Design and Simulation of Powertrain & Battery Subsystems for Adaptable EVCU

BY

AJAYI, Nathaniel Adewale **(ENG2002529)**

EMIHIAN, Daniel Onyekachukwu **(ENG2002553)**

CHINEDUMIJIE, John Izu **(ENG1904885)**

A Project In Partial Fulfillment Of The Requirements For The Award Of A Bachelor's Of Engineering Degree

IN

THE DEPARTMENT OF MECHATRONICS ENGINEERING,

FACULTY OF ENGINEERING,

UNIVERSITY OF BENIN, BENIN CITY, NIGERIA

NOVEMBER 2025

Page Intentionally Left Blank

CERTIFICATION

This is to certify that the work detailed in the contents of this book was carried out by AJAYI, Nathaniel Adewale (ENG2002529) EMIHIAN, Daniel Onyekachukwu (ENG2002553) CHINEDUMIJIE and John Izu (ENG1904885) of the Department of Mechatronics Engineering, Faculty of Engineering, University of Benin, Edo State, Nigeria.

Supervisor: **Prof. O. O. Igbodaro**

_____ _____

Signature *Date*

Project Co-ordinator: **Dr. Godspower Ojariafe**

_____ _____

Signature *Date*

Head of Department: **Prof. O. O. Ighodaro**

_____ _____

Signature *Date*

DEDICATION

This project is dedicated to the pursuit of knowledge and the power of human reason — the undeniable engines of progress.

To our parents and grandparents, whose love, support, and belief in education gave us the foundation to get here in life and to aim even higher.

To my siblings, friends, teachers and lecturers — thank you for the encouragement, the good advice and educational support. This wouldn't have been possible without you.

Lastly, to all those who dare to question, to learn, and to build a better future for Nigeria through critical thought and action.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to everyone who contributed in one way or another to the successful completion of this project.

First and foremost, I extend my deepest appreciation to my project supervisor, Prof. O. O. Ighodaro, for his invaluable guidance, constructive criticism, and unwavering patience throughout the course of this work. His expertise and encouragement were instrumental in shaping the direction of this study.

I am also immensely grateful to my coursemates, whose support, ideas made the research process both enlightening and enjoyable. The countless discussions, shared resources, and mutual encouragement helped keep me motivated till the end.

To my family and friends – thank you for your moral support, and understanding during this period. Your encouragement kept me going even when the journey got tough.

ABSTRACT

The increasing complexity of Electric Vehicle (EV) powertrains necessitates a robust, integrated, and flexible control strategy, centralized within the Electric Vehicle Control Unit (EVCU). This study shows the implementation of the I_d = control strategy using the MATLAB/Simulink Motor Control Blockset under varying loading conditions and speed requirements. This study further goes on to show an implementation of a CC-CV charging controller for a Li-ion battery with multiple current control loops.

The study is designed for compliance with the AUTOSAR (Automotive Open System Architecture) Classic Platform for compliance – ensuring modularity, portability and adherence to industry standards.

The study results validate the performance of the Interior PMSM and the ability to generate C implementation and header files from the model-based engineering (MBE) design approach which can be further used for hardware-in-the-loop testing. This study concludes that the MBE and AUTOSAR approach produces a highly efficient framework for developing, validating and iterating on complex, multi-domain electric vehicle components.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

ADC – Analog-to-Digital Converter

ASW - Application Software

AUTOSAR - Automotive Open System Architecture

BLDC – Brushless Direct Current

BSW - Base Software

CAN – Controller Area Network

CKD - Completely Knocked Down

ECU – Electronic Control Unit

EV - Electrical Vehicles

EVCU – Electrical Vehicle Control Unit

FOC - Field-Oriented Control

GPIO – General Purpose Input/Output

HIL - Hardware-In-The-Loop

HVAC – High-Voltage Alternating Current

I2C – Inter-Integrated Circuit (Communication Protocol)

MBD - Model Based Design

MBE - Model Based Engineering

MCU – Microcontroller Unit

PI – Proportional-Integral

PMSM – Permanent Magnet Synchronous Motor

PWM – Pulse Width Modulation

ACRONYMS

RAM – Random Access Memory

ROM – Read-Only Memory

RPM – Revolutions Per Minute

RTE - Runtime Environment

RTOS – Real-Time Operating System

SKD - Semi Knocked Down

SPI – Serial Peripheral Interface

UART – Universal Asynchronous Receiver/Transmitter

V/F – Voltage/Frequency

VFB - Virtual Functional Bus

CHAPTER 1 – INTRODUCTION

1.1. Background of the Study

The evolution of electric vehicles (EVs) has revolutionized the automotive industry, with companies like Tesla Inc., Polestar, BYD, e.t.c. at the forefront of pioneering innovative solutions for vehicle control systems. As respective leaders in automotive electronics, these manufacturing firms emphasize the development of efficient, reliable, and adaptable technologies to meet the demands of modern electric mobility. Central to this innovation is the Electronic Vehicle Control Unit (EVCU), which serves as the brain of an EV, managing critical functions such as motor control, power regulation, and system communication.

1.1.1 Timeline of Vehicle Manufacturing in Nigeria

In the early 1970s and 1980s, Nigeria's automotive sector was emerging with various government-backed assembly projects with notable partnerships with the likes of Peugeot and some other assemblers.

In 1972, the government had signed a contract with a popular brand, Volkswagen of Germany, to establish an assembly plant in the country (Britishpathe, 1975). By 1975, production was well underway at the factory, which was located along the recently built Lagos-Badagry expressway. These vehicles were assembled using Volkswagen-supplied, CKD parts that were imported directly from Germany. Different models, including the Beetle (1300cc, 1500cc, 1600cc), Audi (100cd), Golf, Kombi bus, Jetta, and Passat were all made at this facility (Wikipedia, 2025).

By the late 1980s and early into the 1990s, Nigeria's automotive sector had started facing multiple crippling and severe headwinds. The rise of economic instability, bolstered by currency devaluation and a high rate inflation, significantly increased the cost of importing raw materials and SKD/CKD components, making local assembly unattractive and increasingly unviable.

The inconsistency with government policies and general mismanagement within multiple state-backed ventures hampered efficiency and productivity. This period also saw the rise of the *tokunbo* (second-hand imported vehicles) market. This influx of relatively cheaper, used vehicles, mostly from Europe and America, provided an affordable alternative for consumers and effectively undercut the demand for locally assembled new cars, ultimately leading to the rapid decline and consequent closure of most of these domestic assembly plants and manufacturing programmes set up by the then administration.

This sad downturn of events led to a decade of stagnation in local automotive manufacturing, particularly in the advancement of vehicle control unit (VCU) technology. While global players moved towards sophisticated electronic control units (ECUs) and eventually, the highly integrated Electronic Vehicle Control Units (EVCUs), Nigeria's industry remained largely reliant on outdated mechanical and rudimentary electronic systems, with little to no investment in research and development for modern automotive electronics.

1.1.2 Nigerian Automotive Industry Today

Since the 2010s, there has been a notable, albeit largely "on paper", revival in Nigeria's automotive industry. This resurgence has been driven by renewed government policy and the emergence of multiple local champions. The Nigerian Automotive Industry Development Plan (NAIDP), launched in 2013, was a significant policy intervention aimed at attracting investment, encouraging local manufacturing, and reducing the importation of fully built vehicles. This policy introduced incentives such as reduced import tariffs on CKD components and higher tariffs on fully built vehicles, intending to make local assembly more competitive (West African Automotive Show, 2024).

This policy framework has fostered the growth of indigenous automotive companies like Innoson Vehicle Manufacturing (IVM), which has become a prominent local champion in EV manufacture and assembly. IVM has expanded its production capabilities to include a range of vehicles, from cars and SUVs to buses and trucks, with a focus on local content development. Other assemblers have also emerged or scaled up operations, often in partnership with international brands, to assemble vehicles within Nigeria. While the aspirations of the NAIDP and the achievements of local manufacturers are commendable,

challenges remain in fully realizing the industry's potential, particularly concerning consistent policy implementation, access to finance, and the competitiveness of locally produced vehicles against imported alternatives.

Despite these promising developments, the Nigerian automotive industry continues to face significant challenges. Local production volumes remain relatively low compared to the market demand, with a heavy reliance on imported fully built units and semi-knocked-down (SKD) parts (BusinessDay, 2025). Infrastructure deficits, inconsistent power supply, and a lack of access to affordable financing also impede growth. Moreover, while there are efforts to increase local content, the supply chain for automotive components is still underdeveloped, leading to higher production costs and a struggle to compete with cheaper imports.

Indigenous companies like the Dangote Group also have recently ventured into the automotive sector, establishing an assembly plant in Kaduna producing trucks and other heavy-duty vehicles in partnership with Sinotruk, aiming to address the demand for commercial vehicles and contribute to local vehicle manufacturing. The National Automotive Design and Development Council (NADDC) continues to play a crucial role in promoting the automotive industry through various initiatives, including the development of automotive testing centers, skills acquisition programs, and research into electric vehicle components (Nanyang Technological University, 2022).

Stallion Group – a prominent player in the Nigerian automotive industry – operates DPAN (formerly known as PAN - Peugeot Automobile Nigeria). DPAN has a long history in Nigeria, having been a significant assembler of Peugeot vehicles. The Lagosian Stallion Group's involvement with DPAN represents a continued effort to support local vehicle assembly and production, contributing to the diversity of the automotive market in Nigeria. While specific details of their current electric vehicle initiatives within DPAN were not immediately available in the context provided, their legacy in assembly positions them as a potential future contributor to EV production in Nigeria, especially given the country's push towards green mobility.

The rapid advancement of electric vehicle (EV) technology has driven the need for efficient, reliable, and adaptable control systems to manage critical vehicle functions such as

motor control, power distribution, and communication. The Electronic Vehicle Control Unit (EVCU) serves as the central processing hub in EVs, orchestrating subsystems to ensure optimal performance, energy efficiency, and safety. As the automotive industry shifts toward sustainable mobility, the development of cost-effective and versatile EVCUs with robust and adaptable control systems has become a focal point for research and innovation, particularly in the design of embedded systems for motor control and battery regulation.

1.2. Statement of Problem

In light of the increasing global shift towards electric mobility, a stronger demand for more robust, efficient, reliable and adaptable solutions for electric vehicle control systems seems to have emerged. However most EV manufacturers develop proprietary ECUs that are tailored to specific vehicle models and hardware architectures. This lack of standardised, adaptable control systems limits interoperability, reusability and scalability amongst platforms. As a result, engineers and researchers face significant challenges when attempting to integrate or modify control systems for various electric powertrains, battery capacities, and vehicle types.

In a lot of developing countries like Nigeria, the situation is further escalated by dependence on imported EV control hardware, which is often costly, difficult to maintain, and not optimised for local driving conditions. These imports are not easily adaptable, customizable for custom builds or extending local manufacturing. The absence of a unified, modular ECU architecture hinders the growth of indigenous EV innovation and slows down development cycles.

This creates a pressing need to design an easily adaptable Electric Vehicle Control Unit (EVCU) for managing multiple EV subsystems — including motor control, battery management, field communication, and safety operations. By addressing these gaps, the proposed design will provide a foundation for adaptable, cost-effective, and locally maintainable EV control solutions, supporting Nigeria’s ongoing transition to clean and intelligent transportation technologies.

1.3. Aims & Objectives

Aim

The aim of this study is to employ model-based engineering design approach to design and validate two electronic vehicle subsystems with focus on efficiency and modularity and adaptability for EV hardware.

Objectives

- Develop modular control models for electric motor drive systems, enabling efficient torque control, speed regulation, and acceleration within the EVCU framework.
- Design models and algorithms for battery state estimation and management, including monitoring of State of Charge (SoC), charging and discharging currents.
- Validate the overall system performance through simulation in MATLAB/Simulink, evaluating response time and efficiency under different driving and load conditions.
- Generate AUToSAR compliant **.c, .h** header files annd .**arxml** files from MATLAB/Simulink Autosar Component Designer with Embedded Coder.

1.4. Scope of Study

This scope of this study is centered around the design, modelling and validation of an Electric Vehicle Control Unit (EVCU) using MATLAB/Simulink and MBE techniques on the AUTOSAR Classic platform. The research is strictly limited to the conceptual and simulation phase of EVCU development rather than full-scale hardware implementation. It aims to demonstrate how an integrated control system can efficiently manage major electric vehicle subsystems while ensuring compliance with basic safety and communication requirements.

The scope encompasses the following key areas:

1. **Motor Control Implementation:** Design and simulation of control algorithms for electric propulsion, including torque regulation, speed and cruise control.

2. **Battery Management:** Modelling of a battery system capable of monitoring key parameters such as voltage, current and State of Charge (SoC), to ensure optimal energy utilization and safety.

Exclusions: The study does not include the physical fabrication of the control unit hardware, detailed PCB design, or full-scale vehicle testing but rather places emphasis on simulation modelling, algorithm development, and performance validation within a virtual MATLAB/Simulink environment. The findings are intended to serve as a foundation for future hardware implementation and optimisation for various electric vehicle platforms.

1.5. Justification of Study

The accelerating global shift toward clean energy and low-emission transportation is reshaping the automotive industry. According to the Global EV Outlook 2025 from the International Energy Agency (IEA), more than one in five new cars sold in 2024 were electric. EVs are central to decarbonising road transport, which accounts for approximately one-sixth of global emissions.

This underlines the urgency of developing robust technologies—such as efficient, safe, and modular Electric Vehicle Control Units (EVCUs)—to meet global climate goals and regulatory demands.

In Nigeria, substantial policy and industrial developments complement this global trend. In July 2025, the Nigerian Senate passed the Electric Vehicle Transition and Green Mobility Bill, 2025, which establishes a legal framework for accelerating EV adoption, promoting green mobility, and supporting industrial capacity building (The Guardian, 2025).

The country has also approved the 2023 Auto Policy aimed at increasing local vehicle production. This policy targets achieving 40% local content in vehicle manufacturing and 30% locally produced electric vehicles (Mobility Rising, 2025).

At the same time, the National Action Plan for EVs is offering financial incentives, including tax breaks and import duty reductions, to encourage both local manufacturing of EVs and establishment of supporting infrastructure.

Moreover, the Nigerian government has committed to 100% zero-emission sales for new cars and vans by 2040, aligning with global zero emission vehicle (ZEV) declarations. (Climate Scorecard, 2025)

Local manufacturers are also stepping up: Innoson Vehicle Manufacturing (IVM), for instance, plans to start producing electric vehicle models locally by mid-2026 with a capacity of about 10,000 units per year (EVWorldAfrica, 2025).

Despite these encouraging developments, gaps remain in technological capacity, especially in advanced control systems such as EVCUs. Many existing EV efforts in Nigeria currently rely heavily on imported components or less flexible control architectures, which may not be optimized for local conditions. Thus, a study exploring the design of a general-purpose EVCU using a modern standard architecture like AUTOSAR can contribute significantly. It would support indigenous capability in safety, energy efficiency, modularity, and reduced dependency on foreign technology.

CHAPTER 2 – LITERATURE REVIEW

2.1 Introduction

Electric vehicles and vehicles in general run on and co-ordinate many subsystems that work separately while locally linked to each other via a FieldBus commnunication system. These subsystems include the PowerTrain, Inverters, Battery Management, Thermal Management, Charge Controllers, Auxiliary Systems, e.t.c. The EVCU is the top-level coordinator for these other subsystems. It is in charge of aggregating data from all the sensors, make supervisory decisions (torque requests from the inverter, battery cooling, e.t.c.). Modern EVCU design must balance real-time control, meet safety standards, achieve modularity and maintainability while interfacing with heterogeneous communication field bus systems (e.g CAN & LIN Bus).

AUTOSAR (Automotive Open System Architecture) — a global development partnership founded in 2003 by automotive manufacturers, suppliers and other companies from the electronics, semiconductor and software industries — provides a software standard architecture, interfaces and development artifacts to improve reusability, adaptability and easy integrability for automotive ECUs (Martínez-Fernández et al., 2015).

Building with AUTOSAR can shorten integration time (possibly reducing overall production and testing cycles) and enable multi-OEM adoption but it imposes constraints (Martínez-Fernández et al., 2015) (runtime environment, standardized communication and component models) that influence real-time and memory design.

2.2 EVCU Design Chronology

2.2.1 Early Systems

The earliest design approaches for Electric Vehicle Control Units (EVCUs), emerging in the late 1990s and early 2000s were designed to be relatively simple supervisory controllers. Their main functions included torque demand calculation, battery protection, and basic communication with inverters and chargers via low-bandwidth CAN or LIN buses. At this time of development, EVCU design was largely OEM-specific, with proprietary design applications and architectures with limited software modularity. These early implementations were fundamentally tightly coupled with their running hardware, limiting reusability and making scaling to new vehicle platforms costly and time-consuming (Gao et al., 2009; Suh et al., 2010).

2.2.2 Growth In the 2000's

With the increasing EV adoption in the mid-2000s, EVCUs began to integrate more subsystems. This ever-growing complexity motivated the adoption of Model-Based Design (MBD) tools such as MATLAB/Simulink for algorithm development and code generation (MathWorks, 2011). Researchers like Rajeshwari Hegde & K S Gurumurthy (2008) and OEMs reported improved simulation capability, faster prototyping, and traceability from model to code. However, issues arose in verifying real-time performance and ensuring interoperability across heterogeneous ECUs.

The introduction of the AUTOSAR partnership founded in 2003 by multiple automobile manufacturers, suppliers and other companies from the electronics, semiconductor and software industries with the sole purpose to develop and establish an open and standardized software architecture for automotive electronic control units (ECUs). . EVCUs increasingly adopted the AUTOSAR Classic Platform to manage BSW, complex drivers, and communication services. This enabled the reusability of software, multi-supplier collaboration, and faster integration cycles (Martínez-Fernández et al., 2015). At the same time, AUTOSAR imposed constraints in terms of runtime overhead, memory footprint, and the need to adapt timing-critical motor control algorithms to fit standardized component

models. Research from this period highlights AUTOSAR's value in meeting ISO 26262 functional safety requirements, but also its challenges in implementing high-frequency control loops.

2.2.2 Control Platforms Today

Recent trends in EVCU design reflects a rise in the utilization of AUTOSAR's Adaptive Platform for Model-Based Design in current EV manufacturing and development coupled with a lot of emerging machine learning and artificial intelligence technologies. Most recent research points toward general-purpose, platform-agnostic EVCU designs that can be adapted across multiple vehicle classes.

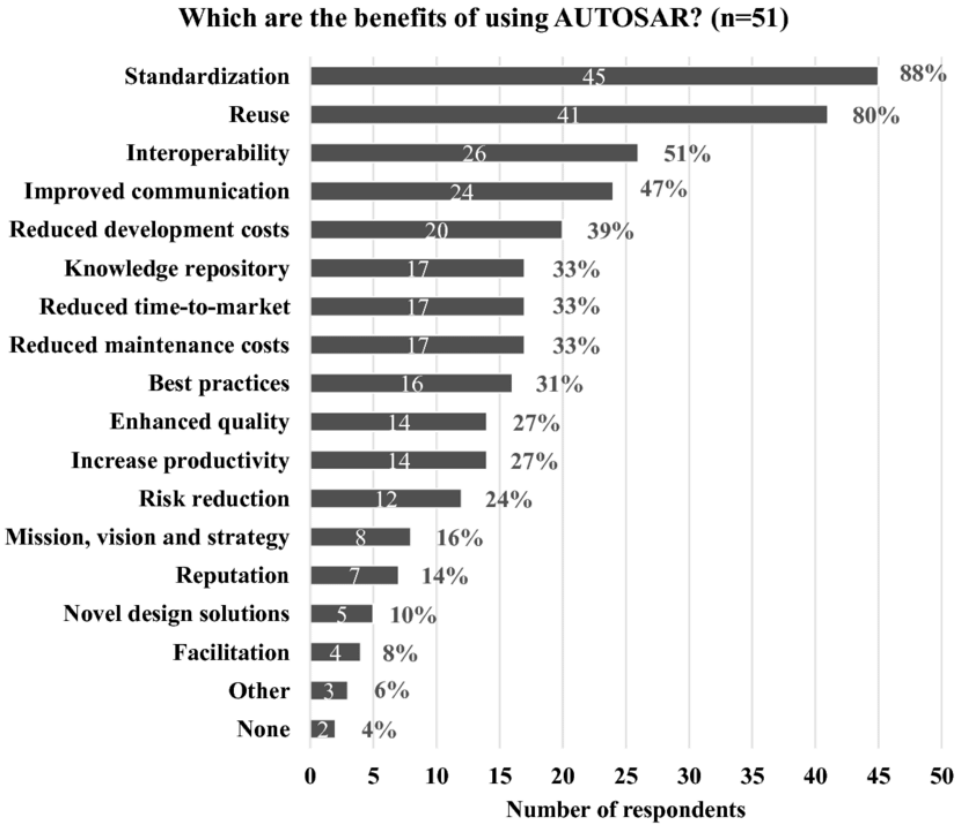


Fig 2.1 Survey Response on AutoSAR benefits from 51 respondents

2.3 AUTOSAR Overview

AUTOSAR, otherwise known as Automotive Open System Architecture, is a global partnership founded in 2003 by various leading automotive manufacturers, parts suppliers, and different companies from electronics, semiconductor, and software industries. It's primary purpose has always been to develop and establish an open and standardized software architecture for automotive ECU development.

As outlined by P. Hansen (2009), this initiative arose from the previously discussed complexity of automotive software development and the need for a common framework to manage the diverse and growing number of electronic systems in vehicles. Before AUTOSAR, automotive software development was mostly proprietary and fragmented, leading to significant challenges in terms of reusability, interoperability, and scalability across different vehicle platforms and platform suppliers.

The key successes of AUTOSAR are a result of its ability to address challenge like these:

- **Standardization and Modularity:** AUTOSAR provides a layered software architecture and standardized interfaces and these promote modularity and allow development of reusable software components. Ravinder Katla (2025) notes that this reduces development time and costs, as engineers can focus on application-specific logic rather than low-level hardware interactions.

- **Interoperability:** Using properly defined uniform communication standards and interfaces, AUTOSAR enables seamless integration of subsystems from various parts suppliers. This fosters multi-supplier collaboration and reduces the need for extensive rework when integrating components from different vendors.

- **Scalability and Flexibility:** The architecture supports scalability across different vehicle classes and hardware platforms, from simple ECUs to complex domain controllers. This flexibility is crucial for adapting to evolving automotive technologies and market demands.

2.3 Applications of Model Based Engineering in EVCU Design

The design of vehicle control subsystems and control logic using model based engineering approaches has been a recent interest for researchers, engineers and software developers.

Motor and Torque Control

The works of Zhiyu Huang and Zhangliang Xiong (2015) developed a motor and torque control on the Infineon TC1767 processor using a Model Based approach. They also implemented the design of the ASW, VFB/RTE layer, runner entities and BSW according to AUTOSAR hierarchical and modularization standards. Recent electric vehicles are equipped with Permanent Magnet Synchronous Motors (PMSM). Zhiyu Huang and Zhangliang Xiong conduct bench testing on a 10 kW Permanent Magnet Synchronous Motor (PMSM) system to demonstrate the capacity of the AUTOSAR-based Electronic Control Unit (ECU) design to manage motor initiation, acceleration, and deceleration, exhibiting satisfactory responsiveness (e.g., from 0 to 600 rpm in 0.6 seconds) within medium to low-speed operational requirements. Some other main parameters listed includes six pair of pole number, a rated voltage 109~190V and stator resistance 0.4Ω with stator inductance of 1.55mH.

This studies shows benefits on modularity, interoperability and hardware agnostic design approach with fast and stable response from the PMSM motor.

Battery Management

Taking a reference from the works of Duck Young Kim, Won Tae Joe, and Hojin Lee (2018) were responsible for the development of a Battery Management System (BMS) for the the Volvo XC90 plug-in hybrid, using AUTOSAR as required by the OEM, and adopting Model-Based Design using MATLAB/Simulink as the development methodology.

Using MBE, they aimed to

1. Increase component reuse
2. Reduce manual coding
3. Improve communication with customers, and

4. Deliver a higher quality BMS.

Employing MBE allowed them to better simulate interactions among software components, visualize signal propagation, and detect points of failure early on.

The BMS software was developed to be AUTOSAR-compliant, with the MBE design workflow driving generation of software components, interfaces and code consistent with the AUTOSAR architecture. MATLAB/Simulink, AUTOSAR Blockset and Embedded Coder were employed to build models, test and generate code.

2.4 Previous Works and Criticisms

Thaker T. Yahia, et al (2023) show an integration of MBE for Motor Speed controller design for torque estimation using transfer function characteristics. The use of models and Separately Excited Direct Current Motor (SEDCM) closely mirrors the PMSM d-q axis FOC control strategy but not exactly. I defer to the trend of EVs not predominantly employing DC Motors in their powertrain.

Chen (2019) provides a comprehensive overview of FOC and its advantages in dynamic response and efficiency. The work demonstrates up to a 5% efficiency gain over standard scalar control methods under variable load conditions. This paper is widely cited as the benchmark for high-performance motor control. While this paper focuses on efficient motor control, it does not primarily take into concern adaptability for EVCU integration or exporting to AUTOSAR compliant code platforms.

Gupta & Weiss (2020) describe the move toward standardized, highly-integrated ECUs. They argue this approach reduces development costs and ensures reliability through rigorous validation. As a result, novel algorithms that require specific hardware accommodations (e.g., unique sensor inputs or faster control loops) cannot be tested or deployed. Bamidele (2022) explicitly criticizes this trend, arguing it "stifles local innovation" in emerging markets. The reliance on imported, non-adaptable ECUs means that vehicles cannot be easily customized for local conditions (e.g., different battery chemistries, road conditions, or lower-cost motor variants). The work highlights that the advancements by Chen (2019) are often impractical for any group outside of the original ECU manufacturer.

2.5 The Research Gap

This literature review highlights that AUTOSAR has emerged as a strong candidate architecture for the development of a general-purpose Electric Vehicle Control Unit (EVCU). Its layered software architecture and standardized interfaces provide modularity, enabling developers to design reusable components that can be more easily maintained and upgraded. Furthermore, AUTOSAR promotes vendor interoperability by defining uniform communication standards, enabling integration of subsystems supplied by different manufacturers without extensive rework. The availability of mature toolchains and support for model-based development (MBD) significantly accelerates the design process, shortens testing cycles, and enhances the division of work across engineering teams. These features collectively reduce time-to-market while ensuring compliance with emerging industry requirements.

Despite these strengths, several practical challenges persist in the application of AUTOSAR to EVCU design. One key issue is the need for precise motor control strategies, such as field-oriented control (FOC) for PMSMs, which demand real-time execution and tight latency guarantees—areas where the AUTOSAR runtime environment can impose constraints. Additionally, optimizing energy efficiency in EV powertrains remains a pressing requirement, requiring integration of advanced algorithms for regenerative braking, battery health management, and thermal optimization. Meeting functional safety certifications (e.g., ISO 26262) introduces further complexity, as AUTOSAR must be tailored to support deterministic behavior and fault-tolerant architectures. Finally, hybrid execution environments, where time-critical tasks coexist with less stringent functions, present partitioning challenges in scheduling and resource allocation within AUTOSAR-based platforms.

CHAPTER 3 - METHODOLOGY

3.1 Research Methodology Overview

This study implements the use of MBE design approach integrated with the AUTOSAR classic platform to design and simulate a general purpose EVCU. This research methodology on the AUTOSAR concepts of modularity, reusability and traceability (Kim, Joe & Lee, 2018; Hansen, 2016).

This project adopts a concept and simulation design cycle rather than hardware implementation focusing on model development and evaluation for electrical motor control and battery state management.

The core of this research methodology is the Model-Based Engineering (MBE) approach, which offers a systematic way to design, analyze and verify complex systems like EVCU and simulate performance and stability. MBE emphasizes the use of abstract models throughout the development lifecycle, from requirements definition to implementation and testing. This approach is particularly advantageous for automotive embedded systems due to its ability to improve clarity, reduce errors, and facilitate early validation and error detection.

The MBE methodology emphasizes the use of abstract, often graphical, models as the primary artifacts throughout the entire system lifecycle, from requirements definition and design to implementation, testing, and deployment. Unlike traditional document-centric approaches, our MBE approach uses formal models created in MATLAB/Simulink to simulate system behavior, structure, and requirements.

3.1.1 System Architecture Definition

The EVCU model, a critical component in electric vehicle operation, is meticulously decomposed into distinct Software Components (SWCs). This decomposition strategy adheres to AUTOSAR principles, promoting clear interfaces between components and enabling parallel development efforts by different teams or individuals. Each of these Software Components is designed to encapsulate specific functionalities, ensuring a clear separation of concerns and contributing to the overall robustness and maintainability of the

EVCU software. This modular approach also significantly simplifies future updates, modifications, and the integration of new features, as changes can often be confined to a single component without affecting the entire system.

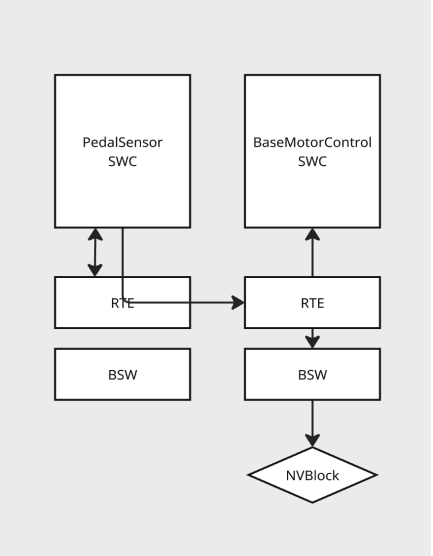


Fig 3.1 RTE Layer for Motor Control SWC

1. **Pedal Sensor Runnable SWC:** The Pedal Sensor Runnable Software Component (SWC) is responsible for calculating pedal displacement, the rate of change/velocity of change, and the equivalent Analog-to-Digital Converter (ADC) gain. The pedal sensor can be integrated with rotary or linear encoders, hall effect sensors or even simple potentiometers for the sake of experimentation.
2. **Motor Control SWC:** This component responsible for managing torque and speed regulation for the Permanent Magnet Synchronous Motor (PMSM) for constant cruise speed or to map pedal input from the PedalSensorRunnable.
3. **Battery Management SWC:** This component monitors State of Charge (SOC), battery voltage and the flow of current.

3.2 Motor Control System Modelling

The Motor Control Subsystem plays a crucial role in electric vehicles (EVs) by translating driver inputs and speed requests into precise motor commands. Its primary function is to

achieve the desired torque and speed control, ensuring a smooth, responsive, and efficient driving experience.

This subsystem receives input from multiple sources, e.g accelerator pedal inputs, which directly reflect the driver's request for acceleration or deceleration and these sources are continuously monitored. Additionally, cruise control, when active or desired, provides a consistent target speed that the motor control must maintain.

Upon receiving these inputs, the Motor Control Subsystem calculates the corresponding motor response from the MotorController system.

The PedalSensorRunnable has one Sender-Receiver port, the voltage representing the magnitude of the displacement from the zero point is translated to a corresponding RPM desired output by passing it through the PedalSensor subsystem block. The rate of change of displacement of the pedal sensor input is also monitored by the system for the PI Controller Gains.

This control paradigm used for efficient speed and torque delivery involves:

- **Motor Control Strategy:** Employing advanced control techniques Field-Oriented Control (FOC) using the ***FOC Default Controller Gains*** block on Simulink to precisely manage the motor's electromagnetic field and produce the desired torque.
- **Inverter Control:** Generating the necessary pulse-width modulation duty signals to control the inverter that converts DC power from the battery into AC power for the PMSM indcution motor.
- **Speed Regulation:** Continuously monitoring motor speed and adjusting torque output to maintain the target speed, especially for cruise control speed requirements.

The output of the Motor Control Subsystem directly governs the electric motor, dictating its rotational speed and the amount of torque it produces. This sophisticated interplay of sensors, software, and power electronics is fundamental to the performance, efficiency, and safety of any modern electric vehicle.

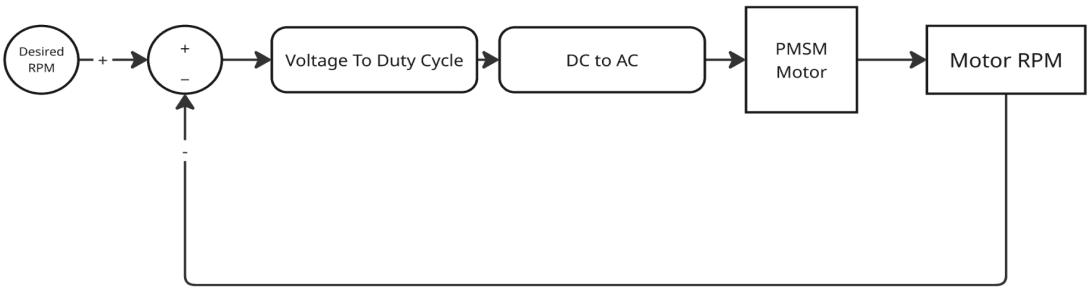


Fig 3.2 Motor Control Block Diagram

The parameters of the PMSM Induction Motor for this study has the values as represented on the table below:

Parameter	**Symbol**	**Value**	**Unit**
Rated Voltage	V	325-350	V
Stator Resistance	R_s	0.02	Ω
d-axis Inductance	L_d	0.61314	μH
q-axis Inductance	L_s	0.12263	mH
Permanent Magnet Flux	ψ_f	0.0725	Wb
Pole Pairs	p	4	-
Rotor Inertia	J	0.0027	kg·m²
Damping Coefficient	B	0.0025	N·m·s

Table 3.1 PMSM Motor Rating (Mathworks, 2023)

The equations used to describe the electrical behavior of a Permanent Magnet Synchronous Motor (PMSM) use the *d-q* rotor reference frame. This mathematical model transforms the

complex, time-varying AC voltages and currents in the motor's stator into two equivalent DC (or slowly changing) components. This simplification is crucial for designing and implementing high-performance control systems, like Field-Oriented Control (FOC) using the $i_d = 0$ control strategy.

The voltage V_d for the d-axis is represented by:

$$V_d = \underbrace{R_s i_d}_{\text{Resistive Drop}} + \underbrace{L_d \frac{di_d}{dt}}_{\text{Inductive Voltage}} - \underbrace{\omega_e L_q i_q}_{\text{Cross-Coupling EMF}}$$

While the q-axis is represented by:

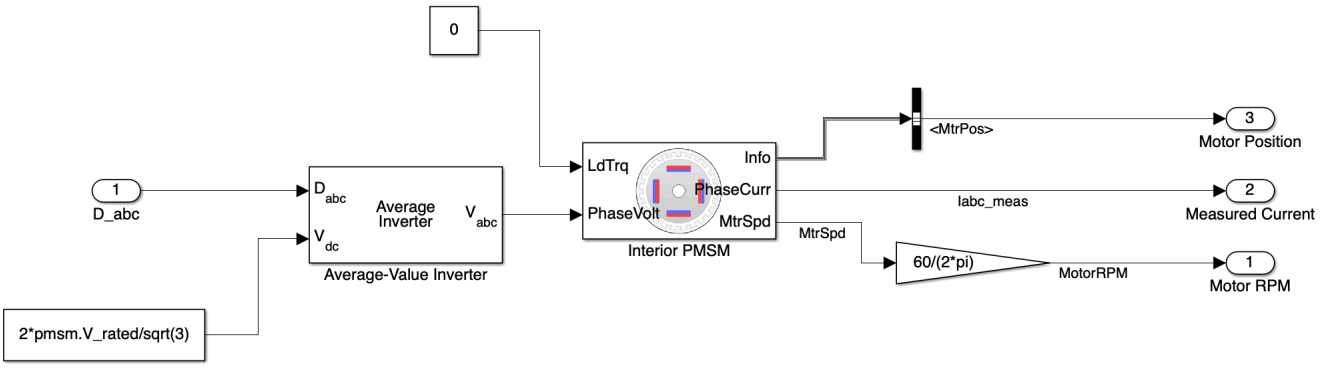
$$V_q = \underbrace{R_s i_q}_{\text{Resistive Drop}} + \underbrace{L_q \frac{di_q}{dt}}_{\text{Inductive Voltage}} + \underbrace{\omega_e L_d i_d}_{\text{Cross-Coupling EMF}} + \underbrace{\omega_e \psi_f}_{\text{Permanent Magnet Back EMF}}$$

This mathematical model transforms the complex, time-varying AC voltages and currents in the motor's stator into two equivalent DC (or slowly changing) components. This simplification is crucial for designing and implementing high-performance control systems, like Field-Oriented Control (FOC).

The *d-q* frame rotates with the motor's rotor. The d-axis (direct axis) is aligned with the permanent magnet's flux. The q-axis leads the d-axis by 90° and is aligned with the direction that produces torque.

Motor Inverter Plant

Fig 3.3 Motor Inverter Plant Subsystem Under Mask



This model shows a high-fidelity, continuous-time simulation of an open-loop motor-inverter plant. This is a high-level modeling of the electromechanical dynamics of the motor and the power-electronic interface.

The primary choice is the use of an **Average-Value Inverter** block. This deliberately simplifies the power electronics to facilitate rapid simulation and control system development, focusing on the motor's core dynamics rather than the high-frequency switching harmonics of the inverter.

The signal flow and methodology of each component are as follows:

1. System Inputs

The plant model has two primary inputs that define its operating condition:

- **D_{abc} (Duty Cycle Commands):** This is the main control input vector (Port 1). It represents the three-phase (A, B, C) duty cycle commands, which are typically generated by a control algorithm (like Space Vector Modulation, not shown). These signals are the averaged command for one switching period, dictating the desired output voltage.
- **V_{dc} (DC Link Voltage):** This input to the inverter block is provided by a constant block, $\frac{2 * pmsm.V_{rated}}{\sqrt{3}}$. This block parameterizes the simulation, simulating a fixed DC bus voltage from the battery based on the motor's rated voltage variable stored in the workspace.

2. Average-Value Inverter

This block is the core of the power-electronic simulation and represents a significant methodological choice.The average value inverter is a simplified model used in power electronics simulations. It represents the behavior of a power electronic circuit over an averaging period, rather than simulating the fast switching transients. This simplification allows for faster simulation times, especially when the details of individual switching events are not critical to the overall system analysis. It focuses on the average voltage and

current values at the input and output of the inverter, providing a general understanding of its operation.

It converts the DC power from the V_{dc} source into a three-phase AC voltage, V_{abc}, based on the duty cycle commands, D_{abc}.

Instead of simulating the discrete, high-frequency (e.g., 10-20 kHz) switching of the six individual transistors (IGBTs or MOSFETs) in a real inverter, this block calculates the average voltage applied to the motor terminals over each switching period.

This approach intentionally omits the high-frequency voltage ripple (harmonics) associated with Pulse-Width Modulation (PWM). The benefit is a massive increase in simulation speed, as the solver can use much larger time steps. This is ideal for developing and tuning control loops (like FOC) and observing the motor's mechanical and fundamental electrical dynamics, which are much slower than the switching events.

3. Interior PMSM

This block models the physical motor, the core of the electric vehicle's propulsion system. It simulates the complex interplay of electromagnetic principles and mechanical forces, including winding resistance, inductance, back EMF generation and torque production. It also accounts for rotor inertia, friction and thermal dynamics - all of which influence efficiency and performance. The motor model's fidelity is crucial for the vehicle control unit's accuracy, as it directly impacts precise control of speed, acceleration and energy consumption.

It simulates the complete electrical and mechanical behavior of our Permanent Magnet Synchronous Motor (PMSM) for the purpose of this speed control modelling.

Some system properties for the PMSM are represented as thus:

Inputs:

1. V_{abc}: The three-phase average voltages from the inverter.
2. T_{load}: This mechanical input is hard-coded to zero (0) via a constant block. This methodology sets the simulation to run under a no-load condition, allowing for

analysis of the motor's spin-up dynamics, back-EMF generation and no-load current draw.

Internal Model (Methodology):

1. Electrical Dynamics: The block uses the V_{abc} inputs to solve the PMSM's dynamic voltage equations in the *d-q* rotor reference frame. It calculates the resulting d-axis and q-axis currents: I_d and I_q.
2. Torque Generation: It computes the total electromagnetic torque T_{em} from the currents. Because it's an Interior PMSM, this model correctly accounts for both the magnetic torque and the reluctance torque.
3. Mechanical Dynamics: The block solves the rotational equation of motion $T_{em} - T_{load} - B\omega = J\frac{d\omega}{dt}$ to determine the rotor's acceleration, angular velocity *MtrSpd*, and position *MtrPos*.

Outputs:

The block bundles all internal states into an Info bus. The Info bus acts like a multiplexer that contains multiples data about the current motor state. The block also outputs two other responses: Phase Current and Motor Speed.

4. System Outputs

The model provides three key outputs, representing the signals that would be available from sensors in a real-world system.

- **Measured Current (Port 2):** This output (I_{abc}) is extracted from the PhaseCurr signal on the Info bus. It represents the three-phase stator currents i_a, i_b and i_c that would be measured by current sensors.
- **Motor Position (Port 3):** This output *MtrPos* is the rotor's electrical angle, extracted from the Info bus. This signal is critical for field-oriented control and would typically come from an encoder or resolver.
- **Motor RPM (Port 1):** This output is derived from the *MtrSpd* signal (in rad/s). The output is connected to a simple gain block with a K-wise multiplication factor of

60/(2*pi), to perform the unit conversion from radians per second to revolutions per minute (RPM) for easier interpretation.

Duty Cycle Plant

The subsystem shown below in the diagram represents the core of the high-performance Field-Oriented Control (FOC) strategy for the Interior Permanent Magnet Synchronous Motor (IPMSM). This subsystem functions as the inner current control loop, which is the foundational element of the complete cascaded motor control system.

The primary objective of this subsystem is to precisely and independently regulate the motor's stator currents in the rotor-based *d*-*q* reference frame. By controlling these currents, the controller achieves decoupled control over the motor's magnetic flux (via the *d*-axis current, i_d) and its electromagnetic torque (via the *q*-axis current, i_q). This methodology transforms the complex AC-driven PMSM into a system that can be controlled with the simplicity of a separately-excited DC motor, enabling high-dynamic performance.

This subsystem takes its reference commands, I_{dqRef}, from an o uter-loop PI speed controller and uses real-time feedback from the motor plant to generate the final D_{abc} (duty cycle) commands for the power inverter.

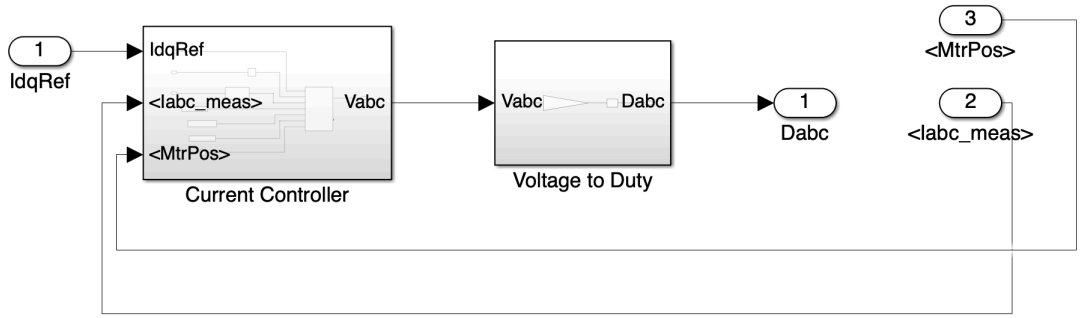


Fig 3.4 Duty Cycle Generator Subsystem

This subsystem is comprised of two principal functional blocks, which execute the FOC algorithm in the appropriate sequence.

1. Inputs

The subsystem requires three inputs to perform its function.

IdqRef: A two-element vector containing the reference (setpoint) values for the *d*-axis and *q*-axis currents, i_{d_ref}, i_{q_ref}. These commands are generated by an outer PI control loop.

$\mathbf{I}_{\mathbf{abc\_meas}}$: A three-element vector containing the measured three-phase stator currents, i_a, i_b, i_c. This signal is the primary feedback for the current-loop, sourced from sensor models in the motor plant.

MtrPos: The measured angle of the motor's rotor θ_e. This signal is critical for performing the Park and Inverse Park transformations, which synchronize the controller's reference frame with the rotor's frame.

2. Current Controller

This block is the computational heart of the FOC implementation. It is responsible for executing the coordinate transformations and the closed-loop current regulation. Its internal methodology follows several key steps:

1. **Clarke Transform (ABC to $\alpha\beta$):** The measured three-phase currents (Iabc_meas) are first transformed from the stationary 3-phase (ABC) frame into a stationary 2-phase orthogonal ($\alpha\beta$) frame.
2. **Park Transform ($\alpha\beta$ to $\mathrm{d-q}$):** Using the measured rotor position MtrPos, the block then rotates the stationary $\alpha\beta$ frame into the synchronous $\mathrm{d-q}$ frame, which is fixed to the rotor. This yields the measured $\mathrm{d-q}$ currents, i_{d_meas}, i_{q_meas}.
3. **Error Calculation:** The measured $\mathrm{d-q}$ currents are compared to the reference $\mathrm{d-q}$ currents (I_{dqRef}) to generate two independent error signals:

$$e_d = i_{d\_ref} - i_{d\_meas}$$

$$e_q = i_{q\_ref} - i_{q\_meas}$$

4. **PI Regulation:** These error signals are fed into two independent Proportional-Integral (PI) controllers, one for the d-axis and one for the q-axis. The PI controllers generate the necessary $\mathrm{d-q}$ voltage commands, v_{d_cmd}, v_{q_cmd}, required to force the current errors toward zero.
5. **Cross-Coupling Decoupling:** A critical step, especially for an IPMSM where $L_d \neq L_q$, is the inclusion of feed-forward decoupling terms. The PI controller outputs are augmented with speed-dependent terms (derived from the motor equations) to compensate for the cross-coupling between the axes. This ensures that a voltage command on the d-axis does not inadvertently affect the q-axis current, and vice-versa, which is essential for true decoupled control.
6. **Inverse Park Transform ($\mathrm{d-q}$ to $\alpha\beta$):** The final, decoupled $\mathrm{d-q}$ voltage commands are transformed back into the stationary $\alpha\beta$ frame using the rotor angle θ_e .
7. **Inverse Clarke Transform ($\alpha\beta$ to ABC):** The $\alpha\beta$ voltage commands are finally transformed into three-phase AC voltage commands, V_{abc}.

Voltage to Duty (Modulator)

This subsystem block represents as the Pulse-Width Modulation (PWM) generator, interfacing the idealized, continuous voltage commands from the Current Controller with the physical, discrete-switching inverter.

1.Input

The three-phase AC voltage commands, Vabc.

2.Output

The block's output, D_{abc}, is a vector of three duty cycle commands (ranging from 0 to 1) that are sent directly to the Average-Value Inverter.

3. System Outputs and Feedback Loop Closure

- D_{abc}: The primary output of the subsystem, representing the duty cycle commands for the inverter's three phases.

- Feedback Signals: The subsystem also passes the measured *MtrPos* and I_{abc_meas} signals through as outputs. This is standard practice in Simulink to make these key plant measurements available to other systems and subsystems.

This subsystem forms a complete, closed-loop $\mathrm{d-q}$ current regulator. It successfully demonstrates the core principles of FOC by transforming the AC motor control problem into a more straightforward DC-like problem, thereby enabling the precise and high-dynamic torque control required for the IPMSM.

Closed Loop Speed Controller

The diagram below shows the implements a complete, closed-loop speed control system for the Interior Permanent Magnet Synchronous Motor (IPMSM) with parameters properly defined above. The control architecture is an FOC control strategy with a P-I controller.

This architecture consists of two main control loops: an inner current loop and an outer speed loop. This cascaded design allows for the independent tuning and robust operation of the system, where the inner loop quickly manages the motor's torque-producing currents, and the outer loop manages the overall mechanical speed.

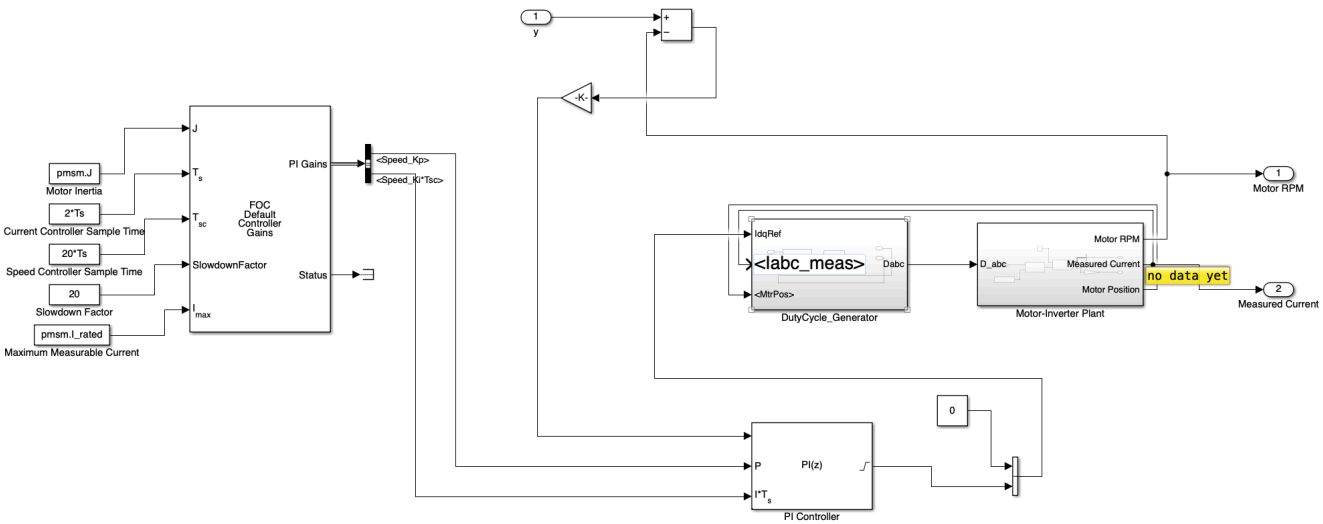


Fig 3.5 Speed Control Plant

Speed Control

The outer loop is responsible for regulating the motor's mechanical speed (Motor RPM) to match a given speed reference (y).

- **Error Calculation:** The primary input to the system is the speed reference y (in RPM). A sum block continuously calculates the speed error by subtracting the measured Motor RPM (fed back from the plant) from this reference.
- **Speed PI Controller:** The resulting speed error signal is fed into a discrete PI Controller. This controller's objective is to drive the speed error to zero. Its output is not a voltage, but rather the torque-producing current command (i_{q_ref}). This command represents the amount of torque the controller believes is necessary to correct the speed error.
- $i_d = 0$ **Control Strategy:** The second component of the current reference, the d-axis current command (i_{d_ref}), is hard-coded to a Constant value of 0. This is a fundamental methodological choice known as the $i_d = 0$ (zero d-axis current) control strategy. For an IPMSM operating below its base (rated) speed, this strategy simplifies the control model and is often used to maximize torque-per-ampere (MTPA) or to simply prioritize torque control via the q-axis.
- **Reference Generation:** A mux block combines the i_{d_ref} (0) and the i_{q_ref} (from the PI controller) into a single two-element vector, *IdqRef*. This vector serves as the setpoint for the inner current loop.

Current Control (FOC)

The inner loop, encapsulated within the ***DutyCycle_Generator*** subsystem, is a complete Field-Oriented Control (FOC) current regulator. Its function is to force the motor's actual $\mathrm{d-q}$ currents to match the I_{dqRef} commands provided by the outer loop.

1. Inputs

This subsystem takes three inputs:

a. I_{dqRef}: The desired i_d, i_q current commands from the outer loop.
b. I_{abc_meas}: The measured three-phase stator currents from the plant.
c. MtrPos: The measured rotor angle from the plant.

2. FOC Strategy

As detailed previously, this block performs the complete FOC algorithm:

a. **Transforms (Clarke & Park):** It uses the MtrPos to transform the measured three-phase currents, I_{abc_meas} into the rotor's $\mathrm{d-q}$ reference frame.
b. **Current Regulation:** It employs two independent PI controllers (one for d-axis, one for q-axis) to calculate the error between the reference IdqRef and the measured $\mathrm{d-q}$ currents. These PI controllers generate the necessary $\mathrm{d-q}$ voltage commands.
c. **Inverse Transforms (Inverse Park):** It transforms the $\mathrm{d-q}$ voltage commands back into three-phase duty cycles (Dabc) via an Inverse Park transform.

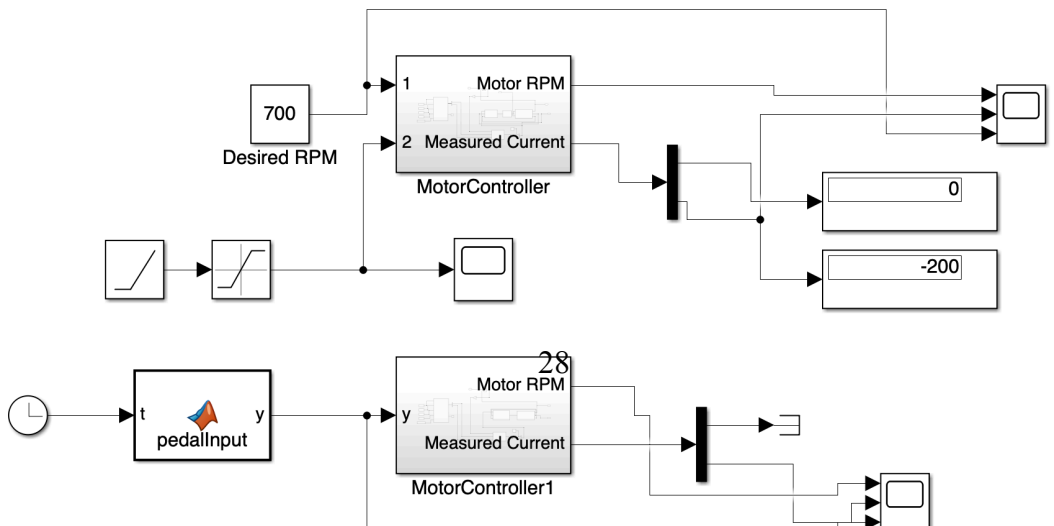
3. Output

The final output is the D_{abc} vector, containing the three-phase duty cycle commands for the inverter.

MotorColler Plant

This Simulink model implements a comprehensive test harness designed to validate the performance of a closed-loop MotorController system under two distinct and critical operating scenarios.

Fig 3.6 Motor Controller Plant Scenario



The primary methodology is to run two independent, parallel simulations:

Scenario 1 - Steady State Test

This is a steady-state test to analyze the controller's accuracy and stability in response to a constant speed command.

This subsystem is a self-contained model that takes a single input—the Desired Speed Reference—and simulates the entire closed-loop response, outputting the ***Motor RPM*** and ***Measured Current*** as results.

To verify the controller's ability to acquire and maintain a fixed speed, and to analyze its steady-state characteristics, we input a constant block provides a step input of 700RPM to the motor controller system. This simulates a driver setting the cruise control to 700RPM.

The MotorController subsystem then takes this 700RPM setpoint input and runs the FOC algorithm.

The ***Measured Current*** output, which contains both i_d and i_q from the FOC, is demultiplexed into two separate outports.

Scenario 2 - Pedal Input:

A dynamic Test to analyze the controller's transient response and tracking performance against a time-varying speed profile. This simulation is used to verify the controller's transient response and its ability to accurately track a time-varying speed command, simulating real-world driving.

A Clock block provides the simulation time (t) to a ***pedalInput*** MATLAB function. This block generates a dynamic speed profile ***y***.

This ***pedalInput*** block models a driver's foot on the accelerator. It generates a signal that likely ramps up, holds, and ramps down.

The MotorController1 instance receives this varying speed reference ***y*** and attempts to make the motor's actual RPM follow it in real-time.

3.3 Battery Charging Controller

Battery Charging Controller

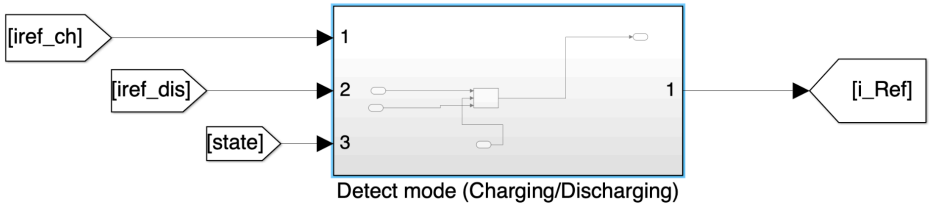


Fig 3.7 Battery mode detector

The subsystem, ***Mode Detector***, functions as a state-based multiplexer. Its primary role is to act as a high-level switch, selecting the correct current reference for the main battery converter's control loop based on the desired mode of operation.

It arbitrates between two distinct current setpoints—one for charging and one for discharging—and, most critically, applies the correct sign convention to the output reference.

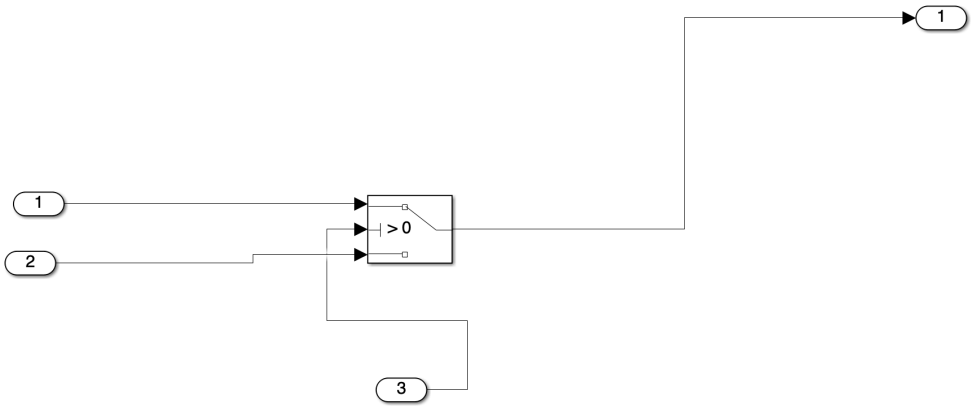


Fig 3.8 Under sub-system mask

Inputs

The subsystem operates based on three distinct inputs:

1. $\mathbf{I_{ref\_ch}}$: This input is the charging current reference and it is a negative-valued signal (e.g., -10*A*) representing the desired current magnitude to be forced into the battery during a charging cycle.

2. $\mathbf{I_{ref\_dis}}$: This input is the discharging current reference also a positive-valued signal (e.g., +20*A*) representing the desired current magnitude to be drawn from the battery during a discharging cycle.

3. **state**: This is a digital mode control integer signal (1 for Charge, 0 for Discharge) that acts as the primary selector. This signal is generated by a higher-level system supervisory logic that decides when to charge or discharge.

Bi-directional DC DC Converter Plant

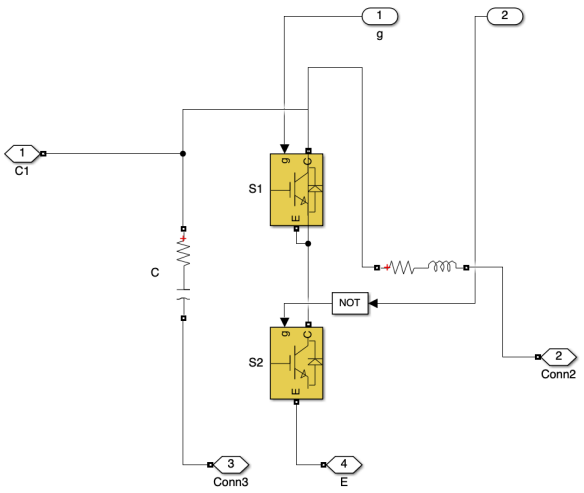


Fig 3.9 DC-DC Buck/Boost Converter

The subsystem models the physical power electronics stage of the battery charging and discharging system. The topology implemented is a classic Non-Inverting Synchronous Buck-Boost Converter, also known as a half-bridge converter.

This block is the power source of the system. It is a high-fidelity plant model that takes a digital control signal and uses it to control the flow of large amounts of current in two directions:

1. **Buck Mode:** From a high-voltage DC bus into the low-voltage battery.
2. **Boost Mode:** From the low-voltage battery onto the high-voltage DC bus for driving motor load and other systems.

1. Analysis of Circuit Topology and Components

1. **Switching Legs:** The core of the converter is a half-bridge composed of two IGBTs (Insulated Gate Bipolar Transistors), S1 (high-side) and S2 (low-side). These are the active switches that control the power flow. The anti-parallel diodes are essential for providing a path for current during switch-off transitions (freewheeling).
2. **Inductor (L):** The inductor (shown with its series resistance) is the primary energy storage element. It stores energy from one side in its magnetic field and then releases that energy to the other side (the battery to DC BUS or the DC bus to battery).
3. **Filter Capacitor (C):** The capacitor (shown with its series resistance, ESR) is on the low-voltage side. It serves to filter the pulsating current from the inductor, smoothing the voltage at the battery terminals and providing a stable DC voltage source/sink.

2. System Interface

1. **Control Input, g:** This is the single PWM (Pulse Width Modulation) gate signal input. This digital signal (a 1 or 0) directly controls the state of the entire converter. The duty cycle (the percentage of time this signal is high) determines the ratio of voltage transfer and thus the direction and magnitude of the current flow.

2. **C1 Port 1:** This is the Low-Voltage (LV) Port, connected to the capacitor C. This port is designed to be connected directly to the battery model.
3. **Conn2 Port 2:** This is the High-Voltage (HV) Port, connected to the inductor L. This port is designed to be connected to the main DC bus of the larger system.
4. **Conn3 Port 3 and E Port 4:** These ports represent the Common Ground or negative rail of the power circuit, to which both the battery and the main DC bus are referenced.

3. Control Logic

The control logic is simple but highly effective: The g signal is fed directly to the gate of the high-side switch S1. The g signal is also fed into a NOT (Logical Operator) block. The output of the NOT block is fed to the gate of the low-side switch S2.

This implements complementary, synchronous switching: when g is HIGH (e.g., 1): S1 is commanded ON, and S2 is commanded OFF and when g is LOW (e.g., 0): S1 is commanded OFF, and S2 is commanded ON.

This synchronous operation (where an active switch S2 is used for the freewheeling path instead of just a diode) dramatically reduces conduction losses and is the key to enabling efficient, high-power bidirectional flow.

In a real-world implementation, a "dead-time" generator would be inserted between the g signal and the NOT block to prevent S1 and S2 from ever being on at the same time, which would cause a "shoot-through" short circuit. Simscape's blocks may simulate this or assume an ideal, instantaneous switch.

4. Bidirectional Modes of Operation

The converter's function is determined by the duty cycle (D) of the g signal and the direction of the current flow commanded by the controller.

1. **Charging (Buck Operation)**
    a. **Goal:** Move power from the HV Port (Conn2) to the LV Port (C1).

b. **Action:** The controller sets a duty cycle D such that $V_{C1} \approx D \times V_{Conn2}$.

c. **g = HIGH (**$S1$ **ON,** $S2$ **OFF):** Power flows from Conn2, through S1, through the inductor L, and into the battery at C1. The inductor current ramps up.

d. **g = LOW (**$S1$ **OFF,** $S2$ **ON):** The inductor's magnetic field collapses, and its current continues to flow in the same direction, "freewheeling" through the low-side switch S2 and into the battery. The inductor current ramps down.

2. **Discharging (Boost Operation)**

a. **Goal:** Move power from the LV Port (C1) to the HV Port (Conn2).

b. **Action:** The controller sets a duty cycle D such that $V_{Conn2} \approx V_{C1}/(1-D)$.

c. **g = LOW (**$S1$ **OFF,** $S2$ **ON):** The battery at C1 is connected across the inductor L (via S2 and ground). The inductor current ramps up, storing energy from the battery.

d. **g = HIGH (**$S1$ **ON,** $S2$ **OFF):** The inductor is now in series with the battery. The inductor's collapsing field (plus the battery voltage) creates a voltage higher than Conn2, forcing current to flow out of the inductor, through S1, and into the HV DC bus at Conn2.

This subsystem is a high-fidelity Simscape Electrical model of the physical power stage. It translates a single PWM duty cycle signal into a complex, bidirectional power flow, enabling the system to function as both a battery charger and a power exporter.

I_{ref_ch} Generation Subsystem

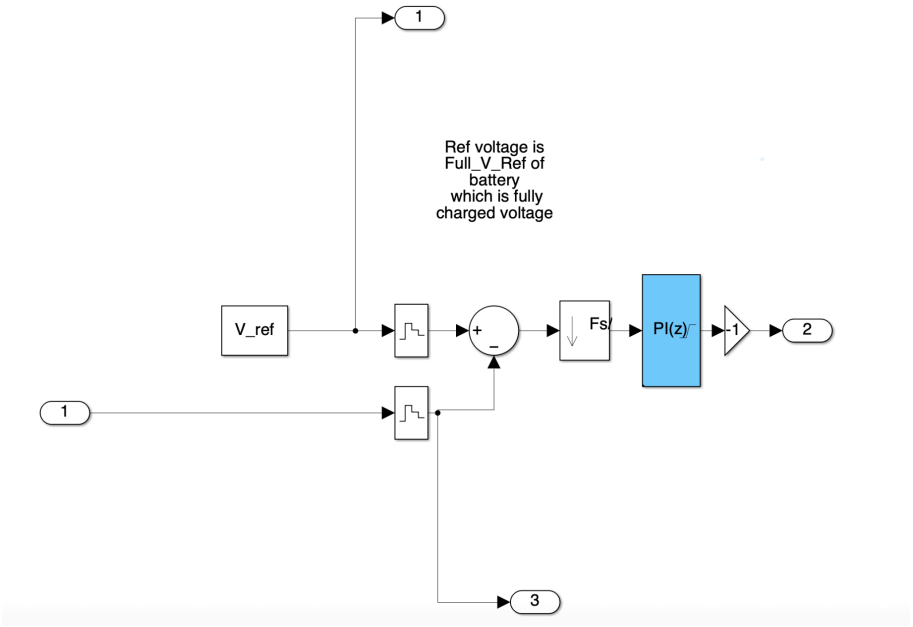


Fig 3.10 I_{ref_ch} Generator

Constant-Voltage (CV) Charging Controller

This subsystem implements the Constant-Voltage (CV) charging phase, which is a critical part of a complete battery charging algorithm.

The subsystem's function is an outer voltage control loop. Its primary goal is not to control current directly, but to calculate the correct charging current reference I_Ref that the inner current loop (in the main converter) must follow to hold the battery's terminal voltage at a precise, fixed setpoint.

1. System Inputs

1. **V_{meas}:** This input is the measured battery terminal voltage. It is the primary feedback signal for this control loop.
2. **V_{ref}:** This block defines the constant voltage setpoint. This is the target voltage that the controller will attempt to maintain at the battery terminals during the CV phase.
3. **$Full_{VRef}$:** This value represents the battery's absolute maximum or fully charged voltage. The V_{Ref} setpoint is intentionally set lower than this maximum to ensure a safe and stable charging taper.

2. Signal Flow

This subsystem is a classic discrete-time PI (Proportional-Integral) feedback controller.

1. **Discretization:** Both the V_meas (Port 1) and the V_Ref (Constant) signals are first passed through Sample and Hold blocks. This is a critical step for digital control, ensuring that the signals being compared are synchronized to the controller's discrete sample time.
2. **Rate Transition (Fs):** The voltage error signal is passed through a Rate Transition block. This is a deliberate design choice indicating that this voltage loop runs at a slower sample rate (Fs) than the faster, main system (which is typically the inner current loop). This is standard practice, as battery voltage changes much more slowly than electrical current.
3. **Error Calculation:** The sampled V_meas and V_Ref are fed into a Sum block to calculate the voltage error. Based on the system's final configuration (specifically the -1 gain), the error is logically calculated as:
   $V_error = V_meas - V_Ref$
4. **PI Control:** The error signal is fed into a discrete PI Controller (PI(z)). This controller's job is to drive the V_error to zero.
   a. Logic: When the charging begins, V_meas is less than V_Ref
   b. This creates a negative V_error (e.g., 24 - 25.955 = -1.955V).
   c. The PI controller's integral action will accumulate this negative error, and its output will become a negative value. The magnitude of this negative output represents the amount of charging current required.
5. **Sign Inversion:** The negative-valued output from the PI(z) block is immediately fed into a Gain block with a value of -1.
   a. This inverts the signal: $(NegativePIOutput) * (-1) = PositiveI_Ref$.
   b. This is the key to the subsystem's logic: It translates the negative error signal into a positive charging current reference, which is the standard convention for charging (current into the battery).

6. **Tapering Operation:** As the PI controller successfully forces a positive current into the battery, its measured voltage V_meas will rise.
    a. As V_meas approaches V_Ref, the $V_error(V_meas - V_Ref)$ approaches zero.
    b. As the V_error approaches zero, the PI controller's output also approaches zero.
    c. This means the final I_{Ref} will naturally and smoothly taper off toward zero Amps. This is the correct and expected behavior of a constant-voltage charger as the battery becomes full.

3. System Outputs

- I_{Ref}: The primary output. This is the dynamically calculated, positive charging current reference that is fed to the inner current loop of the DC-DC converter.
- **Passthrough:** These outputs are likely used for monitoring, logging, or debugging, providing a sampled "snapshot" of the V_{Ref} and V_{meas} signals used by the controller in each step.

I_{ref_dis} Generation Subsystem: Discharging Load-Following Controller

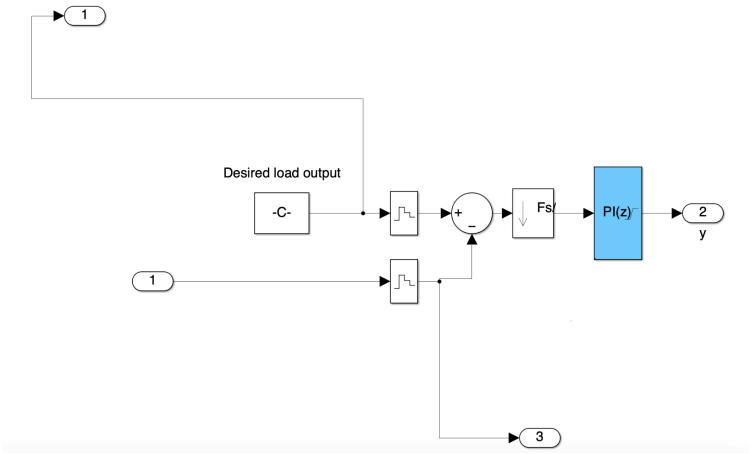


Fig 3.11 I_{ref_dis} Generator

This subsystem implements an outer voltage control loop designed to regulate the main DC bus voltage during a discharging cycle. Its primary function is to maintain a stable voltage supply for the rest of the system (the "load") by drawing the necessary amount of power from the battery.

The subsystem's methodology is to compare the actual DC bus voltage against a desired setpoint. It then calculates the precise discharging current magnitude (I_{ref_dis}) that the inner current loop must pull from the battery to correct any voltage error. This is the core logic for a voltage-source converter operating in boost mode.

1. System Inputs

1. $V_{bus-meas}$**:** This input is the measured voltage of the high-voltage DC bus (i.e., the load side). This is the primary feedback signal for this control loop.
2. **Desired Load Output:** This block defines the DC bus voltage setpoint. It is a Constant block, meaning the controller's objective is to hold the load voltage at this

fixed value, regardless of how much power the load draws (within the battery's capacity).

2. Signal Flow

This subsystem is a classic discrete-time PI (Proportional-Integral) feedback controller.

1. **Discretization:** Both the $V_{bus-meas}$ and the $V_{bus-ref}$ (from the constant block) are passed through Sample and Hold blocks. This is a critical step to synchronize these two signals to the controller's specific sample time, ensuring a stable comparison for the error calculation.
2. **Rate Transition (Fs/):** The resulting error signal is passed through a Rate Transition block. This explicitly indicates that this outer voltage loop runs at a slower sample rate (Fs/) than the inner current loop. This is a standard and robust design practice, as the DC bus voltage (supported by large capacitors) changes much more slowly than the inductor current.
3. **Error Calculation:** The sampled $V_{bus-meas}$ is subtracted from the $V_{bus-ref}$ setpoint: $V_{error} = V_{bus-ref} - V_{bus-meas}$
4. **PI Controller Block:** The error signal is fed into a discrete PI Controller (PI(z)). This controller integrates the error over time and applies a proportional gain to drive the V_{error} to zero.
    a. Logic in Action:
        i. If a large load is connected, the DC bus voltage (V_{bus_meas}) will begin to sag.
        ii. This creates a positive V_{error}.
        iii. The PI controller's output y will become a large positive value, demanding more current from the battery.
        iv. Conversely, if the load is removed, V_{bus_meas} will start to rise. This creates a negative V_{error}, causing the PI controller's output y to decrease, reducing the commanded current.

3. System Outputs

1. **I_{ref_dis} (y):** This is the primary output of the controller. This signal, y, represents the positive magnitude of the discharging current reference. This value is then sent to the ***Mode Detector*** subsystem (as I_{ref_dis}), where it is subsequently inverted to negative current to correctly command the main converter to draw current from the battery.

2. **Passthrough:** This provides the sampled battery voltage ($V_{batt-meas}$) for use by other supervisory or safety functions in the system.

This subsystem is a load voltage regulator. It ensures a stable DC bus for the rest of the application by dynamically adjusting the amount of discharge current it requests from the battery, thereby compensating for any changes in the load.

Current Control Loop

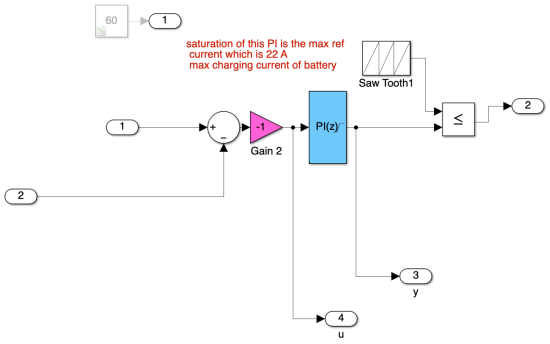


Fig 3.12 Current Control Loop Diagram for Charging/Discharging

This subsystem is the current control loop of the battery charging system. It is the fastest and most critical control loop in the entire architecture.

Its primary function is to take the high-level signed current reference I_{ref} provided by the supervisory ***Mode Detector*** subsystem and forces the actual battery current I_{meas} to match this reference precisely.

To achieve this, the subsystem uses a discrete-time PI Controller that calculates the required control effort for the current but also includes the Pulse-Width Modulation from the saw-tooth generator, creating the final digital gate signal for the power converter.

1. System Inputs

1. I_{ref}: This is the signed current setpoint. This signal is received from the ***Mode Detect*** subsystem and is positive for charging and negative for discharging.
2. I_{meas}: This is the measured battery current. This is the primary feedback signal for this high-speed loop, typically coming from a shunt resistor or any other sensor for detecting the current of the battery.

2. Signal Flow

The core of the system implements the current regulation and signal generation with a streamlined flow.

1) **Sum Block:** The subsystem first calculates the error between the setpoint or desired current value and the measured value:

$$error(t) = I_{ref} - I_{meas}$$

2) **Gain Block:** The calculated error is immediately inverted by a Gain block with a value of -1.

$PI_{input} = (I_{ref} - I_{meas}) * -1$, which simplifies to

$$PI_{input} = I_{meas} - I_{ref}$$

This inversion is a deliberate and common design choice. It ensures the PI controller provides the negative feedback. For example, if the measured current is higher than the reference ($I_{meas} > I_{ref}$), a positive error is fed to the PI controller. The controller's positive response then reduces the duty cycle, which in turn reduces the current, bringing it back to the setpoint.

3) **PI Controller:** This discrete PI controller is the brain of the loop.
    a) It receives the PI_{input} and works to drive it to zero (make $I_{meas} = I_{ref}$).
    b) The output of this controller (y) is not a current, but a control signal that represents the required duty cycle for the PWM.
4) **Saturation Block:** As the text annotation notes, this PI controller has its output saturated (clamped).
    a) This is a critical anti-windup feature. The output y is limited to a valid range.
    b) This prevents the PI controller's integral term from winding up to an impossibly large value if the current cannot reach its setpoint (e.g. during a short circuit or if the requested current I_{ref} is physically impossible). This ensures the controller can recover quickly when the error is resolved. The ***Max_Ch_Curr*** is the physical limit that this saturation enforces.
5) **Saw Tooth Pulse Generator:** This block generates a sawtooth carrier wave. This wave oscillates at the converter's switching frequency and typically has a normalized amplitude.
6) **Less than Operator Block:** This relational operator block is the signal generator. It compares the analog control signal y from the PI controller to the high-frequency Saw Tooth1 carrier wave.

```
IF (y <= Sawtooth_Value) THEN Output = 1 (True)
ELSE Output = 0 (False)
```

This simple comparison generates a digital pulse train where the pulse width (duty cycle) is directly proportional to the control signal y.

3. System Outputs

1. **g:** The primary output of the subsystem. This is the final digital gate signal that is sent directly to the ***Bidirectional DC-DC Converter*** to drive the S1 and S2 IGBT switches.
2. **y:** A passthrough of the PI controller's output (the pre-PWM control signal). This is invaluable for debugging and tuning the loop.

3. **u:** A passthrough of the error signal PI_{input} being fed into the PI controller. This is also used for monitoring loop performance.

Battery Controller System Model

This model implements a complete, closed-loop power management system for an EV battery. The core methodology is a state-based, cascaded control architecture for charging and discharging with a . This design allows the system to intelligently switch between two distinct operational modes—***charging*** and ***discharging***—by activating different outer control loops while sharing a single, fast-acting inner current loop.

The physical plant, encapsulated in the Bidirectional DC-DC Converter subsystem, is modeled as a synchronous, non-inverting buck-boost converter, commonly known as a half-bridge. This topology consists of two IGBTs (S1, S2) in a complementary configuration, an energy transfer inductor (L), and a low-side filter capacitor (C). The converter is controlled by a single gate signal (g), which drives S1 directly and S2 via a logical NOT gate. This synchronous switching enables efficient, bidirectional power flow: it operates in buck mode to charge the low-voltage battery from the high-voltage bus and in boost mode to discharge the battery and supply the high-voltage bus.

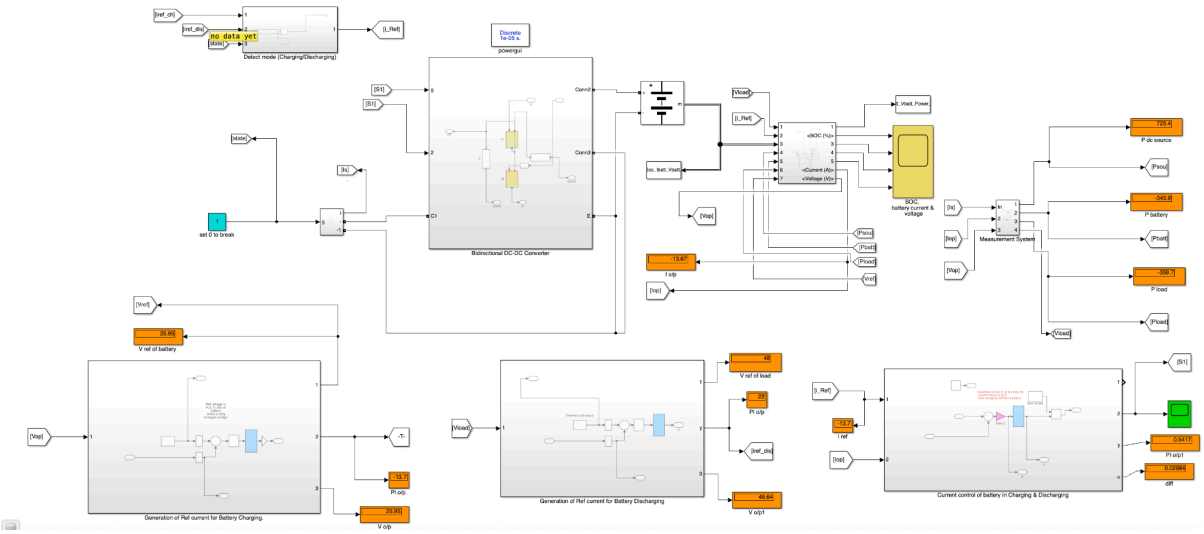


Fig 3.13 Battery Charging Controller Model

The core of the control structure is the ***Current Control Loop*** subsystem, which functions as a discrete-time PI-based Average Current Mode Controller. Its sole objective is to force the measured battery current to precisely track the signed reference current I_{ref}. This loop's methodology is to calculate the error between the reference and measured currents. This error is then inverted by a Gain block that creates a feedback loop. This inverted error is fed to a discrete PI controller, whose output is saturated to a maximum value (***Max_Ch_Curr***) to provide anti-windup and enforce hardware limits. This subsystem also integrates the PWM signal generator, where the PI controller's saturated output (y) is compared to a Saw Tooth carrier wave via a relational operator (<=) to generate the final gate signal (g) for the plant.

The ***Mode Detector*** subsystem provides the supervisory logic, acting as a state-based multiplexer. Based on an external state command, it arbitrates between the two outer loops. If the "charge" is passed, it selects the I_{ref-ch} signal. If the "discharge" is selected, it selects the $I_{ref-dis}$ signal and passes it through an internal gain block, which applies the correct negative sign convention for discharging.

3.4 AUTOSAR Model Definition and Code Generation

The following methodology outlines the systematic process for transitioning the model, MotorController, into a production-intent, AUTOSAR-compliant C Software SWC, after validation. This process utilizes the AUTOSAR Component Designer and Embedded Coder on the Classic Platform to generate the component's descriptive (.arxml) and behavioral (.c/.h) files.

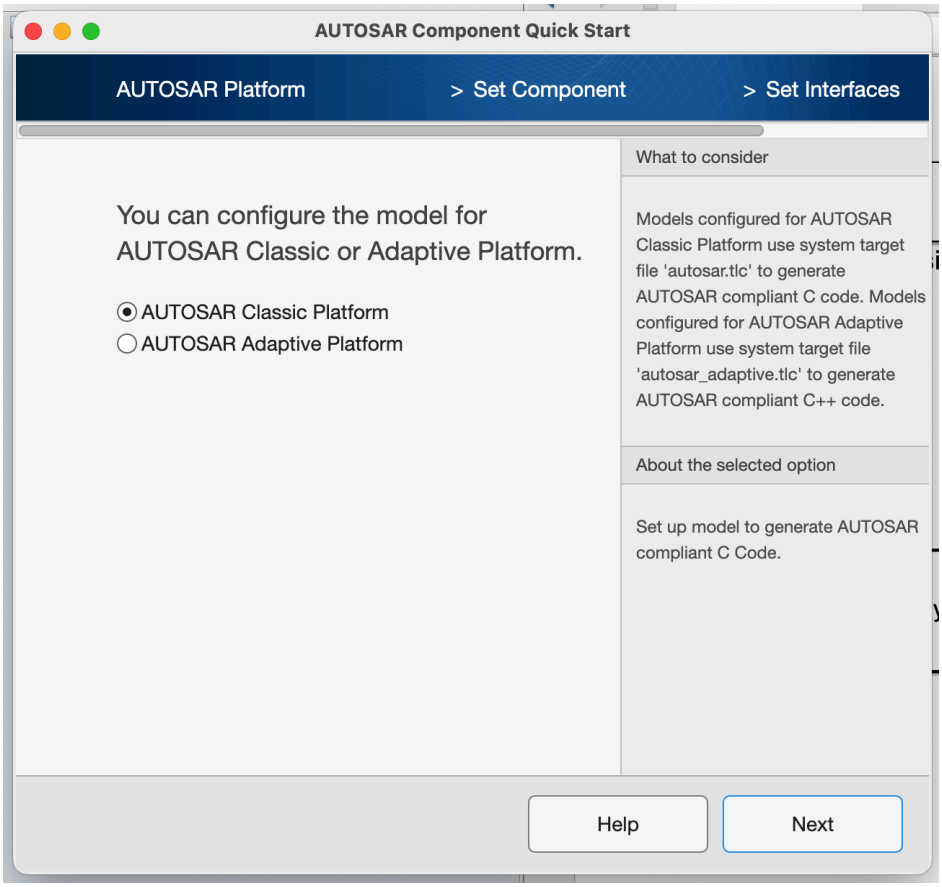


Fig 3.14 AutoSAR Classic Component Quick Start

Code Gen Methodology

The AUTOSAR Component Designer is used to graphically or programmatically configure the SWCs.

1. The Inports and Outports in the Simulink Blocks are mapped to the AUTOSAR ports.
2. The communication between the two loops (I_{dqRef}) is configured as an Inter-Runnable Variable.
3. The model's code generation settings are configured for autosar.tlc, with the language set to C (AUTOSAR-compliant).
4. Upon building the model, Embedded Coder generates the ***.c** and ***.h** files containing the C class implementation of the SWC, along with the ***.arxml** component

description file. This arxml file is then used by a system integrator to connect the SWC to the rest of the AUTOSAR stack.

CHAPTER 4 — RESULTS AND DISCUSSION

4.1 Introduction

This chapter presents the simulation results for the two primary systems developed in this work: the Field-Oriented Control (FOC) motor controller and the bidirectional battery charging system. The results are generated and validated entirely within the Simulink environment.

The findings for each system will be presented, followed by a detailed discussion and analysis. This discussion will correlate the observed behavior with the control methodologies detailed in Chapter 3, verifying the system's performance against its design objectives and highlighting key operational characteristics.

4.2 Motor Control

Scenario 1: Cruise Speed Test (No Load)

The primary objective of this test is to validate the FOC motor controller's ability to acquire and maintain a constant speed setpoint under no-load conditions. This scenario evaluates the controller's transient response to a step input, its steady-state accuracy, and the correct implementation of the $i_d = 0$ control strategy.

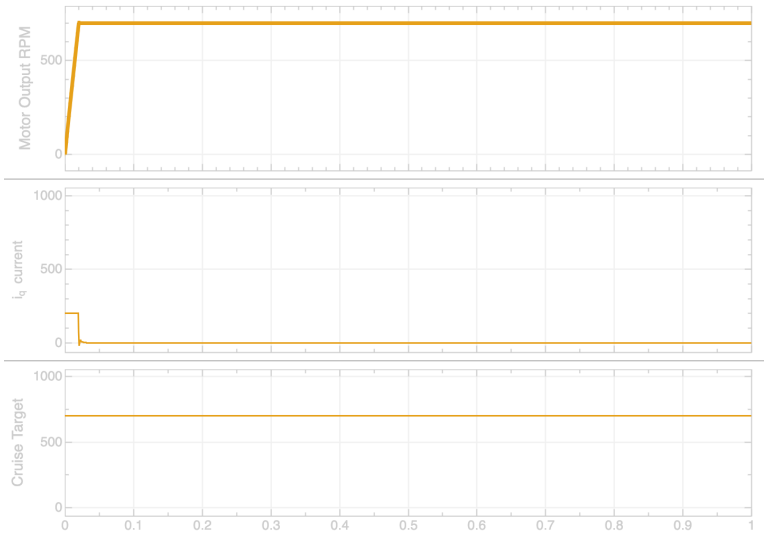


Fig 4.1 Motor response under no load conditions

Results:

The system's response to a 700 RPM step command as presented in Figure 4.1.

1. **Motor Output RPM:** The motor speed exhibits a rapid, smooth acceleration, achieving the 700 RPM setpoint in approximately 0.05 seconds. A minor, well-damped overshoot is observed, after which the controller quickly settles the speed precisely at the 700 RPM target within 0.1 seconds. The controller successfully maintains this speed with zero steady-state error for the remainder of the 1-second simulation.
2. i_q **current:** The q-axis current, which is directly proportional to torque, shows an initial peak corresponding to the high torque demand during acceleration. As the motor speed reaches the setpoint and the speed error approaches zero, the i_q current drops immediately to a negligible, non-zero value. This residual current represents the minimal torque required to overcome the motor's internal friction and windage at no-load.

The results from the cruise speed test strongly validate the cascaded PI control methodology. The top plot demonstrates the outer speed loop's effectiveness. The fast rise time and minimal overshoot indicate a stable and well-tuned response. The zero steady-state error is a direct and expected outcome of the integral action within the speed PI controller, confirming its ability to eliminate persistent errors.

The i_q current plot provides a clear visualization of the controller's logic. The initial current spike is the outer speed loop commanding maximum torque request from the inner loop to correct the large speed error. The moment this error is nullified, the outer loop correctly reduces its command to the minimal i_q current necessary to maintain speed against no-load friction. This behavior confirms the controller's high precision.

Scenario 2: Cruise Speed Test (Linearly Increasing Load)

This test is designed to evaluate the controller's robustness and dynamic load rejection capabilities. The system is subjected to a constant speed command while a linearly increasing

external load torque is applied. The objective is to verify the controller's ability to maintain the speed setpoint by actively counteracting this dynamic disturbance.

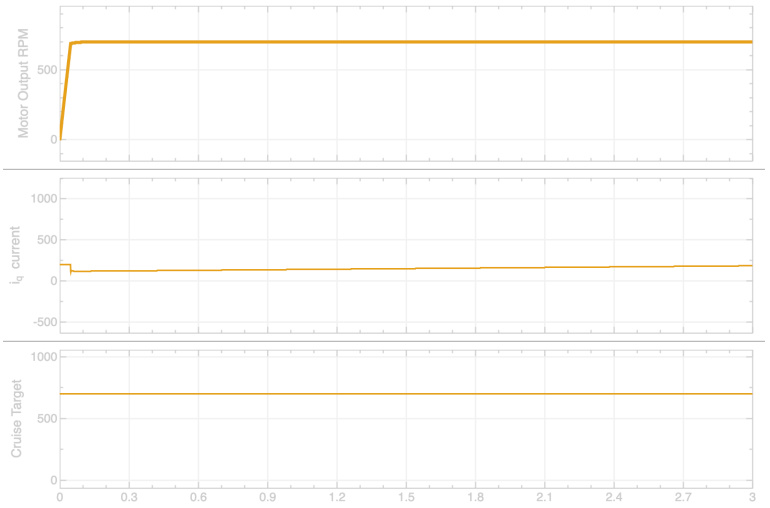


Fig 4.2 Motor response under linearly increasing load

Results:

The system's response to a 700 RPM Cruise Target command over a 3-second simulation is presented in the figure.

1. **Cruise Target:** The speed setpoint is held at a constant 700 RPM for the entire simulation.
2. **Motor Output RPM:** The motor speed accelerates to the 700 RPM setpoint in approximately 0.15 seconds. After this initial transient, the controller maintains the speed precisely at the 700 RPM target with zero perceptible steady-state error for the entire 3-second duration.
3. i_q **current:** The q-axis current exhibits an initial positive spike, corresponding to the torque required for acceleration. After the motor reaches the target speed (at $t = 0.15_s$), the i_q current settles to a low, near-zero value. From this point, it demonstrates a clear and steady linear increase for the remainder of the simulation, rising from near 0A to a higher positive value by $t = 3.0_s$.

Scenario 3 - Driver Simulated Input

This test evaluates the controller's dynamic tracking performance. Unlike the previous tests, this scenario provides a time-varying speed reference, simulating a driver's "pedal input" for acceleration, cruising, and deceleration. The objective is to verify the controller's tracking fidelity and transient response to a complex, multi-stage command profile.

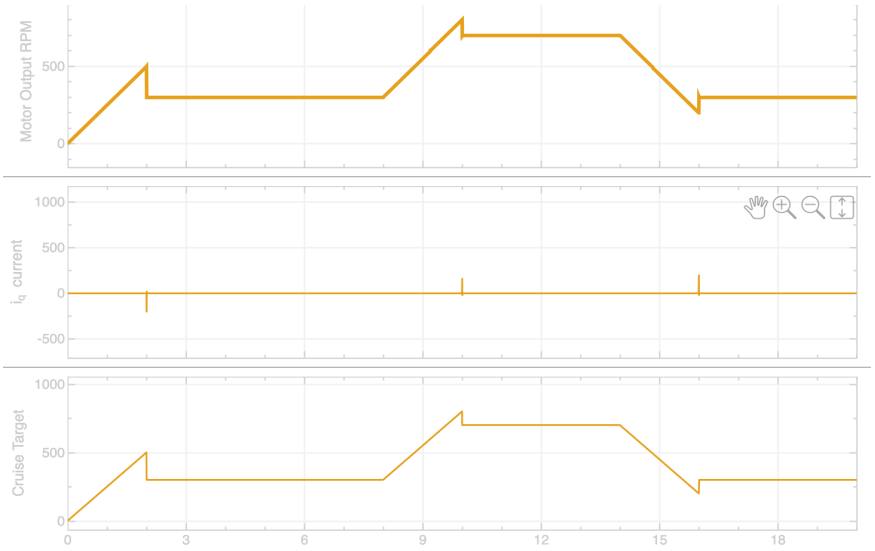


Fig 4.3 Motor response to simulated pedal input

Results:

The simulation results are presented in the figure, showing the system's response to the ***pedalInput*** profile.

1. **Cruise Target:** This plot shows the reference speed command, which serves as the "pedal input." It consists of several distinct phases: an initial acceleration ramp (0-2s), a constant-speed cruise (2-8.5s), a second acceleration ramp (8.5-10s), a second, faster cruise (10-14.5s), a deceleration ramp (14.5-16s), and a final, slower cruise (16s onward).

2. **Motor Output RPM:** The actual motor speed demonstrates exceptionally high-fidelity tracking of the Cruise Target. The RPM output trace follows the commanded ramps and constant-speed segments almost perfectly, with minimal to no perceptible error or lag.

3. i_q **current:** The q-axis current provides a clear insight into the controller's effort.

    a. During both acceleration ramps, the i_q current is a constant positive value, indicating a constant positive torque is being applied.

    b. During both constant-speed cruise segments, the i_q current drops to near-zero, reflecting the no-load condition where only minimal torque is needed to overcome friction.

    c. During the deceleration ramp, the i_q current becomes a constant negative value.

    d. At each transition in the command (e.g., at $t = 2_s, 8.5_s, 10_s$), small, sharp transient spikes are visible in the i_q current.

This test provides a comprehensive validation of the cascaded FOC controller's dynamic performance. The near-perfect tracking shown in the top plot confirms that the outer speed loop is accurately calculating the required torque command (I_{dqRef}) at every time step, and the inner current loop is successfully executing that command.

The i_q current plot is particularly interesting. The controller applies positive torque (positive i_q) for acceleration and near-zero torque for cruising. Noticeably, during the deceleration phase (14.5-16s), the controller automatically commands a negative torque (negative i_q). This demonstrates the system's inherent capability for regenerative braking, where the controller actively slows the motor to precisely follow the deceleration ramp, rather than simply coasting.

The small transient spikes in i_q at the transitions are the expected, correct behavior of the PI controller. They represent the responses of the proportional and integral action required to

instantaneously change the motor's state of acceleration, further highlighting the controller's high responsiveness.

4.3 Battery Charging/Discharging Controller

Scenario 1: Discharging Mode with Constant Load

This test validates the system's performance in the discharging mode. The objective is to verify that the cascaded controller can successfully regulate the system and supply a constant load, drawing the required power from the battery while maintaining stability.

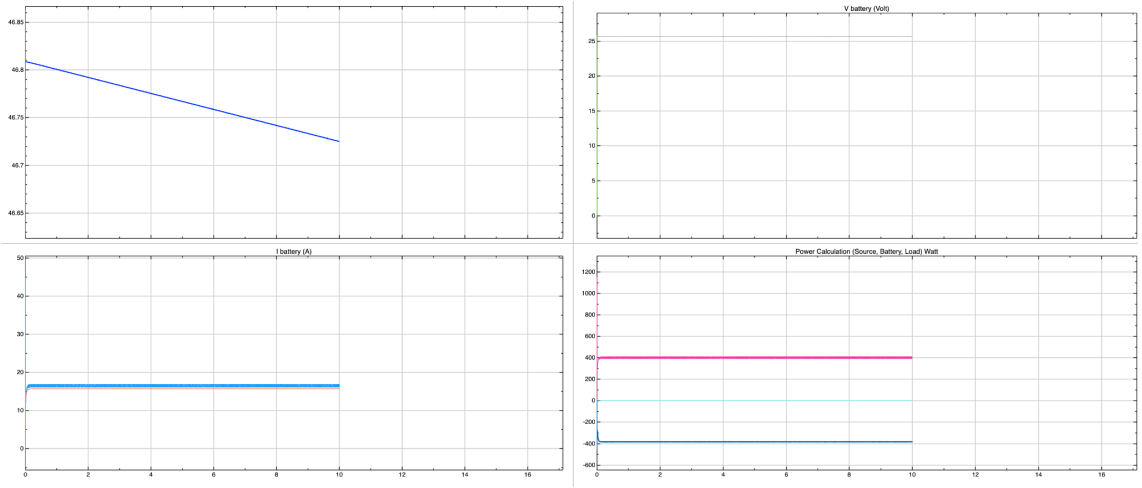


Fig 4.4 Battery Discharging Behaviour

Results:

The system's response over a 10-second simulation in discharging mode is presented in the four plots.

1. **Power Calculation:** This plot provides the clearest overview of the system's state. The P Load oscillates around 380-410W. In response, the P Battery (dark blue line) hovers around -390 and -400 W, indicating the battery is sourcing roughly the exact amount of power consumed by the load. The P Source (light blue line) is at 0 W, confirming the main source is inactive, as expected.
2. **I - Battery:** This plot shows the current drawn from the battery. It is constant at approximately 15 A. Given that the battery power is ~390 W (from the power plot)

and the voltage is 25 V (from the voltage plot), the actual current is -15 A ($I = P/V$). This confirms the plot is showing the magnitude of the discharge current.

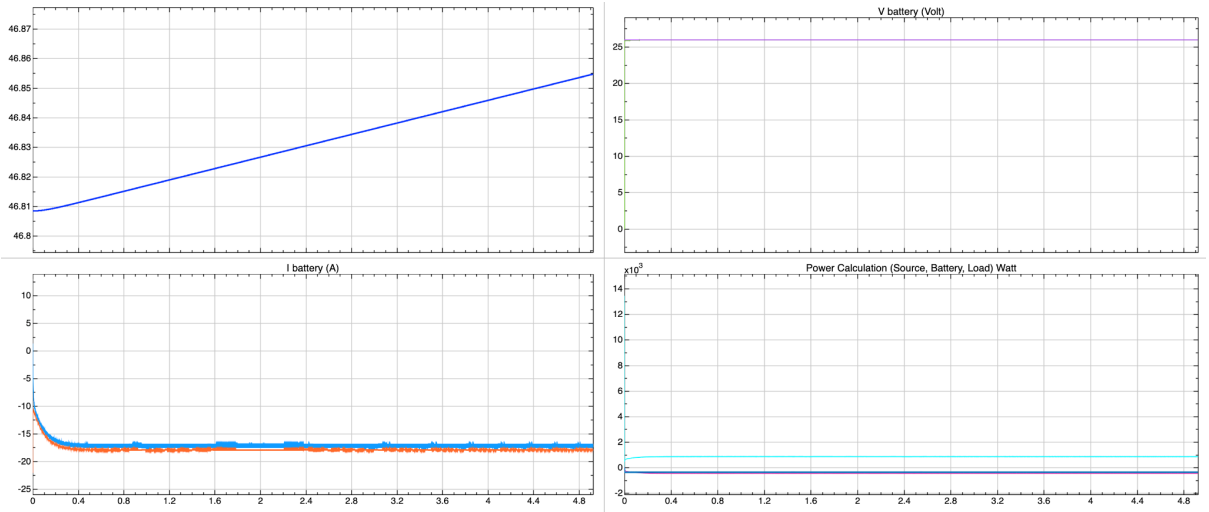
3. **V Battery:** The battery's terminal voltage is shown to be stable at 25 V throughout the 10-second discharge.
4. **SOC:** This plot (y-axis from 46.85% to 46.7%) shows a slow, linear decrease in the battery's State of Charge. This directly corresponds to the constant power (375 W) being drawn, confirming a net energy outflow from the battery.

The stability of all key metrics—constant load power, constant battery power, constant battery current, and constant battery voltage, resulting in a smooth, linear decay in SOC—demonstrates that the cascaded control system is stable and performing precisely as designed. The ***Mode Detector*** subsystem has also correctly applied the negative sign convention to the current reference, enabling the system to draw power from the battery.

Scenario 2: Charging Mode

This test is intended to validate the system's performance in the charging mode. The expected behavior is that the controller will follow a positive current reference (I_{ref_ch}), forcing current into the battery and resulting in a corresponding increase in the battery's State of Charge (SOC).

Fig 4.5 Battery Charging Behaviour



Results

1. **I battery (A):** The inner loop successfully tracks a constant current reference that goes from -10A to approximately -17.83 A. Both the reference (blue) and the measured current (orange) are stable and closely matched.
2. **State of Charge:** The State of Charge shows a steady, linear increase from 46.81% to 46.86%, corresponding to the energy being added to the battery.
3. **V battery:** The battery voltage is stable at 25 V. This is characteristic of the Constant-Current (CC) phase, where the voltage has not yet reached its upper limit.
4. **Power Calculation (Watt):** The P Battery (dark blue line) is stable at a constant negative value representing power being absorbed by the battery.

The simulation results are fully consistent and demonstrate a stable, correctly-operating system. The I battery plot confirms that the inner current loop is precisely tracking its reference of -17.83 A. This negative current, per the passive sign convention, correctly represents the state of charging.

This charging operation is unequivocally confirmed by the other system metrics. The SOC plot's steady, linear increase is the correct physical response to a constant charging current. The stability of the V battery plot is also correct for the CC charging phase.

Finally, the power calculation plot aligns perfectly. The constant negative P Battery value confirms that the battery is absorbing power at the rate defined by the controller. This test successfully validates that the Generation of ref current for battery Charging (outer loop) is successfully passing a constant reference, the ***Mode Detector*** logic is correctly selecting this reference and the system is properly executing the command.

4.3 AUTOSAR-Compliant Code Generation

This section validates the final stage of the model-based design (MBD) workflow: the automatic generation of production-intent code from the validated MotorController model.

Component Description:

The generated XML files like the implementation **.arxml** included below is the standardized AUTOSAR component description. It contains no executable logic. Instead, it formally defines the component's contract with the outside world. It specifies the ports, data types and runnable entities.

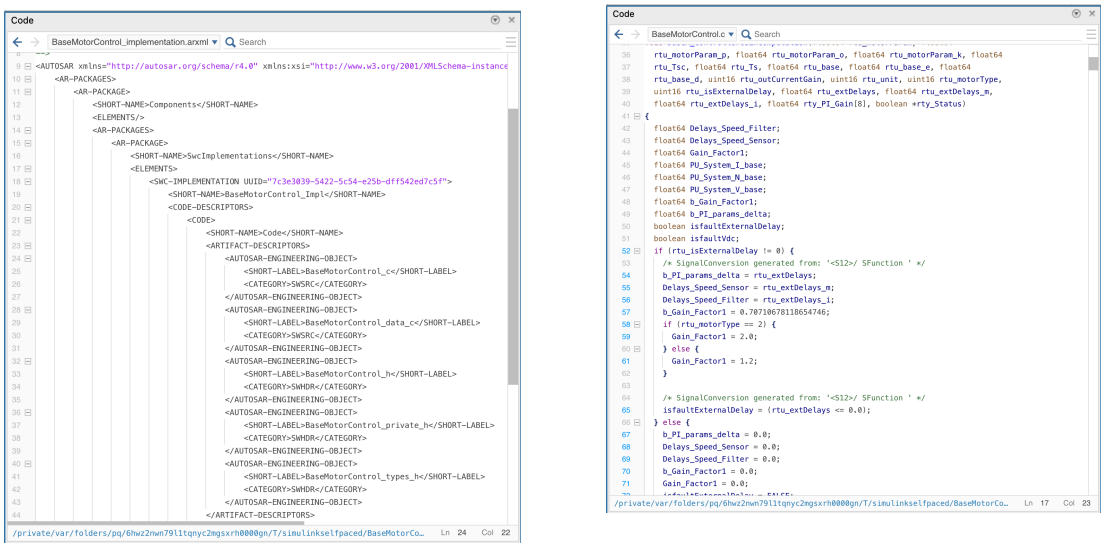


Fig 4.6 AUTOSAR ARXML component and C implementation files

Behavioral Implementation:

These C source and header files contain the actual, platform-independent implementation of the control algorithm. The Simulink model's logic—the cascaded PI loops, FOC mathematics, and multi-rate execution—is translated directly into C class methods. These methods are the concrete implementation of the runnables defined in the .arxml file.

The successful generation of these compliant artifacts provides a clear validation of the MBD and AUTOSAR methodology. The advantages over traditional, hardware-dependent implementation are significant, particularly in development velocity and reliability.

4.4 Summary

This chapter presented the simulation results for the two core EVCU systems approached with Model Based Engineering Design, and the implementation of the FOC Motor Controller with the AutoSAR SWC Architecture. The FOC motor controller was shown to be highly accurate and responsive in both steady-state (cruise) and dynamic (pedal) tracking scenarios. The bidirectional battery controller was also validated in both its charging and discharging modes, with all results aligning with the system's passive sign convention. The findings confirm that both cascaded control systems are stable, robust, and meet their design objectives.

CHAPTER 5 — CONCLUSION

5.1 Objective

This study undertook the design, modeling, and simulation-based validation of a general-purpose Electric Vehicle Control Unit (EVCU). The primary objective was to demonstrate the efficacy of a Model-Based Engineering (MBE) workflow, using MATLAB/Simulink and adhering to AUTOSAR Classic Platform principles, to create and validate a complex, integrated control system entirely within a virtual environment. The research was strictly limited to the conceptual and simulation phase to prove the viability of the control logic before any hardware commitment.

5.2 Summary of Findings

The research successfully resulted in the development and validation of two primary, high-fidelity subsystems, which form the core of the EVCU.

First, a high-performance, cascaded Field-Oriented Control (FOC) system for the propulsion motor was designed. Simulation results confirmed the controller's stability and robustness. It demonstrated high-fidelity tracking in both steady-state (cruise) scenarios, with and without load, and in dynamic (pedal input) scenarios. The controller also correctly implemented the $i_d = 0$ strategy and exhibited the expected regenerative braking (negative i_q current) during deceleration.

Second, a state-based, cascaded bidirectional battery controller was successfully modeled. This system demonstrated stable operation in its two key modes: Constant-Current (CC) charging and constant-power discharging (load regulation). A key finding during this development was the validation of the system's passive sign convention (negative current for charging) and the corresponding negative feedback logic, which was essential for achieving a stable design.

The successful generation of AUTOSAR-compliant C and ARXML artifacts from the validated motor control model confirmed the architectural soundness and portability of the design, bridging the gap from simulation to a production-intent implementation.

5.3 Conclusion and Significance

This study successfully demonstrates that a complex, multi-domain EVCU can be effectively designed, integrated, and verified purely through simulation. The use of an AUTOSAR-compliant, model-based workflow was proven to be highly effective. The successful simulation of both the FOC motor controller and the bidirectional battery controller provides a high-confidence validation of the control algorithms.

The significance of this simulation-only approach lies in its ability to significantly accelerate the development and iteration cycle. It de-risks the entire project by allowing for the identification and correction of fundamental design flaws, such as the feedback logic in the battery controller, at the earliest possible stage, long before any costly hardware is procured or built. The MBD methodology provides a "single source of truth" that directly translates a validated design into deployable code, which is an indispensable capability in modern automotive engineering.

5.4 Limitations and Recommendations for Future Work

This study was, by design, strictly limited to the conceptual and simulation phase. The findings do not include verification from Hardware-in-the-Loop (HIL) or real-world hardware testing. The component models (motor, battery) are simplifications and do not, for example, comprehensively model all thermal dynamics.

Based on the successful results of this study, the following recommendations for future work are made:

1. **HIL Testing and Implementation:** The next logical step is to deploy the generated AUTOSAR C code from the MotorController SW-C onto a target ECU (such as the

TI TMS320F28069) and perform comprehensive HIL testing to validate real-time performance.

2. **Model Expansion:** The EVCU model should be expanded to include the other critical subsystems besides the ones outlined in the initial scope, particularly a comprehensive thermal management system for the battery and motor, as well as vehicle communication network models (e.g. CAN bus).
3. **Advanced Control:** The controllers themselves can be enhanced, for example, by implementing a full CC-CV (Constant-Current Constant-Voltage) charging algorithm for the battery and MTPA (Maximum Torque Per Ampere) for the motor controller.

In conclusion, this study provides a robust and validated simulation framework for a general-purpose EVCU, offering a strong foundation for future hardware development and system expansion.

REFERENCES

1. British Pathé. *Nig — General Gowon Opens Multi-Million Naira Car Assembly Plant*. [Film/T news footage] British Pathé. Available at: https://www.britishpathe.com/asset/201014/

2. Nanyang Technological University, Singapore. (2022) *A new car assembly plant begins operation in Nigeria*. NTU-SBF Centre for African Studies, 18 February. Available at: https://www.ntu.edu.sg/cas/news-events/news/details/a-new-car-assembly-plant-begins-operation-in-nigeria

3. West Africa Automotive. (2024) *Federal Government Urged to Approve NAIDP 2024-2034 Act to Strengthen Automotive Industry*, 25 July. Available at: https://westafricaautomotive.com/federal-government-urged-to-approve-naidp-20242034-act-to-strengthen-automotive-industry

4. Omogbolagun, T. (2025) 'Senate pushes for use of locally-made vehicles', *BusinessDay*, 15 May. Available at: https://businessday.ng/news/article/senate-pushes-for-use-of-locally-made-vehicles/

5. Olatunji, K. (2025) 'Senate adopts electric vehicle bill, considers green future', *The Guardian*, 2 July. Available at: https://guardian.ng/news/senate-adopts-electric-vehicle-bill-considers-green-future

6. Mobility Rising. (2025) *Nigeria passes law to promote electric mobility*, 3 July. Available at: https://www.mobility-rising.com/p/nigeria-passes-law-to-promote-electric-mobility

7. Hegde, R. and Gurumurthy, K.S. (2008) 'Model Based Approach for the Integration of ECUs', *Proceedings of the World Congress on Engineering 2008 Vol I*, London, U.K., 2-4 July. London: International Association of Engineers.

8. Hansen, P. (2004) ‘AUTOSAR Could Transform the Auto Electronics Industry’, *Hansen Report on Automotive Electronics*, October 2004 https://www.researchgate.net/publication/294401982_Hansen_AUTOSAR_could_transform_the_auto_electronics_industry

9. Katla, R. (2025) 'AUTOSAR Framework: Enabling Scalable and Modular Software Architecture in Contemporary Automotive Systems', *Journal of Information Systems Engineering and Management*, 10(60s).

10. Huang, Z. & Xiong, Z., 2015. *Design of Electric Vehicle Drive Motor ECU Control System Based on AUTOSAR*. In: *Proceedings of the 2015 International Conference on Intelligent Systems Research and Mechatronics Engineering*. Atlantis Press, pp. 1612-1615. DOI: 10.2991/isrme-15.2015.327

11. Kim, D.Y., Joe, W.T. & Lee, H., 2018. *Developing AUTOSAR- and ISO 26262-Compliant Software for a Hybrid Vehicle Battery Management System with Model-Based Design.* MathWorks. https://www.mathworks.com/company/technical-articles/developing-autosar-compliant-software-for-a-hybrid-vehicle-battery-management-system-with-model-based-design.html

12. Yahia, T.T., Antar, R.K. and Saleh, A.A. (2023) 'DC Motor Speed/Torque Estimation and Control Based on Transfer Function Characteristics', *2023 International Conference on Engineering, Science and Advanced Technology (ICESAT)*, 21-22 June. IEEE, pp. 156–161.

13. Chen, L. (2019). *Advanced Control of PMSM Drives*. IEEE Transactions on Industrial Electronics, 66(4), 2845-2856.

14. Gupta, R., & Weiss, M. (2020). *Standardization and Challenges in Automotive ECU Design*. SAE International Journal of Engines, 13(2), 245-258.

15. Bamidele, F. O. (2022). *The 'Hardware-Software Gap' in Emerging EV Markets*. Journal of Engineering for Development, 18(3), 112-125.

16. Martínez-Fernández, S., Gómez, A., Ayala, C.P. and Franch, X. (2015) 'A survey on the benefits and drawbacks of AUTOSAR', *Proceedings of the 37th International Conference on Software Engineering,* Florence, Italy, 16–24 May. ACM. DOI: 10.1145/2752489.2752493

17. Gao, J., Sun, F., He, H., Zhu, G. G., & Strangas, E. G. (2009) 'A comparative study of supervisory control strategies for a series hybrid electric vehicle', *Asia-Pacific Power and Energy Engineering Conference (APPEEC 2009)*, Wuhan, China, 27-31 March. doi:10.1109/APPEEC.2009.4918038.

18. Han, K. & Cho, J. (2012) "*Design Exploration Technique for Software Component Mapping of AUTOSAR Development Methodology"* — useful for component partitioning and design trade-offs in AUTOSAR.