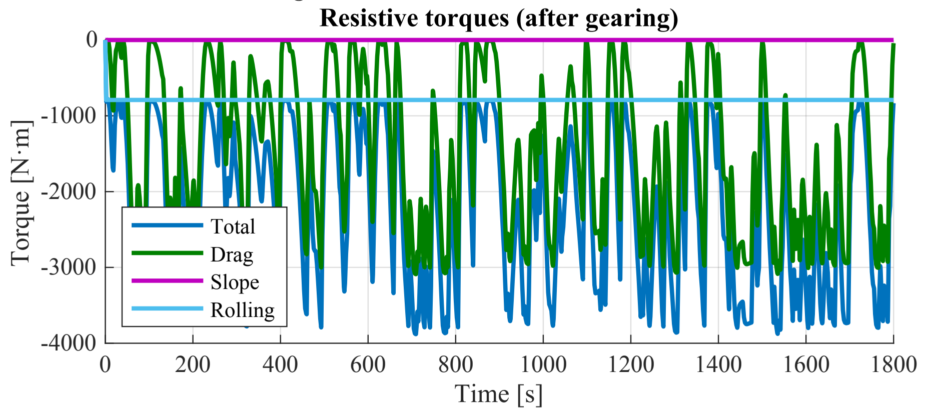


Figure B.118: SoC start = 95%

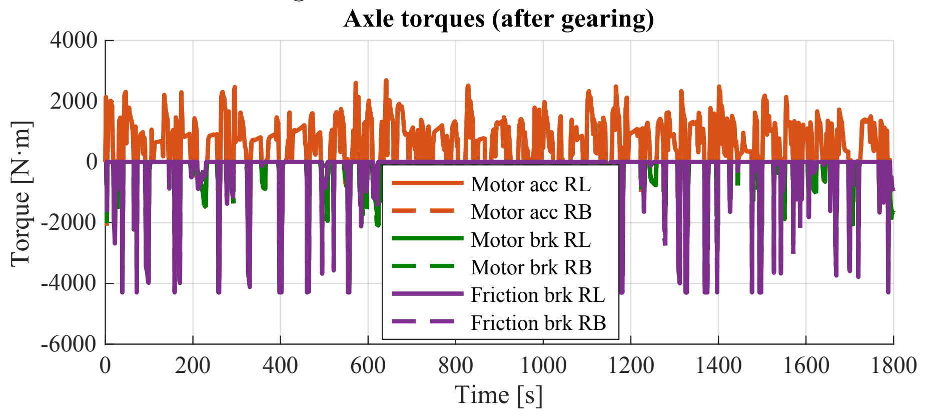


Figure B.119: SoC start = 95%

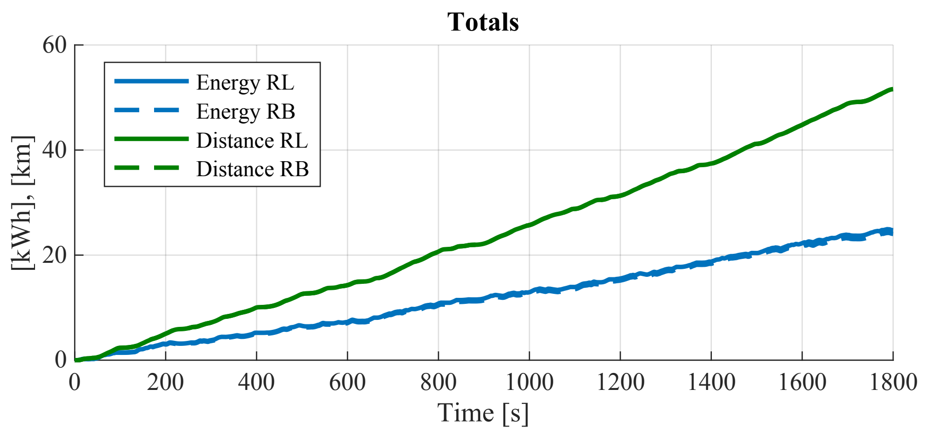


Figure B.120: SoC start = 95%

B.3.3 SOC start at 75%

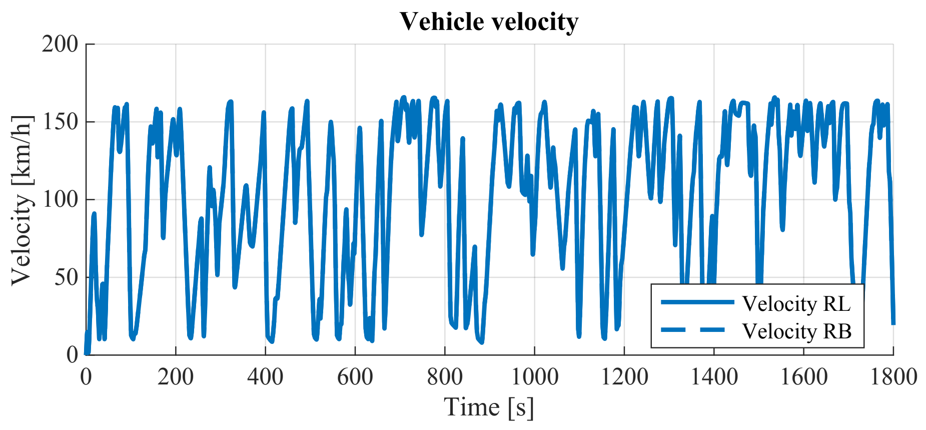


Figure B.121: SoC start = 75%

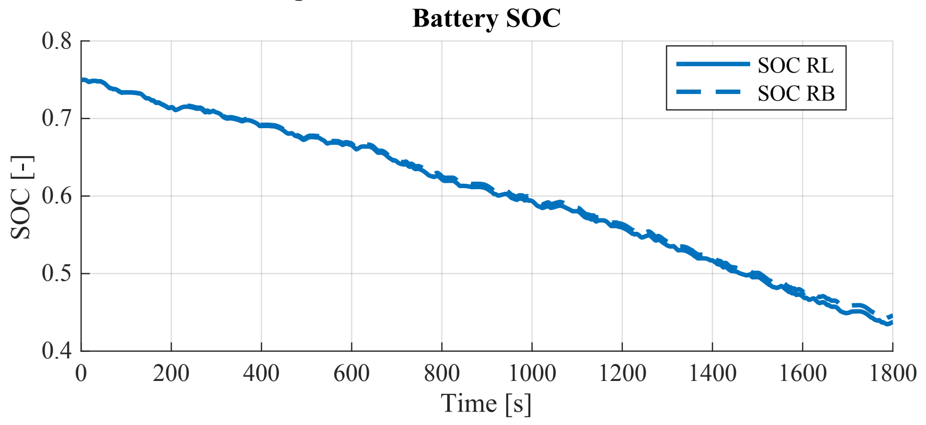


Figure B.122: SoC start = 75%

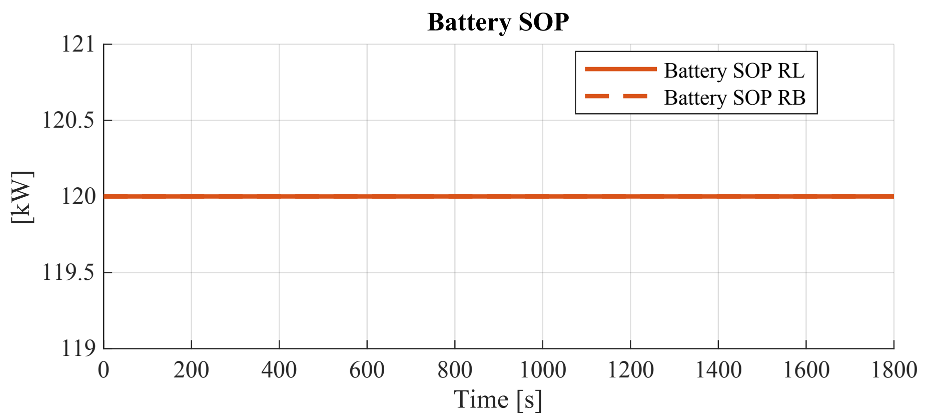


Figure B.123: SoC start = 75%

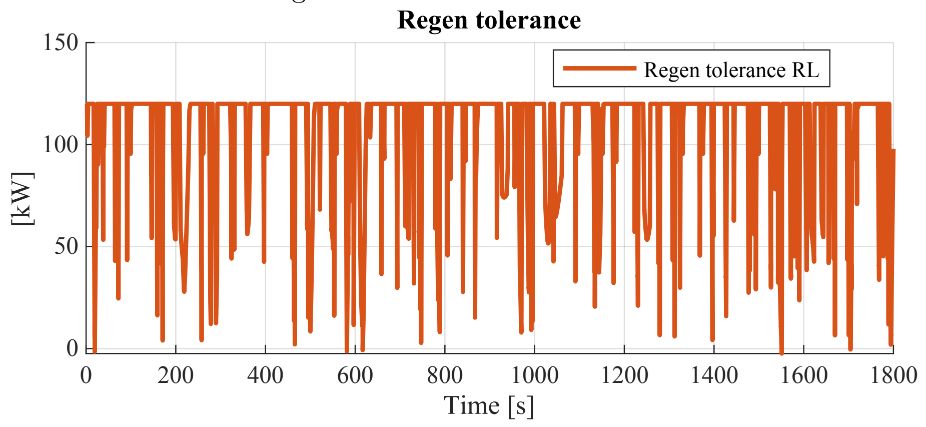


Figure B.124: SoC start = 75%

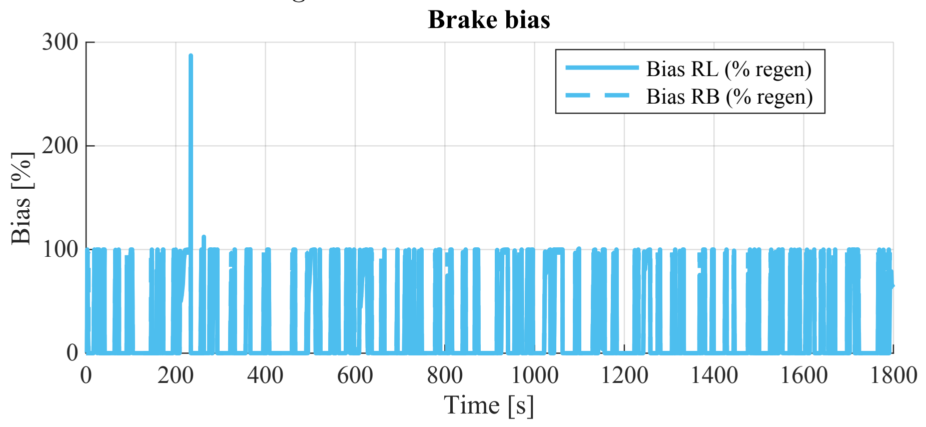


Figure B.125: SoC start = 75%

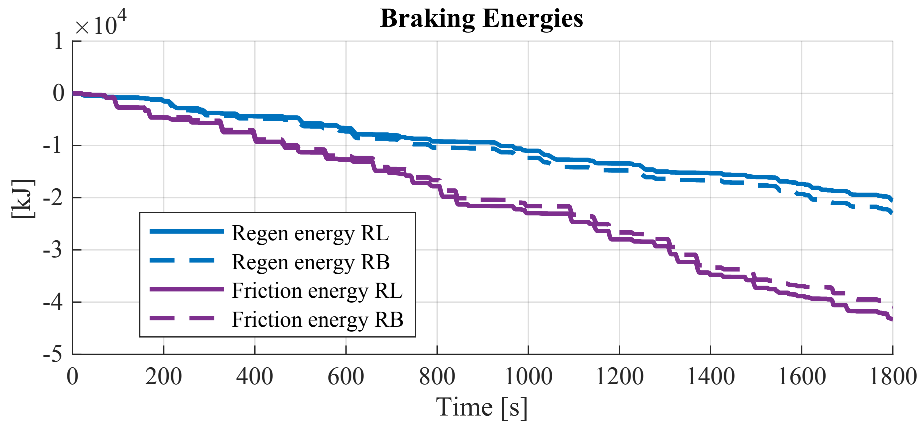


Figure B.126: SoC start = 75%

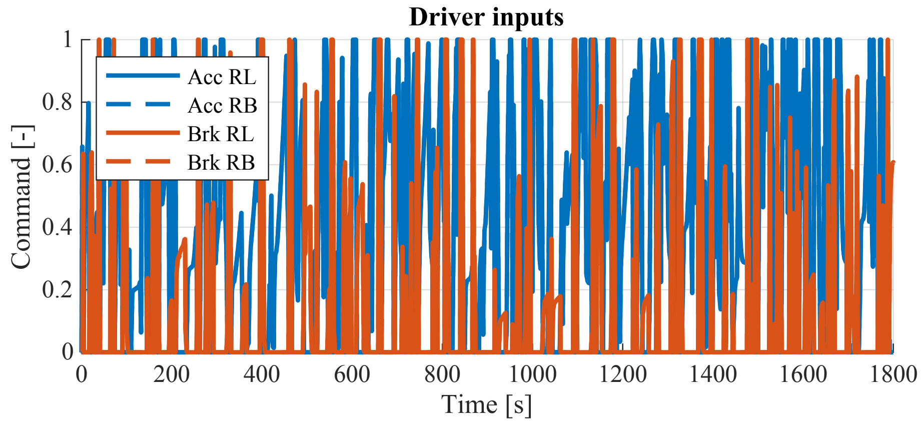


Figure B.127: SoC start = 75%

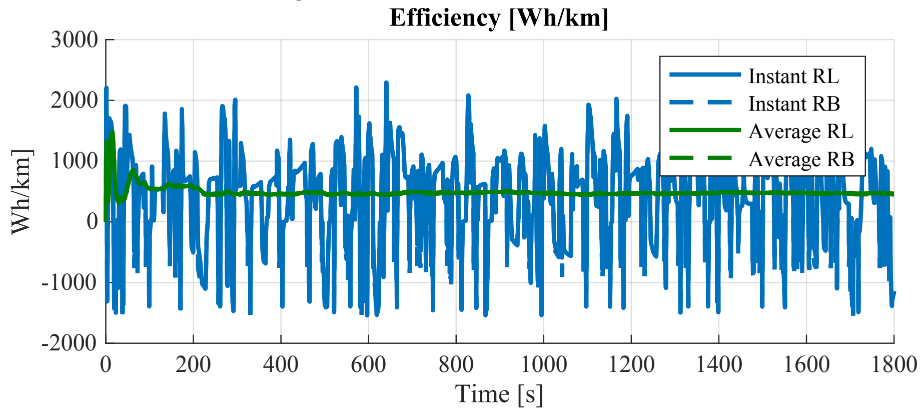


Figure B.128: SoC start = 75%

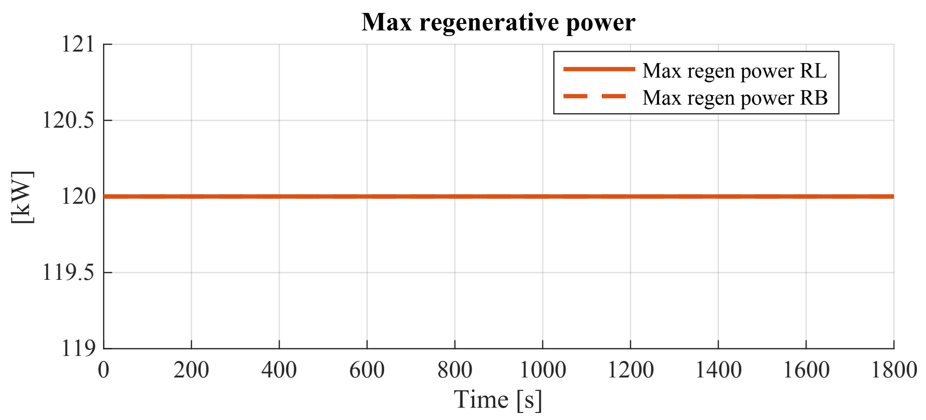


Figure B.129: SoC start = 75%

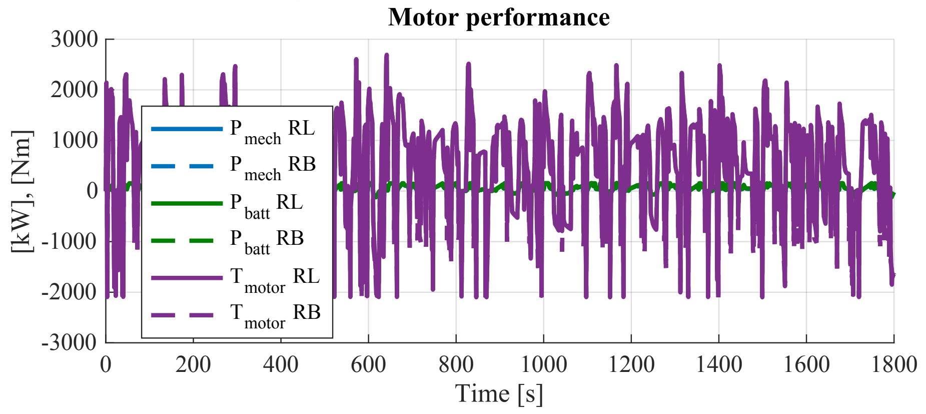


Figure B.130: SoC start = 75%

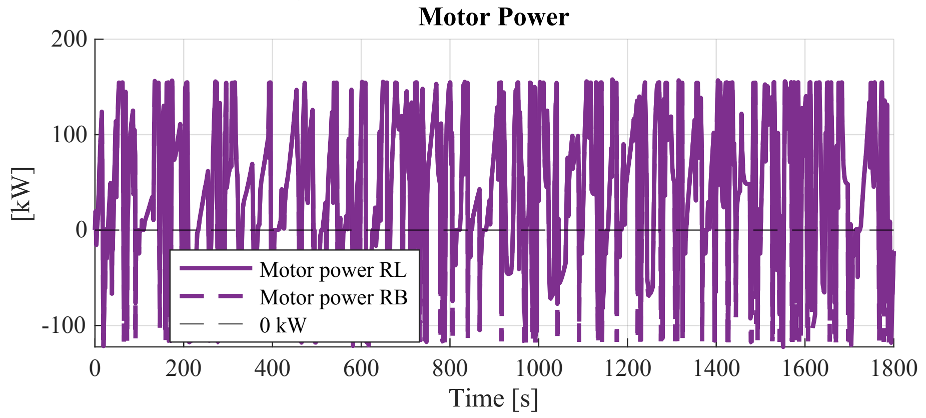


Figure B.131: SoC start = 75%

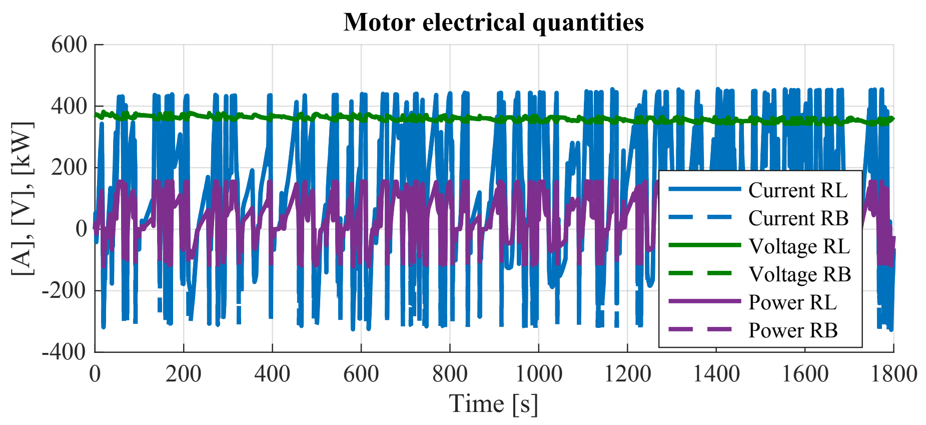


Figure B.132: SoC start = 75%

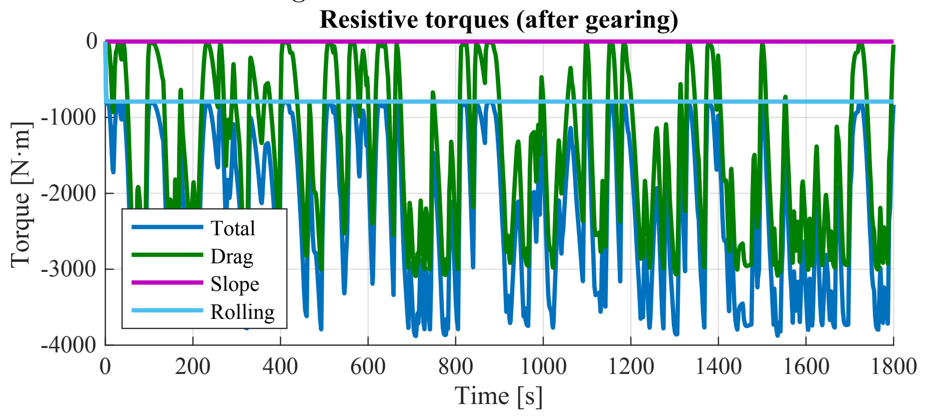


Figure B.133: SoC start = 75%

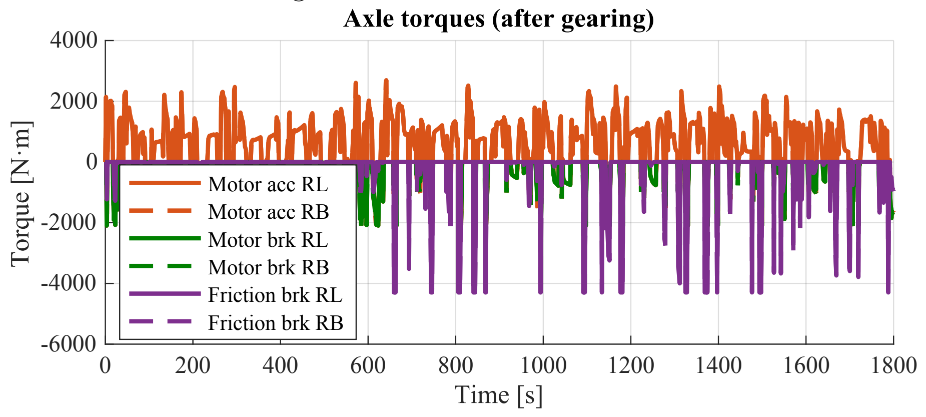


Figure B.134: SoC start = 75%

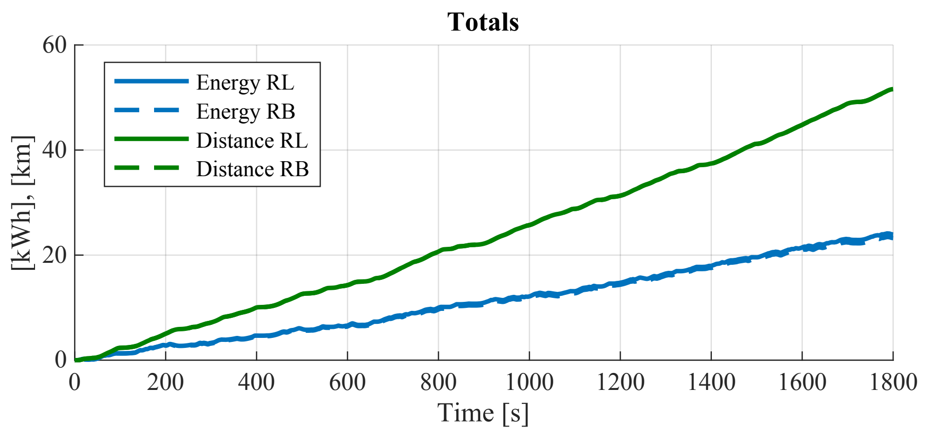


Figure B.135: SoC start = 75%

B.4 Random drivecycle different driver dynamics

B.4.1 SOC start at 100%

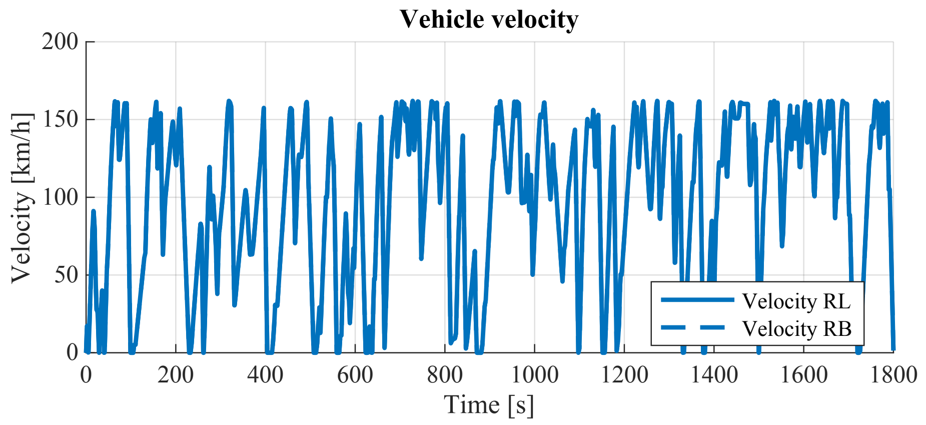


Figure B.136: SoC start = 100%

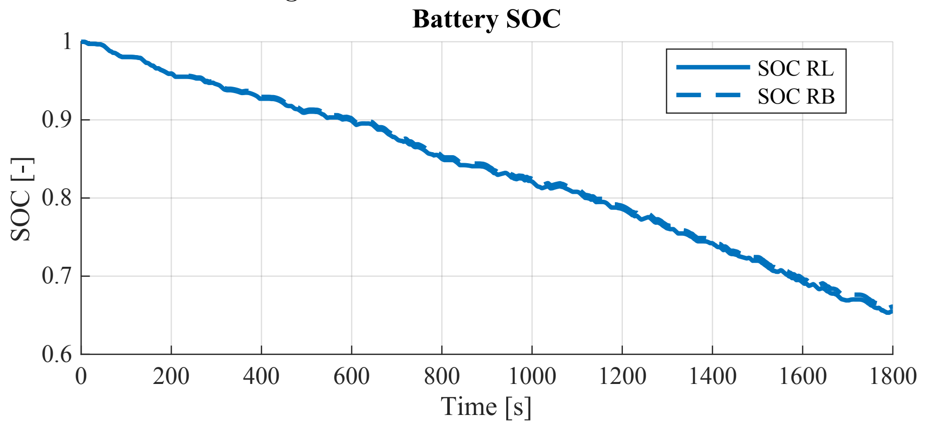


Figure B.137: SoC start = 100%

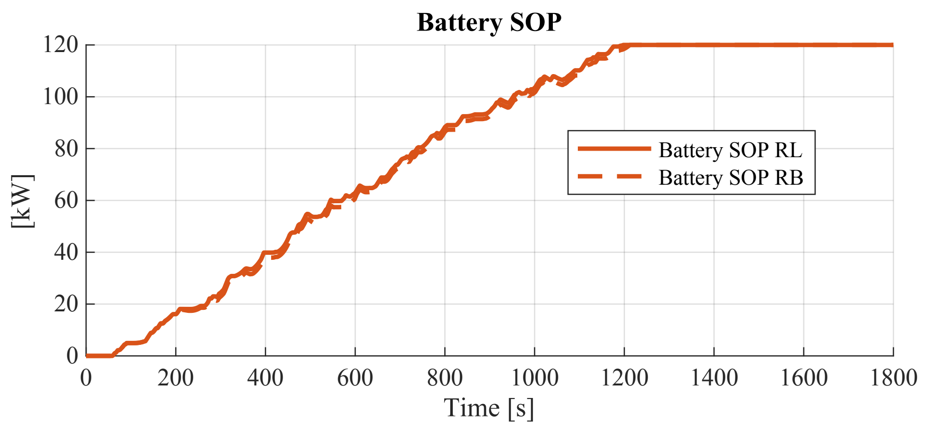


Figure B.138: SoC start = 100%

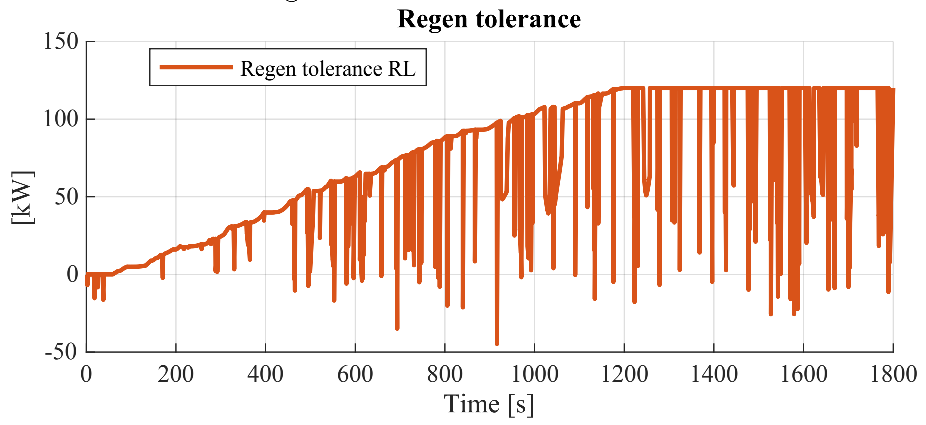


Figure B.139: SoC start = 100%

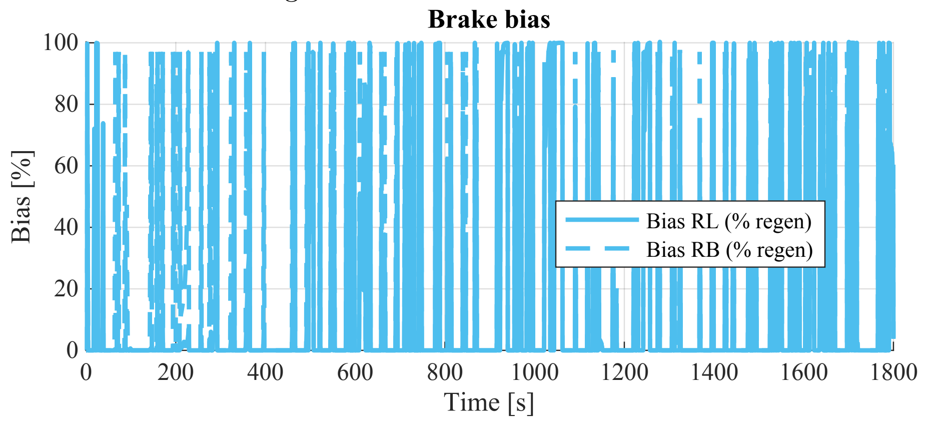


Figure B.140: SoC start = 100%

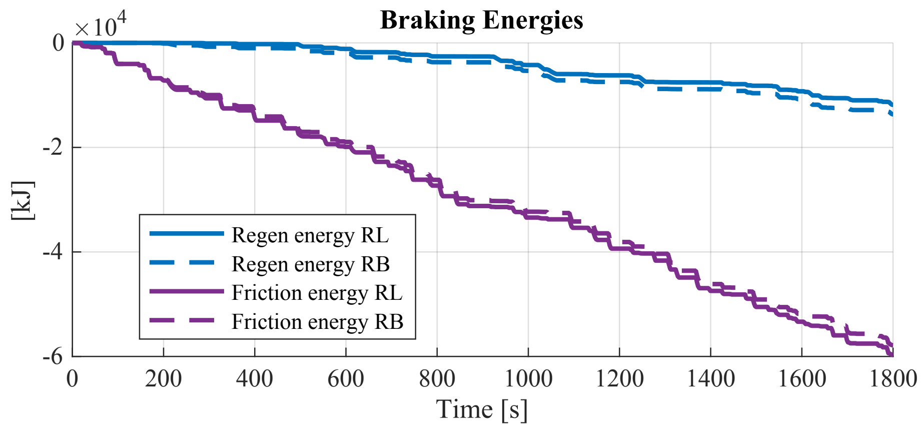


Figure B.141: SoC start = 100%

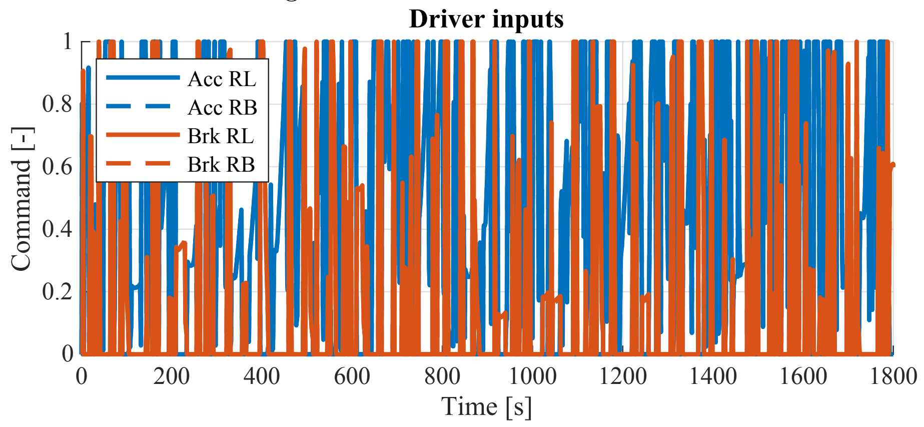


Figure B.142: SoC start = 100%

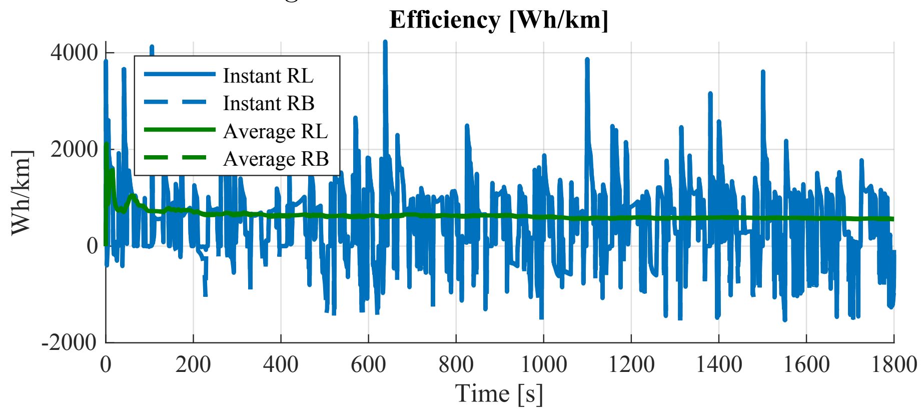


Figure B.143: SoC start = 100%

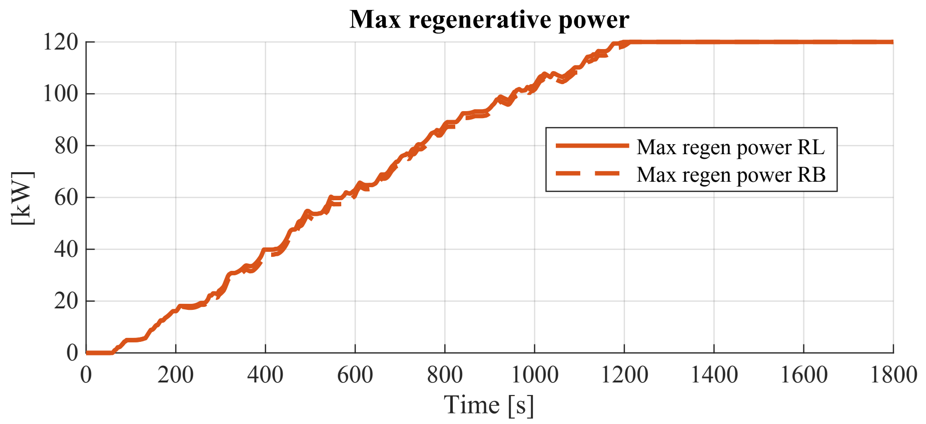


Figure B.144: SoC start = 100%

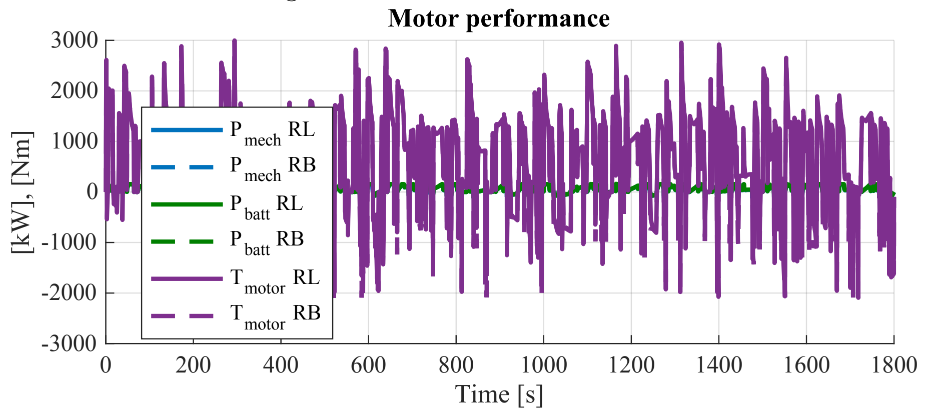


Figure B.145: SoC start = 100%

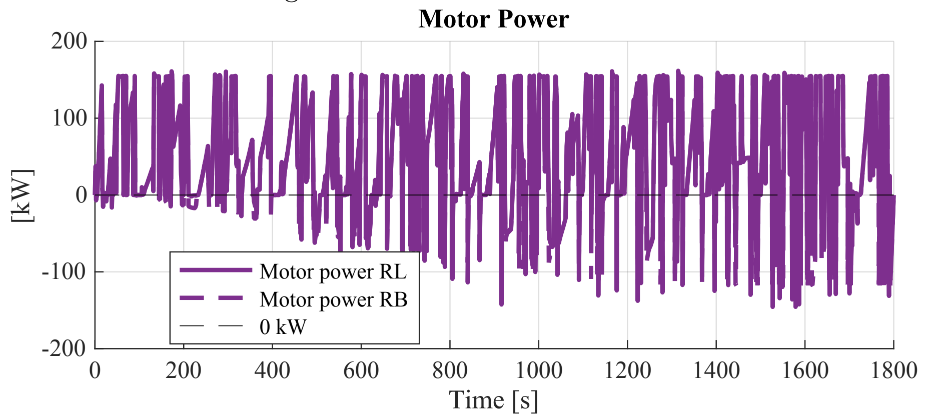


Figure B.146: SoC start = 100%

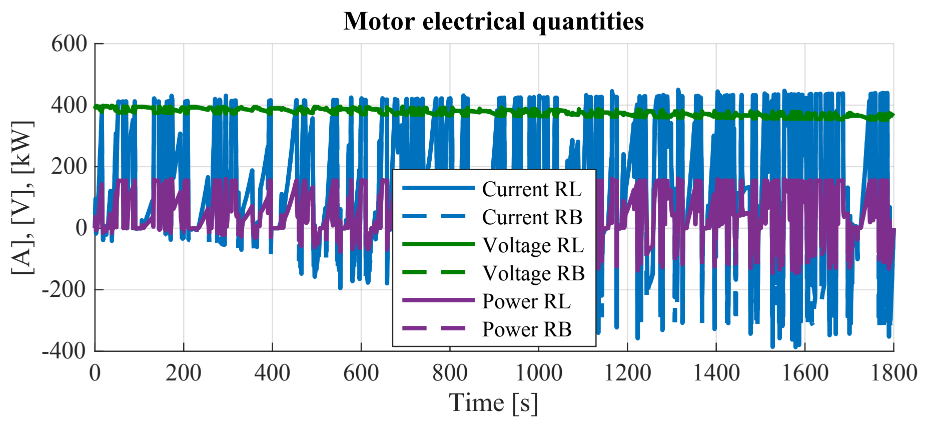


Figure B.147: SoC start = 100%

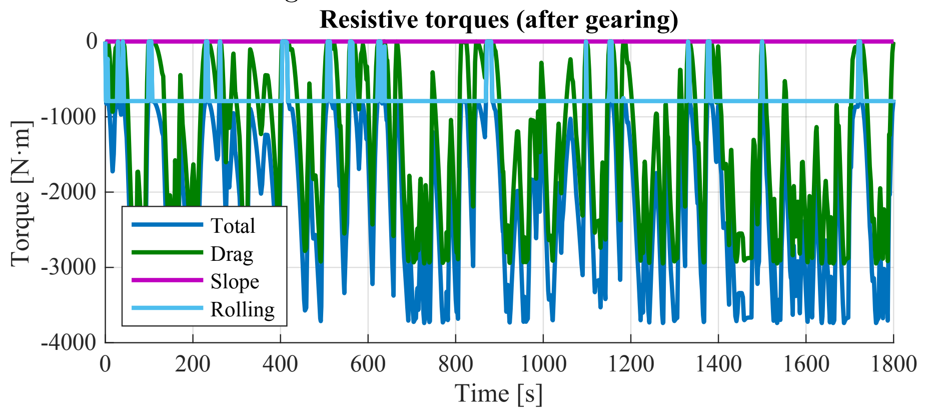


Figure B.148: SoC start = 100%

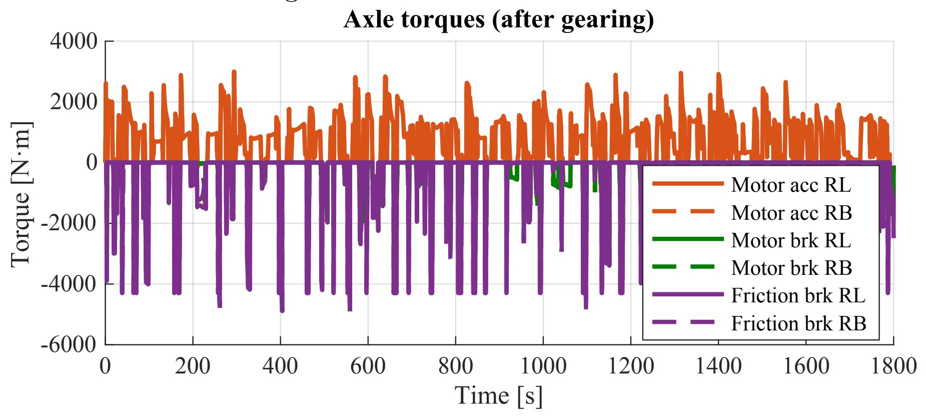


Figure B.149: SoC start = 100%

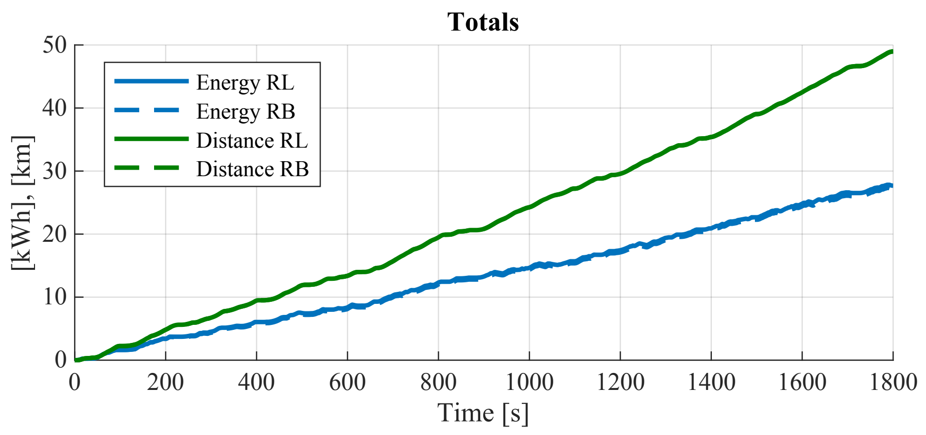


Figure B.150: SoC start = 100%

B.4.2 SOC start at 95%

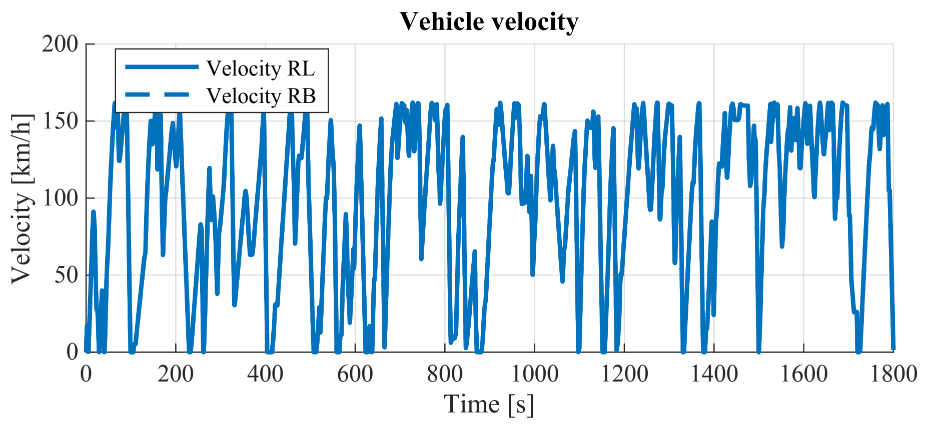


Figure B.151: SoC start = 95%

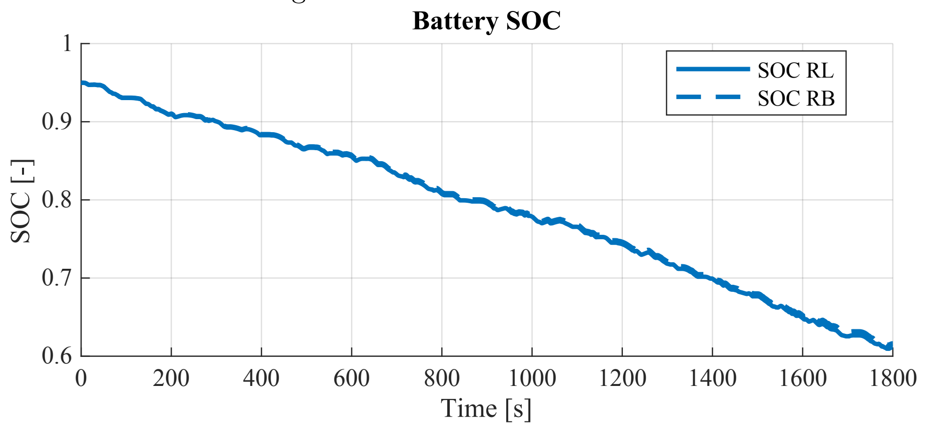


Figure B.152: SoC start = 95%

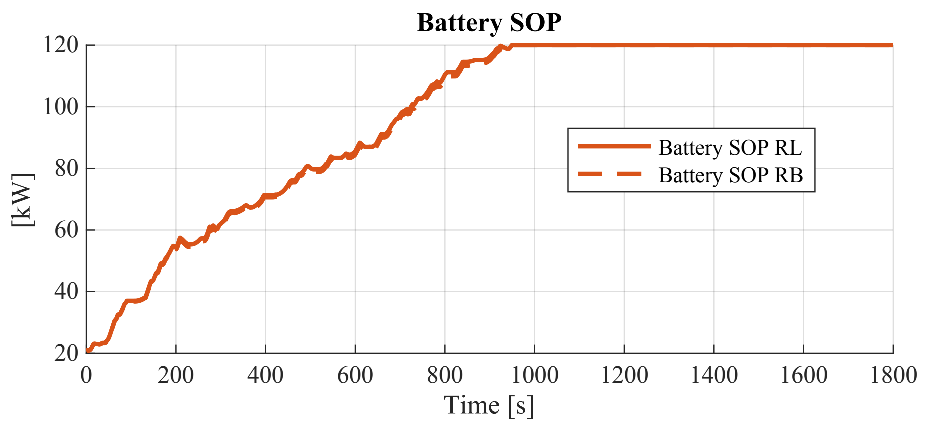


Figure B.153: SoC start = 95%

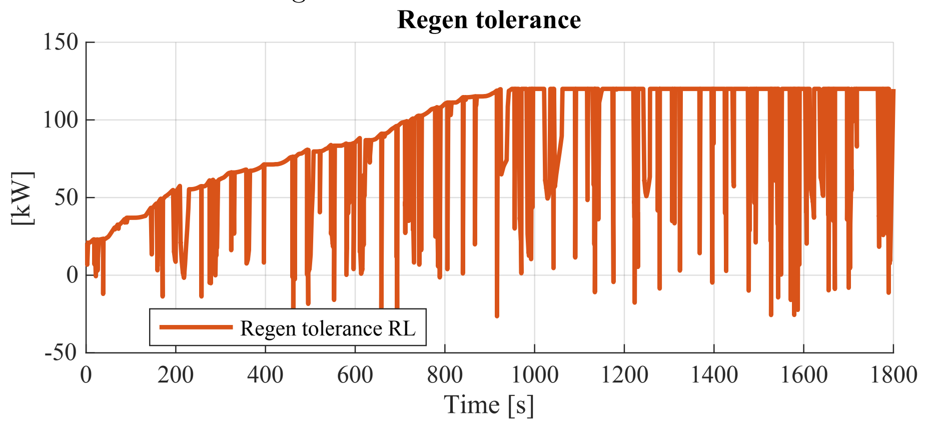


Figure B.154: SoC start = 95%

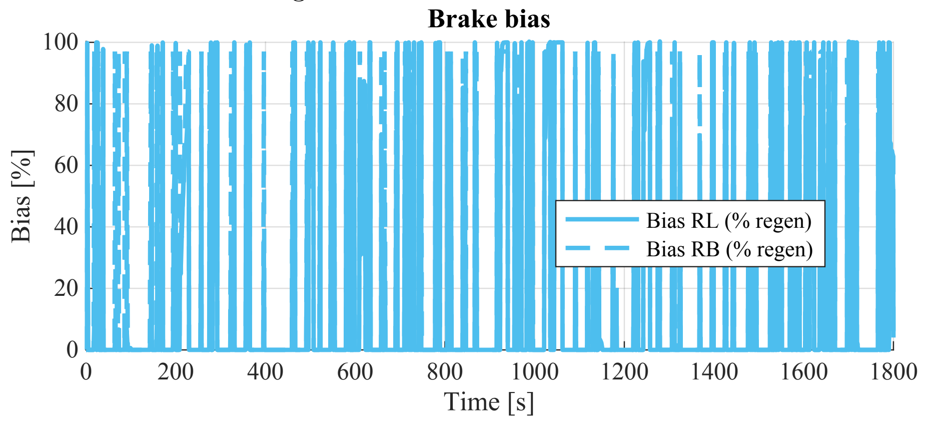


Figure B.155: SoC start = 95%

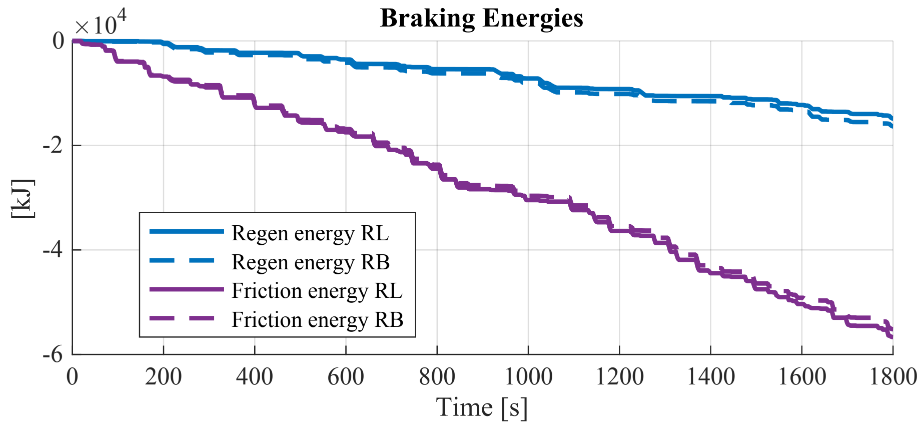


Figure B.156: SoC start = 95%

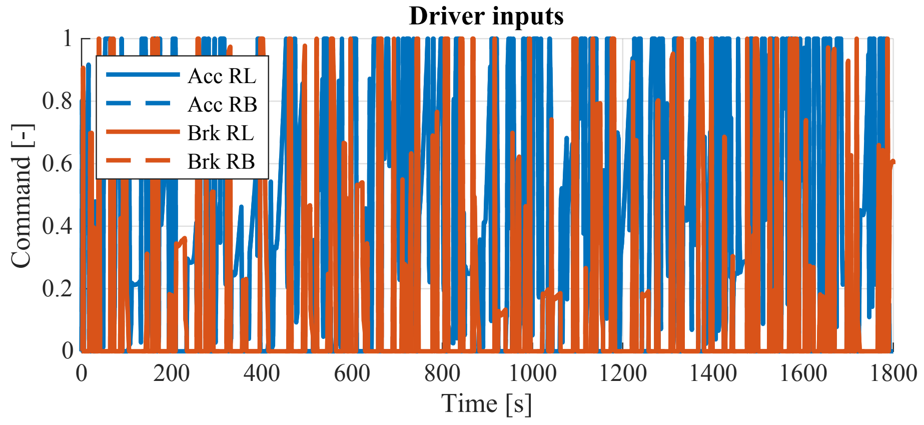


Figure B.157: SoC start = 95%

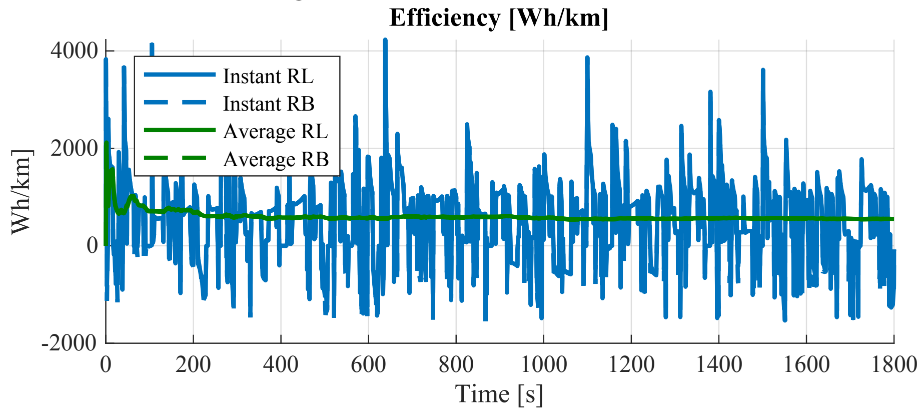


Figure B.158: SoC start = 95%

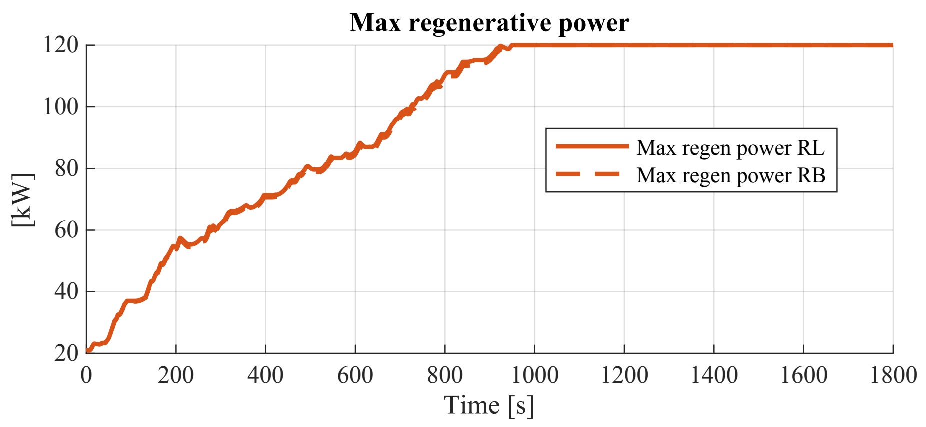


Figure B.159: SoC start = 95%

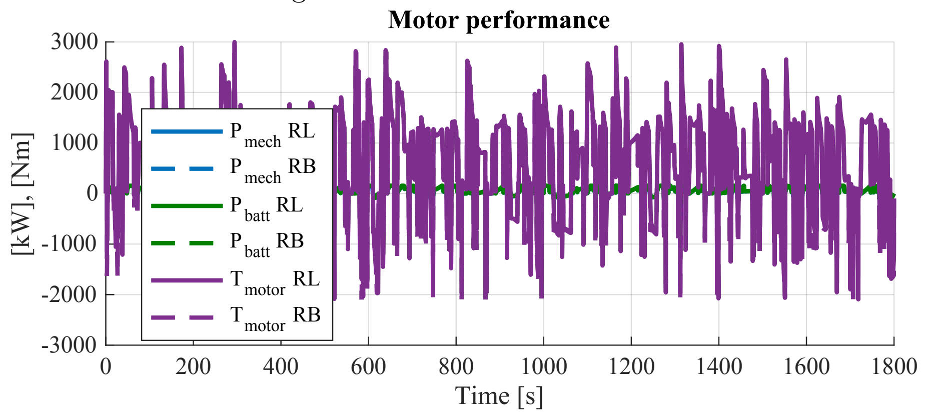


Figure B.160: SoC start = 95%

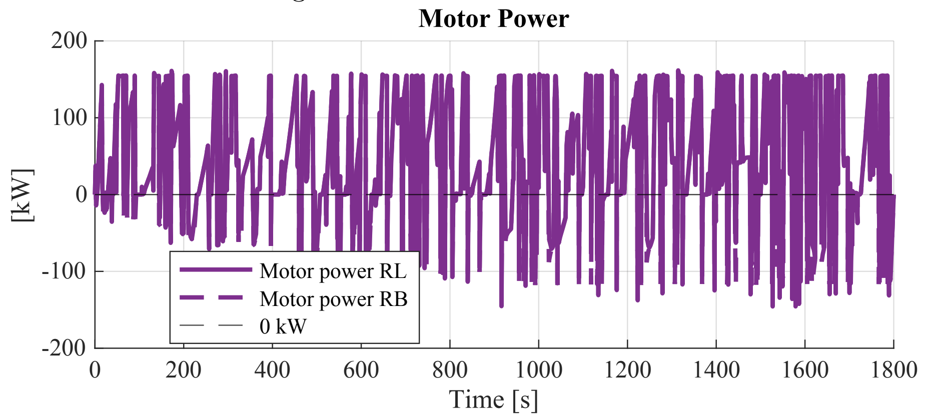


Figure B.161: SoC start = 95%

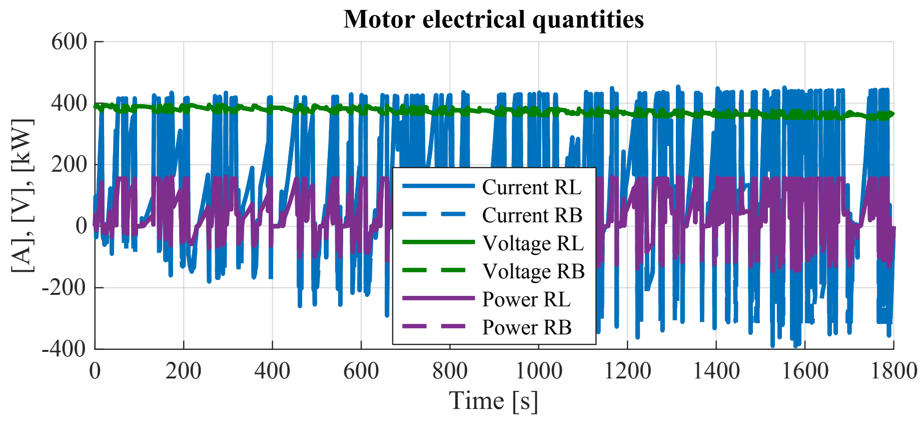


Figure B.162: SoC start = 95%

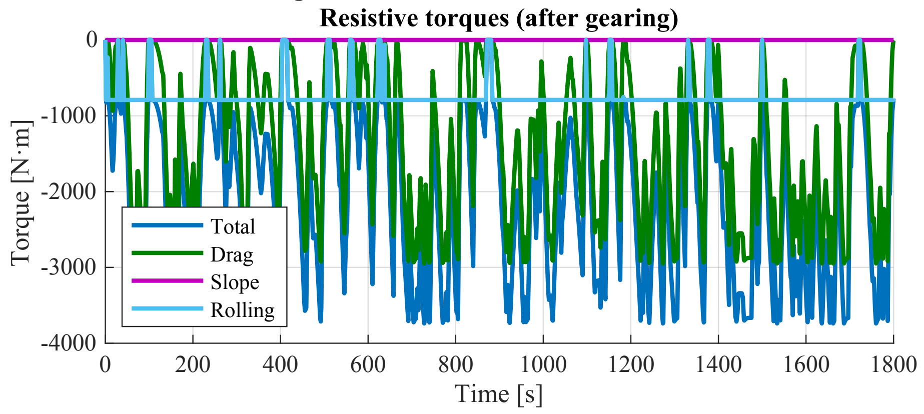


Figure B.163: SoC start = 95%

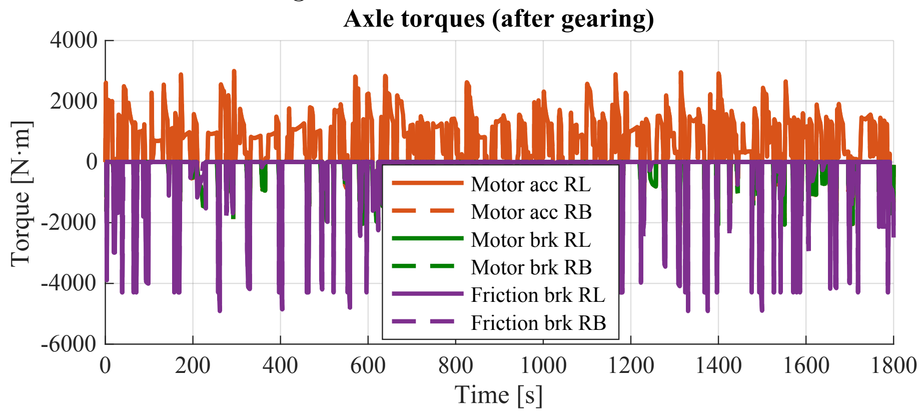


Figure B.164: SoC start = 95%

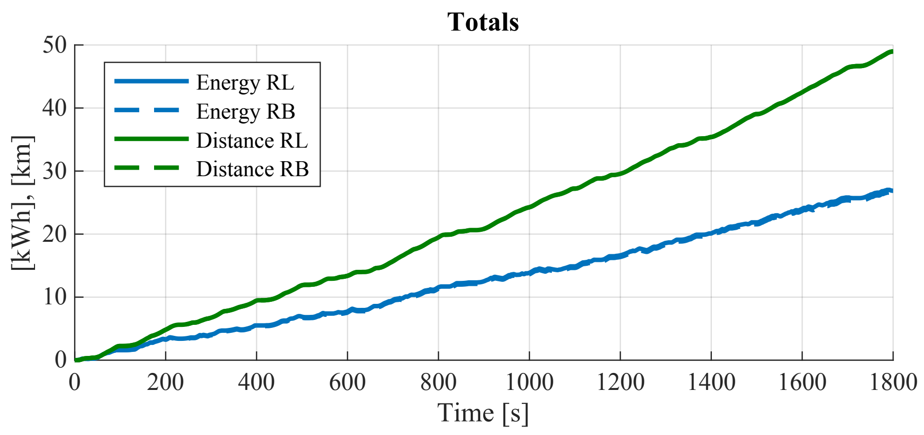


Figure B.165: SoC start = 95%

B.4.3 SOC start at 75%

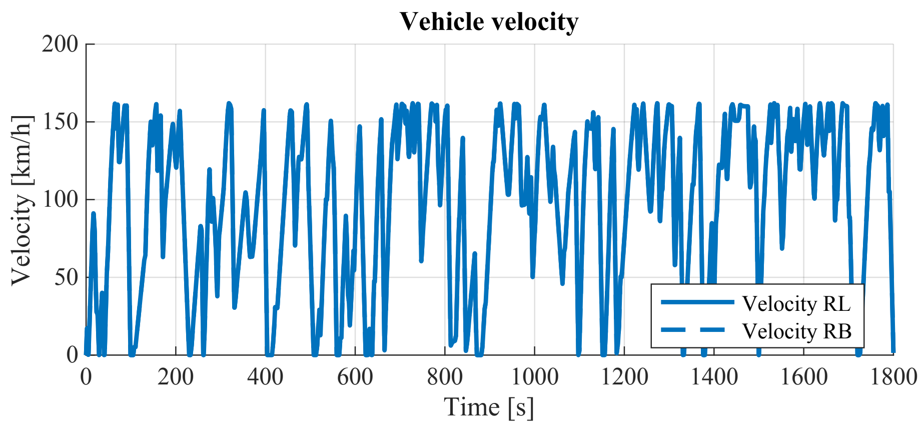


Figure B.166: SoC start = 75%

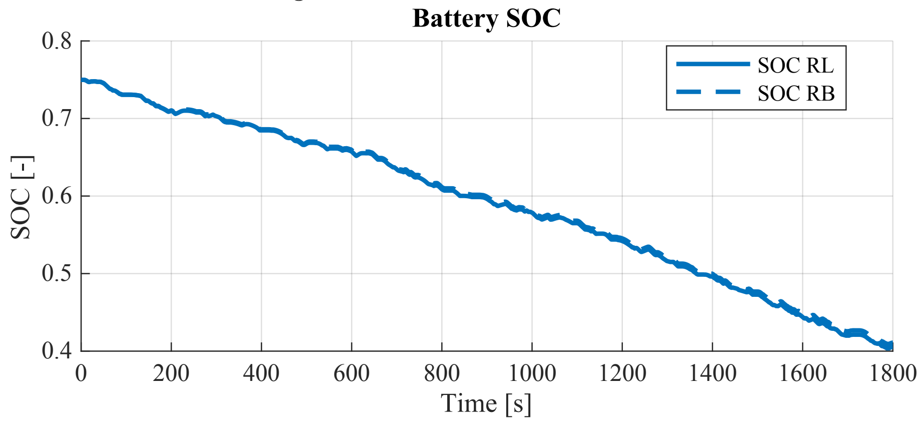


Figure B.167: SoC start = 75%

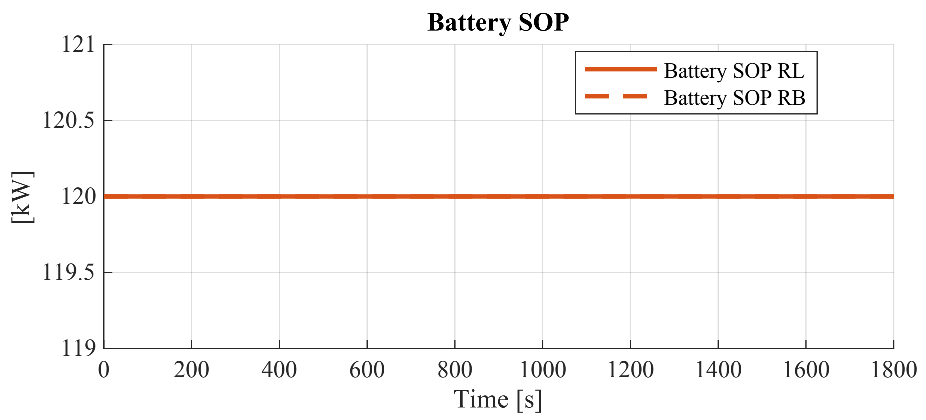


Figure B.168: SoC start = 75%

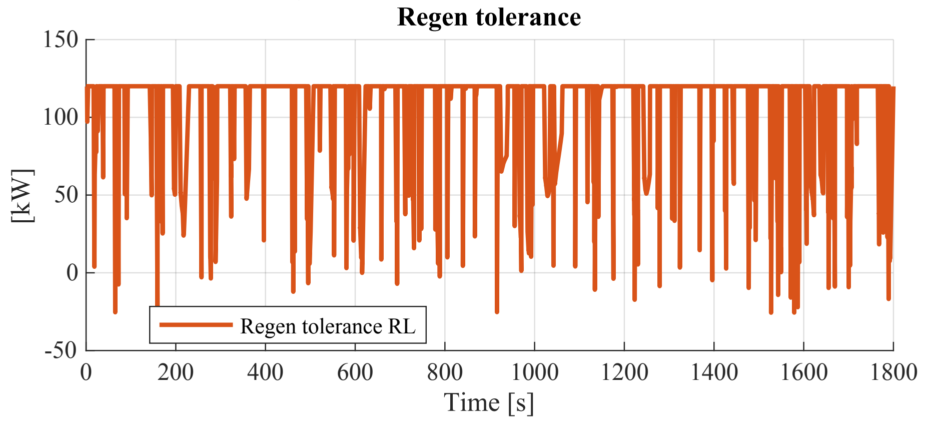


Figure B.169: SoC start = 75%

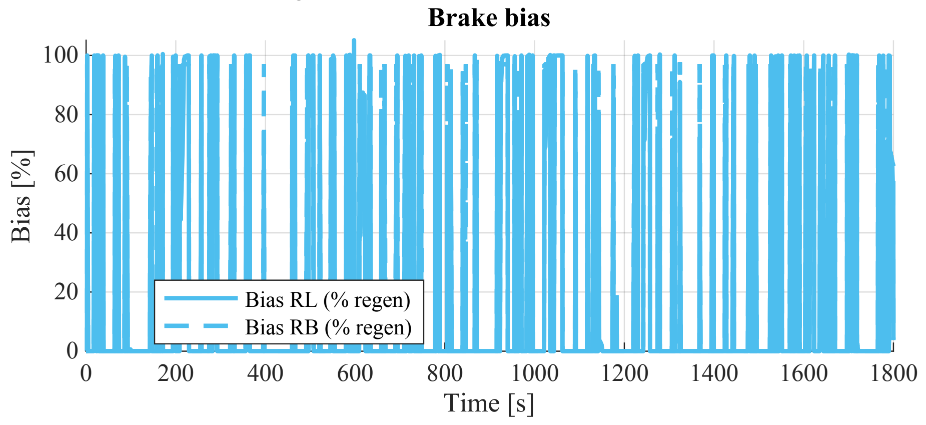


Figure B.170: SoC start = 75%

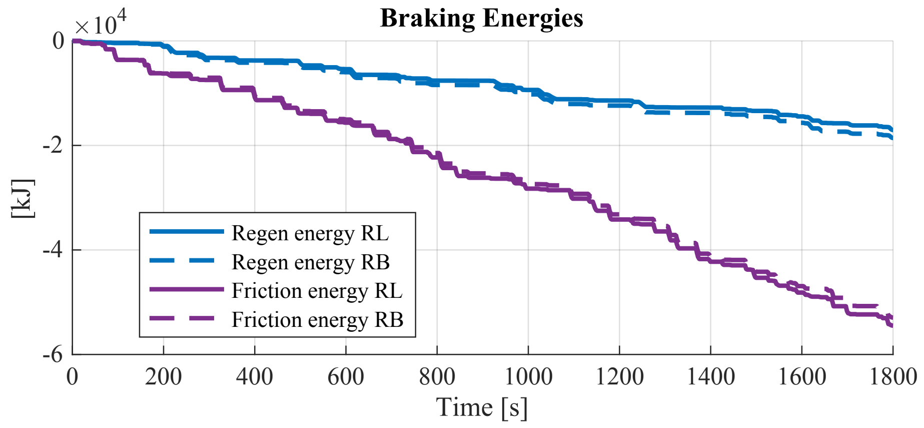


Figure B.171: SoC start = 75%

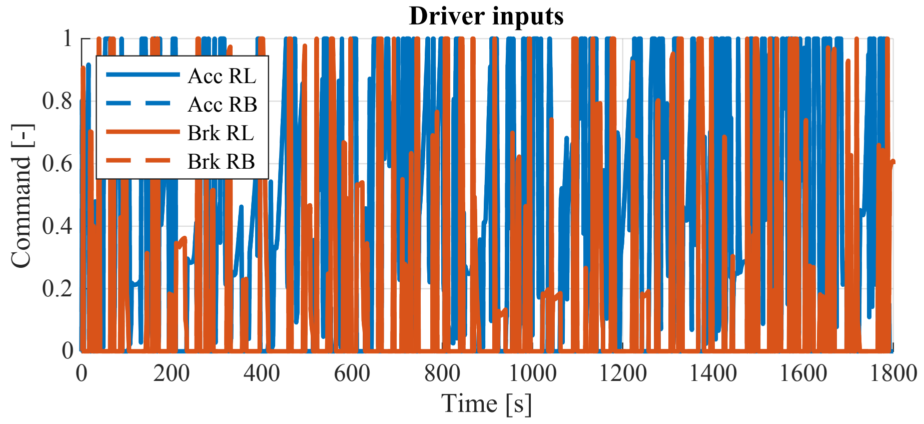


Figure B.172: SoC start = 75%

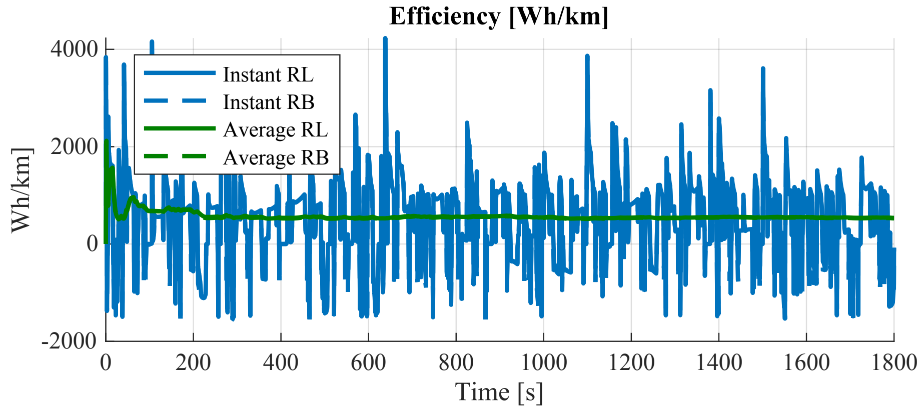


Figure B.173: SoC start = 75%

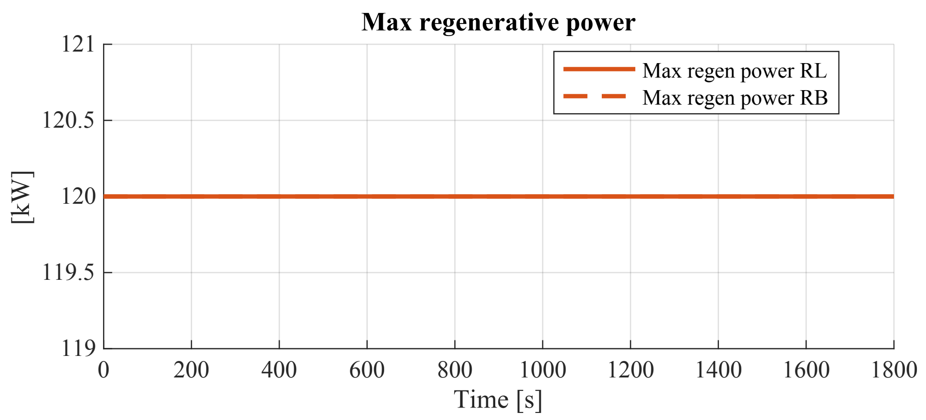


Figure B.174: SoC start = 75%

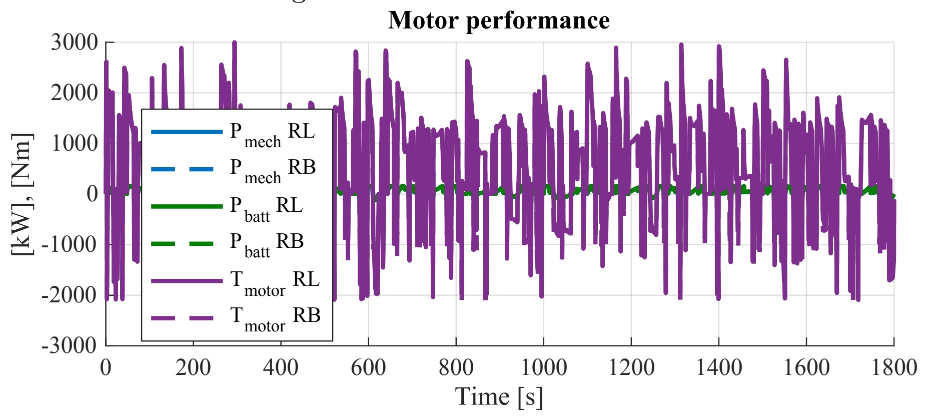


Figure B.175: SoC start = 75%

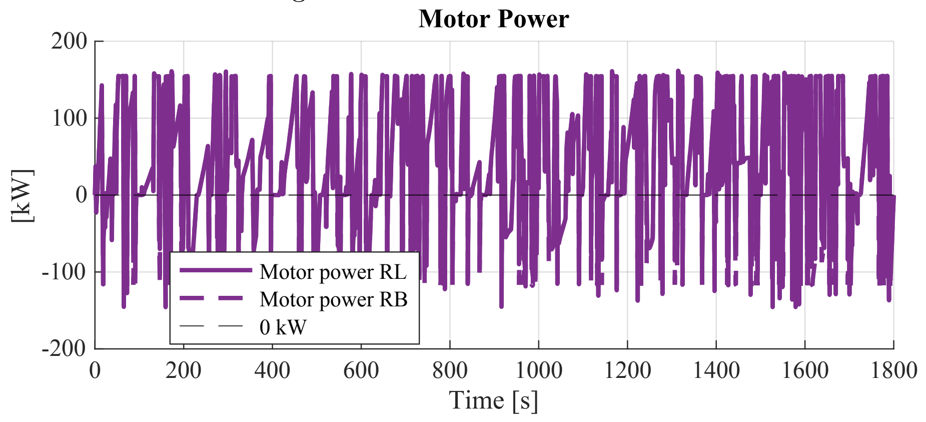


Figure B.176: SoC start = 75%

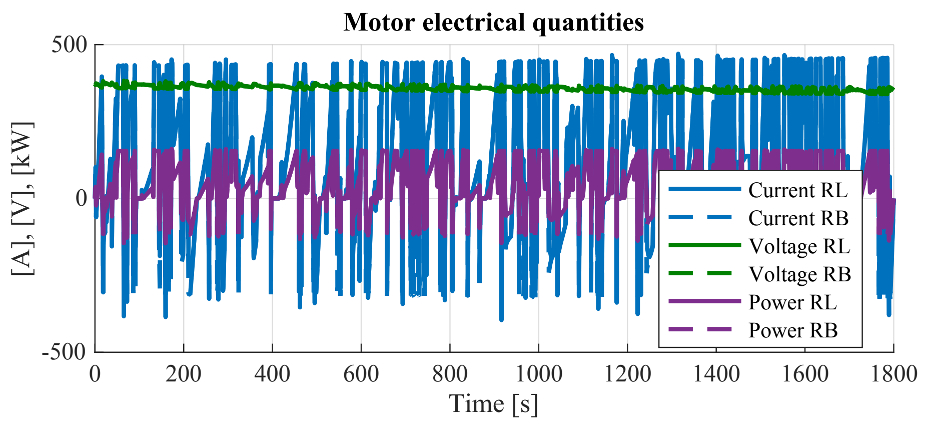


Figure B.177: SoC start = 75%

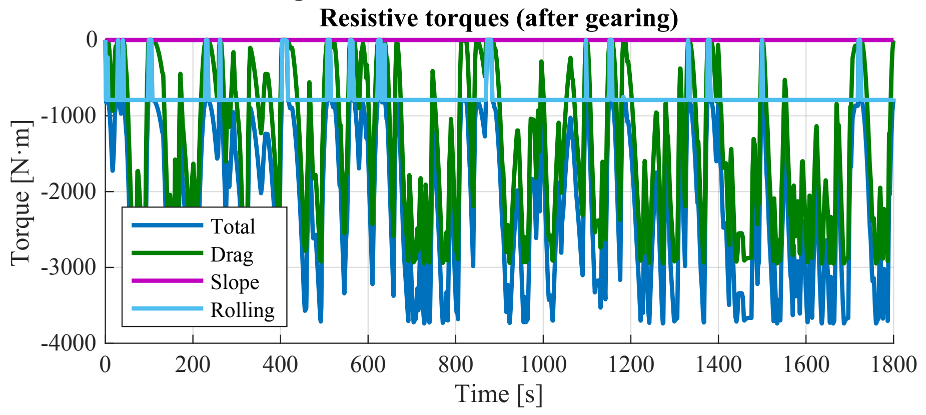


Figure B.178: SoC start = 75%

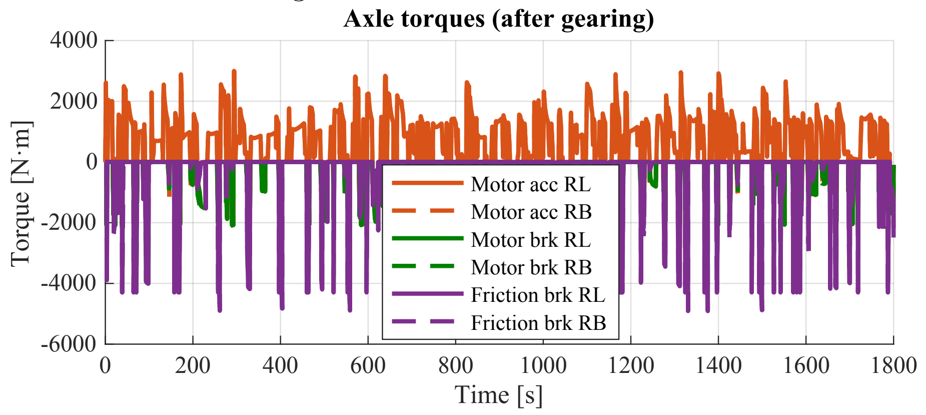


Figure B.179: SoC start = 75%

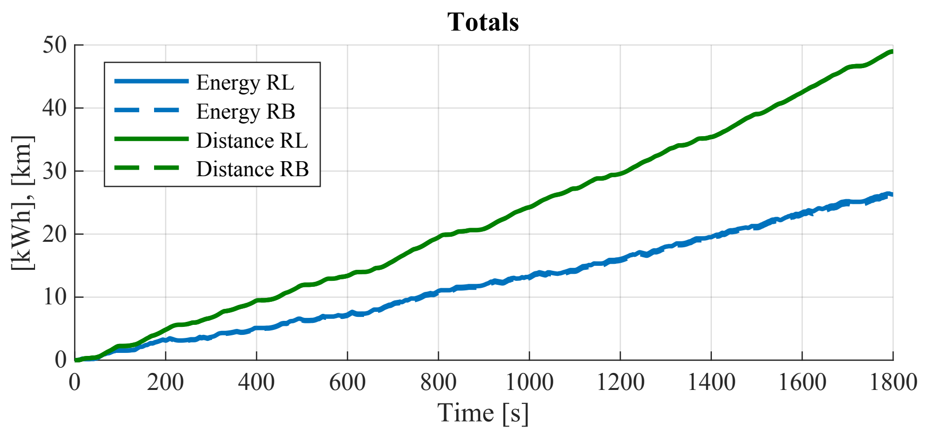


Figure B.180: SoC start = 75%

C | MATLAB code

C.1 Reinforcement Learning Reward and Observation Function

This section presents the MATLAB implementation used to generate the observation vector and reward signal for the reinforcement learning agent. The function is executed at each simulation time step and enforces physical constraints related to battery power capability, braking smoothness, and numerical stability.

```matlab
function [obs, r, isdone, r_energy, r_smooth, x] = makeObsReward( ...
    Brake_cmd, Pee, n_rpm, T_req, SOC, Pcap, a_act, a_prev)
%codegen

epsP      = 1e-6;
n_base    = 4000.0;
T_rated   = 300.0;
Pcap_max  = Pcap;
w_smooth  = 0.01;

Brake_cmd = min(max(Brake_cmd, 0.0), 1.0);
a_prev    = min(max(a_prev,    0.0), 1.0);
a_act     = min(max(a_act,     0.0), 1.0);
SOC       = min(max(SOC,       0.0), 1.0);
Pcap      = max(Pcap, 1e-3);
n_rpm     = max(n_rpm, 0.0);
T_req     = max(T_req, 0.0);
Pee       = max(Pee,   0.0);

obs = zeros(6, 1, 'double');
obs(1,1) = Brake_cmd;
obs(2,1) = min(Pee / max(Pcap_max, epsP), 1.0);
obs(3,1) = min(n_rpm / n_base, 1.0);
obs(4,1) = min(T_req / max(T_rated, epsP), 1.0);
obs(5,1) = SOC;
obs(6,1) = min(Pcap / max(Pcap_max, epsP), 1.0);

x = Pee / (Pcap + epsP);

if x <= 0.8
    r_energy = x;
elseif x <= 1.0
    margin = (x - 0.8) / 0.2;
    r_energy = 0.8 + (margin * 0.15);
```

```
else
    overage = (x - 1.0);
    r_energy = 0 - (2.0 * overage);
end

da = a_act - a_prev;
r_smooth = -0.05 * abs(da);

r = r_energy + r_smooth;

r = max(min(r, 1.0), -2.0);
if ~isfinite(r)
    r = -2.0;
end

isdone = false;

end
```

C.2 Torque Split Function for Regenerative and Friction Braking

This section provides the MATLAB implementation used to split the total braking torque demand into regenerative machine torque and friction brake torque. The allocation is shaped by braking intensity and motor speed, and the regenerative torque is saturated according to both a power-based limit and an absolute machine torque limit.

```
function [Tr, Tf] = fcn(T_break, a, Brk_cmd, w)
% Torque split function for regenerative and friction braking.
%
% This function computes the regenerative braking torque (Tr) commanded to
% the electric machine and the corresponding friction braking torque (Tf)
% such that the total demanded braking torque is satisfied while respecting
% speed-dependent and actuator-related constraints.
%
% Inputs:
%   T_break  : Total requested braking torque [Nm]
%   a        : Agent action (normalized regenerative torque share) [-]
%   Brk_cmd  : Normalized braking command from driver/controller [-]
%   w        : Motor angular speed [rad/s]
%
% Outputs:
%   Tr       : Regenerative braking torque command [Nm]
%   Tf       : Friction braking torque command [Nm]

rpm_off    = 100;
rpm_full   = 600;
w_off      = rpm_off  * 2*pi/60;
w_full     = rpm_full * 2*pi/60;

if Brk_cmd <= 0.5
    share = 1.0;
elseif Brk_cmd <= 0.8
```

```matlab
    share = (0.8 - Brk_cmd) / 0.3;
else
    share = 0.0;
end

if w <= w_off
    fade = 0.0;
elseif w >= w_full
    fade = 1.0;
else
    fade = (w - w_off) / (w_full - w_off);
end

Tr_des = T_break * a * share * fade;

Pmax   = 150e3;
w_safe = max(abs(w), 1);
T_pmax = Pmax / w_safe;

T_tmax = 310;

T_cap  = min(T_pmax, T_tmax);

Tr = max(min(Tr_des,  T_cap), -T_cap);

Tf = T_break - Tr;

end
```

C.3 Rule-Based Torque Split for Regenerative Braking

This section presents the rule-based torque allocation strategy used as a baseline regenerative braking controller. The total braking torque demand is deterministically split between regenerative braking torque and friction braking torque based on braking command intensity, motor speed, and fixed actuator constraints. No learning or adaptation is involved in this controller.

```matlab
function [Tr, Tf] = fcn(T_break, a, Brk_cmd, w)

rpm_off    = 100;                   % regenerative braking disabled below
    this speed
rpm_full   = 600;                   % full regenerative capability above
    this speed
w_off      = rpm_off  * 2*pi/60;    % [rad/s]
w_full     = rpm_full * 2*pi/60;    % [rad/s]

% Base regenerative share determined by braking command intensity
if Brk_cmd <= 0.5                   % light braking: prioritize
    regenerative braking
    share = 1.0;
elseif Brk_cmd <= 0.8               % medium braking: linear blending
    region
    share = (0.8 - Brk_cmd) / 0.3;   % linear reduction from regen to
        friction
else                                 % hard braking: friction braking only
    share = 0.0;
end

% Speed-dependent fade-in of regenerative braking
% Ensures zero regenerative torque near standstill
if w <= w_off
    fade = 0.0;
elseif w >= w_full
    fade = 1.0;
else
    fade = (w - w_off) / (w_full - w_off);
end

% Desired regenerative torque prior to physical constraints
Tr_des = T_break * a * share * fade;

% Power-based torque limitation derived from P = T * w
Pmax   = 150e3;          % maximum allowable regenerative power [W]
w_safe = max(abs(w), 1); % numerical safeguard at low speed
T_pmax = Pmax / w_safe;  % torque limit imposed by power constraint [Nm]

% Absolute motor torque limitation
T_tmax = 310;            % maximum motor torque magnitude [Nm]

% Combined regenerative torque cap
T_cap  = min(T_pmax, T_tmax);

% Saturation of regenerative torque command
Tr = max(min(Tr_des, T_cap), -T_cap);

% Remaining braking torque supplied by friction brakes
Tf = T_break - Tr;

end
```

C.4 Drive Torque Command and Torque–Speed Envelope

This section presents the function used to translate the normalized accelerator command into a motor drive torque request. The commanded torque is limited by a speed-dependent torque–speed envelope that accounts for both the maximum torque capability of the electric machine and the maximum allowable power at higher rotational speeds. A smooth taper is applied near the no-load speed to avoid discontinuities in the torque command.

```
function [T_drive_cmd, Tmax] = drive_torque_cmd(acc_cmd, w_mot)
% Drive torque request and torque-speed envelope
%
% Inputs:
%   acc_cmd : Normalized accelerator command [-]
%   w_mot   : Motor mechanical speed [rad/s]
%
```

```
% Outputs:
%   T_drive_cmd : Requested motoring torque [Nm]
%   Tmax        : Maximum available torque at current speed [Nm]

acc = min(max(acc_cmd, 0), 1);

Tmax = torque_envelope_local(w_mot);

T_drive_cmd = acc * Tmax;

end


function Tmax = torque_envelope_local(w_mot)
% Torque-speed envelope with power limitation and smooth high-speed taper

peakT   = 310;             % Maximum torque in flat region [Nm]
peakP   = 150e3;           % Maximum electrical power [W]
w_no    = 14000*pi/30;     % No-load speed [rad/s]
w_taper = w_no * 0.95;     % Start of torque taper region [rad/s]

w = max(abs(w_mot), 1e-3);

T_pow = peakP / w;

Tmax_unclamped = min(peakT, T_pow);

if w < w_taper
    taper = 1.0;
elseif w < w_no
    taper = 0.5 * (1 + cos(pi * (w - w_taper) / (w_no - w_taper)));
else
    taper = 0.0;
end

Tmax = Tmax_unclamped * taper;

Tmax = max(0, Tmax);

end
```

C.5 Brake Command to Torque Mapping

This section presents the function used to convert the normalized brake command into a mechanical braking torque request. The mapping is deterministic and applies a fixed pedal-to-torque scaling, ensuring a bounded and physically interpretable braking demand that can be combined with regenerative braking logic and friction brake actuation.

```
function Tbr_cmd = acc_to_torque(B, Gear_ratio)
% Brake command to torque conversion
%
% Inputs:
%   B          : Normalized brake command [-]
%   Gear_ratio : Drivetrain gear ratio
```

```matlab
%
% Output:
%   Tbr_cmd    : Requested braking torque [Nm]

T_max = 4200 / 10.5;    % Pedal-to-torque scaling constant [Nm]

Tbr_cmd = max(0, min(1, B)) * T_max;

end
```

C.6 Road-Load Torque Model at the Wheel

This section presents the road-load model used to compute the total resisting torque acting at the wheel shaft. The formulation captures the main longitudinal resistive effects acting on the vehicle, including rolling resistance, road grade, aerodynamic drag, and viscous losses. All contributions are expressed as torques opposing the direction of motion and are smoothly defined near zero speed to ensure numerical robustness.

```matlab
function [T_rr, T_g, T_d, T_v, T_road] = road_load( ...
    omega, grade_pct, Vehicle_mass, Vehicle_Wheel_Radius, ...
    Cd, A, bvis, Crr, rho)

% Road-load torque at the wheel shaft
%
% Inputs:
%   omega      : Wheel angular speed [rad/s]
%   grade_pct  : Road grade [%]
%   Vehicle_mass         : Vehicle mass [kg]
%   Vehicle_Wheel_Radius : Effective wheel radius [m]
%   Cd         : Aerodynamic drag coefficient [-]
%   A          : Frontal area [m^2]
%   bvis       : Viscous friction coefficient
%   Crr        : Rolling resistance coefficient [-]
%   rho        : Air density [kg/m^3]
%
% Outputs:
%   T_rr   : Rolling resistance torque [Nm]
%   T_g    : Grade-induced torque [Nm]
%   T_d    : Aerodynamic drag torque [Nm]
%   T_v    : Viscous loss torque [Nm]
%   T_road : Total resisting torque at wheel [Nm]

vel = omega .* Vehicle_Wheel_Radius;

m     = Vehicle_mass;
r     = Vehicle_Wheel_Radius;
g     = 9.81;
b_lin = bvis / r;
omega0 = 0.5;

grade = atan(grade_pct / 100.0);

Ka  = 0.5 * rho * Cd * A * r;
```

```matlab
Krr = Crr * m * g * r * cos(grade);
Kg  = m * g * r * sin(grade);

sgn = tanh(omega / omega0);

T_rr  = -Krr * sgn;
T_g   = -Kg  * sgn;
T_d   = -Ka  * vel .* abs(vel);
T_v   = -b_lin * vel;

T_road = T_rr + T_g + T_d + T_v;

end
```

C.7 Electric Motor Efficiency Model

This section documents the motor efficiency model used to approximate the efficiency of the electric machine as a function of shaft torque and angular speed. The model is based on a polynomial loss formulation expressed in per-unit quantities and calibrated to represent the efficiency behavior of the reference electric drivetrain. The resulting efficiency is bounded between zero and unity and is suitable for system-level energy flow calculations.

```matlab
function eta = SkodaMotorEfficiency(Torque, omega)
% Motor efficiency model
%
% Inputs:
%   Torque : Motor shaft torque [Nm]
%   omega  : Motor angular speed [rad/s]
%
% Output:
%   eta    : Motor efficiency [-]

T_b = 310;                    % Peak motor torque [Nm]
w_b = 2*pi*14000/60;          % Base motor speed [rad/s]
P_loss_base = 9000;           % Base loss scaling factor [W]
Pmax = 150000;                % Maximum mechanical power [W]

Tpu = Torque / T_b;
wpu = omega  / w_b;

Ploss_pu = ...
    -0.002 ...
    + 0.175*wpu ...
    - 0.065*Tpu ...
    + 0.181*wpu.^2 ...
    + 0.577*Tpu.*wpu ...
    + 0.697*Tpu.^2 ...
    + 0.443*wpu.^3 ...
    - 0.542*Tpu.^2.*wpu ...
    - 1.043*Tpu.*wpu.^2 ...
    + 0.942*Tpu.^3;

Ploss = max(Ploss_pu * P_loss_base, 0);
```

```

Pmech = Torque * omega;

eta = Pmech / (Pmech + Ploss);

eta = min(max(eta, 0), 1);

end
```

D | Original simulink model

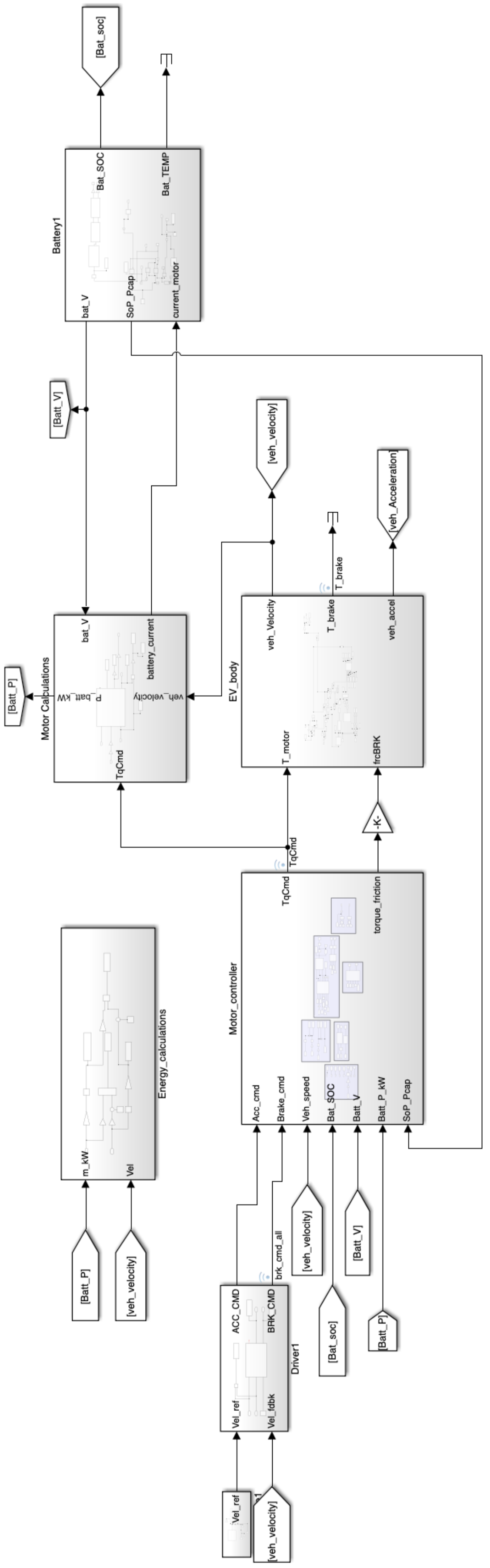


Figure D.1: Original Simulink full model

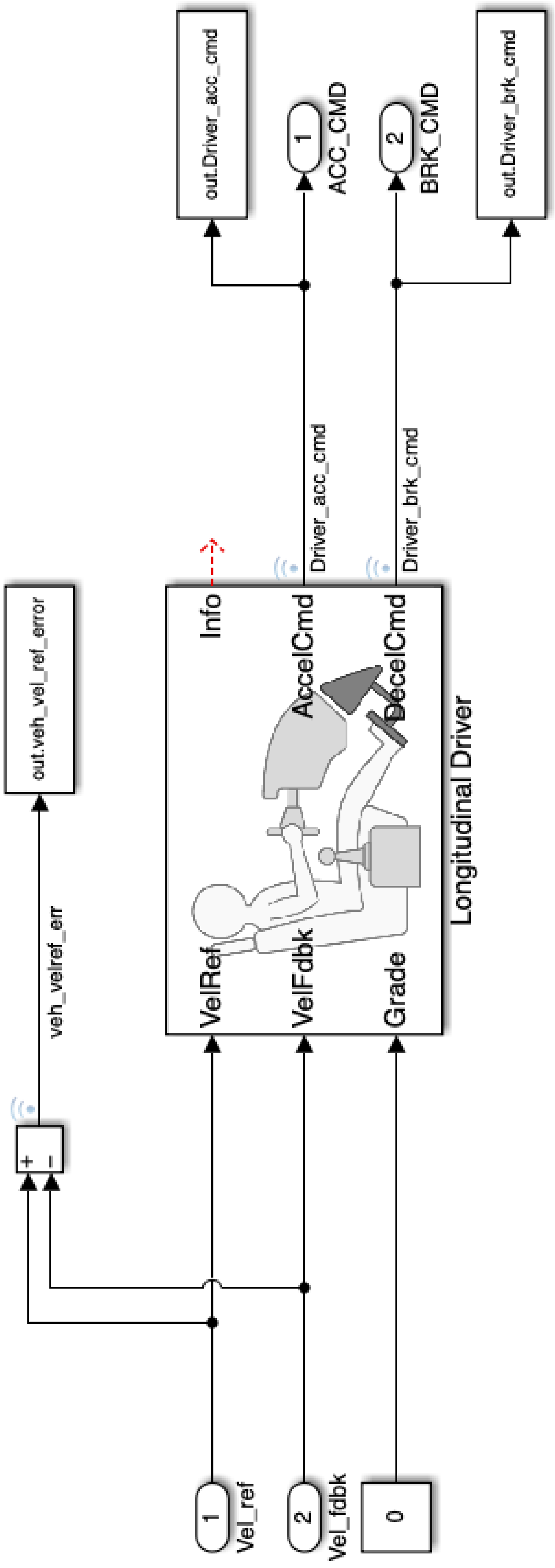


Figure D.2: Original Simulink Driver

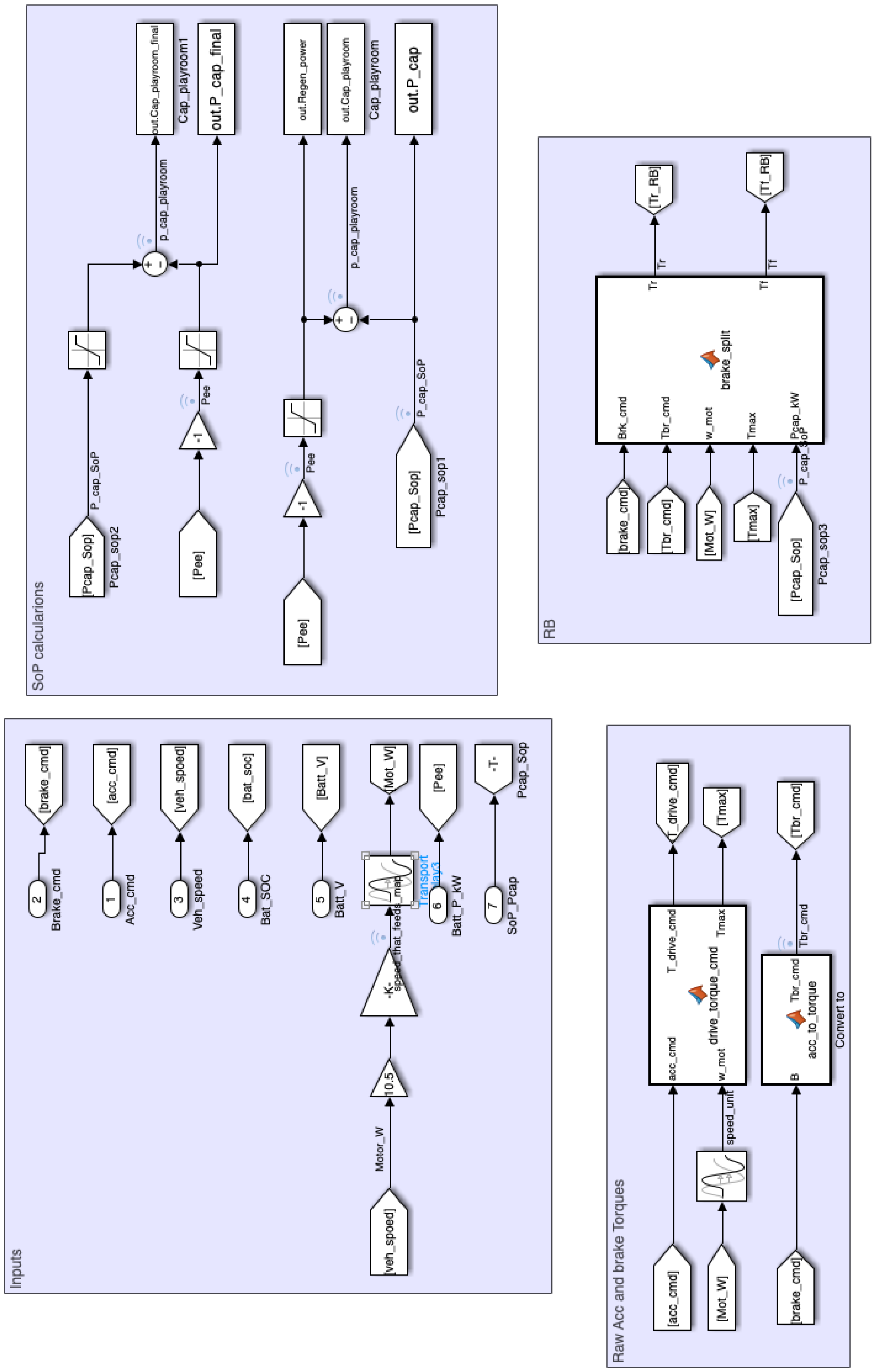


Figure D.3: Original Simulink motor controller (Part A)

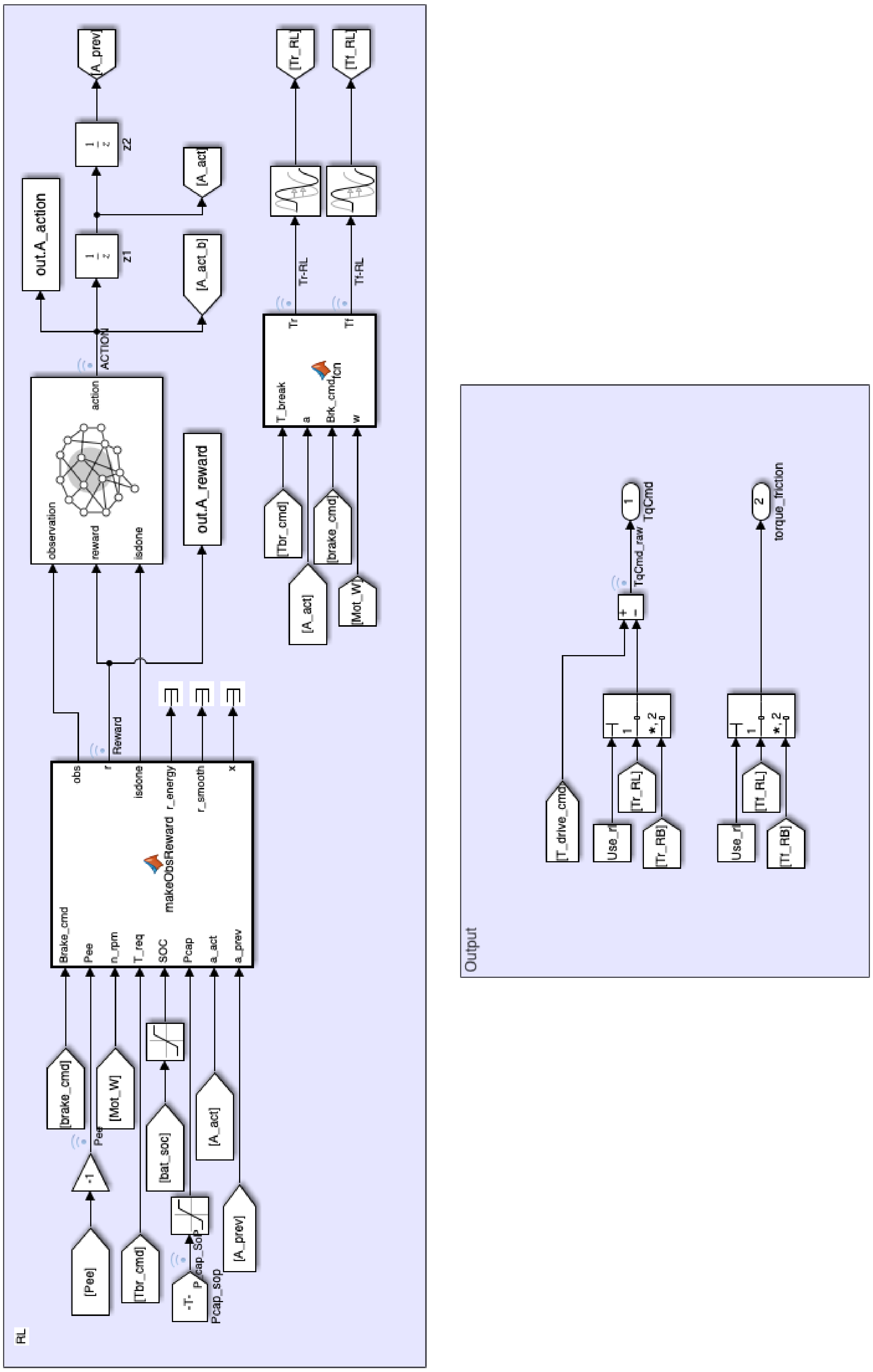


Figure D.4: Original Simulink motor controller (Part B)

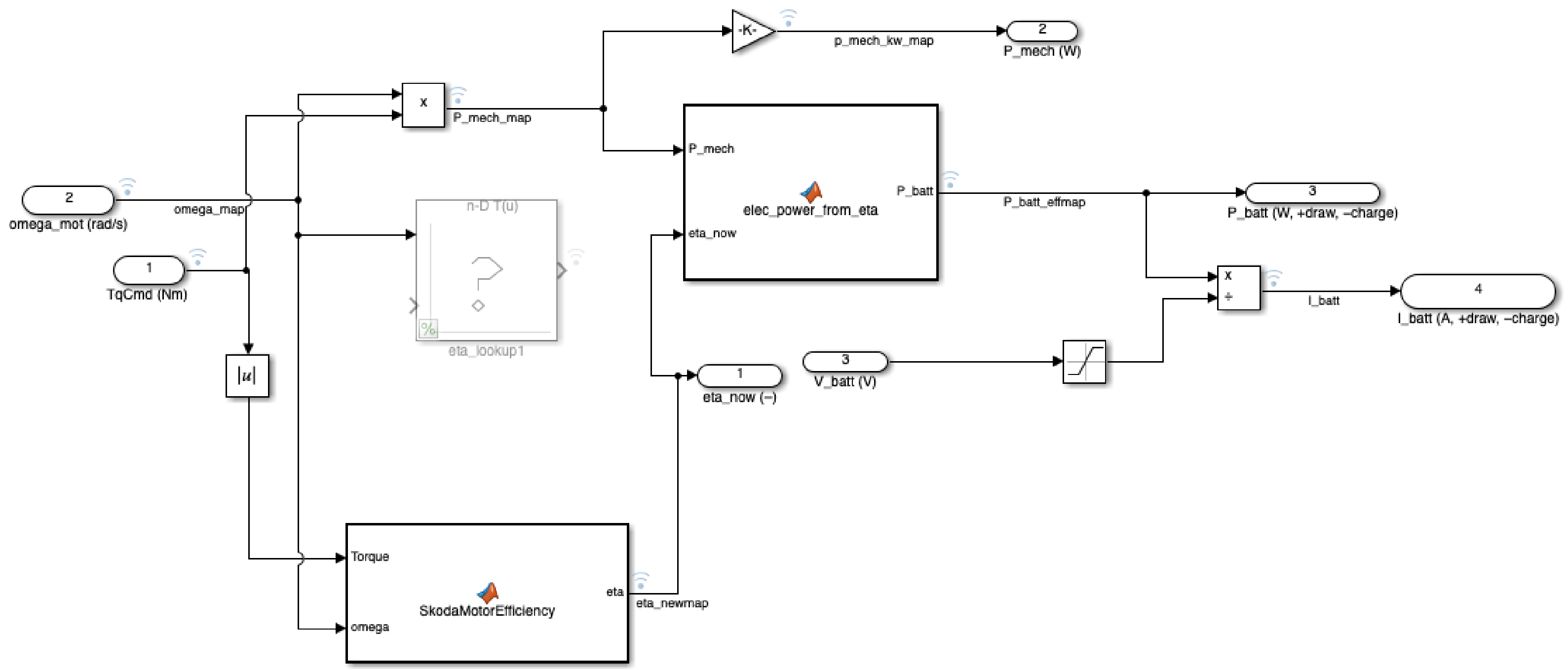


Figure D.5: Original Simulink Motor calculations (Part A)

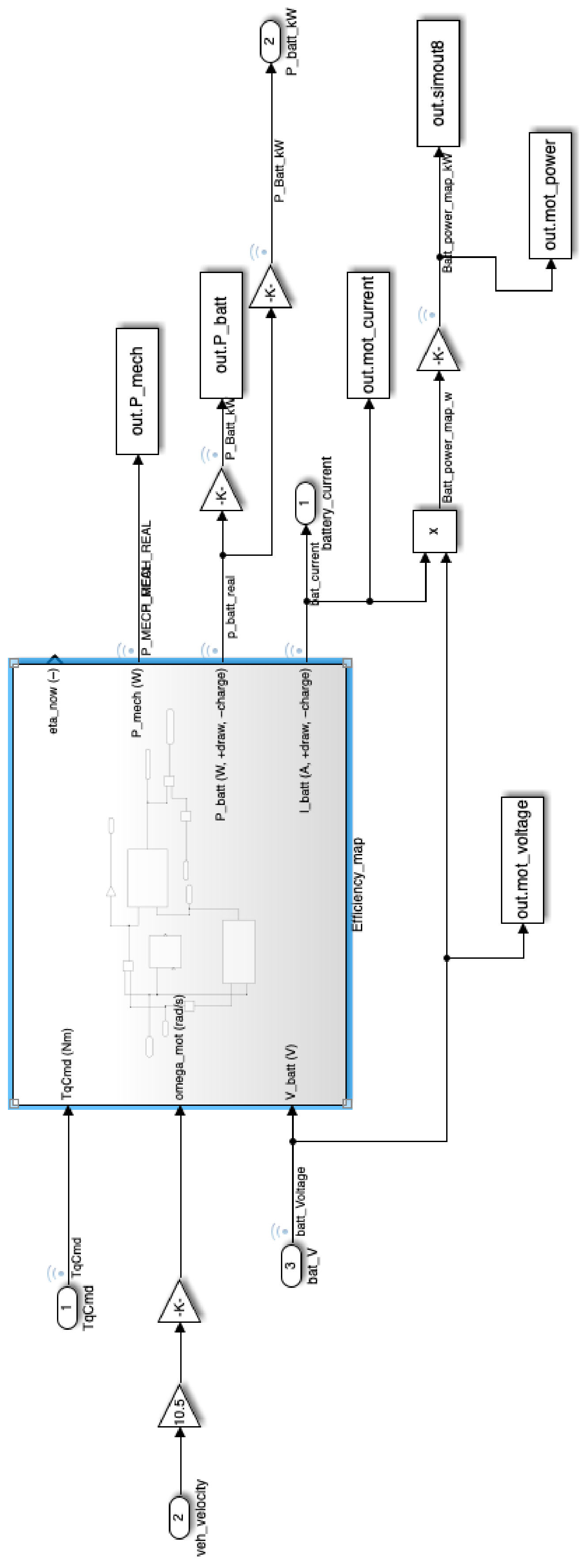


Figure D.6: Original Simulink Motor calculations (Part B)

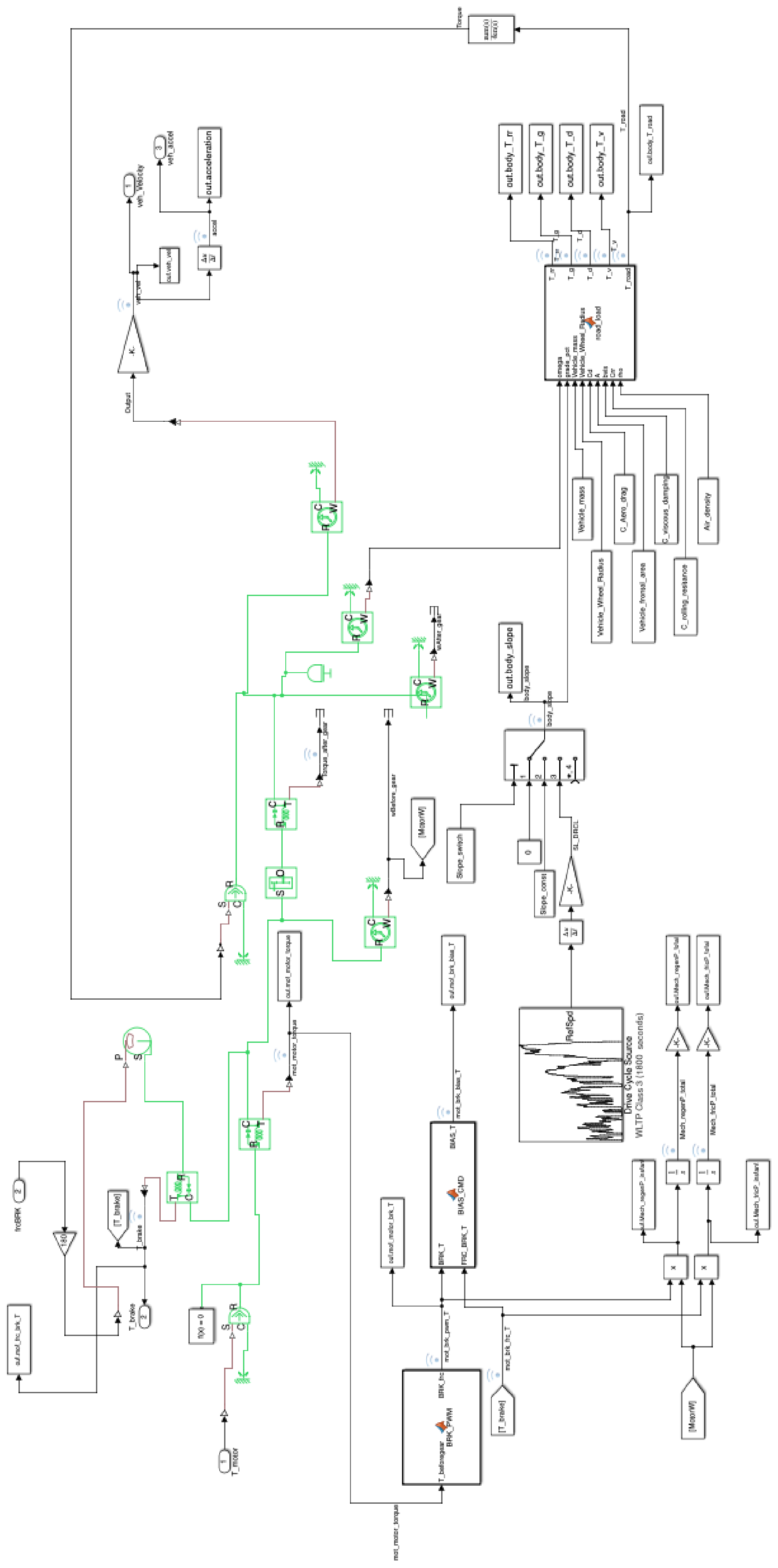


Figure D.7: Original Simulink EV body

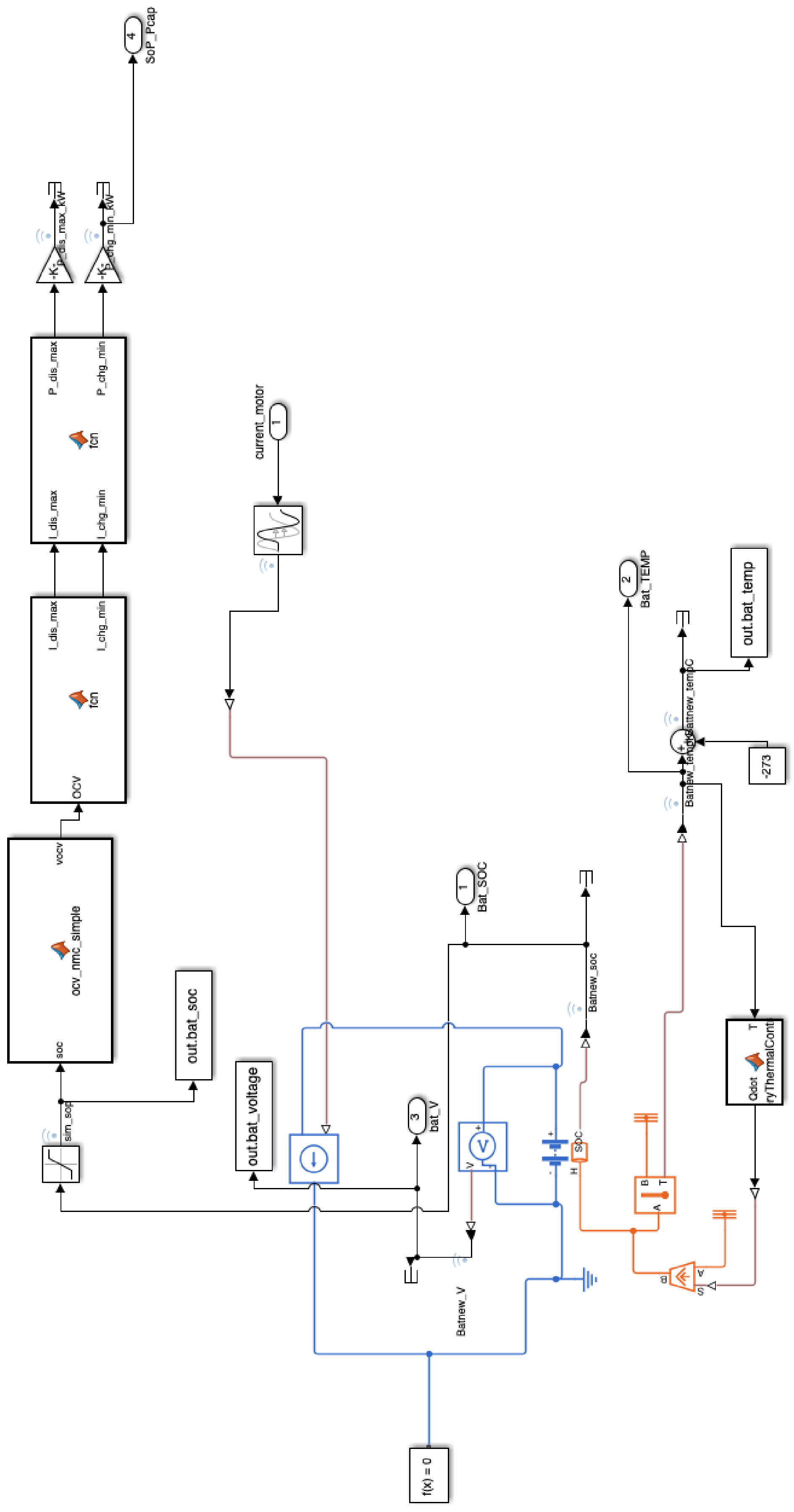


Figure D.8: Original Simulink battery