POLITECNICO DI TORINO

Degree in Automotive Engineering

Master's Degree Thesis

Electric Vehicles Powertrain Control and Optimization

Gabriel Jenner De Faria Orsi

Supervised by
Prof. Dr. Carlo Novara
Prof. Dr. Angelo Bonfitto Phd. Michele Pagone

July 2024

Summary

In recent years, electric vehicles have gained significant attention as a sustainable transportation solution. The shift towards electric vehicles has become increasingly important in addressing environmental concerns and reducing dependence on traditional fossil fuels. Vehicle manufacturers are now challenged to develop advanced powertrain technologies that offer compact, energy-efficient, and environmentally friendly solutions at affordable costs. This requires extensive research and development efforts to design innovative technological solutions that cater to the growing demand for low carbon emission transportation, essential for combating global warming and enhancing urban air quality. This thesis specifically focuses on the modeling and control of electric vehicle powertrain, using a model predictive control as methodology for minimizing the battery consumption and optimizing the acceleration performance.

Keywords: Electric vehicles, powertrain, Model Predictive Control, longitudinal dynamics, optimization.

Table of Contents

List of Tables VI

List of Figures VII

Acronyms XI

1 Introduction 1

1.1 Objectives . . . 2

1.2 Literature review . . . 3

2 Electric Vehicle Powertrain Components 6

2.1 Power Source . . . 7

2.2 Battery Management System . . . 7

2.3 Inverters and Converters . . . 7

2.4 Electronic Controllers . . . 8

2.5 Electric Motor . . . 8

2.6 Transmission system . . . 8

2.7 Onboard Charger . . . 9

3 Model Predictive Control (MPC) 10

3.1 Introduction to Model Predictive Control . . . 10

3.2 MPC Methodology . . . 11

4 Modeling 13

4.1 Foreword Approach . . . 14

4.2 Backward approach . . . 14

4.3 Driving Cycle - WLTP3 . . . 15

4.4 Vehicle Parameters . . . 16

4.5 Longitudinal Dynamics Model . . . 17

4.6 Gearbox Model . . . 19

4.7 Wheel Model . . . 20

4.8 Electric Motor Model 21
4.9 Battery Model 24
4.10 Polynomial fits 27
4.11 Car-Following Scenario Block 30
4.12 CTG Policy Controller 31
4.13 State Space Model 33
4.14 MPC Block 35
4.15 Model in the Loop (MIL) 500e 37

5 Simulations 40
5.1 Backward Model 41
5.2 Backward-Forward Reference Model 44
5.3 MPC Controlled Model 47
5.3.1 Velocity Reference 49
5.3.2 Position Reference 52
5.3.3 ACC Simplified Scenario 56
5.3.4 ACC Complete Scenario 59
5.4 Model in the Loop 500e with MPC controller 62

6 Conclusion 70
6.1 Next Steps 71

Bibliography 73

A Complete cycle plots 76

B MATLAB scripts 84

C Python scripts 115

List of Tables

4.1 Vehicle parameters 16
4.2 Battery parameters 25

List of Figures

2.1 Electric vehicle powertrain architecture 6
2.2 Battery pack with management system 7

4.1 Forword approach 14
4.2 Backward approach 15
4.3 WLTP3 driving cycle 16
4.4 Longitudinal dynamics diagram block - backward modeling 18
4.5 Longitudinal dynamics diagram block - forward modeling 18
4.6 Gearbox block - backward modeling 19
4.7 Gearbox block - forward modeling 20
4.8 Wheel block - forward modeling 21
4.9 Electric motor block 22
4.10 2D efficiency map 23
4.11 3D efficiency map 24
4.12 Battery Simulink model 26
4.13 Interpolation of battery voltage and resistance 26
4.14 3D efficiency map generated by polynomial equation 28
4.15 3D efficiency map with limits adjusted 29
4.16 2D efficiency map generated by polynomial equation 30
4.17 Car-following scenario Simulink model 31
4.18 Simplified CTG controller Simulink model 32
4.19 Complete CTG controller Simulink model 32
4.20 MPC Simulink block 35
4.21 Interpreted MATLAB function block parameters 36
4.22 MIL setup - Controller and Plant Simulink blocks 38
4.23 MIL setup - Controller Simulink model 38
4.24 MIL setup - High level controller Simulink model 39
4.25 MIL setup - Low level controller Simulink model 39

5.1 Vehicle and powertrain Simulink model - backward approach 41
5.2 Wheel torque and angular speed 42

5.3 Electric motor shaft torque and angular speed 43
5.4 Battery power and State of Charge 44
5.5 Backward-Forward Simulink model 45
5.6 Backward-Forward model states evolution 46
5.7 Backward-Forward model torque and acceleration evolution 47
5.8 MPC controlled Simulink model 48
5.9 MPC controlled Simulink powertrain plant 48
5.10 MPC controlled model states evolution - MPC default parameters . 49
5.11 MPC controlled model torque and acceleration evolution - MPC default parameters 50
5.12 MPC controlled model states evolution - velocity reference 51
5.13 MPC controlled model control torque and vehicle acceleration evolution - velocity reference 52
5.14 MPC controlled model states evolution - position reference 54
5.15 MPC controlled model control torque and vehicle acceleration evolution - position reference 55
5.16 MPC controlled model ego and lead positions - position reference . 56
5.17 MPC controlled model states evolution - simplified ACC scenario . 57
5.18 MPC controlled model control torque and vehicle acceleration evolution - simplified ACC scenario 58
5.19 MPC controlled model ego and lead positions - simplified ACC scenario 59
5.20 MPC controlled model states evolution - complete ACC scenario . . 60
5.21 MPC controlled model control torque and vehicle acceleration evolution - complete ACC scenario 61
5.22 MPC controlled model ego and lead positions - complete ACC scenario 62
5.23 MIL setup with MPC - controller Simulink model 63
5.24 MIL setup with MPC - MPC Simulink model 63
5.25 MIL setup with MPC - low level controller Simulink model 64
5.26 MIL results - MPC and reference powertrain signals 65
5.27 MIL results - MPC and reference battery signals 66
5.28 MIL results - MPC and reference battery signals zoomed in 67
5.29 MIL results - MPC and reference relative distance signals 68
5.30 MIL results - MPC and reference relative distance signals zoomed in 69

A.1 MPC controlled simplified model results states evolution - velocity reference 1800 s 76
A.2 MPC controlled simplified model results torque and acceleration evolution - velocity reference 1800 s 77
A.3 MPC controlled simplified model results states evolution - position reference 1800 s 78

A.4 MPC controlled simplified model results torque and acceleration evolution - position reference 1800 s 79
A.5 MPC controlled simplified model results relative distance - position reference 1800 s 80
A.6 MPC controlled simplified model results states evolution - simplified ACC 1800 s 81
A.7 MPC controlled simplified model results torque and acceleration evolution - simplified ACC 1800 s 82
A.8 MPC controlled simplified model results relative distance - simplified ACC 1800 s 83

Acronyms

EV
: Electric Vehicle

HEV
: Hybrid Electric Vehicle

MPC
: Model Predictive Control

PMSM
: Permanent Magnet Synchronous Motor

DTC
: Direct Torque Control

FOC
: Field Oriented Control

PMC
: Power Management Control

MCU
: Micro Controller Unit

SOC
: State Of Charge

ACC
: Adaptive Cruise Control

CTG
: Constant Time Gap

MIL
: Model in the Loop

Chapter 1

Introduction

In the past decades, electric vehicles (EVs) have emerged as a transformative force, promising a cleaner and more sustainable future for transportation. With the pressing need to combat climate change and reduce our dependence on fossil fuels, the adoption of EVs has gained momentum worldwide and has witnessed significant growth over the past decade, owing to advancements in battery technology, supportive government policies, and a growing awareness of the environmental issues among consumers.

The powertrain, a crucial component of any vehicle, plays the main role in determining the overall performance and efficiency of an electric vehicle. An EV powertrain is composed of several interconnected subsystems, including the electric motor, power electronics (usually a DC/AC converter), and the energy storage system (usually lithium-ion batteries). Efficiently managing and optimizing these subsystems is vital to achieve superior vehicle performance, extended driving range, and enhanced energy efficiency.

In recent years, substantial advancements have been made in powertrain control strategies and optimization techniques, owing to breakthroughs in technology, computing power, and an increasing understanding of EV dynamics. The implementation of sophisticated control algorithms, intelligent energy management systems, and real-time optimization has revolutionized the way electric vehicles perform on the road.

The study of new methods and advancement of existing ones regarding control to increase performance of electric vehicle powertrain is essential for the future of the transportation field. One of the main reasons for the advancements in the field is the European Union Regulation of 2019, Regulation 2019/631 [1], which states as target of reducing 100% of CO2 emissions from passengar cars and light commercial vehicles by 2035 in the whole European Union territory.

In this Master's thesis, the purpose is to use a non linear Model Predictive Control (MPC) in order to control the whole powertrain system with the objective of optimizing the energy consumption of the battery to increase its range, and, thus, the energetic efficiency of the powertrain as a whole. By analyzing the existing literature and experimental data, this study seeks to provide comprehensive insights into the following key aspects:

1. Powertrain components: description of the main powertrain components of an electric vehicle and its functionalities, as well as the state-of-the art usage of such components, highlighting their advantages and limitations.

2. Powertrain and vehicle dynamics models: description of the mathematical, physical and computational modeling of the powertrain and vehicle daynamics, that will be used for the implementation, tuning and testing of the controller, and for the simulation of different scenarios to generate the desired results.

3. Model Predictive Control (MPC): general description of the non linear MPC strategy, followed by its application for the EV powertrain control, emphasizing the system states to be inputted and the control variables to be outputted by the controller.

1.1 Objectives

The objectives of this Master Thesis is to elucidate the functionality of a Model Predictive Controller, applied in the automotive field. The focus here is to use the control strategy to predict the states of the longitudinal dynamics of a vehicle with an electric powertrain, with the objective of minimizing the battery consumption when following a given speed profile (WLTP3). Furthermore, car following scenarios are studied, simulating a realistic ACC case, with the ego vehicle being controlled by the MPC, and the reference being generated using a CTG policy. Finally, the tuned MPC is used to generate the required torque for a complete 500e model in the loop (MIL) provided by Politecnico di Torino. For that, the following objectives are achieved:

- Description of MPC strategy;
- development of longitudinal dynamics model using a backward approach;
- development of the electric powertrain with Electric Machine working as motor and generator, and a battery model, that will provide SOC information as a system state;
- simulation of the model following the reference speed profile to validate the equations;

- implementation of a forward model that has as input the EM torque provided by the backward model. This dual approach model has the physical causality of a real case scenario, and will serve as the states reference;
- implementation of the MPC controller with the powertrain plant modeled with the forward approach, so that the plant has as input the control torque provided by the controller;
- further implementation of the tuned MPC in a ACC scenario using CTG policy;
- comparison of the results of the uncontrolled reference model, and the MPC controlled model.
- simulation of the complete MIL 500e setup with the MPC controller generating the required torque signal.

1.2 Literature review

There have been a lot of studies in a number of different fields of Electric and Hybrid Vehicles in the past decade, and a great number of different control and optimization strategies along multiple parts of the powertrain have been implemented. One of the main problems with electric vehicles is related to the energy storage system (battery system) in terms of weight and energy available for long distance driving (driving range). Trying to overcome that main matter, different strategies of control and optimization are available in the literature, as well as articles making a review on the state of the art components of such powertrains.

Among the optimization strategies, [2] sets the travel time as a target, allowing the transmission ratio to be adapted along the route. This approach is applicable to transport vehicles meanly in different given routes, this way, the shifting strategy or the optimal transmission itself can be designed depending on the route. The main design objective in this article is the minimization of the battery weight. Also in [3] the gear ratio is the object of the optimization, and the optimization results are presented for two design examples presented - Tesla Model S and Mini Cooper SE, the first with an induction motor, and the last with the most common PMSM (Permanent Magnet Synchronous Motor) equipped. Furthermore, in [4] the powertrain parameters are the objects of optimization, in a New European Driving Cycle scenario, with dynamic and economic optimizations being the main goals. The vehicle modeling is presents, the transmission control unit model for dynamic and economic shift decisions is made, the vehicle control unit is modeled as well, and the parameters of the powertrain are optimized based on a genetic algorithm. Article [5] presents a platform for electric powertrain simulation, also a powertrain architecture with 3 degrees of freedom is presented and optimized with the objective of inspect typical study cases, methodology and results. Furthermore a for the degree of freedom is introduced to the model for scenarios when the battery is partially discharged. Also, different results from different control strategies are presented, a software optimization is made only adjusting control laws with a fixes EV architecture, and a hardware optimization is performed by introducing the extra degree of freedom in the architecture.

Furthermore, [6] and [7] discuss and make reviews on advanced traction motor control strategies for the first case, and general trend for HEV's and EV's. [6] makes a review on the evolution of different control techniques and concludes that there is a great number of researches involving the application of Direct Torque Control (DTC) and Field Oriented Control (FOC) to traction motors, also pros and cons of the researches are presented. On the other hand, in [7] a overview on the current researches on hardware optimization of HEV's and EV's, suggesting the challenges and future researches that could be made.

Also in [8] a review on the state of the art control strategies for each component in EV powertrain architectures is made. It is discussed that the main control issues reside in the HEV torque management, EV battery management system, motor drive technique and control of the energy recovery. For the torque management in HEV's the strategy is to coordinate the torque provided by the engine and the electric motor, to supply the required torque by the driver, while optimizing the efficiencies of the engine, the electric motor and battery consumption - it is a constrained optimization problem to be addressed. In the battery management system side, the main problem is the estimation of the State Of Charge (SOC) of the battery, the main method are open circuit voltage measurement, resistance method, Fuzzy logic method, neural network method and Kalman filtering method. For the motor drive control (electric motor control) the main control strategies are voltage/frequency ratio control, slip frequency control, vector control and Direct Torque Control (DTC). In the regenerative breaking control field, three braking force distribution control strategies are proposed: parallel regenerative breaking control, ideal regenerative breaking control and maximum regenerative breaking control.

Different control strategies for the powertrain of EV's are found in the literature. Different Fuzzy control strategies are applied in the scope of EV's as reported in [9] and [10]. The first presents the Fuzzy control logic applied in a Indirect Vector Control technique, which calculates the slip speed of the electric machine, an EV powertrain is controlled with the logic described, and the evaluation is made using a FUDS driving cycle. The last performs a Fuzzy Control Multi-Objective Optimization in a Dual Hybrid Energy Storage System (Dual-HESS) with the objective of optimizing the batteries and ultracapacitors size. A novel Dual-HESS is also proposed in the article, with one energy storage system for each axis (front and rear). A Power Management Control (PMC) with Fuzzy logic is applied to the proposed architecture. The simulation model developed in [10] is similar to the one developed in this thesis, with the longitudinal model, EM model with efficiency map and the modeling of both the battery and the ultracapictors.Three driving cycles are used to generate the speed profile used as reference for the simulation, the driving cycles used are: FTP-75 (urban driving), HWFET (highway driving) and USo6 (high speed and required acceleration). The reference acceleration is obtained by the derivative in time of the speed profile, and this acceleration is inputted as the required force for the simulation, as it is proposed in this thesis. The Fuzzy PMC receives as inputs the required frontal and rear torques from the drive systems, and also the overall efficiencies (electric motor and inverter) for each axis. The output of the controller is the percentage of the required torque to be fulfilled by each drive axis. The torques multiplied by their respective percentage are applied to the EM's equations to define the required torques from the machines.

Using Hardware in The Loop Platform, [11] developed a controller for EV powertrain. First a mathematical model of a PMSM (Permanent Magnet Synchronous Motor) is developed, also a plant model of a power battery is constructed. A real MCU is used to communicate with the electric motor model, and a real battery management system is used to communicate with the battery plant model. In the paper, all these components are combined with a real vehicle controller to provide a complete test environment. A control software was developed.

Furthermore, the modeling and simulation of EV powertrain is discussed in more detail in [12]

Predictive optimization strategies are also implemented in [13] with the focus on the energy management system in a modeled random traffic scenario. A stochastic MPC strategy is done to co optimize both the speed and the powertrain energy management system in a driving environment with uncertainties.

Also a Constant Time Gap (CTG) policy for ACC and its modeling is discussed both in [14] and [15].

The complete Model in the Loop (MIL) of the 500e has its modeling discussed in [16].

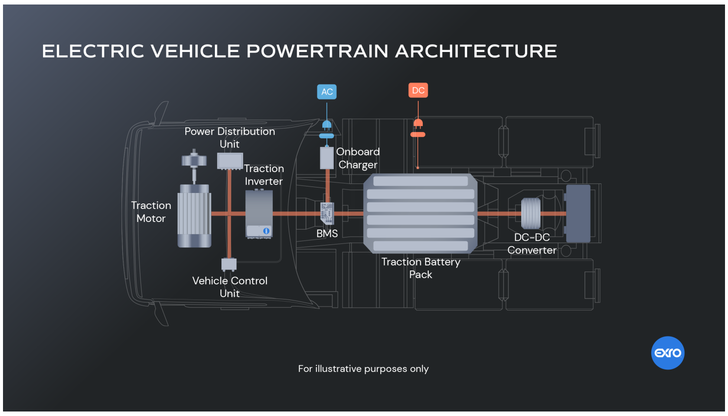
Chapter 2

Electric Vehicle Powertrain Components

This chapter provides an overview of the main components of an electric vehicle powertrain.

The increasing market share of hybrid electric vehicles and electric vehicles has elevated the importance of electric machines in powertrain development [17]. Examples of electrified powertrains include hybrid, plug-in hybrid, electric vehicles, and fuel cell vehicles [18]. The primary constituents of an electric powertrain are the power source with an management system (Battery Management System), electronic controllers, electric motor, transmission system, and onboard charger for batteries [19]. These components work together to provide the necessary power and control for the operation of the electric vehicle. Figure 2.1 shows a generic architecture of a electric vehicle powertrain.

Figure 2.1: Electric vehicle powertrain architecture



Source: [20]

2.1 Power Source

The power source is a crucial component of an electric powertrain as it provides the necessary energy to propel the vehicle. The more used power source is battery cells, usually lithium-ion batteries, due to their high energy density and long cycle life. Other power sources, such as fuel cells or supercapacitors, are also being explored and implemented in certain electric vehicle models.

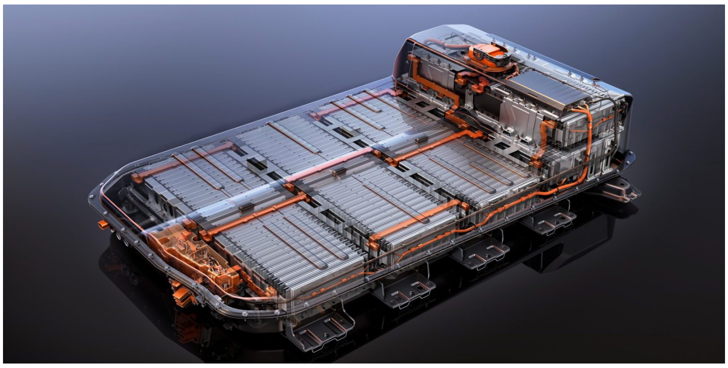
2.2 Battery Management System

The battery management system plays a vital role in the powertrain of electric vehicles. It is responsible for monitoring and controlling the performance, efficiency, and safety of the battery pack.

This system ensures that each individual battery cell is operating within its optimal range and prevents overcharging or discharging, which can lead to reduced battery life and degraded performance.

Figure 2.2 elucidates the dimension of the battery pack along with the management system.

Figure 2.2: Battery pack with management system



Source: [21]

2.3 Inverters and Converters

The traction converter is a electronic component responsible for converting the energy DC output (voltage and current) into a AC input for the electric machine.

The DC-DC converter, on the other hand, does not change the nature of the signal itself, but it steps down the DC voltage of the battery pack (usually very high tension values - 100 up to 400 V) to much smaller values to be used in other auxiliary electronics in the vehicle (such as air conditioning, sound system, etc) or to charge the 12V auxiliary battery.

2.4 Electronic Controllers

The electronic controllers play a key role in managing and controlling the flow of electrical energy within the powertrain. These controllers include the Battery Management System, which monitors and controls the charging and discharging of the battery, ensuring its optimal performance and longevity. Other electronic controllers, such as the motor controller and power electronics, are responsible for controlling the speed and torque of the electric motor, converting DC energy from the battery to AC energy for the motor, and managing the overall power distribution within the powertrain.

2.5 Electric Motor

The electric motor is the heart of the electric powertrain. It is responsible for converting electrical energy into mechanical energy to propel the vehicle. Various types of electric motors can be used in an electric powertrain, including permanent magnet synchronous motors, induction motors, and switched reluctance motors. The choice of motor depends on factors such as power requirements, efficiency, and cost considerations.

The electric machine also works as a generator when the torque is negative (breaking manouver) making it possible to generate power and charge the battery while breaking - generative breaking. This is a great advantage of electric vehicles powertrains.

2.6 Transmission system

The transmission system in an electric powertrain is responsible for transferring the mechanical power from the electric motor to the wheels of the vehicle. This system often consists of a single-speed or multi-speed transmission, depending on the specific requirements of the vehicle. The transmission system plays a crucial role in optimizing the power and torque delivery to the wheels, ensuring efficient and smooth acceleration of the vehicle.

2.7 Onboard Charger

The onboard charger is another important component of the electric powertrain. It is responsible for converting the AC power from an external power source, such as a charging station, into DC power to charge the vehicle's battery. This component allows for convenient charging of the electric vehicle and is essential for maintaining the battery's state of charge.

In summary, the main components of an electric vehicle powertrain include the driving motor, electronic controllers, electric motor, transmission system, and onboard charger . These components work together to convert electrical energy into mechanical energy, control the flow of power, and ensure efficient operation of the electric vehicle.

Chapter 3

Model Predictive Control (MPC)

3.1 Introduction to Model Predictive Control

Model Predictive Control is a control strategy widely used in various domains, including the control of electric vehicle powertrains. This control strategy takes into account the dynamic nature of the system and predicts future states and inputs based on a mathematical model of the system. By explicitly considering the system dynamics, MPC is able to optimize control inputs over a finite time horizon to achieve desired objectives such as energy efficiency, performance, and constraint satisfaction.

The use of Model Predictive Control in industrial processes is highly advantageous due to its ability to handle constraints such as input saturation and rate limits [22]. Model Predictive Control is capable of dealing with complex multi-input multi-output systems with hard state and input constraints, making it suitable for controlling electric vehicle powertrains with longitudinal dynamics [23]. Furthermore, Model Predictive Control has the ability to handle model uncertainty and disturbances, which is essential in achieving robust and reliable control performance.

One of the major advantages of Model Predictive Control is its ability to incorporate constraints on the inputs and outputs of the system [22].

This is particularly important in the context of controlling electric vehicle powertrains, where constraints on the battery state of charge and motor torque limits need to be taken into account. Using MPC for controlling an electric vehicle powertrain with longitudinal dynamics allows for the optimization of control inputs over a finite time horizon, taking into account system constraints and the varying nature of the driving conditions.

Incorporating MPC in the control of electric vehicle powertrains allows for the prediction and optimization of future states and inputs based on the system's dynamic model. This proves to be especially advantageous in achieving energy efficiency and effective management of the powertrain components.

Furthermore, the ability of MPC to continuously update the model and control strategy allows it to handle changes in system parameters, such as variations in battery capacity or motor efficiency, without the need for extensive re-calibration.

3.2 MPC Methodology

Model Predictive Control is an optimization-based control method that uses a dynamic model of the system to predict and optimize future states and inputs based on a cost function and subject to constraints. The cost function typically aims to minimize a combination of control effort, system error, and deviation from desired operating conditions.MPC works by solving an optimization problem at each control interval, where the objective is to find the optimal control inputs that minimize the cost function while satisfying system constraints.

The optimization problem is solved over a finite time horizon, also known as the prediction horizon, which allows for considering future system behavior and making informed decisions. During the optimization process, the control inputs are calculated for the current time step, but only the first set of inputs is applied to the system. The remaining inputs are discarded, and the process is repeated at the next control interval with updated measurements and predictions. This repeated optimization process is known as a moving horizon approach. By using a moving horizon approach, MPC can effectively handle changes in the system parameters and adapt to varying driving conditions.

By incorporating a dynamic model of the electric vehicle powertrain into the MPC framework, the control algorithm can effectively predict and optimize future system states and inputs. The ability to handle changes in the system parameters and adapt to varying driving conditions makes MPC a powerful tool for electric vehicle powertrain control. It enables the prediction and optimization of future states and inputs, allowing for efficient energy management and effective control of the powertrain components.

To explain the math of MPC, let's start with the basic formulation. In Model Predictive Control, the optimization problem seeks to minimize a cost function subject to system dynamics and constraints [24]. This can be expressed as:

$$minJ = \Sigma_{i=0}^{N-1} L(x(i), u(i)) + \phi(x(N)) \tag{3.1}$$

subject to:

$$x(i+1) = f(x(i), u(i))g(x(i), u(i))0$$

where:

- J is the cost function to be minimized;
- N is the prediction horizon;
- x(i) represents the system state at time i;
- u(i) represents the control input at time i;
- L is the stage cost function that quantifies the performance of the system at each time step;
- ϕ is the terminal cost function that captures the desired final state of the system- f represents the system dynamics, which describe how the state evolves over time based on the current state and control input;
- g represents the system constraints, which limit the feasible state and input space.

By solving this optimization problem iterative, the MPC algorithm generates a sequence of control inputs that minimizes the cost function while satisfying the system dynamics and constraints. This approach allows the MPC algorithm to effectively control the electric vehicle powertrain by dynamically adjusting the control inputs based on real-time measurements and predictions.

In summary, Model Predictive Control is a control method that optimizes a cost function based on system dynamics and constraints.

Chapter 4

Modeling

The modeling is made in Simulink and MATLAB scripts. The longitudinal dynamics equations of the vehicle and of the electric motor are done based on [25] and [26]. The battery model equations are also based on [25].

Backward and forward approaches are used in the simulations. First, for the validation of the model and parameters, a backward model is developed with the reference speed profile and acceleration as inputs of the system. Then, for the uncontrolled reference model, a backward model is used to produce the electric motor torque command, which serves as the input for the complete forward powertrain model (vehicle dynamics with electric powertrain components - electric motor and battery). For the MPC controlled model, the complete system model is developed in a state space form, which is set as a parameter of the MPC and will serve as the prediction model for the controller. The controller result is a electric motor torque, that will be the input of the forward complete powertrain model, where the states (SOC, position and velocity) will be computed and compared with the uncontrolled model.

The driving cycle that it is being imposed to the vehicle as a reference speed profile is the WLTP3, which has a duration of 1800 seconds, and has a low speed scenario, simulating urban driving, and a high speed scenario which simulates highway driving condition.

The model of each powertrain component used in the simulation is depicted in this section, as well as the interpolations, and polynomial fits done for the battery and electric motor models.

The MATLAB scripts for all the models and parameters are available in the appendix of this thesis B. And the polynomial fits necessary are made in Python, and are available in the appendix C.

The MPC code is a closed .p file, and provided by Politecnico di Torino.

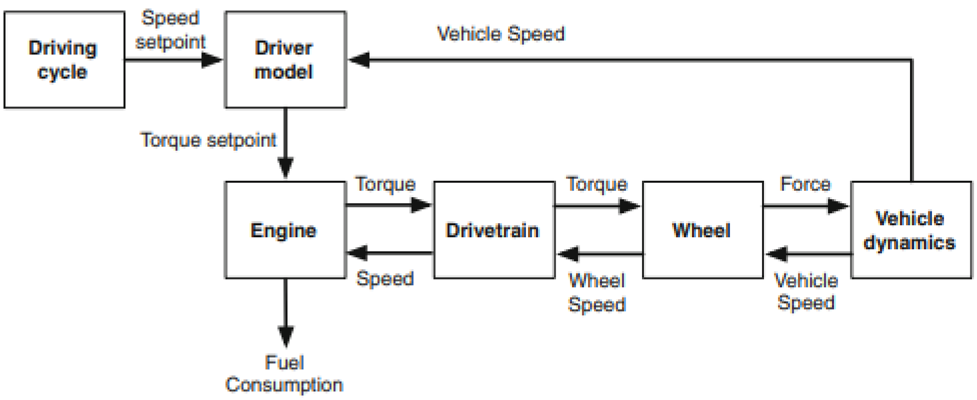
4.1 Foreword Approach

In the foreword approach the physical casulity of the system is reproduced, so the reference desired speed is compared to the actual vehicle speed and acceleration or breaking commands are produced to achieve the desired reference, a driver model is necessary to provide such commands, and a supervisor block is responsible for issuing the actuators set points to the rest of the powertrain components which is responsible to produce the traction force, such force is applied to the vehicle dynamics. The acceleration is determined by the equation 4.1. Figure 4.1 shows the scheme of such approach [25].

$$M_{veh}\frac{dv_{veh}}{dt} = F_{inertia} = F_{trac} - F_{roll} - F_{aero} - F_{grade} \tag{4.1}$$

$F_{inertia}$ is the inertial force, F_{trac} is the traction force, F_{roll} is the rolling resistance force, F_{aero} is the aerodynamic resistance and F_{grade} is the slope/inclination force - weight component.

Figure 4.1: Forword approach



Source: [25]

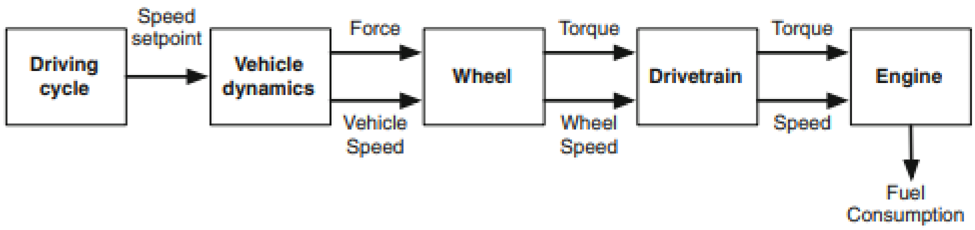
4.2 Backward approach

The backward approach no driver model is used and equation 4.1 is rearranged to calculate the traction force that need to be produced for the vehicle to follow the desired speed profile. In that way, the desired speed and acceleration is directly inputted in the traction force equation 4.2, this way the motor torque and energy consumption are the outputs. The tractive force to be applied is based on the provided velocity, payload, grade profiles and vehicle characteristics. Based on that information the toque that the powertrain should provide is calculated and the power/speed characteristics of the components are taken into account to determine the operating point of the motor and the energy consumption, consequently.

$$F_{trac} = F_{pwt} - F_{brake} = F_{inertia} + F_{grade} + F_{roll} + F_{aero} \tag{4.2}$$

Figure 4.2 shows the backward approach modeling.

Figure 4.2: Backward approach

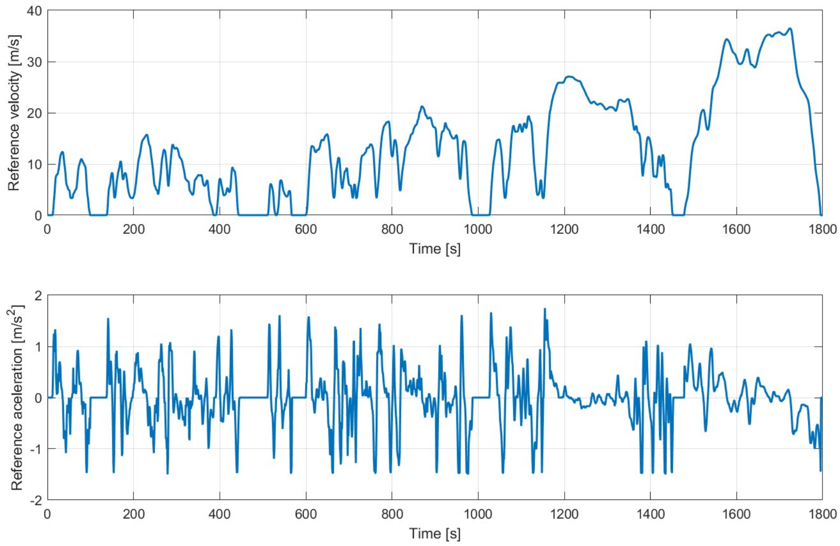


Source: Onori 2016 [25]

4.3 Driving Cycle - WLTP3

The driving cycle that is used is the WLTP3, its speed and acceleration profile are depicted in figure 4.3. The acceleration is obtained directly from the differentiation of the speed profile.

Figure 4.3: WLTP3 driving cycle



Source: MATLAB (2023).

4.4 Vehicle Parameters

The vehicle used as reference is the Fiat 500e, and the parameters, provided by [16] and Politecnico di Torino are exposed in table 4.1.

Table 4.1: Vehicle parameters

Parameter	**Symbol**	**Value**	**Unit**
Vehicle Mass	M_{veh}	1400	kg
Front axle - CoG	a	1	m
Rear axle - CoG	b	1.3	m
Height CoG	h	0.3	m
Static rolling coefficient	f_0	4.5	N/kN
Wheel radius	r_w	0.3	m
Drag coefficient	C_d	0.33	-
Frontal area	A_f	2.15	m2
Gear ratio	τ_{gb}	9.6	-
Gearbox efficiency	η_{gb}	0.97	-

Source: [16]

4.5 Longitudinal Dynamics Model

Based on the backward approach, the longitudinal dynamics of the vehicle can be described by the inertial force ($M_{veh}a_{ref}$) and by the resistive forces: grade force, due to road inclination 4.6; aerodynamics resistance force 4.6; rolling resistance forces 4.5. By summing all the contributions and given a desired acceleration it is possible to compute the output or wheel torque by the backward approach 4.4.

$$T_{wheel} = (F_{grade} + F_{roll} + F_{aero} + M_{veh}a_{ref})r_w \tag{4.3}$$

$$F_{roll} = M_{veh}g\ f_0 \tag{4.4}$$

$$F_{grade} = M_{veh}g\ sin(\alpha) \tag{4.5}$$

$$F_{aero} = 0.5\rho A_f C_d v_{ref}^2 \tag{4.6}$$

In the forward approach, on the other hand, the equation can be rearranged, so that the vehicle acceleration is computed as consequence of the tractive forces provided by the powertrain, and also considering the effect of the resistive forces. So the vehicle dynamic equation in the forward approach is exposed in 4.7.

$$M_{veh}\frac{dv_{veh}}{dt} = \frac{T_{wheel}}{r_w} - F_{roll} - F_{grade} - F_{aero} \tag{4.7}$$

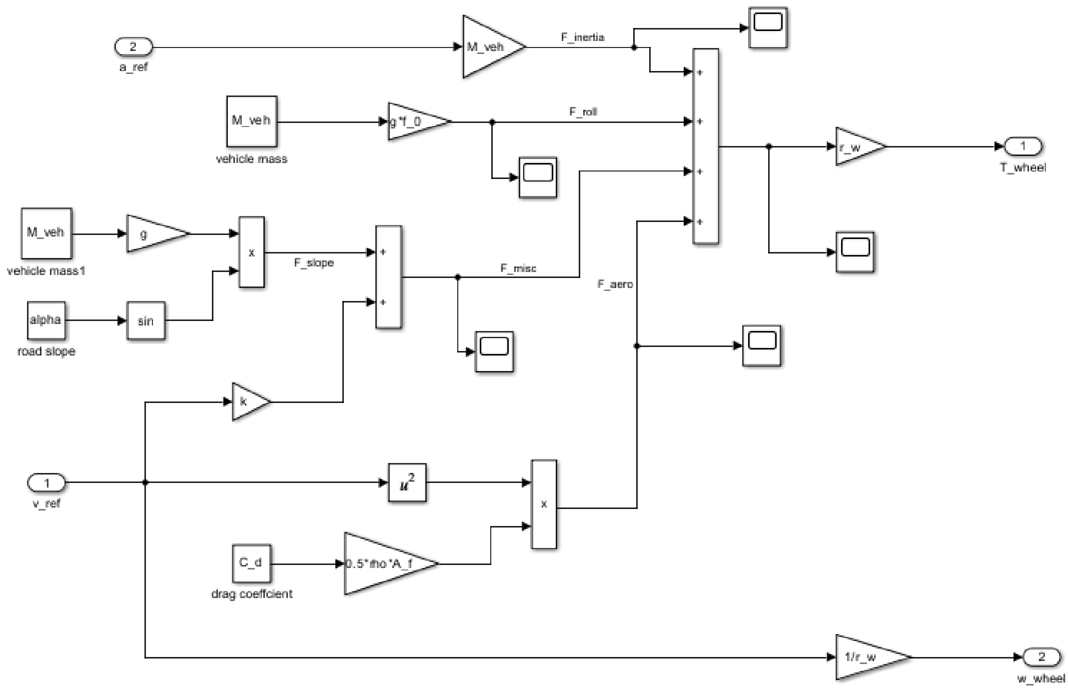
The wheel angular speed is given by equation 4.8.

$$\omega_w = v_{ref}/r_w \tag{4.8}$$

T_{wheel} is the output torque that must be provided to make the vehicle follow the reference, accounting for the resistive forces. r_w is the wheel radius, ω_w is the wheel angular speed, M_{veh} is the total vehicle mass, g is the gravity acceleration, α is the road slope in radians, f_0 is the static rolling coefficient, ρ is the air specific mass, A_f is the vehicle frontal area, C_d is the drag coefficient, and v_{ref} and a_{ref} are the reference speed and acceleration provided by the driving cycle.

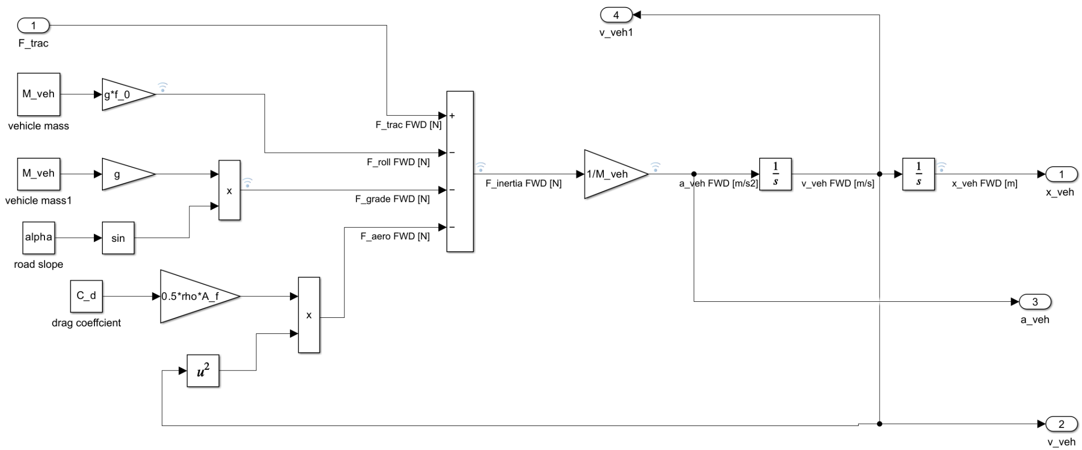
The block diagram for the vehicle dynamics equation using the backward modeling approach is shown in figure 4.4, while the forward modeling is shown in figure 4.5.

Figure 4.4: Longitudinal dynamics diagram block - backward modeling



Source: Own authorship (2024).

Figure 4.5: Longitudinal dynamics diagram block - forward modeling



Source: Own authorship (2024).

4.6 Gearbox Model

The gearbox model is a simple computation of the gearbox efficiency and gear ratio for both the toque and angular speeds coming from the wheel. The torque and wheel speed at the motor shaft level - after the gearbox - are given by equations 4.10 and 4.10 respectively.

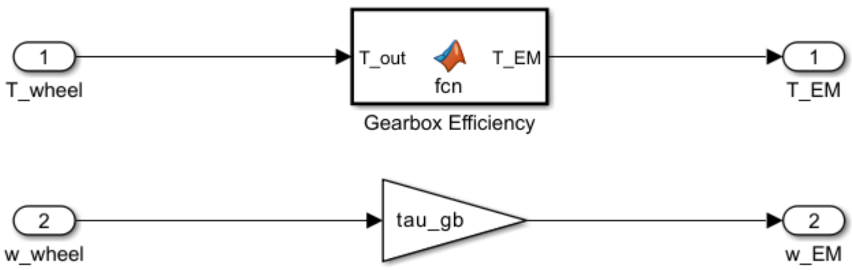
$$T_{EM} = \frac{T_{wheel}}{\eta_{tr}^{\text{sign}(T_{wheel})}\tau_{gb}} \tag{4.9}$$

$$\omega_{EM} = \omega_{wheel}\tau_{gb} \tag{4.10}$$

T_{EM} and ω_{EM} are the torque and angular speed of the electric motor shaft, τ_{gb} is the gearbox ratio, and η_{gb} is the gearbox efficiency.

The Simulink block with the equations for the gearbox dynamics using the backward approach is shown in figure 4.6.

Figure 4.6: Gearbox block - backward modeling

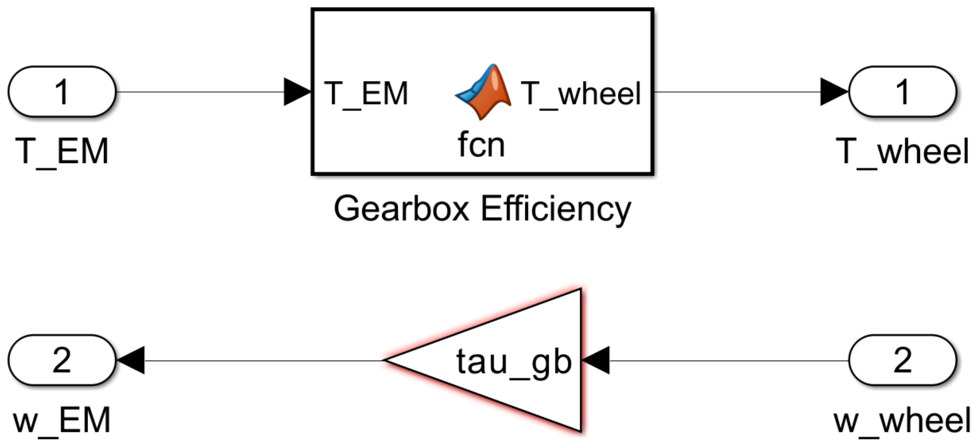


Source: Own authorship (2024).

The model showed in figure 4.7, the angular speed relation is the same, but the efficiency is calculated by equation 4.11.

$$T_wheel = T_EM(\tau_{gb}\eta_{gb}^{\text{sign}(T_{EM})})) \tag{4.11}$$

Figure 4.7: Gearbox block - forward modeling

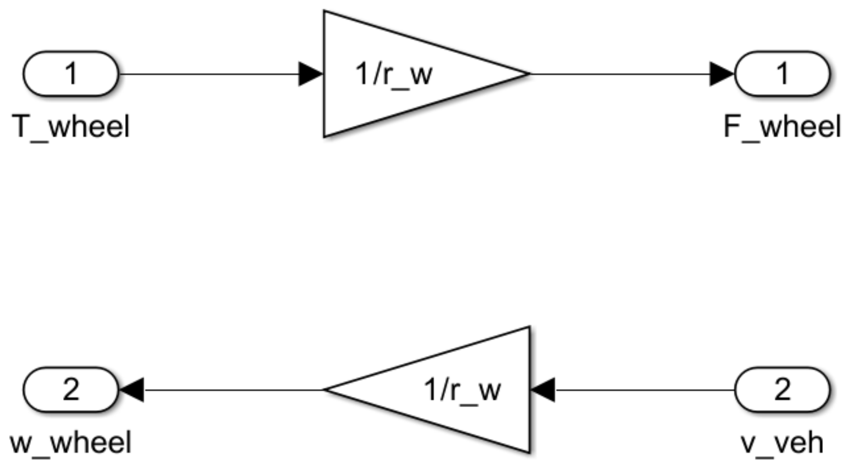


Source: Own authorship (2024).

4.7 Wheel Model

This simulink block is mainly used in the forward models, and it simply relates the wheel torque with the wheel force, and the vehicle speed with the wheel angular speed. Figure 4.8 shows the Simulink black that relates the described variables.

Figure 4.8: Wheel block - forward modeling



Source: Own authorship (2024).

It is a basic division by the wheel radius, as exposed in equations 4.13 and 4.13

$$F_{wheel} = \frac{T_{wheel}}{r_w} \tag{4.12}$$

$$w_{wheel} = \frac{v_{veh}}{r_w} \tag{4.13}$$

F_{wheel} is the wheel force, w_{wheel} is the wheel angular speed, v_{veh} is the vehicle speed and r_w is the wheel radius in meters.

4.8 Electric Motor Model

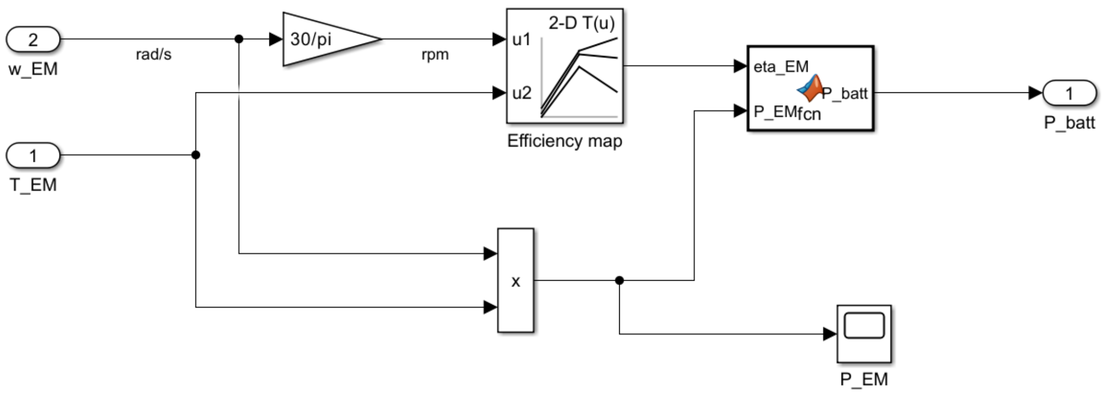
The electric motor modeling is mainly dependent on the efficiency mapping based on a given angular speed and shaft torque.

In the electric motor block of the model (elucidated in figure 4.9), both the motor power and the battery power are the outputs. The motor power is given by equation 4.15 and the battery power that is inputted into the battery block is given by equation 4.15.

$$P_{EM} = T_{EM}\omega_{EM} \tag{4.14}$$

$$P_b = \frac{P_{EM}}{[\eta_{EM}(\omega_{EM}, T_{EM})\eta_{inv}]^{\text{sign}(P_{EM})}} \tag{4.15}$$

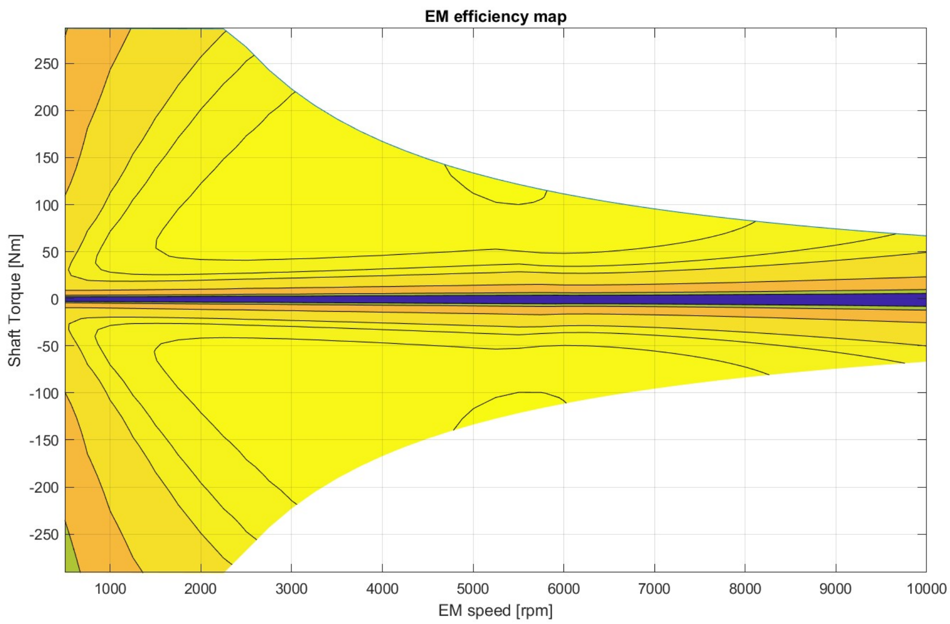
Figure 4.9: Electric motor block



Source: Own authorship (2024).

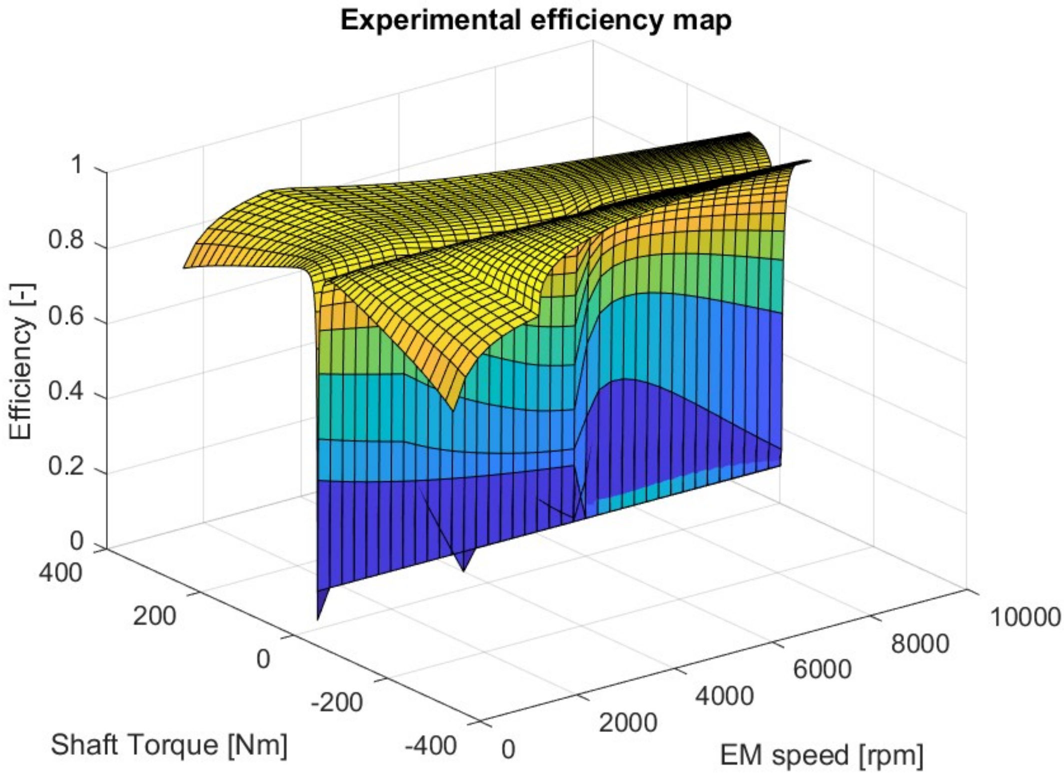
The efficiency map for the electric motor is a generic one provided by Politecnico di Torino, for simulation purposes, the real efficiency map of the Fiat 500e is not available. The 2D and 3D (surface plot) maps are exposed in figures 4.10 and 4.11.

Figure 4.10: 2D efficiency map



Source: Own authorship (2024).

Figure 4.11: 3D efficiency map



Source: Own authorship (2024).

The electric motor model is the same for both the forward and backward modeling approaches.

4.9 Battery Model

The battery model follows the reference modeling exposed in [25]. The battery parameters are shown in table 4.2.

Table 4.2: Battery parameters

Parameter	**Symbol**	**Value**	**Unit**
Number of series cells	N_s	108	-
Number of parallel cells	N_p	1	-
Number of total cells	N_b	108	-
Nominal capacity	Q_{nom}	60	Ah
Coloumbic efficiency	η_c	0.95	-

Source: [16]

The State of Charge (SOC), modeled by equation 4.17, is defined as the ration between the current battery charge Q_b and the nominal battery capacity Q_{nom}. When differentiating both sides it is possible to obtain the derivative of the SOC as a function of the battery current I_b, as exposed in equation 4.17.

$$SOC = \xi = \frac{Q_b}{Q_{nom}} \tag{4.16}$$

$$SOC = \xi = -\frac{1}{\eta_c^{\text{sign}(I_b)}} \frac{I_b}{Q_{nom}} \tag{4.17}$$

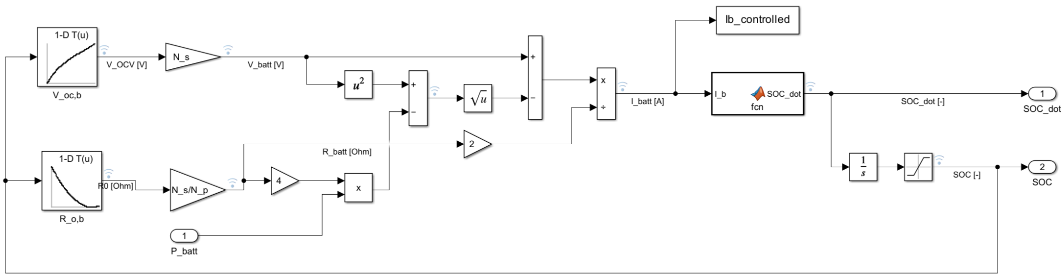
In this modeling, when the battery current is greater than 0, it is in discharge mode, and when is lower than zero, it is in charge mode.

The battery is modeled as an ideal voltage source $V_{oc,b}$ with series of input resistance $R_{o,b}$, so it is possible to solve for the current [25] - equation 4.18.

$$I_b = \frac{V_{oc,b} - \sqrt{V_{oc,b}^2 - 4R_{o,b}P_{batt}}}{2R_{o,b}} \tag{4.18}$$

Furthermore, the Simulink model that implements the equations is exposed in figure 4.12. The equations are for one cell only. To perform a calculation for the whole battery, the tension $V_{oc,b}$ must be multiplied by the number of series cell, and the resistance by N_s/N_p.

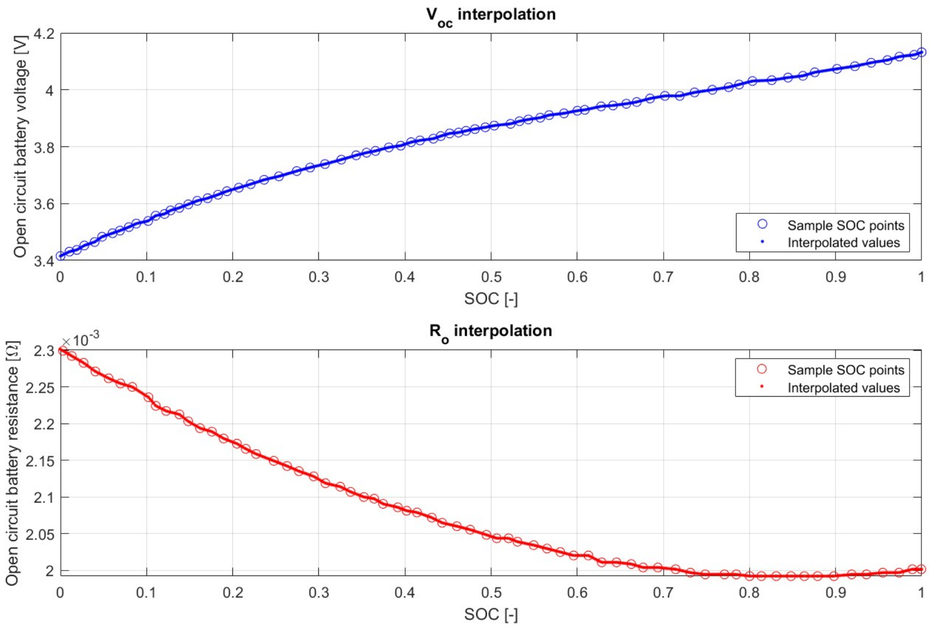
Figure 4.12: Battery Simulink model



Source: Own authorship (2024).

For obtaining the values a resistance and voltage for a single cell, experimental value of resistance and voltage variation as function of the SOC where provided by Politecnico di Torino. The interpolation was done using the *griddedInterpolant* function from MATLAB, and the results are shown in figure 4.13.

Figure 4.13: Interpolation of battery voltage and resistance



Source: Own authorship (2024).

The same battery model is used for both the for the forward and the backward modeling approaches.

4.10 Polynomial fits

Polynomial fits of the battery voltage and resistance equations, and of the electric machine efficiency are developed in this section. These polynomials are used in the state space equations of the system, which is used as prediction model by the MPC controller. The polynomials generation code are done in Python and exposed in C. The polynomial expressions are 4.19 through 4.21.

$$\begin{aligned} E_{EM}(\omega, T) &= 0.95 + (-1.11 \times 10^{-4})\omega + (1.61 \times 10^{-5})T + (2.05 \times 10^{-8})\omega^2 \\ &+ (-8.74 \times 10^{-9})\omega T + (-5.05 \times 10^{-6})T^2 + (-1.14 \times 10^{-12})\omega^3 \\ &+ (7.30 \times 10^{-13})\omega^2 T + (4.52 \times 10^{-9})\omega T^2 + (2.87 \times 10^{-7})T^3 \end{aligned} \tag{4.19}$$

$$V(\xi) = -0.40666 * \xi^2 + 1.0703 * \xi + 3.4385 \tag{4.20}$$

$$R(\xi) = 0.00041627 * \xi^2 - 0.00071804 * \xi + 0.0023018 \tag{4.21}$$

For the electric machine efficiency polynomial, different degrees are tested and the Mean Absolute Error and Root Mean Squared Error are computed. The chosen polynomial is the one with the lower value for both errors

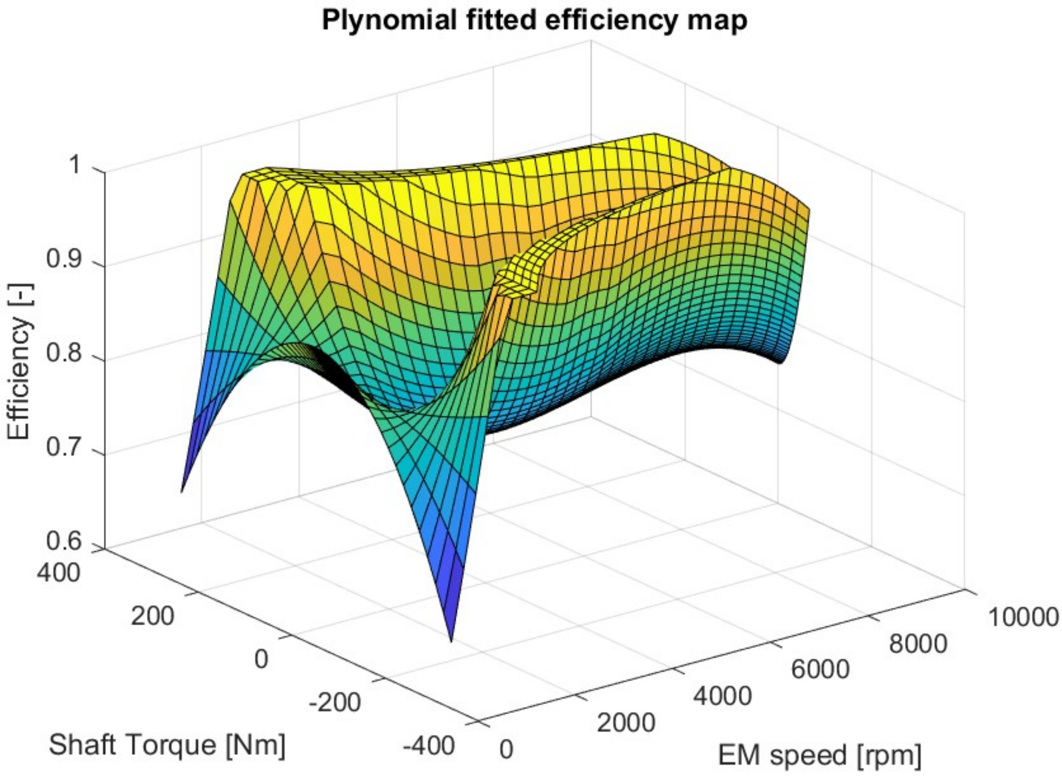
$$MAE = 0.13471$$

$$RMSE = 0.20122$$

and corresponds to the 3rd degree. The 3D and 2D (contour) efficiency maps generated by the polynomial equation (as function of the shaft torque in [Nm] and angular speed in [rpm]) are illustrated in 4.14 and 4.16 respectively.

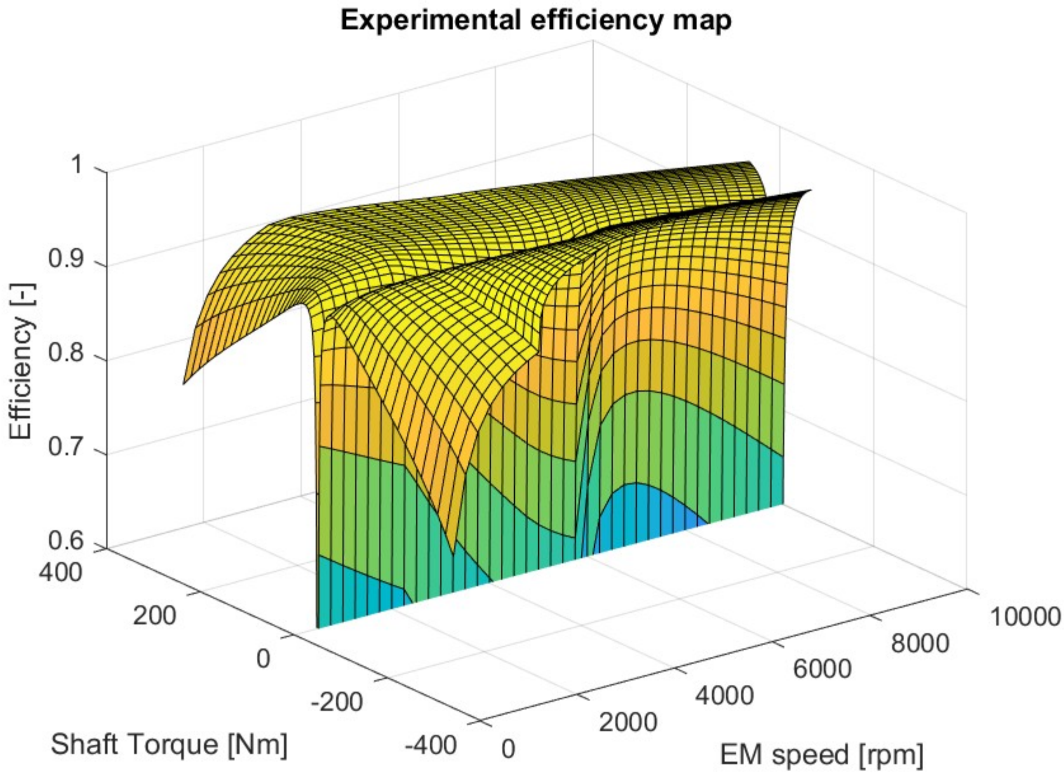
For comparison, the experimental surface plot with the adjusted Z-axis limits - same limits of the polynomial result - is showed in figure 4.15

Figure 4.14: 3D efficiency map generated by polynomial equation



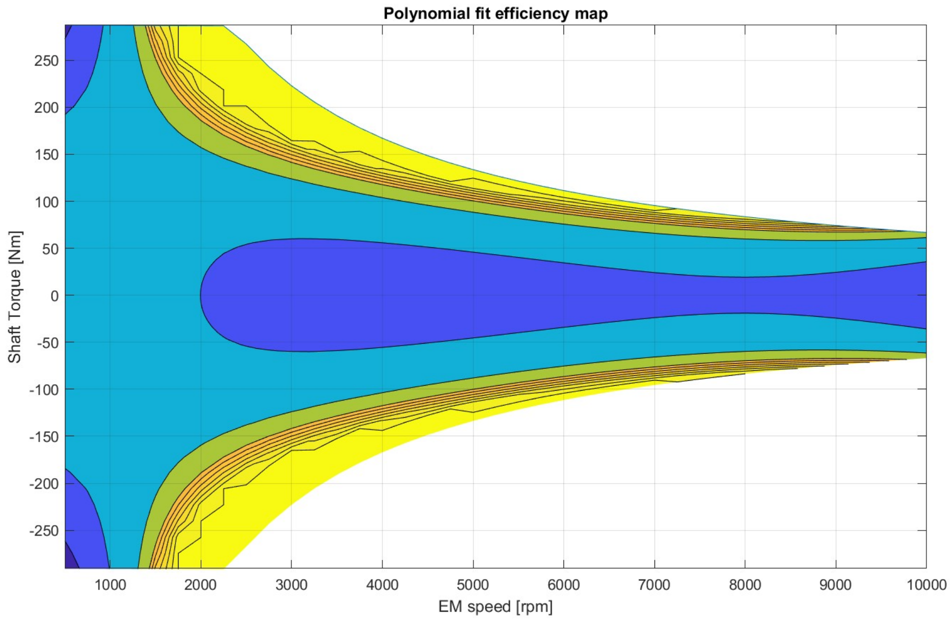
Source: Own authorship (2024).

Figure 4.15: 3D efficiency map with limits adjusted



Source: Own authorship (2024).

Figure 4.16: 2D efficiency map generated by polynomial equation



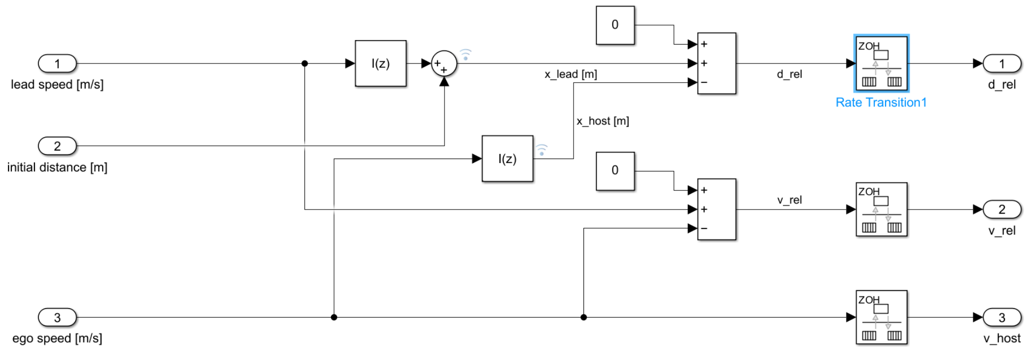
Source: Own authorship (2024).

The main reason for the difference is the 0 efficiency points in the experimental data , when the torque is close to 0. But, as it will be showed in the simulation sections, the MPC controlled performed well with the described polynomial fit for the EM efficiency.

4.11 Car-Following Scenario Block

This simple block is available in [16] in the model in the loop of the 500e, and it's objective is to simulate a more realistic response from the by adding a delay into the lead vehicle via simple integrator blocks and rate transition blocks. The Simulink model is exposed in figure 4.17.

Figure 4.17: Car-following scenario Simulink model



Source: [16].

The integrator blocks all have a unitary integrator gain.

4.12 CTG Policy Controller

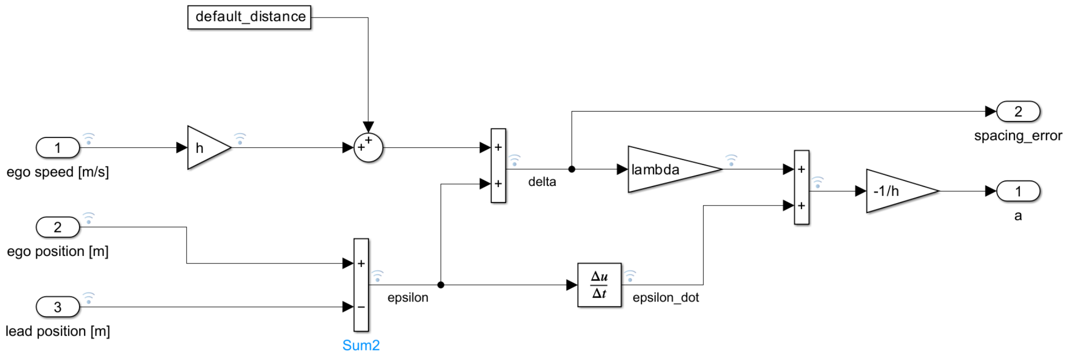
The CTG controller used is for the car following more realistic scenarios on the MPC controlled model, and are based on the model in the loop of the 500e provided by [16].

The CTG policy is mainly used for providing a acceleration reference based on a constant time gap between the leading and the ego vehicles. It also ensures a platoon stability, which a simple PID controller based only on the single vehicle sensor information (without V2V communication) is not capable of doing.

Since this controller is not the main scope of this thesis, only the modeling and the motivation will be described in the present section.

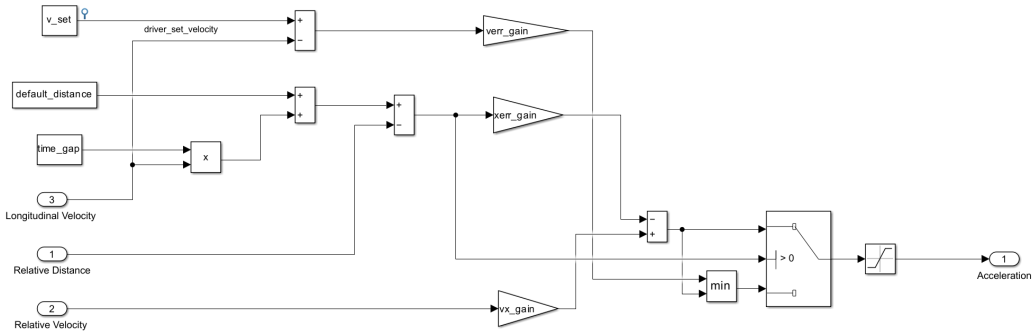
Figures 4.18 and 4.19 show the simpler and the more complete CTG model. From the models, the CTG equations are exposed in 4.23 and 4.23.

Figure 4.18: Simplified CTG controller Simulink model



Source: [16]

Figure 4.19: Complete CTG controller Simulink model



Source: [16]

$$a_{CTG} = -\frac{\lambda}{h}(\dot{x}_{ego}h + d_{default} + \epsilon) - \frac{\dot{\epsilon}}{h} \tag{4.22}$$

$$a_{CTG} = -(d_{default} + \dot{x}_{lead}h - \Delta d)K_{xerr} + vx_{gain}\Delta v \tag{4.23}$$

h is the time gap in seconds, x_{ego} is the ego/following vehicle position, x_{lead} is the leading vehicle position, λ is a tuned parameter, $\epsilon = x_{ego} - x_{lead}$ is the relative distance, Δd is the relative distance and Δv is the relative velocity, both from the car-following scenario block described in 4.11, $d_{default}$ is the default distance set to be maintained, K_{xerr} is tuned spacing error gain, and vx_{gain} is the relative velocity gain. All the parameter are tuned or provided by [16], and are exposed in the code snipped below, and in the appendix B.

```
% ACC and CTG Contoller parameters
default_distance = 50; % reference distance from leading vehicle
[m]
tau = 0.5; % vehicle LTI model [s]
h = 4*tau; % time gap [s] (h > 2*tau)
lambda = 0.5; % CTG parameter [-]
Td = 0.01;
s = tf('s');
P = 1/(tau*s + 1);  % Vehicle simplified plant
v_set = 40;         % ACC set velocity [m/s]
time_gap  = 3;      % ACC time gap [s]
verr_gain = 0.1;    % ACC velocity error gain - CTG
xerr_gain = 0.3;    % ACC spacing error gain - CTG
vx_gain   = 0.5;    % ACC relative velocity gain - CTG
max_acc   = 2;      % Maximum acceleration [m/s^2]
min_acc   = -3;     % Minimum acceleration [m/s^2]
```

The saturation of the acceleration in the complete CTG model is defined by the minimum value of -3 and 2 m/s^2. The transfer function defined by P is the vehicle plant for the simplified ACC scenario, the plant is given by 4.24, and it is the responsible for inserting the delays in the model.

$$\ddot{x} = \frac{1}{\tau s + 1} a \tag{4.24}$$

4.13 State Space Model

To be set as the prediction model of the MPC, the complete system is described in a state space form, with the electric motor torque - that is the control output - set as the input of the system. The three states set to the system, as mentioned before, are:

- Battery State of Charge (SOC): $x_1 = SOC$.
- Vehicle position: $x_2 = x$.
- Vehicle velocity: $x_3 = \dot{x}$.

The SS model input is the electric motor torque (control output) in the previous time instant - T_{EM}.

$$u = T_{EM}$$

The rolling resistance force F_{roll} is given by equation 4.5 and the aerodynamic force F_{aero} is adapted to be computed as function of the state:

$$F_{aero} = 0.5 C_d \rho A_f x_3^2 \tag{4.25}$$

the grade force is not considered in this case, the road inclination is not considered for the sake of simplicity.

The electric motor (EM) equations as functions of the states are showed in equations 4.27 through 4.29.

$$\omega_{EM} = \frac{x_3}{\phi} \tag{4.26}$$

$$\omega_{EM,rpm} = \omega_{EM} \frac{30}{\pi} \tag{4.27}$$

$$\eta_{EM} = E_{EM}(\omega_{EM,rpm}, T_{EM}) \tag{4.28}$$

$$P_{EM} = \omega_{EM} T_{EM} \tag{4.29}$$

The parameter ϕ is used to facilitate the development of the system equations and is given by 4.30.

$$\phi = \frac{r_w}{\tau_{gb}} \tag{4.30}$$

With respect to the battery, its power is given by 4.15. And the battery voltage, resistance and current as function of the states are exposed in 4.32 through 4.34.

$$P_b = \frac{P_{EM}}{(\eta_{EM}\eta_{inv})^{\mathrm{sign}(P_{EM})}} \tag{4.31}$$

$$V_{oc} = N_s V(x_1) \tag{4.32}$$

$$R_o = \frac{N_s}{N_p} R(x_1) \tag{4.33}$$

$$I_b = \frac{V_{oc} - \sqrt{V_{oc}^2 - 4R_o P_b}}{2R_o} \tag{4.34}$$

The functions E_{EM}, V and R are polynomial fits of the efficiency, voltage and resistance interpolated experimental data and are discussed in 4.10.

Finally, the state equations of the complete system are exposed in 4.36, 4.37 and 4.37.

$$\dot{x}_1 = f(x_1, x_3, u) = -\frac{I_b}{Q_{nom}\eta_{batt}^{\text{sign}(I_b)}} \tag{4.35}$$

$$\dot{x}_2 = f(x_3) = x_3 \tag{4.36}$$

$$\dot{x}_3 = f(x_3) = \frac{(\eta_{gb}/\phi)T_{EM} - F_{roll} - F_{aero}}{M_{veh}} \tag{4.37}$$

The state derivative vector is exposed in 4.38.

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} -\frac{I_b}{Q_{\text{nom}}\eta_{\text{batt}}^{\text{sign}(I_b)}} \\ x_3 \\ \frac{(\eta_{\text{gb}}/\phi)T_{\text{EM}} - F_{\text{roll}} - F_{\text{aero}}}{M_{\text{veh}}} \end{bmatrix} \tag{4.38}$$

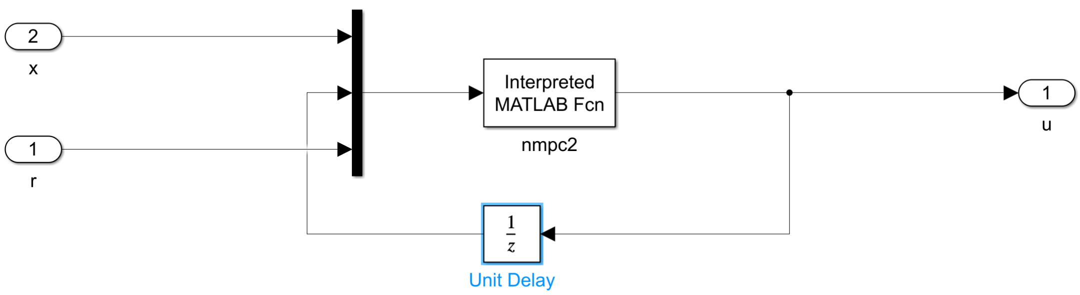
And the output vector is given by 4.39

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{4.39}$$

4.14 MPC Block

The MPC block from Simulink is illustrated in figure 4.20. The unit delay block has an initial condition of 0 and a sampling time as the one defined in the MPC parameters $Ts = 0.05$ s.

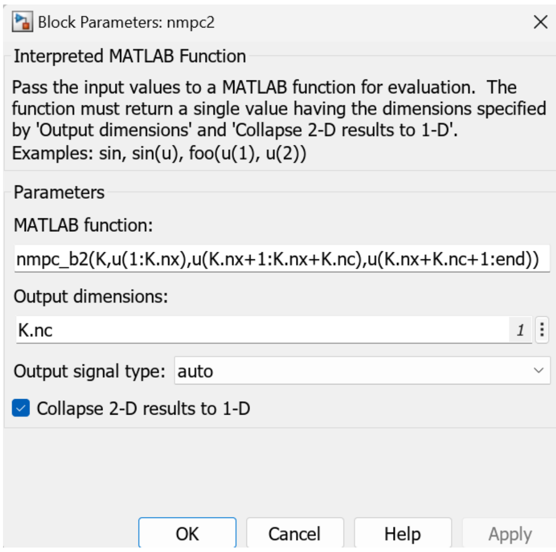
Figure 4.20: MPC Simulink block



Source: Own authorship (2024).

The reference block input (r) is the values of the reference states in each simulation time instant, while the states input (x) is feedback from the model plant. The interpreted MATLAB function block has "nmpc_b2.p" file as parameter, and its arguments are the MATLAB struct "K" assembled using the "nmpc_design_4b.p" and the Mux virtual vector divided in states, control feedback and reference. Figure 4.21 shows the parameters of the MATLAB function block with the .p file in it.

Figure 4.21: Interpreted MATLAB function block parameters



Source: Own authorship (2024).

"K.nc" is the control output dimension, and the parameters of the "nmpc_b2" function follows the dimensions of the input Mux block, being the first argument the states dimension, then the control output dimension that is feedback to the block, and finally the reference dimension, that has the same dimension as the system states (x).

The "K" struct parameters, or MPC parameters are the following:

```
Ts = 0.05; % Sampling time
par.nx          = 3; % number of states
par.nu          = 1; % control elements number
par.ny          = 3; % number of outputs
par.model = @prediction_longitudinal_model; %Modello di predizione
par.ub          = 250; % Upper bound saturazione input -> maximum value for control output
par.lb          = -250; % Lower bound saturazione input -> minimum value for control output
par.tol         = 1; % Reference tolerance
par.Nfev        = 150; % Iteration number of fmincon in cost function (default 200)
```

```
    par.Ts          = Ts;
    par.R = 1; % definite positive matrix for cost function
    par.P = diag([0;0;1]); % definite positive matrix for cost
function
    par.Q = diag([0;0;1]); % definite positive matrix for cost
function
    par.Tp = 5*Ts; % Prediction horizon

    K = nmpc_design_4b(par); %Generazione parametri design NMPC
```

as described in the code, 'nx', 'nu', and 'ny' are the number of states, control outputs (same as 'nc') and number of outputs. The 'model' parameter is the prediction model function, or the state-space model of the powertrain, receiving the control input as the EM torque, with its code exposed in B. Upper and lower bounds of the torque control output are set in 'ub' and 'lb' respectively. 'tol' is the refernce tolerance, 'Nfev' the number of iterations in the cost function, 'Ts' is the sampling time, 'Tp' the prediction horizon (which is always an integer multiple of the sampling time).

Finally, matrices P, Q and R are the ones that come from the Ricatti equation, as in LQR controllers. They are all diagonal matrices, where Q and P control the energy of the state error, while R matrix control the energy of the control input.

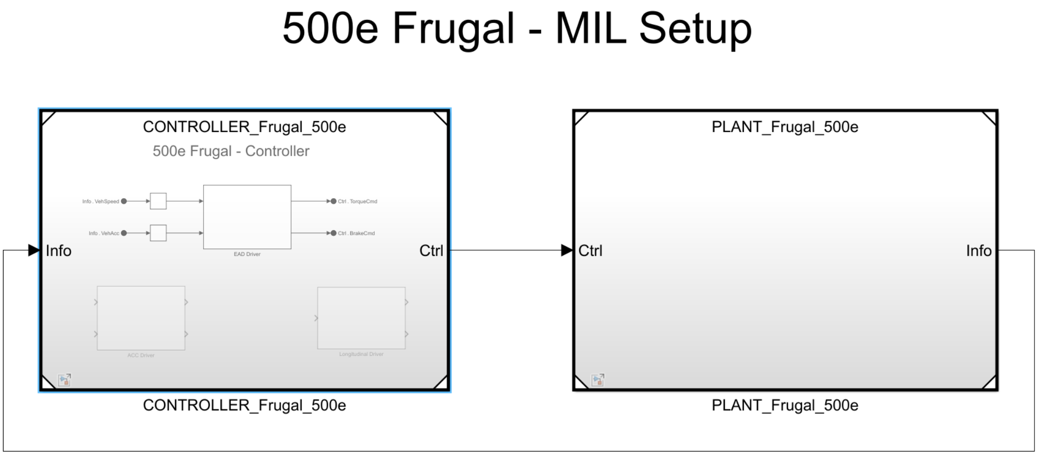
Since the P and Q matrices control the states, and the reference is a state in a time instant, the values are set to zero in the matrices diagonals according to the set reference. For example in the first MPC controlled model simulation setup, the velocity (third state) is set as the only reference, thus, matrices P and Q should have zero for the other states, since they are inputted as null values in the state reference vector.

Matrix R is the responsible to account for the control output/feedback input.

4.15 Model in the Loop (MIL) 500e

The complete model provided by Politecnico di Torino and [16] is briefly discussed. The model is divided into the Controller block and the Plant block, as exposed in figure 4.22.

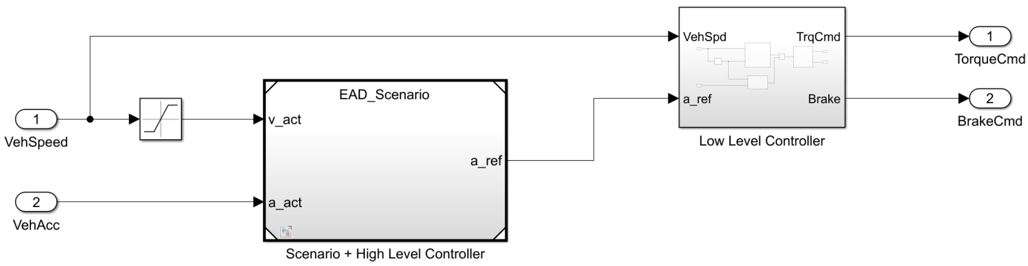
Figure 4.22: MIL setup - Controller and Plant Simulink blocks



Source: [16]

Figure 4.23 shows the Controller block. The high level controller block is very similar to the complete ACC scenario setup discussed in section 5, it has the car following block with the CTG policy block in it, like the ones discussed in sections 4.11 and 4.12.

Figure 4.23: MIL setup - Controller Simulink model

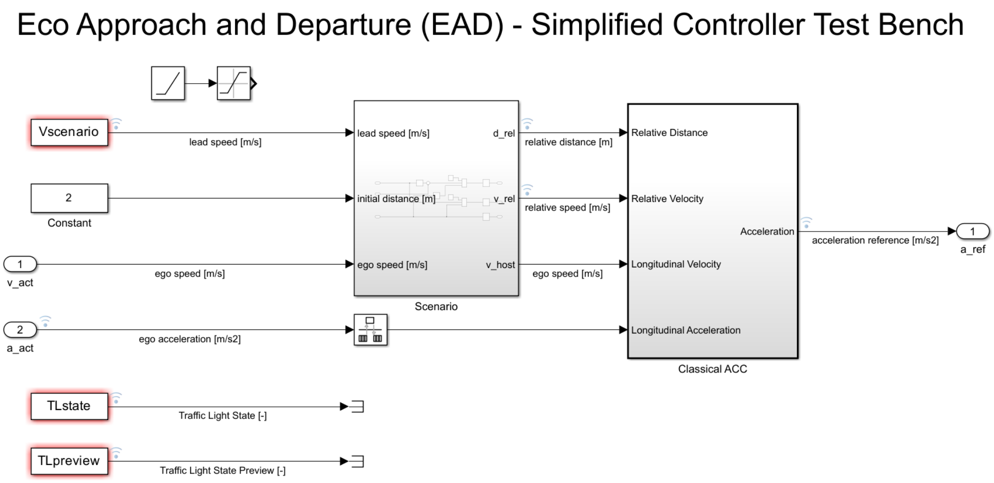


Source: [16]

The low level controller block has as input the acceleration provided by the high level controller, and through a backward model (very similar to the one discussed in this thesis. The required torque is computed, and a driver model with a PI controller simulates the driver required torque. These signals are summed and the torque and brake commands are generated by the Torque Distribution block. These commands are sent to the complete vehicle plant and the simulations are done.

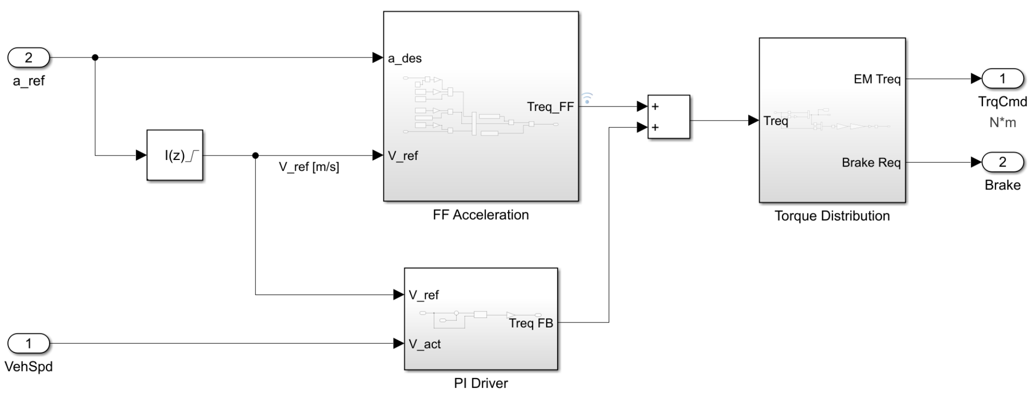
The high level controller block is exposed in figure 4.24, and the low level controller is illustrated in figure 4.25.

Figure 4.24: MIL setup - High level controller Simulink model



Source: [16]

Figure 4.25: MIL setup - Low level controller Simulink model



Source: [16]

Chapter 5

Simulations

The simulations made in this thesis where the following:

- Backward (BW) model: for a first validation of the model parameters and equations, a simple backward approach simulation is made, using the WLTP3 driving cycle speed and accelerations profiles as reference and inputs of the model.

- Backward and forward (BW-FW) model simulation: prior to the MPC controller implementation, the backward model is used as an electric motor shaft torque provider - which is the MPC control output. The EM shaft torque is the input of the forward model (with the same parameters and force equations as the backward model) and the resultant states of this model are used as reference for the controlled model. The backward-forward model serves the purpose of being the states reference provider.

- MPC controlled model: done with the MPC controlled model, and the reference states for the comparison are provided by the BW-FW model. 4 different setups are simulated in this case. A first setup with the WLTP3 velocity profile as reference; second setup with the WLTP3 position profile as reference; third setup with an ACC (Adaptive Cruise Control) simplified scenario using simplified CTG (Constant Time Gap) controller and vehicle plant to compute the positions and velocities; final scenario with an ACC realistic scenario with the complete longitudinal dynamics vehicle plant to compute position and velocities and a more complete CTG controller.

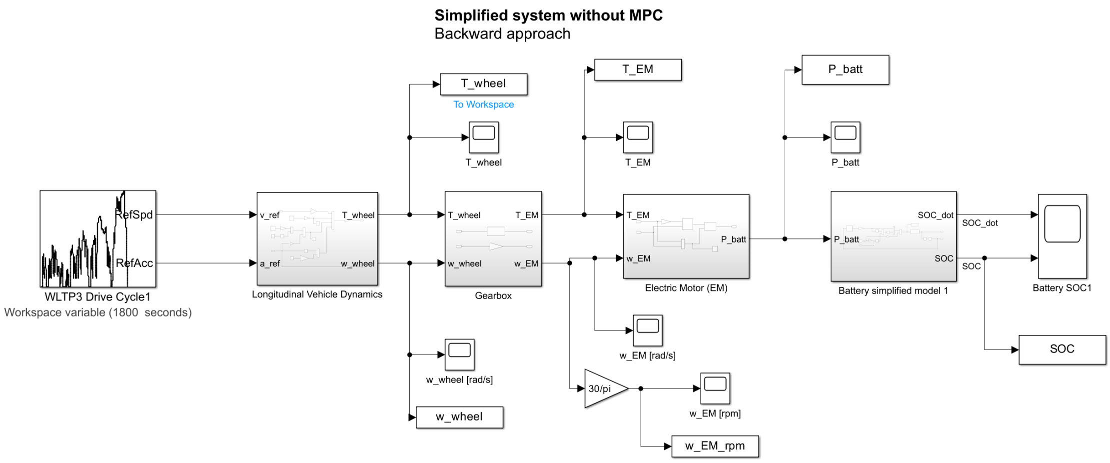
- MIL 500e with MPC: in the final simulation, the tuned MPC controller is used in the complete 500e provided model, and it is responsible for generating the required torque command.

In the figures where there is a comparison between the controlled and uncontrolled results, the signals go until the 400 seconds time instant of the cycle. This zoomed in plot is done for better visualization and analysis of the comparison plots. The complete cycle simulations are available in the appendix A of this thesis.

5.1 Backward Model

The backward model is depicted in figure 5.1, all the blocks have the equation explained in chapter 4, and here, the longitudinal vehicle dynamics block is the one defined in 4.4, using the backward approach.

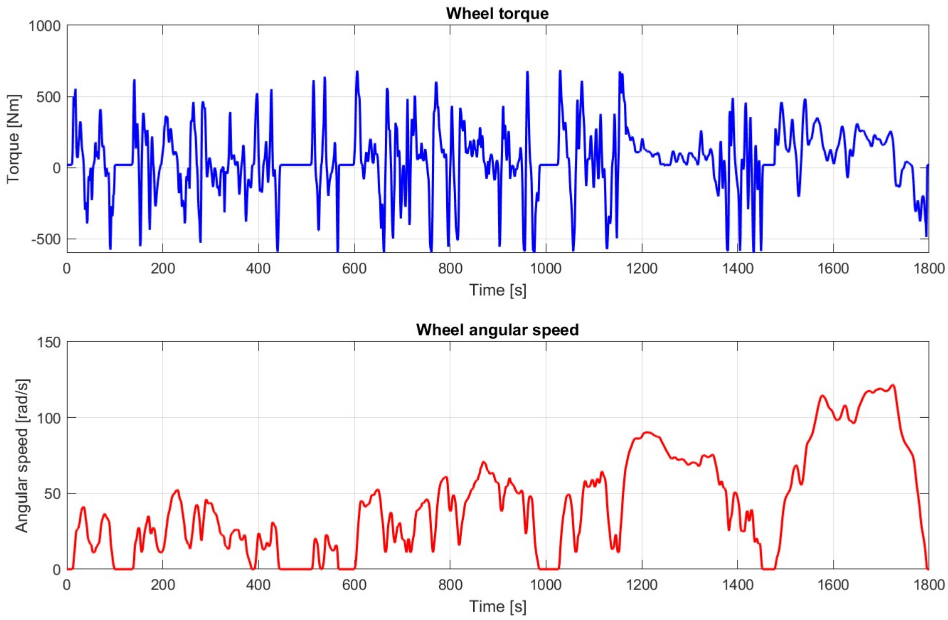
Figure 5.1: Vehicle and powertrain Simulink model - backward approach



Source: Own authorship (2024).

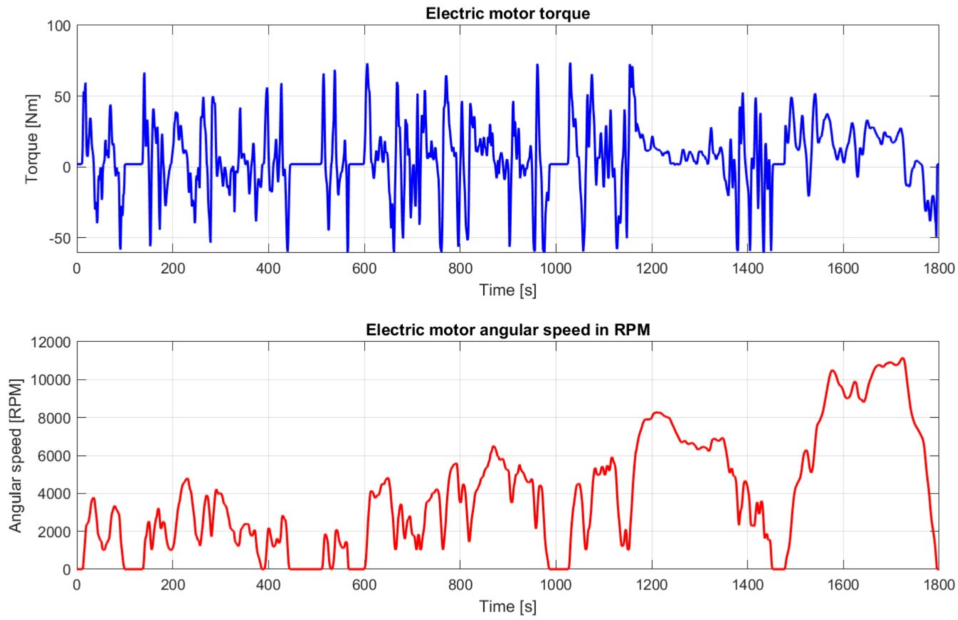
From the simulation, the obtained signals are wheel and electric motor torque, angular speeds, battery power and State of Charge. The wheel torque and angular speed are exposed in 5.2, the electric motor shaft torque and angular speed (in RPM) are shown in figure 5.3, and, finally, the battery signals of power and state of charge are depicted in figure 5.4.

Figure 5.2: Wheel torque and angular speed



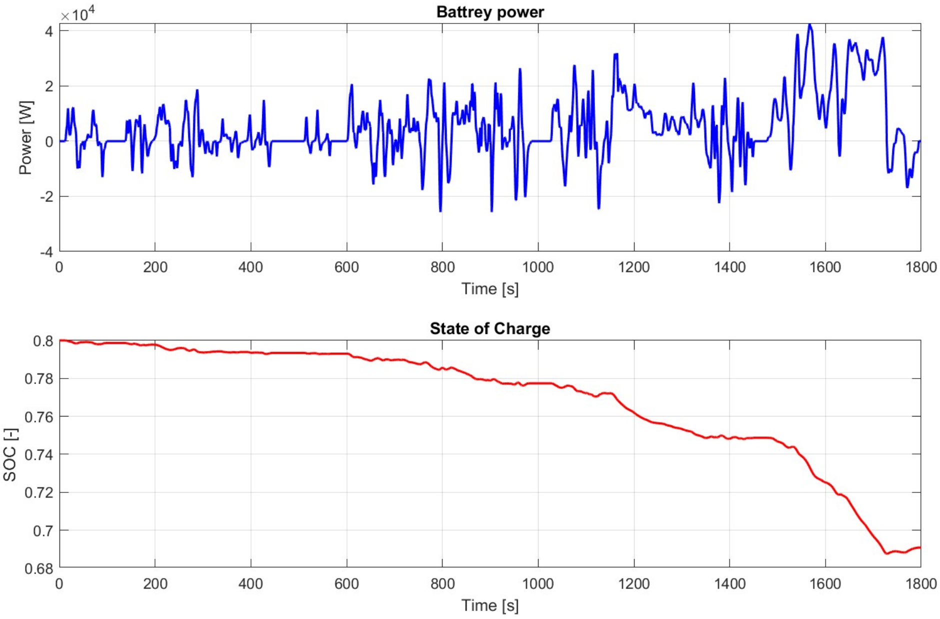
Source: Own authorship (2024).

Figure 5.3: Electric motor shaft torque and angular speed



Source: Own authorship (2024).

Figure 5.4: Battery power and State of Charge



Source: Own authorship (2024).

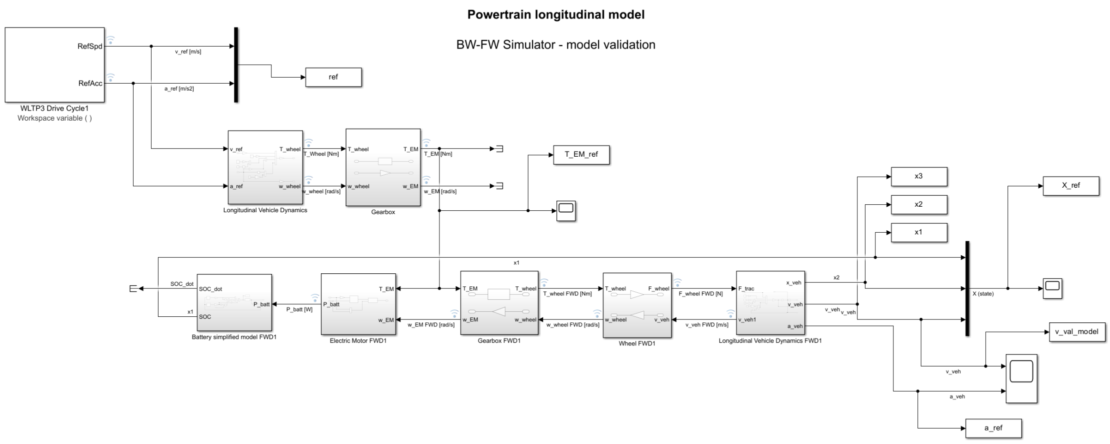
By analyzing the simulation results it is possible to see that the torque and angular speeds both follow the reference speed and acceleration as expected, but with different magnitudes due the gearbox gains, vehicle mass and wheel radius. The EM shaft torque has maximum absolute values around 75 Nm, which is bellow the EM maximum torque of 250 Nm. Considering the battery, its signals of power and SOC are the ones with more variations and don't follow the reference profiles, due to the higher non linearity in the model, and dependency on all the other powertrain components combined. The state of charge finishes the cycle with around 0.7 or 70% charge, with the highest consumption closest to the end of the cycle, due to the higher velocities and consequent higher power and consumption from the battery.

The EM torque result from this model is used as input for the backward-forward (BW-FW) model, that serves as state reference generator and lead vehicle states for the MPC controlled model scenarios.

5.2 Backward-Forward Reference Model

In this case, the backward model explained in 5.1 will be set as the electric motor torque generator - which will be the role of the MPC controller in the final simulations. The torque is the input of a forward complete powertrain model, described in section 4, and depicted in figure 5.5.

Figure 5.5: Backward-Forward Simulink model



Source: Own authorship (2024).

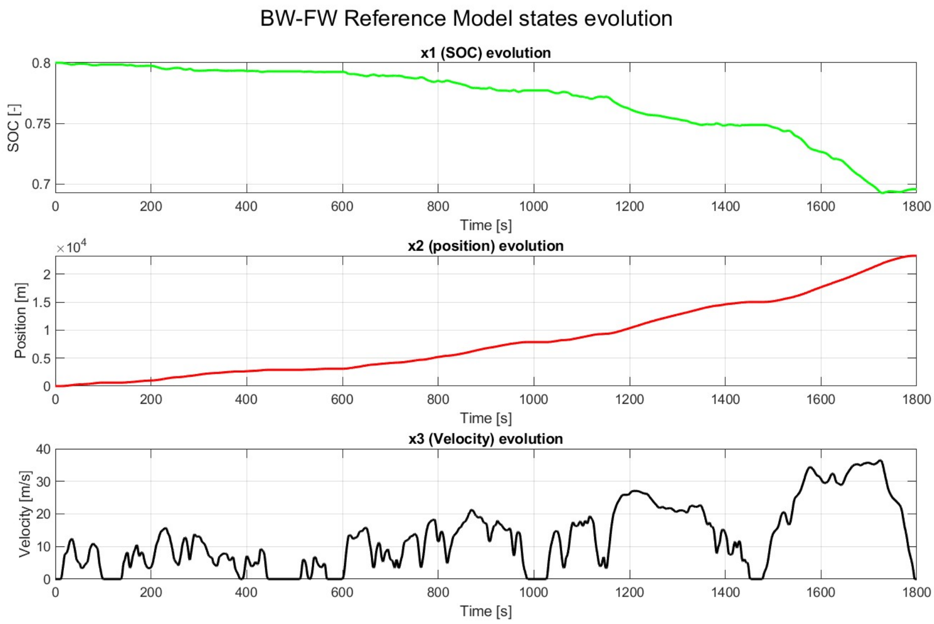
All the blocks present in this model are the ones explained in section 4, the only difference would be the inputs and outputs of the blocks themselves, not the equations. The inputs and outputs ought to be different to respect the physical causality of the system in the forward modeling approach.

The vehicle longitudinal dynamics equation and Simulink blocks for the forward model are shown in 4.5 and 4.7.

In this simulation setup, the evolution of the states are obtained, and provided as reference for the controlled model afterwards. It is also used as the leading vehicle complete model for the ACC scenario comparison between ego and lead vehicle position and speed. The forward model is the complete plant of the vehicle and from where the states are computed.

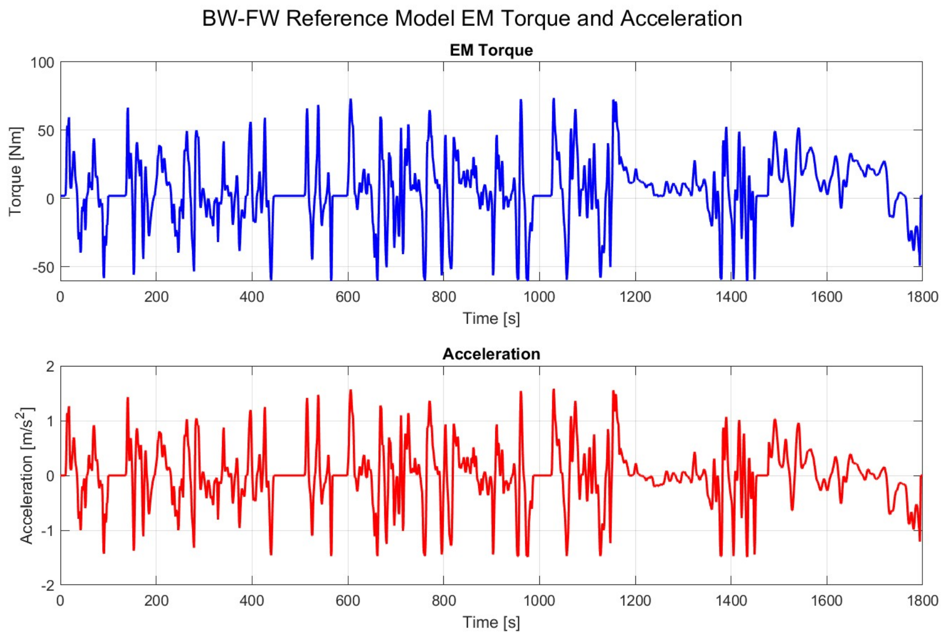
Figures 5.6 and 5.7 show respectively the states evolution of the model and the vehicle acceleration (third state derivative) and EM torque evolution in time. The torque is the same obtained in 5.3, since it is obtained from the backward model.

Figure 5.6: Backward-Forward model states evolution



Source: Own authorship (2024).

Figure 5.7: Backward-Forward model torque and acceleration evolution



Source: Own authorship (2024).

Since there is no delay, or control command, the model states just replicate the reference, and the SOC is the same as obtained in the backward model. That is the reason why this model is used as state reference and as the leading vehicle in the ACC scenarios.

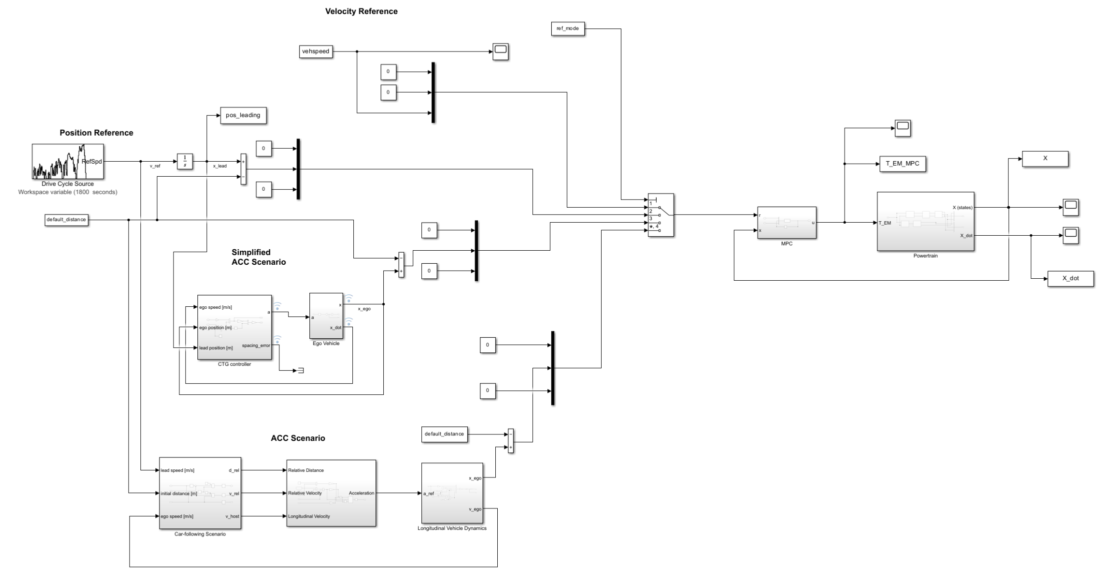
5.3 MPC Controlled Model

The controlled model Simulink is showed in figure 5.8. The different inputs that set the simulation scenario via MPC reference states are controlled by a variable in the script and by the switch Simulink block. There are four diferente scenarios that are simulated, as explained in the beginning of this chapter:

- Velocity reference: velocity profile as the third state reference, other states reference set to 0.
- Position reference: position profile directly integrated from the WLTP3 cycle speed profile set as second state reference. Other states set to 0.
- Simplified ACC: simplified CTG controller and simplified vehicle plant. The simplified vehicle plant output position is the second state reference. Other states set to 0.

- ACC: realistic car following scenario with complete CTG controller. The position resultant from the CTG acceleration integration is the second state reference. Other states are set to 0.

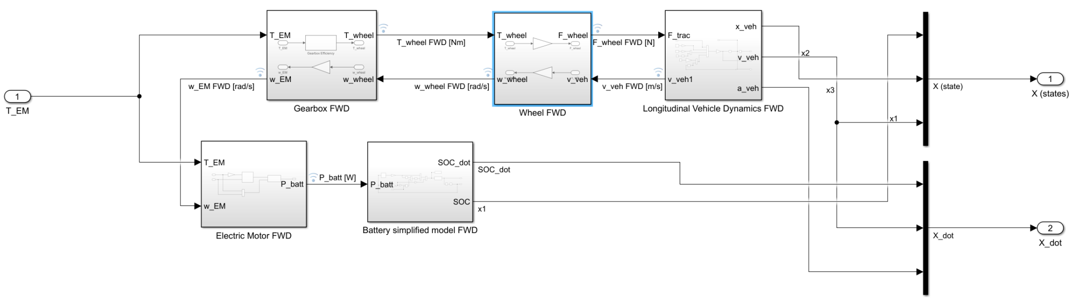
Figure 5.8: MPC controlled Simulink model



Source: Own authorship (2024).

The plant of the system, defined in the Powertrain block, has its Simulink model exposed in 5.9. The blocks of the powertrain are defined in section 4, and are modeled in the forward approach, as the forward model explained in 5.2.

Figure 5.9: MPC controlled Simulink powertrain plant



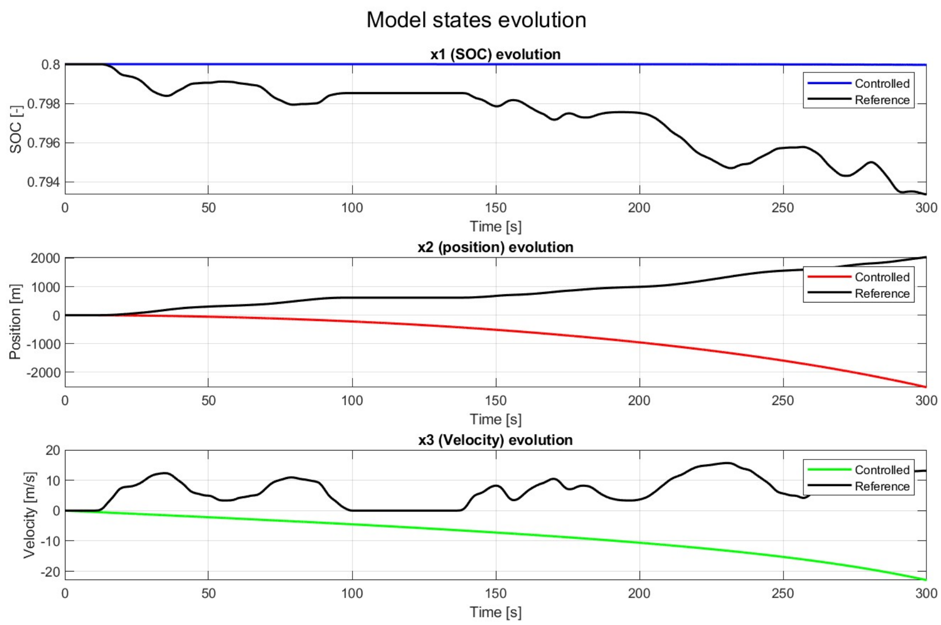
Source: Own authorship (2024).

5.3.1 Velocity Reference

In this first simulation setup, the speed profile from the WLTP3 cycle is the direct input of the third state x_3 reference. The other reference states are set to 0, as it is possible to see in the Simulink model illustrated in 5.8.

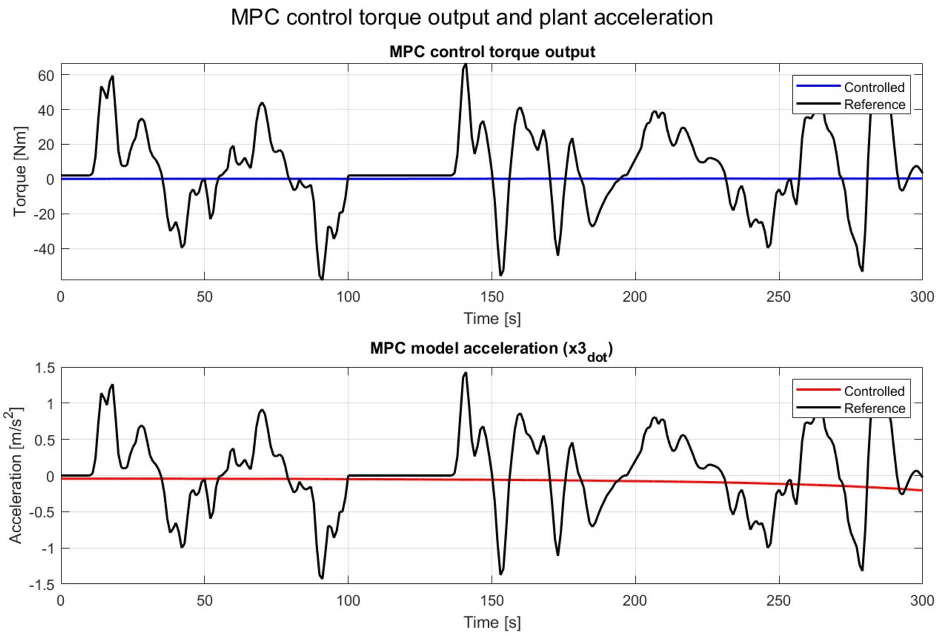
First, to show the importance of the MPC parameters tuning, this first setup with velocity as reference is simulated with the standard parameters showed in section 4.14. The results of the states evolution of the controlled plant compared with the reference model are exposed in figures 5.10 and 5.11. A shorter simulation time is set, just for illustrate the importance of the tuning - 300 seconds from the 1800 second cycle.

Figure 5.10: MPC controlled model states evolution - MPC default parameters



Source: Own authorship (2024).

Figure 5.11: MPC controlled model torque and acceleration evolution - MPC default parameters



Source: Own authorship (2024).

It is possible to see that basically there is no control action, and there is no track of the reference. The states go to negative position and velocities, while the battery SOC remains the same. The torque control and acceleration both decrease slowly.

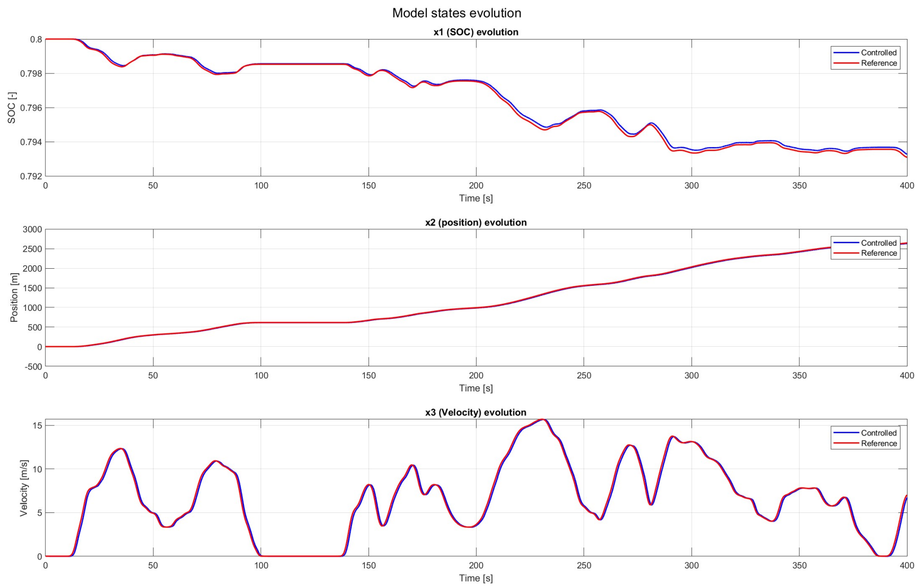
For this setup, the tuned parameters for the controller are exposed in the code section below.

```
Ts = 0.05; % Sampling time
par.nx        = 3; % number of states
par.nu        = 1; % control elements number
par.ny        = 3; % number of outputs
par.model = @prediction_longitudinal_model; %Modello di predizione
par.ub        = 250; % Upper bound saturazione input -> maximum value for control output
par.lb        = -250; % Lower bound saturazione input -> minimum value for control output
par.tol       = 1; % Reference tolerance
par.Nfev      = 150; % Interation number of fmincon in cost function (default 200)
```

```
    par.Ts          = Ts;
    par.R = 0.05; % matrice diagonale definita positiva per cost function
    par.P = diag([0;0;10000]); % matrice diagonale definita positiva per cost function
    par.Q = diag([0;0;1]); % matrice diagonale definita positiva per cost function
    par.Tp = 10*Ts; % Prediction horizon (sempre multiplo intero del Ts)
    K = nmpc_design_4b(par); %Generazione parametri design NMPC
```

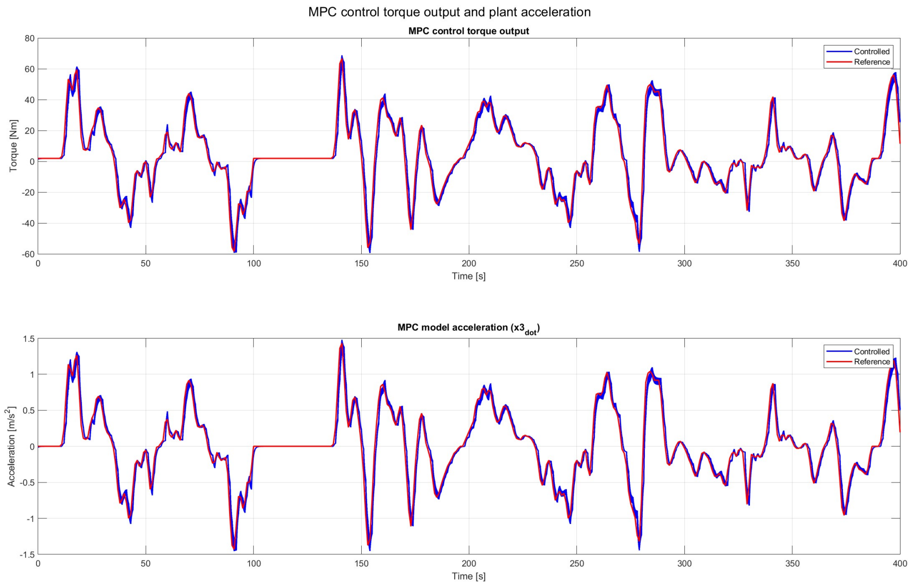
The parameters are tuned via trial and error, adjusting the results to better follow the reference states. Figure 5.12 and 5.13 show, respectively, the states evolution and the electric motor torque and vehicle acceleration - being the derivative of the system third state.

Figure 5.12: MPC controlled model states evolution - velocity reference



Source: Own authorship (2024).

Figure 5.13: MPC controlled model control torque and vehicle acceleration evolution - velocity reference



Source: Own authorship (2024).

By analyzing the results it is possible to conclude that the MPC control torque command made the vehicle follow the reference velocity as desired, and the states evolution are vary similar to the reference uncontrolled models. The torque commands are a bit smoother than the ones from the uncontrolled models, but the trace and the magnitudes are pretty similar.

The purpose of this first simulation by inputting the velocity profile directly as reference, it is possible to conclude that the MPC controller works and it is able to control the powertrain plant to follow a given reference.

The scenario is not realistic, but it accomplished its purposes.

5.3.2 Position Reference

Now, instead of setting directly the velocity from the profile as reference, the desired leading vehicle position is set as reference, by integrating the WLTP3 profile directly. The initial condition for the integrator is set to the desired default distance of 10 meters. This value is kept the same for the ACC simulation scenarios.

This is also a non realistic scenario, since the profile is being directly inputted with no delays or proper treatment to mimic a car following case. It is just to test the MPC controller when the second state (position) is set as refernece, and the other states are set to 0.

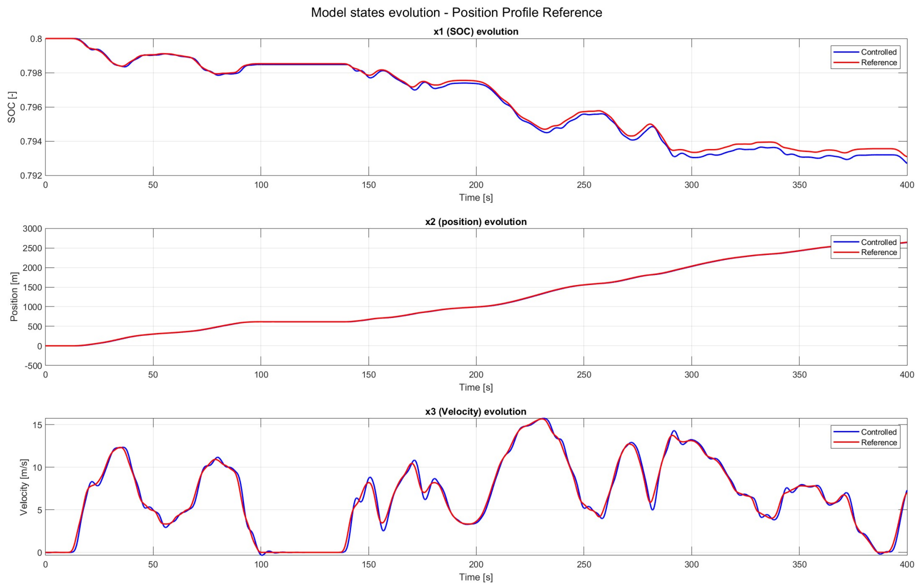
The parameters in this scenario are defined below:

```
Ts = 0.05; % Sampling time

par.nx        = 3; % number of states
par.nu        = 1; % control elements number
par.ny        = 3; % number of outputs
par.model = @prediction_longitudinal_model; %Modello di predizione
par.ub        = 250; % Upper bound saturazione input -> maximum value for control output
par.lb        = -250; % Lower bound saturazione input -> minimum value for control output
par.tol       = 1; % Reference tolerance
par.Nfev      = 150; % Interation number of fmincon in cost function (default 200)
par.Ts        = Ts;
par.R = 0.05; % matrice diagonale definita positiva per cost function
par.P = diag([0;50000;0]); % matrice diagonale definita positiva per cost function
par.Q = diag([0;1;0]); % matrice diagonale definita positiva per cost function
par.Tp = 10*Ts; % Prediction horizon (sempre multiplo intero del Ts)
K = nmpc_design_4b(par); %Generazione parametri design NMPC
```

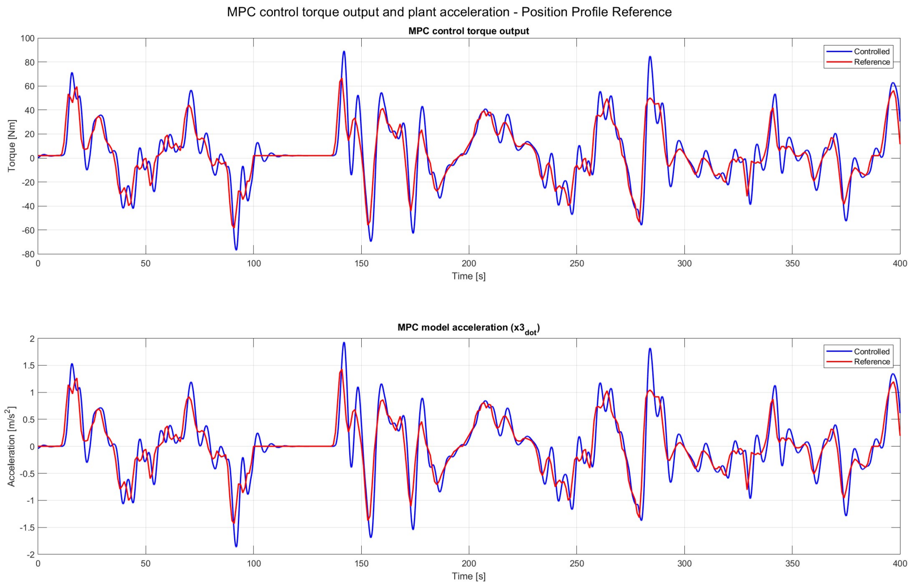
Figure 5.14 and 5.15 show, respectively, the states evolution in comparison with the reference, and the control torque and vehicle acceleration evolution. Also, the ego (controlled) and leading vehicle positions are computed in 5.16.

Figure 5.14: MPC controlled model states evolution - position reference



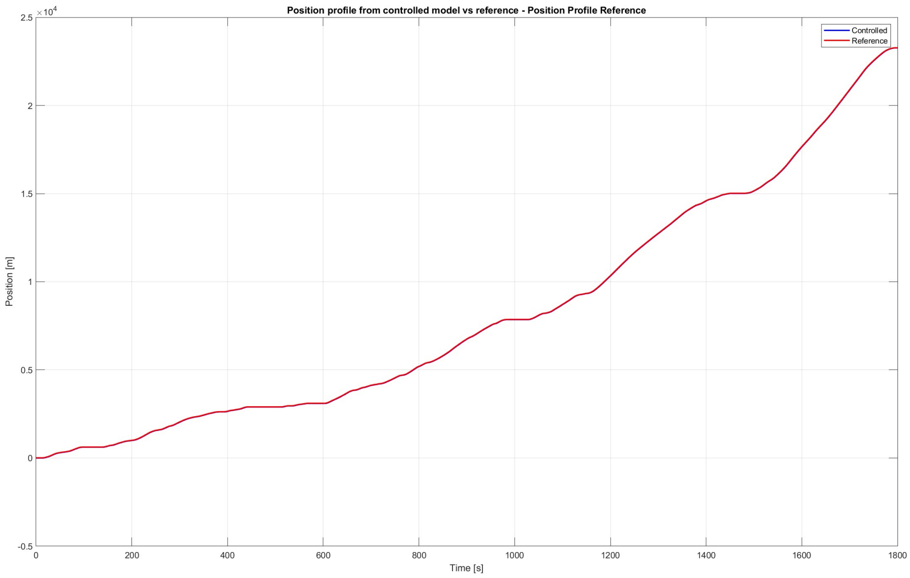
Source: Own authorship (2024).

Figure 5.15: MPC controlled model control torque and vehicle acceleration evolution - position reference



Source: Own authorship (2024).

Figure 5.16: MPC controlled model ego and lead positions - position reference



Source: Own authorship (2024).

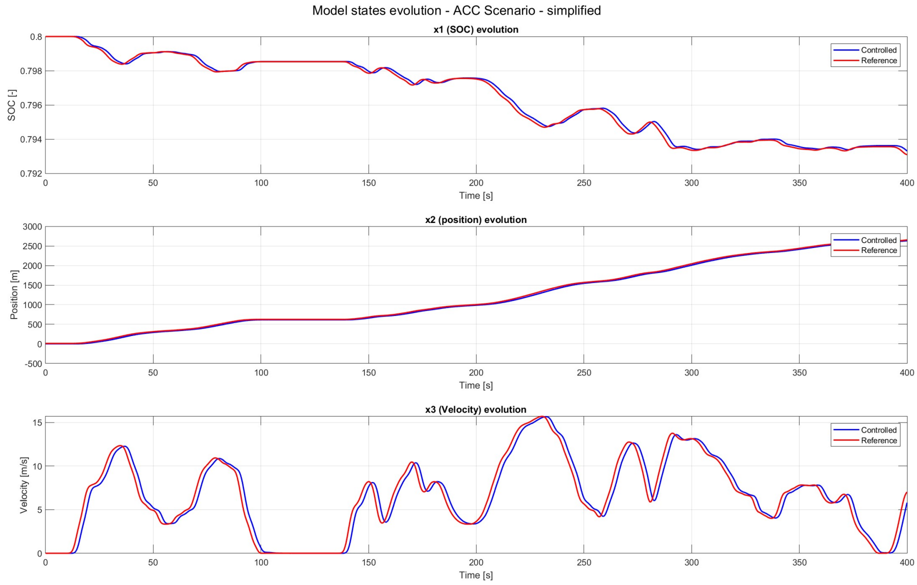
From the results, it is possible to conclude that the MPC controller works properly and the parameters are properly tuned when the given reference is the second state (position) and the others are set to 0.

5.3.3 ACC Simplified Scenario

For a first ACC approach and to properly tune the parameters with a simpler model, a simpler CTG controller is used to generate the acceleration reference which is the input of the transfer function that represents the plant of the vehicle response to the acceleration - described in equation 4.24 - and also simulates a delay for the computation of the position profile.

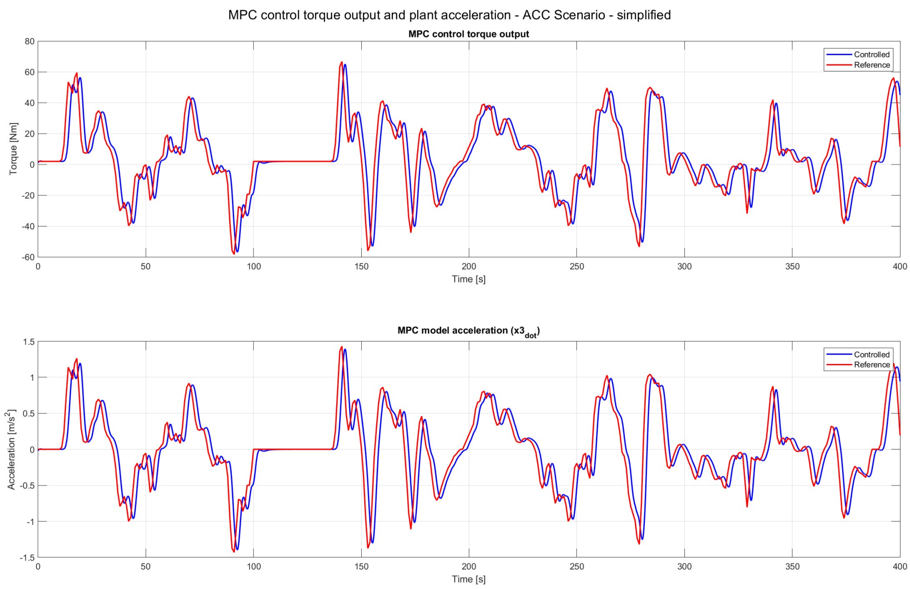
The CTG controller inputs are the lead vehicle position, which is the same used in the position reference case, and the feedback position and velocity from the ego vehicle, generated from the simple transfer function from 4.24. This ego position from the transfer function model is the one inputted as second state reference in the MPC.

Figure 5.17: MPC controlled model states evolution - simplified ACC scenario



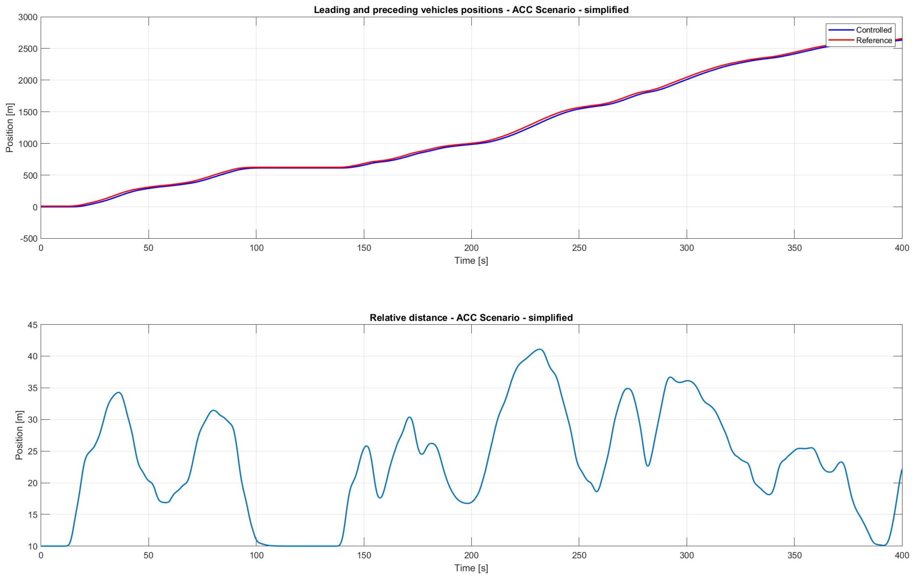
Source: Own authorship (2024).

Figure 5.18: MPC controlled model control torque and vehicle acceleration evolution - simplified ACC scenario



Source: Own authorship (2024).

Figure 5.19: MPC controlled model ego and lead positions - simplified ACC scenario



Source: Own authorship (2024).

From the results exposed in 5.17, 5.18 and 5.19 it is possible to see the states evolution, control torque and vehicle acceleration, and position traces and relative distances, respectively. In this more realistic scenario, the MPC successfully followed the reference, and there was no collision - relative distance is always positive and the position traces do not intercept.

It is a more realistic scenario then the other two presented so far, and the tuned parameters have been demonstrated as satisfactory for this simpler ACC scenario.

5.3.4 ACC Complete Scenario

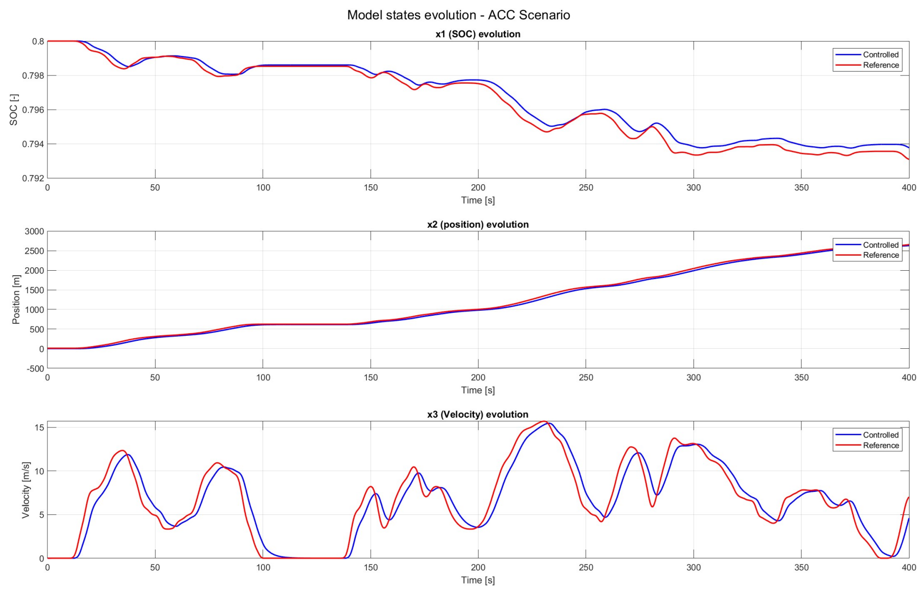
In the more realistic case, the reference is given by the complete CTG policy block, attached to a car-following scenario block and the complete longitudinal dynamics of the vehicle.

The car-following block is detailed in 4.11, and it is responsible to generate the inputs of the complete CTG policy block. By using simple Integral controls with unitary integral gain and rate transition blocks, delays to the data signals and smother transitions are obtained, simulating a more realistic sensor data acquisition and transmission. It generates relative distance and velocity signals, as well as lead velocity. Those information are provided as input for the CTG block.

The complete CTG block, explained in section 4.12, provides the acceleration reference that is integrated to generate the position reference for the MPC block.

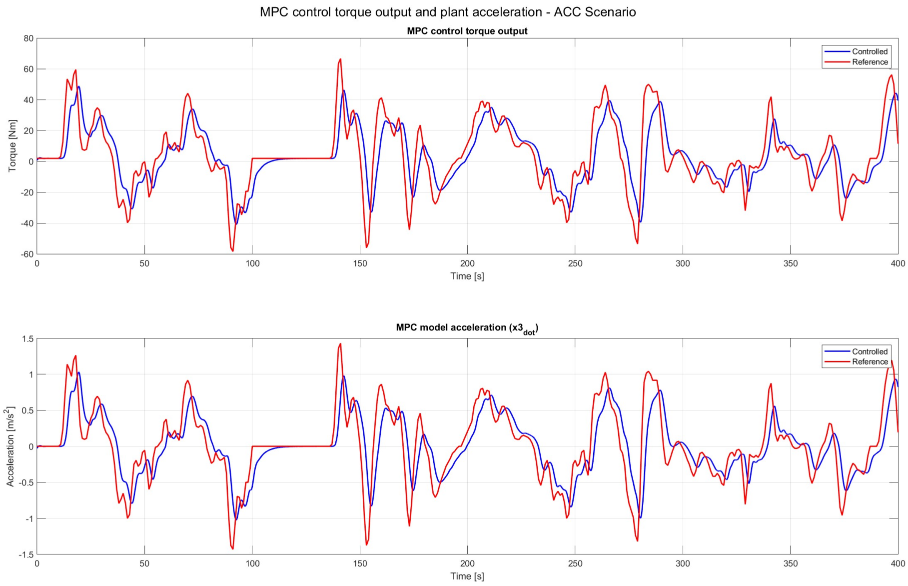
Results for this simulation setup are exposed in 5.20, 5.21 and 5.22.

Figure 5.20: MPC controlled model states evolution - complete ACC scenario



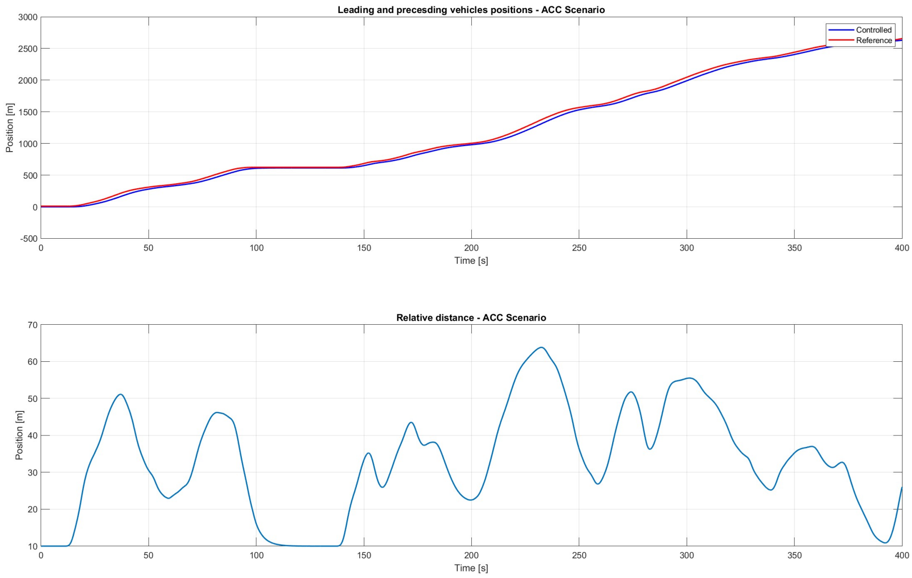
Source: Own authorship (2024).

Figure 5.21: MPC controlled model control torque and vehicle acceleration evolution - complete ACC scenario



Source: Own authorship (2024).

Figure 5.22: MPC controlled model ego and lead positions - complete ACC scenario



Source: Own authorship (2024).

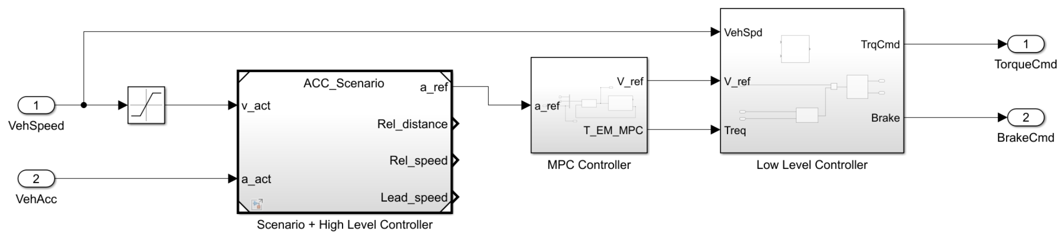
It is possible to see that the MPC produced a torque signal capable of making the vehicle follow the lead vehicle. The stability around the 10 meters default spacing is not obtained in this simulation, a better tuning of the CTG parameters may enhance the capability of the model to keep the desired distance, but this is not the focus of this thesis. Overall, the MPC strategy showed satisfactory results in this more realistic case.

5.4 Model in the Loop 500e with MPC controller

The MPC controller with the simplified plant is inputted in the Controller block, following the EAD Scenario block with the scenario and high level controller modeling, which provides the reference acceleration signal. The simplified plant is responsible for providing the states feedback for the controller, as in the simplified plant simulations discussed previously.

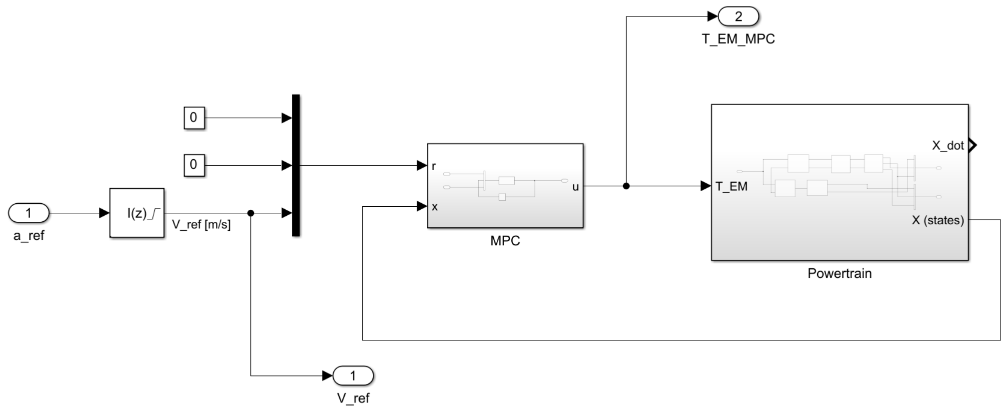
Thus, for the MPC implementation, the original controller block exposed in 4.23 is modified, and the resultant controller model is exposed in 5.23. The acceleration reference provided by the high level controller is now the reference input of the MPC. Its integral is computed and the resultant velocity is set as the second state reference. The MPC block is exposed in figure 5.24, the components are the ones discussed along this thesis, with the MPC block that has as input the reference state, and the EV simplified plant that provides the feedback states. The low level controller is modified, and it receives directly the control torque provided by the MPC, there is no backward model anymore to compute the torque command. The modified low level controlled block is exposed in figure 5.25.

Figure 5.23: MIL setup with MPC - controller Simulink model



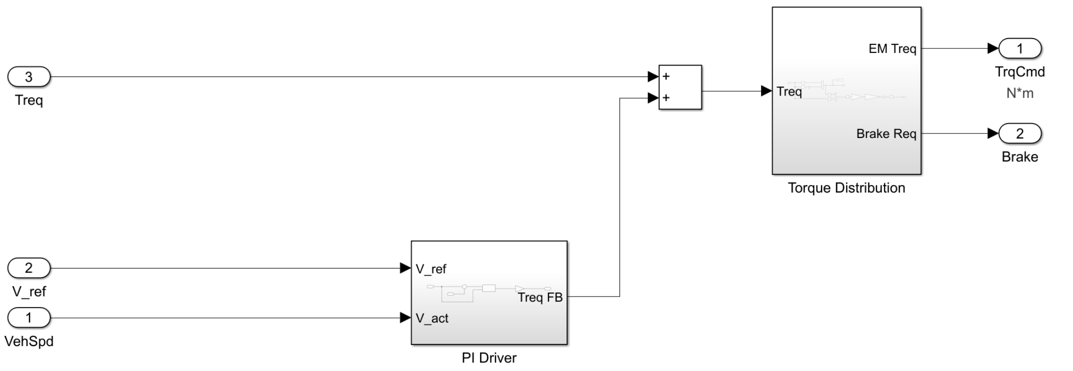
Source: Own authorship (2024).

Figure 5.24: MIL setup with MPC - MPC Simulink model



Source: Own authorship (2024).

Figure 5.25: MIL setup with MPC - low level controller Simulink model

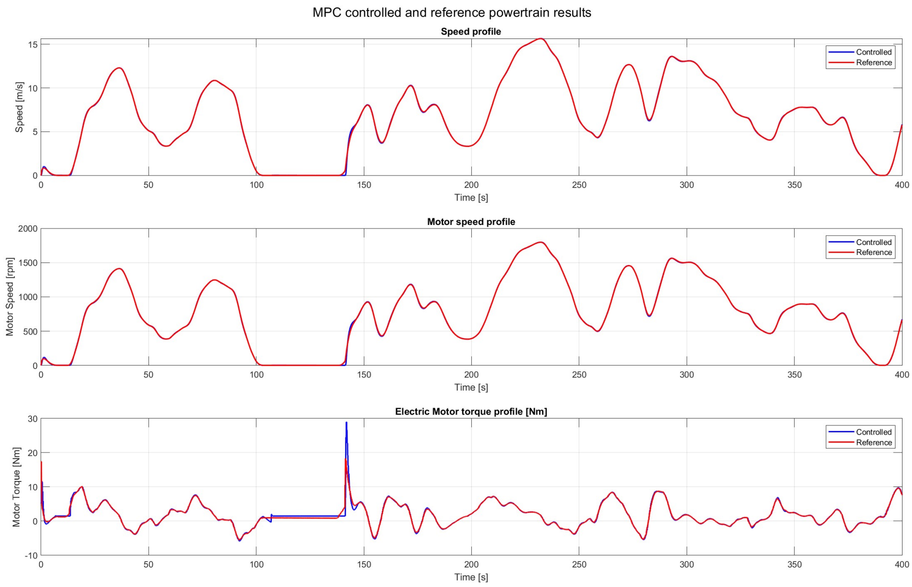


Source: Own authorship (2024).

Finally, the whole MIL setup is simulated with the MPC controller providing the required torque. The signals acquired from the powertrain provided by the Simulink model developed in [16] are: vehicle speed in m/s, electric motor speed in RPM and torque in Nm, battery charge in Ah and electrical efficiency in kWh/100km, and relative distance in meters. All this signals are obtained for the reference (unmodified) model and for the MPC controlled model, where the controller is responsible for generating the required torque from the inputted position reference, as discussed previously. In both models the ACC scenario is tested with the CTG policy. The results are exposed together, comparing the controlled with the uncontrolled model.

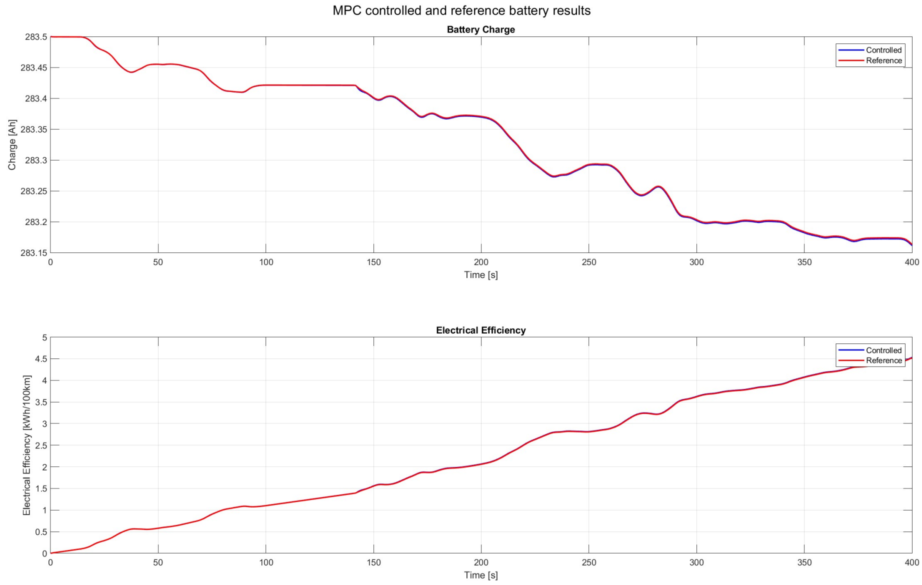
Figure 5.26 show the powertrain signals comparison (vehicle speed, and electric motor torque and angular speed). Figures 5.27 and 5.28 shows the result of the battery signals, and a zoomed in signal, respectively. They expose the battrey charge and electrical efficiency. Finally, figures 5.29 and 5.30 elucidate the relative distance signals, and the zoomed in trace.

Figure 5.26: MIL results - MPC and reference powertrain signals



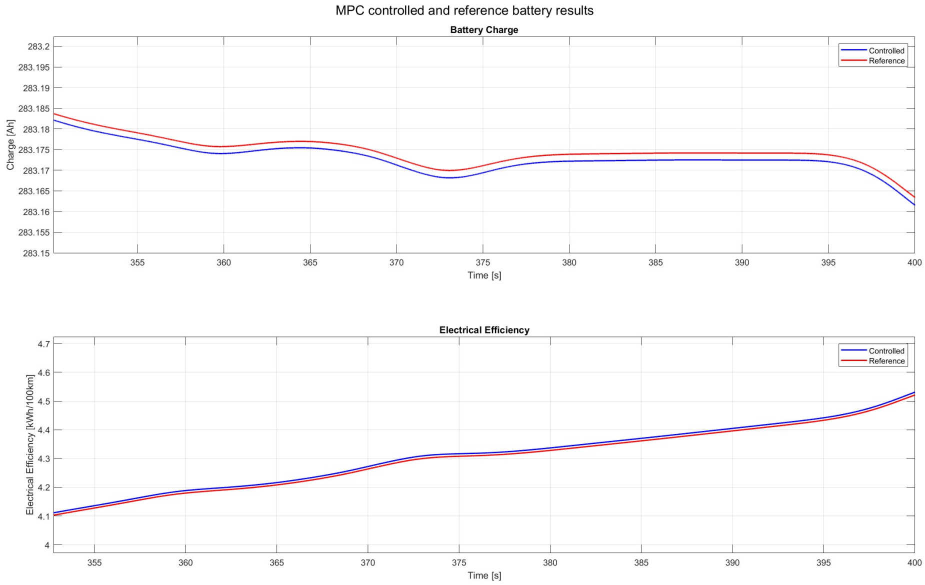
Source: Own authorship (2024).

Figure 5.27: MIL results - MPC and reference battery signals



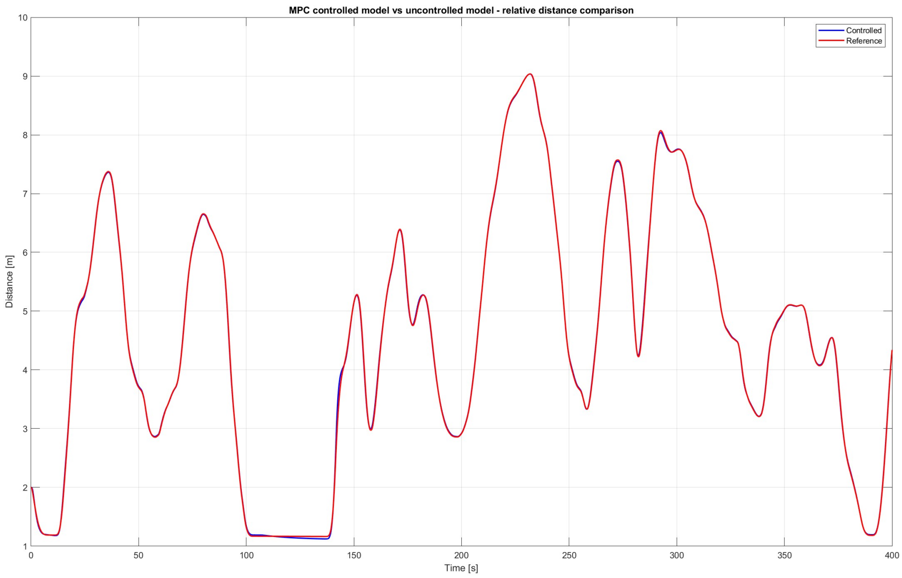
Source: Own authorship (2024).

Figure 5.28: MIL results - MPC and reference battery signals zoomed in



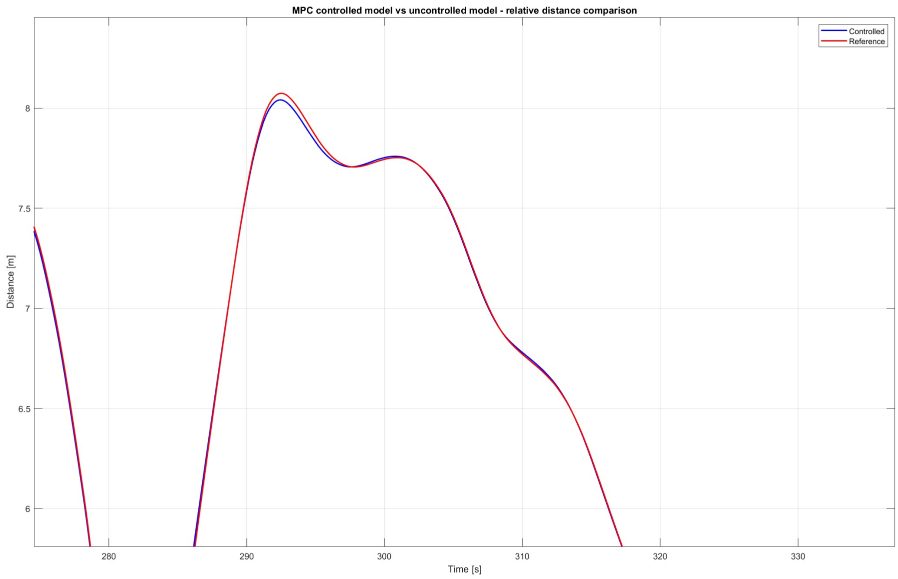
Source: Own authorship (2024).

Figure 5.29: MIL results - MPC and reference relative distance signals



Source: Own authorship (2024).

Figure 5.30: MIL results - MPC and reference relative distance signals zoomed in



Source: Own authorship (2024).

It is possible to conclude from the plots, that the signals are very similar, since the same position is used as reference for both simulations. In the zoomed images it is possible to see that the controlled model the battery charge is 0.005 Ah lower, while the electrical efficiency is 0.1 kWh/100km higher. This difference is negligible and the signals can be considered the same, for the powetrain, the electric motor torque present a higher peak around 145 second mark of the cycle, but the overall behavior is the same. The same happens for the vehicle speed profile and relative distances.

As stated earlier in this report, all the simulation scripts are present in the appendix of this report - appendix B.

Chapter 6

Conclusion

In this thesis, the longitudinal dynamics of the vehicle are developed and validated with different modeling approaches.

First, for the model equations validation, a backward model is developed. The backward model has as input the velocity and acceleration reference and produce the torque signals along the powertrain. In this approach, the acceleration and velocity are use in the vehicle longitudinal dynamics equations to compute the resistive and inertial forces, then the powertrain force necessary to reach the desired acceleration and overthrow the resistive forces is calculated, and, consequently, the wheel torque. Then the wheel torque is converted into the shaft torque after the gain of the gearbox, and the Electric Machine dynamics are computed with the shaft torque and angular speed information and efficiency map. The EM power is obtained and passed to the battery block, hence this power must be provided by the battery. In the battery dynamics equation, State of Charge is computed. This backward model is used as a EM torque generator to validate the forward modeling approach, which is used in the MPC controlled model as plant.

The backward-forward model is used as states, EM torque and vehicle acceleration reference signals to be compared with the controlled models. The forward part is modeled with the opposite physical causality presented in the backward one. The EM torque is the input of the vehicle dynamic equation, and the acceleration of the vehicle is calculated from it, as the velocity and position by integration. The torque provided as input also serves to feed the dynamic of the EM model, which outputs a electric power signal (as in the backward model), which is inputted to the battery dynamics. This backward forward model is used as reference or uncontrolled plant, since it accounts for all the dynamics and reproduces the reference speed and acceleration in the states, accounting for all the losses and non linearity of the model.

The results of both models for equations validation and for reference (uncontrolled model) purposes are satisfactory, since the speed profile is followed successfully, and the torque signals are generated properly and present reasonable values bellow the EM shaft torque limit of around 250 Nm. And the wheel torque are proportional to the acceleration profile of the WLTP3 cycle, as expected. The battery has a small loss in SOC.

Then, for a first development and tuning of the MPC controller, the state space model of the powertrain dynamics is developed and inputted into the controller as a parameter, and the equations are modeled for the EM torque to be the control output and feedback input of the controller. The forward model is used as plant to be controlled and for the states computation. For a first tuning and testing of the MPC, the speed profile is directly inputted as reference, and the other states have no reference signal. Then, to test if the controller would respond well in position set as reference, in a second case the speed profile from the cycle is integrated and the position trace is provided as second state reference for the MPC block. Finally, in the last 2 and more realistic car-following scenarios, a CTG policy is implemented to provide the position reference. First a very simple modeling with no delays or simulations of data acquisition and a very simple transfer function is used as vehicle plant to compute the reference position profile. In the end, for the last test case, a realistic CTG is implemented, with data acquisition delay simulation and complete vehicle longitudynal dynamics block to compute the position reference fed to the MPC block.

Furthermore, the MIL complete model is simulated with the MPC for a final validation of the simplified powertrain modeling, and to check if the controller would behave properly in a more complex and realistic vehicle model. The results were very satisfactory, once that the MPC controller was able to generate a EM required torque signal that made the vehicle follow the reference and keep the desired distance from the leading vehicle.

In all the controlled model simulation, the controller performed well and produced a smoother torque/acceleration signal then the reference model (except in the MIL approach). The torque was very similar to the reference and the position and speed traces are followed successfully in all cases. The SOC showed similar values in the controlled and uncontrolled models, with the controlled one presenting higher SOC values at the end of the cycle in some simulations. In the ACC scenarios, the distance between the vehicles oscillated around the set distance, and there was no collision in any point of the cycle.

6.1 Next Steps

For the next steps of this thesis, the MPC controller can be tested in vehicle model with lateral dynamics as well and more complex electric components modeling to check if it works properly in a more complete setup prior to be tested in a real vehicle. Also, the desired battery profile can be provided as reference to the MPC controller, to check if it can follows two state's references. A platoon stability test can also be done with the CTG policy providing the acceleration signals, with all the vehicles in the platoon being controlled by a Model Predictive Controller.

Bibliography

[1] EC-European Commission et al. «Regulation (EU) 2019/631 of the European parliament and of the council of 17 April 2019 setting CO2 emission performance standards for new passenger cars and for new light commercial vehicles, and repealing Regulations (EC) No 443/2009 and (EU) No 510/2011 (recast)». In: *Official Journal of the European Union L* 111 (2019), pp. 13–53 (cit. on p. 1).

[2] Pablo Luque, Daniel A Mántaras, Jorge Roces, Luis Castejón, Hugo Malón, and David Valladares. «Optimization of the powertrain of electric vehicles for a given route». In: *Transportation Research Procedia* 58 (2021), pp. 246–253 (cit. on p. 3).

[3] Grzegorz Sieklucki. «Optimization of Powertrain in EV». In: *Energies* 14.3 (2021), p. 725 (cit. on p. 3).

[4] Yuhang Li, Bo Zhu, Nong Zhang, Hao Peng, and Yongzhong Chen. «Parameters optimization of two-speed powertrain of electric vehicle based on genetic algorithm». In: *Advances in Mechanical Engineering* 12.1 (2020), p. 1687814020901652 (cit. on p. 3).

[5] Noëlle Janiaud, François-Xavier Vallet, Marc Petit, and Guillaume Sandou. «Electric vehicle powertrain architecture and control global optimization». In: *World Electric Vehicle Journal* 3.4 (2009), pp. 682–693 (cit. on p. 3).

[6] Matthew Liam De Klerk and Akshay Kumar Saha. «A comprehensive review of advanced traction motor control techniques suitable for electric vehicle applications». In: *IEEE Access* 9 (2021), pp. 125080–125108 (cit. on p. 4).

[7] Mehrdad Ehsani, Krishna Veer Singh, Hari Om Bansal, and Ramin Tafazzoli Mehrjardi. «State of the art and trends in electric and hybrid electric vehicles». In: *Proceedings of the IEEE* 109.6 (2021), pp. 967–984 (cit. on p. 4).

[8] Chen Hong, Gong Xun, HU Yun-Feng, Liu Qi-Fang, Gao Bing-Zhao, and Guo Hong-Yan. «Automotive control: the state of the art and perspective». In: *Acta Automatica Sinica* 39.4 (2013), pp. 322–346 (cit. on p. 4).

[9] Joycer Osorio, Pedro Ponce, and Arturo Molina. «Electric Vehicle Powertrain Control with Fuzzy Indirect Vector Control». In: *2012 11th Mexican International Conference on Artificial Intelligence*. IEEE. 2012, pp. 122–127 (cit. on p. 4).

[10] Jony Javorski Eckert, Ludmila Correa de Alkmin Silva, Franco Giuseppe Dedini, and Fernanda Cristina Corrêa. «Electric vehicle powertrain and fuzzy control multi-objective optimization, considering dual hybrid energy storage systems». In: *IEEE Transactions on Vehicular Technology* 69.4 (2020), pp. 3773–3782 (cit. on pp. 4, 5).

[11] Wang Hongyu, Yuwen Zhiqiang, Fang Yong, Qiao Yunqian, and Li Zhiming. «Development of pure electric vehicle powertrain controller based on hardware in the loop platform». In: *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. IEEE. 2015, pp. 498–502 (cit. on p. 5).

[12] Jian Zhou, Xiangming Shen, and Dong Liu. «Modeling and simulation for electric vehicle powertrain controls». In: *2014 IEEE Conference and Expo Transportation Electrification Asia-Pacific (ITEC Asia-Pacific)*. IEEE. 2014, pp. 1–4 (cit. on p. 5).

[13] Xingyu Zhou, Fengchun Sun, Chuntao Zhang, and Chao Sun. «Stochastically predictive co-optimization of the speed planning and powertrain controls for electric vehicles driving in random traffic environment safely and efficiently». In: *Journal of Power Sources* 528 (2022), p. 231200 (cit. on p. 5).

[14] Zhenhai Gao, Wei Yan, and Hongjian Li. «Design of the time-gap-dependent robust headway control algorithm for ACC vehicles». In: *International journal of vehicle design* 70.4 (2016), pp. 325–340 (cit. on p. 5).

[15] Junmin Wang and Rajesh Rajamani. «Should adaptive cruise-control systems be designed to maintain a constant time gap between vehicles?» In: *IEEE Transactions on Vehicular Technology* 53.5 (2004), pp. 1480–1490 (cit. on p. 5).

[16] Raffaele Manca, Eugenio Tramacere, Stefano Favelli, Andrea Tonoli, and Luca Camosi. «Impact of Rolling Resistance Modeling on Energy Consumption for a Low Voltage Battery Electric Vehicle». In: *2023 International Symposium on Electromobility (ISEM)*. IEEE. 2023, pp. 1–6 (cit. on pp. 5, 16, 25, 30–32, 37–39, 64).

[17] Federico Millo, Luciano Rolando, and Maurizio Andreata. «Numerical Simulation for Vehicle Powertrain Development». In: (Sept. 2011). URL: `https://scite.ai/reports/10.5772/24111` (cit. on p. 6).

[18] Alam Ms. «Golf Car Application Based Performance Analysis of a Generic Neighborhood Fuel Cell Vehicle (NFCV) Powertrain». In: (Jan. 2014). URL: https://scite.ai/reports/10.4172/2167-7670.1000107 (cit. on p. 6).

[19] Hicham El Hadraoui, Mourad Zegrari, Ahmed Chebak, Oussama Laayati, and Nasr Guennouni. «A Multi-Criteria Analysis and Trends of Electric Motors for Electric Vehicles». In: (Apr. 2022). URL: https://scite.ai/reports/10.3390/wevj13040065 (cit. on p. 6).

[20] Data by you. *Electric Vehicle Powertrain Architecture*. Accessed in November 2023. URL: https://www.exro.com/industry-insights/ev-power-electronics-explained (cit. on p. 6).

[21] Data by you. *Battery Management System*. Accessed in November 2023. URL: https://www.transportenvironment.org/discover/ev-batteries-are-getting-cleaner-%20and-cleaner-2-3-times-better-2-years-ago/ (cit. on p. 7).

[22] Shiquan Zhao, Ricardo Cajo, Robain De Keyser, and Clara-Mihaela Ionescu. «The Potential of Fractional Order Distributed MPC Applied to Steam/Water Loop in Large Scale Ships». In: (Apr. 2020). URL: https://scite.ai/reports/10.3390/pr8040451 (cit. on p. 10).

[23] Johannes Köhler, Elisa Andina, Raffaele Soloperto, Matthias A. Müller, and Frank Allgöwer. «Linear robust adaptive model predictive control: Computational complexity and conservatism». In: (Dec. 2019). URL: https://scite.ai/reports/10.1109/cdc40024.2019.9028970 (cit. on p. 10).

[24] Karam Abughalieh and Shadi Alawneh. «A Survey of Parallel Implementations for Model Predictive Control». In: (Jan. 2019). DOI: 10.1109/access.2019.2904240. URL: https://scite.ai/reports/10.1109/access.2019.2904240 (cit. on p. 11).

[25] Simona Onori, Lorenzo Serrao, and Giorgio Rizzoni. *Hybrid electric vehicles: Energy management strategies*. Springer, 2016 (cit. on pp. 13–15, 24, 25).

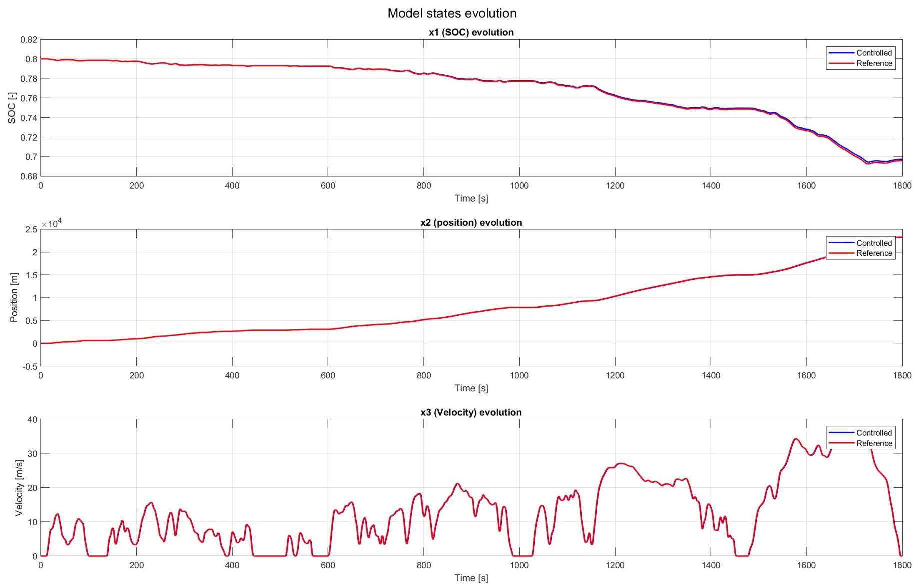
[26] Pier Giuseppe Anselma and Giovanni Belingardi. «Enhancing Energy Saving Opportunities through Rightsizing of a Battery Electric Vehicle Powertrain for Optimal Cooperative Driving». In: *SAE International Journal of Connected and Automated Vehicles* 3.12-03-02-0007 (2020), pp. 71–83 (cit. on p. 13).

Appendix A

Complete cycle plots

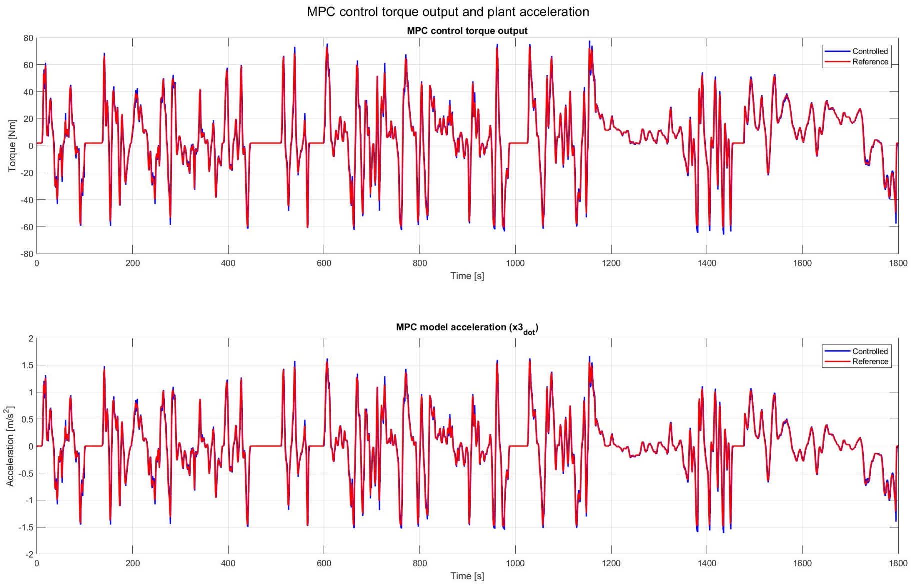
The complete simulation for the whole 1800 seconds from the WLTP3 cycle are exposed in this appendix.

Figure A.1: MPC controlled simplified model results states evolution - velocity reference 1800 s



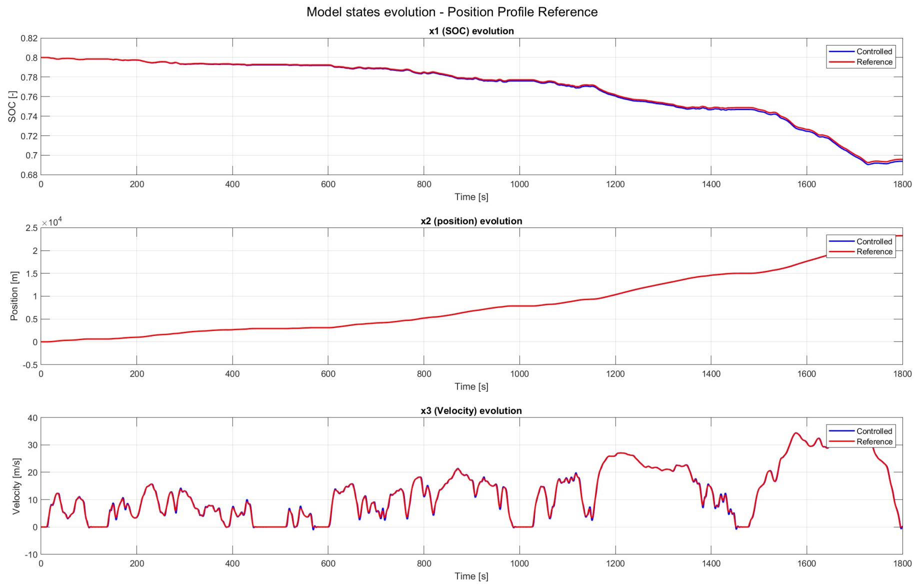
Source: Own elaboration (2024).

Figure A.2: MPC controlled simplified model results torque and acceleration evolution - velocity reference 1800 s



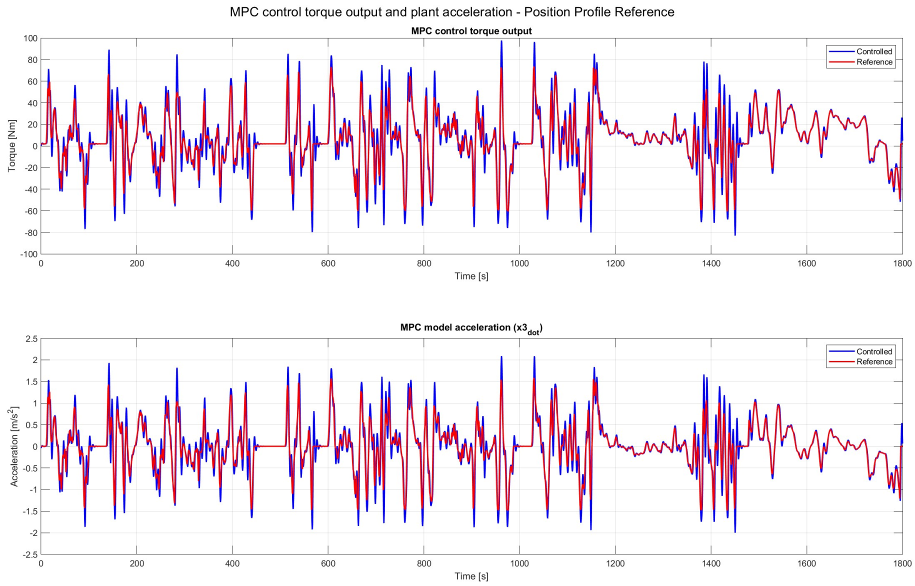
Source: Own elaboration (2024).

Figure A.3: MPC controlled simplified model results states evolution - position reference 1800 s



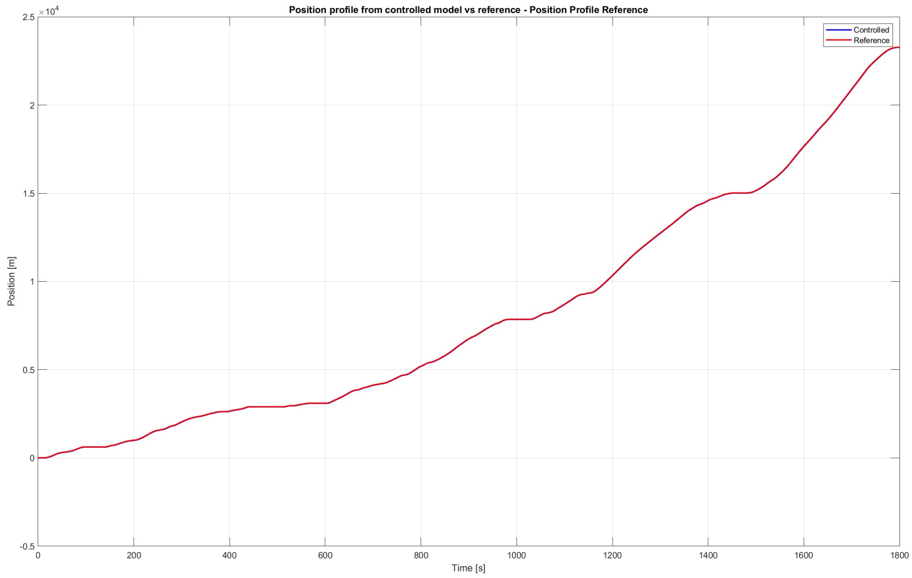
Source: Own elaboration (2024).

Figure A.4: MPC controlled simplified model results torque and acceleration evolution - position reference 1800 s



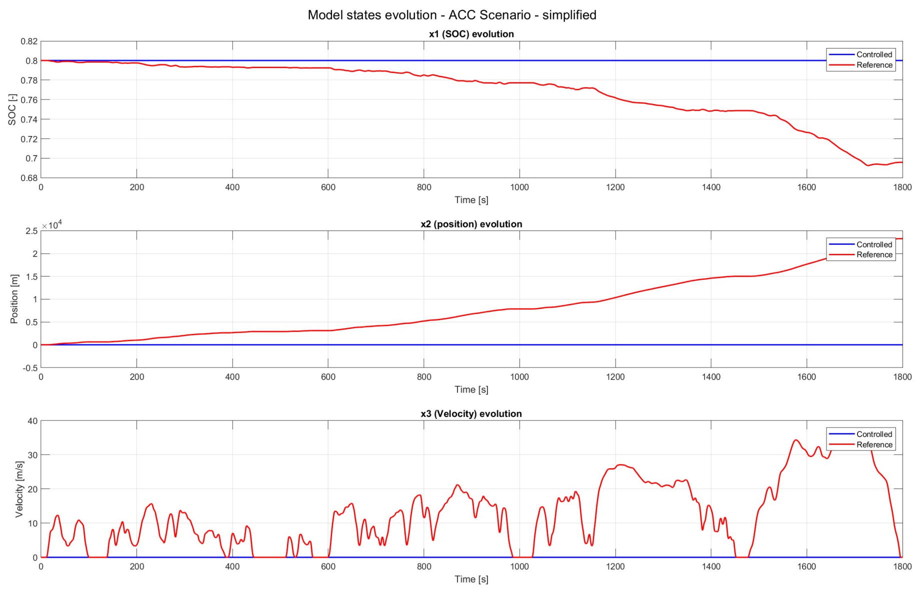
Source: Own elaboration (2024).

Figure A.5: MPC controlled simplified model results relative distance - position reference 1800 s



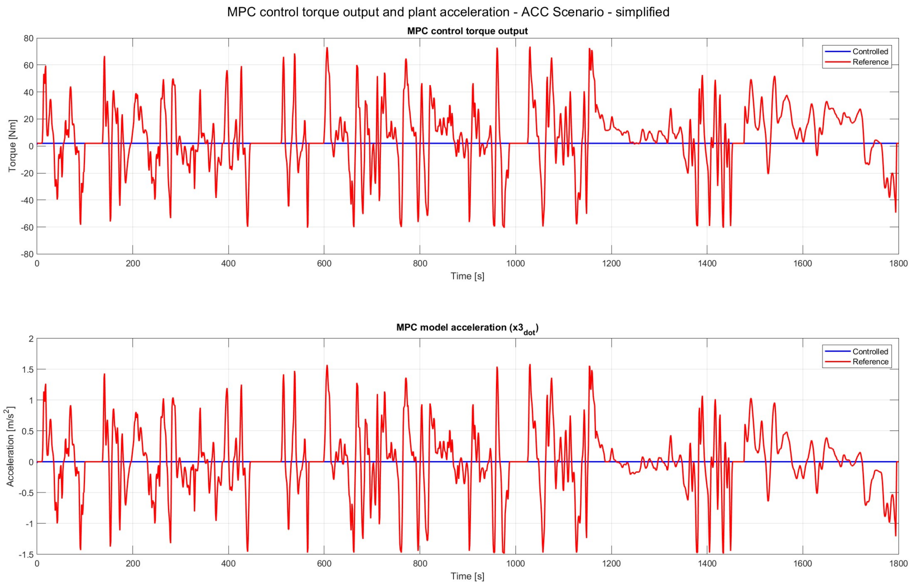
Source: Own elaboration (2024).

Figure A.6: MPC controlled simplified model results states evolution - simplified ACC 1800 s



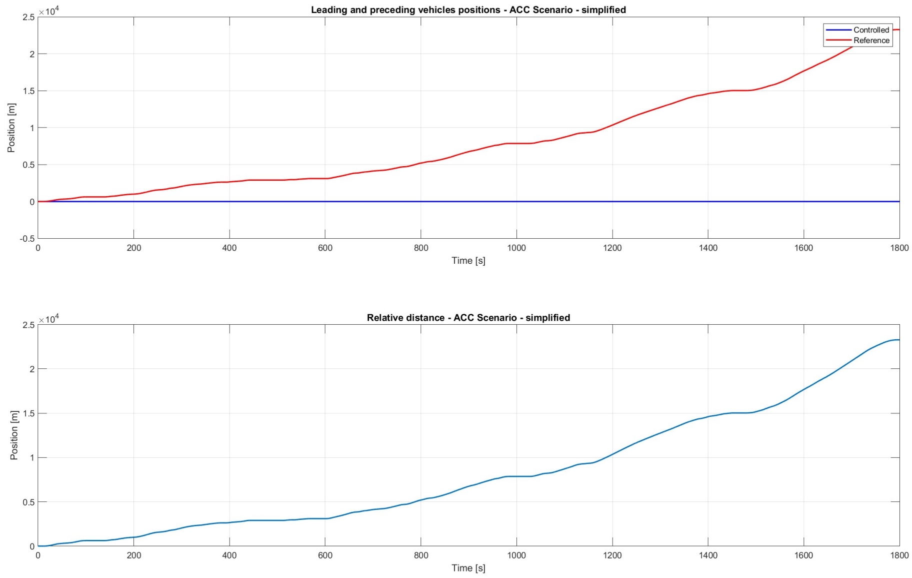
Source: Own elaboration (2024).

Figure A.7: MPC controlled simplified model results torque and acceleration evolution - simplified ACC 1800 s



Source: Own elaboration (2024).

Figure A.8: MPC controlled simplified model results relative distance - simplified ACC 1800 s



Source: Own elaboration (2024).

Appendix B

MATLAB scripts

Script for plotting the EM efficiency map.

```
close all;
clear;
clc;

load("Efficiency.mat");
load("Shaft_torque.mat");
load("Speed.mat");

Speed_max = Speed(:,1);
Torque_max = Shaft_Torque(:,1);

F = scatteredInterpolant(Speed(:),Shaft_Torque(:),Efficiency(:));

% levels plot
figure
levels = [0:0.70:0.70 00.70:0.1:0.94 0.94:0.01:1]*100;
M = contourf(Speed,Shaft_Torque,Efficiency,levels);
hold on
plot(Speed_max,Torque_max)
grid on
title('EM efficiency map')
xlabel('EM speed [rpm]')
ylabel('Shaft Torque [Nm]')

Efficiency(Efficiency==0) = 1;

% surface plot
figure
```

```
%surf([Speed(:,1:27) Speed(:,32:end)],[Shaft_Torque(:,1:27)
    Shaft_Torque(:,32:end)], [Efficiency(:,1:27) Efficiency(:,32:end)
    ])
surf(Speed,Shaft_Torque,Efficiency)
title('EM efficiency surface')
xlabel('EM speed [rpm]')
ylabel('Shaft Torque [Nm]')
zlabel('Efficiency [%]')
```

Backward model script.

```
clear
close all
clc

%% PARAMETERS

% adding folders to path and loading data
addpath('data')
addpath('Eff_map')
load(fullfile('data','WLTP3.mat')) % loading the speed profile (WLTP3
    )
load("Eff_map\Efficiency.mat"); % loading the EM efficeincy map
load("Eff_map\Shaft_torque.mat"); % loading shaft torque map
load("Eff_map\Speed.mat"); % loading speed map

% setting efficiency to 100% when T = 0
Efficiency(Efficiency==0) = 100;

% velocity and acceleration profiles
vehspeed = [time_s,speed_kmh/3.6]; % speed profile in [m/s]
dt = 1;
vehacc = (vehspeed(2:end,2)-vehspeed(1:end-1,2))./dt; %vehicle
    acceleration [m/s^2]
vehacc = [0; vehacc];

% simulation parameters
t_sim = 1800; % simulation time [s]
g = 9.81; % Gravity acceleration [m/s^2]
alpha = 0*pi/180; % road slope [rad]
rho = 1.25; % air density [kg/m^3] - from Onori HEV book

% battery parameters
N_s = 108; % Number of series
N_p = 1; % Number of parallels
N_b = N_s*N_p; % number of battery cells
Q_nom = 60*N_p; % nominal battery capacity [Ah]
eta_c = 0.95; % Coloumbic efficiency
```

```
load('bat_Ro_vs_SOC_data.mat'); % Ro variation with SOC - single
    battery cell
load('bat_Voc_vs_SOC_data.mat'); % Voc variation with SOC - single
    battery cell
F_Voc_s = griddedInterpolant(SOC_Voc_data(:,1),SOC_Voc_data(:,2)); %
    interpolated Voc data for a single cell
F_Ro_s = griddedInterpolant(SOC_Ro_data(:,1),SOC_Ro_data(:,2)); %
    interpolated Ro data for a single cell
SOC_vec = linspace(0,1,500);
Voc_s = F_Voc_s(SOC_vec); % vector with interpolated values of Voc_s
    (single cell)
Ro_s = F_Ro_s(SOC_vec); % vector with interpolated values of Ro_s (
    single cell)
Voc = N_b*Voc_s;
Ro = N_b*Ro_s;

% vehicle parameters
M_veh = 1400; % vehicle mass [kg]
a = 1; % Front axle - CoG  Front axle - CoG  [m]
b = 1.3; % Rear axle - CoG  [m]
h = 0.3; % Height of CoG [m]
f_0 = 4.5*1e-3; % Static rolling coefficient [N/kN]
k = 0; % miscellaneous loss coeff [Ns/m]
r_w = 0.3; % Wheel radius [m]
C_d = 0.33; % Drag coeff
A_f = 2.15; % Frontal area [m^2]
tau_gb = 9.6; % Gear ratio
eta_gb = 0.97; % Gearbox efficiency
eta_inv = 1; % Inverter efficiency

% electric motor parameters
Speed_max = Speed(:,1);
Torque_max = Shaft_Torque(:,1);
M_eff = [Speed(:),Shaft_Torque(:),Efficiency(:)];
F = scatteredInterpolant(Speed(:),Shaft_Torque(:),Efficiency(:));

% initial condition
SOC0 = 0.8;

%% SIMULATION

open('simplified_EV_model.slx')
sim('simplified_EV_model.slx')

%% PLOTS

directory = "C:\Users\gabri\Documents\TCC - EV powertrain control\
    Text\images";

```

```
fig = figure();
fig.Position = [100, 100, 1000, 600];
subplot(2,1,1);
plot(T_wheel.Time,T_wheel.Data,'b','LineWidth',1.5);
grid on
title('Wheel torque');
xlabel('Time [s]');
ylabel('Torque [Nm]');
subplot(2,1,2);
plot(w_wheel.Time,w_wheel.Data,'r','LineWidth',1.5);
xlabel('Time [s]');
ylabel('Angular speed [rad/s]');
grid on
title('Wheel angular speed');

filename = 'result_BW_T_and_w_wheel.jpg';
fullFileName = fullfile(directory, filename);
saveas(fig, fullFileName);


fig = figure();
fig.Position = [100, 100, 1000, 600];
subplot(2,1,1);
plot(T_EM.Time,T_EM.Data,'b','LineWidth',1.5);
grid on
title('Electric motor torque');
xlabel('Time [s]');
ylabel('Torque [Nm]');
subplot(2,1,2);
plot(w_EM_rpm.Time,w_EM_rpm.Data,'r','LineWidth',1.5);
xlabel('Time [s]');
ylabel('Angular speed [RPM]');
grid on
title('Electric motor angular speed in RPM');

filename = 'result_BW_T_and_w_EM.jpg';
fullFileName = fullfile(directory, filename);
saveas(fig, fullFileName);


fig = figure();
fig.Position = [100, 100, 1000, 600];
subplot(2,1,1);
plot(P_batt.Time,P_batt.Data,'b','LineWidth',1.5);
grid on
title('Battrey power');
xlabel('Time [s]');
ylabel('Power [W]');
subplot(2,1,2);
```

```
plot(SOC.Time,SOC.Data,'r','LineWidth',1.5);
xlabel('Time [s]');
ylabel('SOC [-]');
grid on
title('State of Charge');

filename = 'result_BW_P_b_and_SOC.jpg';
fullFileName = fullfile(directory, filename);
saveas(fig, fullFileName);
```

Backward-Forward model script.

```
clear
close all
clc

%% PARAMETERS

% adding folders to path and loading data
addpath('data')
addpath('data\Eff_map')
load(fullfile('data','WLTP3.mat')) % loading the speed profile (WLTP3
    )
load("Eff_map\Efficiency.mat"); % loading the EM efficeincy map
load("Eff_map\Shaft_torque.mat"); % loading shaft torque map
load("Eff_map\Speed.mat"); % loading speed map

% setting efficiency to 100% when T = 0
Efficiency(Efficiency == 0) = 10;
eff = Efficiency/100;

% velocity and acceleration profiles
vehspeed = [time_s,speed_kmh/3.6]; % speed profile in [m/s]
dt = 1;
vehacc = (vehspeed(2:end,2)-vehspeed(1:end-1,2))./dt; %vehicle
    acceleration [m/s^2]
vehacc = [0; vehacc];

% simulation parameters
t_sim = 1800; % simulation time [s]
g = 9.81; % Gravity acceleration [m/s^2]
alpha = 0*pi/180; % road slope [rad]
rho = 1.25; % air density [kg/m^3] - from Onori HEV book

% battery parameters
N_s = 108; % Number of series
N_p = 1; % Number of parallels
N_b = N_s*N_p; % number of battery cells
```

```
Q_nom = 60*N_p; % nominal battery capacity [Ah]
eta_c = 0.95; % Coloumbic efficiency
load('bat_Ro_vs_SOC_data.mat'); % Ro variation with SOC - single
    battery cell
load('bat_Voc_vs_SOC_data.mat'); % Voc variation with SOC - single
    battery cell
F_Voc_s = griddedInterpolant(SOC_Voc_data(:,1),SOC_Voc_data(:,2)); %
    interpolated Voc data for a single cell
F_Ro_s = griddedInterpolant(SOC_Ro_data(:,1),SOC_Ro_data(:,2)); %
    interpolated Ro data for a single cell
SOC_vec = linspace(0,1,500);
Voc_s = F_Voc_s(SOC_vec); % vector with interpolated values of Voc_s
    (single cell)
Ro_s = F_Ro_s(SOC_vec); % vector with interpolated values of Ro_s (
    single cell)
Voc = N_b*Voc_s;
Ro = N_b*Ro_s;

% vehicle parameters
M_veh = 1400; % vehicle mass [kg]
a = 1; % Front axle - CoG  Front axle - CoG  [m]
b = 1.3; % Rear axle - CoG  [m]
h = 0.3; % Height of CoG [m]
f_0 = 4.5*1e-3; % Static rolling coefficient [N/kN]
k = 0; % miscellaneous loss coeff [Ns/m]
r_w = 0.3; % Wheel radius [m]
C_d = 0.33; % Drag coeff
A_f = 2.15; % Frontal area [m^2]
tau_gb = 9.6; % Gear ratio
eta_gb = 0.97; % Gearbox efficiency
eta_inv = 1; % Inverter efficiency

% electric motor parameters
Speed_max = Speed(:,1);
Torque_max = Shaft_Torque(:,1);
M_eff = [Speed(:),Shaft_Torque(:),Efficiency(:)];
F = scatteredInterpolant(Speed(:),Shaft_Torque(:),Efficiency(:));

% initial condition
SOC0 = 0.8;

%% SIMULATION

open('EV_BW_FW_reference_model.slx')
sim('EV_BW_FW_reference_model.slx')

%% PLOTS

x1 = X_ref.Data(:,1);
```

```
x2 = X_ref.Data(:,2);
x3 = X_ref.Data(:,3);
sim_time = X_ref.Time;

directory = "C:\Users\gabri\Documents\TCC - EV powertrain control\Text\images";

fig1 = figure();
fig1.Position = [100, 100, 1000, 600];
sgtitle("BW-FW Reference Model states evolution");
subplot(3,1,1);
plot(sim_time ,x1,'g','LineWidth',1.5);
ylabel('SOC [-]');
xlabel('Time [s]');
title('x1 (SOC) evolution')
grid on;

subplot(3,1,2);
plot(sim_time ,x2,'r','LineWidth',1.5);
ylabel('Position [m]');
xlabel('Time [s]');
title('x2 (position) evolution')
grid on;

subplot(3,1,3);
plot(sim_time ,x3,'k','LineWidth',1.5);
ylabel('Velocity [m/s]');
xlabel('Time [s]');
title('x3 (Velocity) evolution');
grid on;

filename = 'result_BW_FW_ref_model_states.jpg';
fullFileName = fullfile(directory, filename);
saveas(fig1, fullFileName);

% control torque and state derivative

fig2 = figure();
fig2.Position = [100, 100, 1000, 600];
sgtitle("BW-FW Reference Model EM Torque and Acceleration");
subplot(2,1,1)
plot(sim_time, T_EM_ref.Data,'b', 'LineWidth',1.5);
grid on;
xlabel("Time [s]")
ylabel("Torque [Nm]")
title("EM Torque")

subplot(2,1,2)
plot(sim_time, a_ref.Data,'r','LineWidth',1.5);
```

```
grid on;
xlabel("Time [s]")
ylabel("Acceleration [m/s^2]")
title('Acceleration')


filename = 'result_BW_FW_ref_model_acc_T.jpg';
fullFileName = fullfile(directory, filename);
saveas(fig2, fullFileName);
```

Prediction model function script.

```
function [xdot, y] = prediction_longitudinal_model(t,X,U) % add y to
    output eventually
% arguments: t, X: states,
% U: command in the previous instant

% enviroment parameters
g = 9.81; % Gravity acceleration [m/s^2]
alpha = 0*pi/180; % road slope [rad]
rho = 1.25; % air density [kg/m^3] - from Onori HEV book

% vehicle parameters
M_veh = 1400; % vehicle mass [kg]
a = 1; % Front axle - CoG  Front axle - CoG  [m]
b = 1.3; % Rear axle - CoG  [m]
h = 0.3; % Height of CoG [m]
f_0 = 4.5/1000; % Static rolling coefficient [N/kN]
k = 0; % miscellaneous loss coeff [Ns/m]
r_w = 0.3; % Wheel radius [m]
C_d = 0.33; % Drag coeff
A_f = 2.15; % Frontal area [m^2]
tau_gb = 9.6; % Gear ratio
eta_gb = 0.97; % Gearbox efficiency
eta_inv = 1; % Inverter efficiency
phi = r_w/tau_gb;

% battery parameters
N_s = 108; % Number of series
N_p = 1; % Number of parallels
N_b = N_s*N_p; % number of battery cells
Q_nom = 60*N_p*3600; % nominal battery capacity [As]
eta_batt = 0.95; % Coloumbic efficiency

% state variables
x1 = X(1); % SOC
x2 = X(2); % Position
x3 = X(3); % Velocity
```

```

% control input
T_EM = U;

% Forces
 F_roll = M_veh*g*f_0; % rolling resistance force
 F_aero = 0.5*C_d*rho*A_f*x3^2; % aerodynamic force

% EM equations
w_EM = x3/phi;
w_EM_rpm = w_EM*30/pi;
eta_EM = eff_poly(w_EM_rpm, T_EM);
P_EM = w_EM*T_EM;

% battery equations
Pb = P_EM/((eta_EM*eta_inv)^(sign(P_EM)));
V_oc = N_s*V_poly(x1); % vector with interpolated values of Voc_s (
    single cell)
R_o = N_s/N_p*R_poly(x1); % vector with interpolated values of Ro_s (
    single cell)
Ib = (V_oc - sqrt(V_oc^2 - 4*R_o*Pb))/(2*R_o);

% state equations
soc_dot = -Ib/(Q_nom*eta_batt^(sign(Ib)));
x_dot = x3;
v_dot = ((eta_gb/phi)*T_EM - F_roll - F_aero)/M_veh;

% state derivative
xdot = [soc_dot;x_dot;v_dot];

y = [x1;x2;x3];
end
```

Polynomial fit functions and testing script.

```
clear;
clc;

% testing polinomial fit with battery and EM efficiency data

%% adding folders to path
addpath('data\Eff_map');
addpath('data');

%% loading data into vectors

%%% loading EM parameters
load('Efficiency.mat'); % loading the EM efficeincy map
```

```
load("Shaft_torque.mat"); % loading shaft torque map
load("Speed.mat"); % loading speed map

%%% loading battery params
load("bat_Ro_vs_SOC_data.mat");
load("bat_Voc_vs_SOC_data.mat");

%% EM efficiency polynomial polynomial fitting and testing

% EM efficiency ref (compare the results with the polinomial)
eff_flat = Efficiency(:)/100;
torq_flat = Shaft_Torque(:);
speed_flat = Speed(:);

F = scatteredInterpolant(Speed(:),Shaft_Torque(:),Efficiency(:)/100);
figure
surf(Speed,Shaft_Torque,Efficiency);


speed_points =
    [500;750;1000;1250;1500;1750;2000;2250;2500;2750;3000;3250;

    3500;3750;4000;4250;4500;4750;5000;5250;5500;5750;6000;6250;6500;6750;

    7000;7250;7500;7750;8000;8250;8500;8750;9000;9250;9500;9750;10000];


torque_points = [287.315725265347;287.233717032146;287.151724602149;

    287.069743739971;286.987771986552;286.905807734913;286.823849853203;

    286.741897501612;267.378269069311;243.071146629631;222.814866998334;

    205.674858958375;190.983337167297;178.250632202600;167.109517497491;

    157.278993072683;148.540714541255;140.726295202330;133.690149986783;

    127.323954482924;121.536502005985;116.252306265388;111.408460170112;

    106.952121758109;102.838578612008;99.0297423697019;95.4928054975261;
```

```
    92.1999538366269;89.1266218638583;86.2515686389848;83.5562072631545;
    81.0242030664065;78.6411424126642;76.3942584067584;74.2722022887362;
    72.2648518732074;70.3631501887848;68.5589689496418;66.8449924891107];

%%% testing and comparing the results of polynimial fit and
    interpolant

poly = [];
eval = [];
number_points = [];
for i=1:length(speed_points)
    poly = [poly; eff_poly(speed_points(i),torque_points(i))];
    eval = [eval; F(speed_points(i),torque_points(i))];
    number_points = [number_points;i];
end

% plotting the diferentece between the evaluations

figure
plot(number_points,poly - eval, 'LineWidth',1.5);
title('Difference from polynomial fit and interpolant evaluation of
   EM efficiency');
xlabel('Evaluations');
ylabel('Difference');

%% Battery voltage and resistance polynomial fitting and testing

% interpolants for comparison and tests
F_Voc_s = griddedInterpolant(SOC_Voc_data(:,1),SOC_Voc_data(:,2)); %
    interpolated Voc data for a single cell
F_Ro_s = griddedInterpolant(SOC_Ro_data(:,1),SOC_Ro_data(:,2)); %
    interpolated Ro data for a single cell

% Extracting data from files
SOC_R = SOC_Ro_data(:,1);
R_data = SOC_Ro_data(:,2);

SOC_V = SOC_Voc_data(:,1);
V_data = SOC_Voc_data(:,2);

% Perform a 2nd-degree polynomial fit
degree = 2;
R_coefficients = polyfit(SOC_R, R_data, degree);
```

```
V_coefficients = polyfit(SOC_V, V_data, degree);

% Polynomial functions
R_function = poly2str(R_coefficients, 'SOC');
V_function = poly2str(V_coefficients, 'SOC');
disp(['Polynomial Equation: R(SOC) = ' R_function]);
disp(['Polynomial Equation: V(SOC) = ' V_function]);

% Create a polynomial model using polyval
SOC_range_R = linspace(min(SOC_R), max(SOC_R), 100); % Adjust the
    range as needed
Resistance_fit = polyval(R_coefficients, SOC_range_R);

SOC_range_V = linspace(min(SOC_V), max(SOC_V), 100); % Adjust the
    range as needed
Voltage_fit = polyval(V_coefficients, SOC_range_V);

% Plot the original data points and the fitted curve
figure;
plot(SOC_R, R_data, 'o', 'DisplayName', 'Original Data');
hold on;
plot(SOC_range_R, Resistance_fit, 'r-', 'DisplayName', '2nd-degree
    Polynomial Fit');
xlabel('State of Charge (SOC)');
ylabel('Resistance');
title('2nd-degree Polynomial Fit: Resistance as a function of SOC');
legend('Location', 'Best');
grid on;


figure;
plot(SOC_V, V_data, 'o', 'DisplayName', 'Original Data');
hold on;
plot(SOC_range_V, Voltage_fit, 'r-', 'DisplayName', '2nd-degree
    Polynomial Fit');
xlabel('State of Charge (SOC)');
ylabel('Voltage');
title('2nd-degree Polynomial Fit: Voltage as a function of SOC');
legend('Location', 'Best');
grid on;

%% Resultant polynomial functions

% EM polynomial fit

function eff = eff_poly(w, T)
    % eff = 2.07338239e-6*T - 3.84695020e-6*w + 8.63478964e-1;
     e = 3.86559103e-6*T^2 -1.15468531e-9*w^2 - 6.64654450e-9*T*w +
    3.14229679e-5*T + 1.63013222e-5*w + 7.71598709e-1;
```

```
    if e > 1
        eff = 1;
    elseif e < 0
        eff = 0;
    else
        eff = e;
    end
end

% Battery resistance polynomial fit

function V_oc = V_poly(SOC)
    V_oc = -0.40666*SOC^2 + 1.0703*SOC + 3.4385;
end

% Battery voltage polynomial fit

function R_o = R_poly(SOC)
    R_o = 0.00041627*SOC^2 - 0.00071804*SOC + 0.0023018;
end
```

MPC controlled model script.

```
    clear
close all
clc

%% PARAMETERS

% polynomial functions for battery voltage, resistance and EM
    efficiency
% defined in the polynimial_fits.m file, and the funcions are saved
% separately to be used directly inside this model

addpath("polynomial_fits")
addpath("data\Eff_map")
addpath("data")
addpath("tests\")
addpath("simulation_results_const_v_ref")
load(fullfile('data','WLTP3.mat')) % loading the speed profile (WLTP3
    )

% velocity and acceleration profiles
vehspeed = [time_s,speed_kmh/3.6]; % speed profile in [m/s]
dt = 1;
vehacc = (vehspeed(2:end,2)-vehspeed(1:end-1,2))./dt; %vehicle
    acceleration [m/s^2]
vehacc = [0; vehacc];
```

```matlab
speed_profile = vehspeed(:,2);
time_vector = vehspeed(:,1);

% simulation parameters
g = 9.81; % Gravity acceleration [m/s^2]
alpha = 0*pi/180; % road slope [rad]
rho = 1.25; % air density [kg/m^3] - from Onori HEV book

% battery parameters
N_s = 108; % Number of series
N_p = 1; % Number of parallels
N_b = N_s*N_p; % number of battery cells
Q_nom = 60*N_p; % nominal battery capacity [Ah]
eta_c = 0.95; % Coloumbic efficiency
load('bat_Ro_vs_SOC_data.mat'); % Ro variation with SOC - single battery cell
load('bat_Voc_vs_SOC_data.mat'); % Voc variation with SOC - single battery cell
F_Voc_s = griddedInterpolant(SOC_Voc_data(:,1),SOC_Voc_data(:,2)); % interpolated Voc data for a single cell
F_Ro_s = griddedInterpolant(SOC_Ro_data(:,1),SOC_Ro_data(:,2)); % interpolated Ro data for a single cell
SOC_vec = linspace(0,1,500);
Voc_s = F_Voc_s(SOC_vec); % vector with interpolated values of Voc_s (single cell)
Ro_s = F_Ro_s(SOC_vec); % vector with interpolated values of Ro_s (single cell)

% vehicle parameters
M_veh = 1400; % vehicle mass [kg]
a = 1; % Front axle - CoG Front axle - CoG [m]
b = 1.3; % Rear axle - CoG [m]
h = 0.3; % Height of CoG [m]
f_0 = 4.5*1e-3; % Static rolling coefficient [N/kN]
k = 0; % miscellaneous loss coeff [Ns/m]
r_w = 0.3; % Wheel radius [m]
C_d = 0.33; % Drag coeff
A_f = 2.15; % Frontal area [m^2]
tau_gb = 9.6; % Gear ratio
eta_gb = 0.97; % Gearbox efficiency
eta_inv = 1; % Inverter efficiency

% electric motor parameters
load("Eff_map\Efficiency.mat"); % loading the EM efficeincy map
load("Eff_map\Shaft_torque.mat"); % loading shaft torque map
load("Eff_map\Speed.mat"); % loading speed map
Speed_max = Speed(:,1);
Torque_max = Shaft_Torque(:,1);
Efficiency(Efficiency == 0) = 10;
```

```
eff = Efficiency/100;
M_eff = [Speed(:),Shaft_Torque(:),Efficiency(:)];
F = scatteredInterpolant(Speed(:),Shaft_Torque(:),Efficiency(:));


% ACC and CTG Contoller parameters
default_distance = 50; % reference distance from leading vehicle [m]
tau = 0.5; % vehicle LTI model [s]
h = 4*tau; % time gap [s] (h > 2*tau)
lambda = 0.5; % CTG parameter [-]
Td = 0.01;
s = tf('s');
P = 1/(tau*s + 1);  % Vehicle simplified plant
v_set = 40;         % ACC set velocity [m/s]
time_gap  = 3;      % ACC time gap [s]
verr_gain = 0.1;    % ACC velocity error gain - CTG
xerr_gain = 0.3;    % ACC spacing error gain - CTG
vx_gain   = 0.5;    % ACC relative velocity gain - CTG
max_acc   = 2;      % Maximum acceleration [m/s^2]
min_acc   = -3;     % Minimum acceleration [m/s^2]


% initial conditions
SOC0 = 0.8;
x0 = -default_distance;
v0 = vehspeed(1);
xx0 = [SOC0; x0; v0];

%% PLOTS
directory = "C:\Users\gabri\Documents\TCC - EV powertrain control\
    Text\images";

% plotting Voc and Roc interpolations
fig = figure();
fig.Position = [100, 100, 1000, 600];
subplot(2,1,1);
plot(SOC_Voc_data(:,1),SOC_Voc_data(:,2),'bo',SOC_vec,Voc_s,'b.');
grid on
title('V_{oc} interpolation');
xlabel('SOC [-]');
ylabel('Open circuit battery voltage [V]');
legend('Sample SOC points','Interpolated values');
subplot(2,1,2);
plot(SOC_Ro_data(:,1),SOC_Ro_data(:,2),'ro',SOC_vec,Ro_s,'r.');
xlabel('SOC [-]');
ylabel('Open circuit battery resistance [\Omega]');
grid on
title('R_o interpolation');
legend('Sample SOC points','Interpolated values', 'Location','best');
```

```
filename = 'battrey_interpolation.jpg';
fullFileName = fullfile(directory, filename);
saveas(fig, fullFileName);


% plotting driving cycle velocity and acceleration profiles
fig = figure();
fig.Position = [100, 100, 1000, 600];
subplot(2,1,1)
plot(vehspeed(:,1),vehspeed(:,2),'LineWidth',1.5)
grid on
xlabel('Time [s]');
ylabel('Reference velocity [m/s]');
subplot(2,1,2)
plot(vehspeed(:,1),vehacc,'LineWidth',1.5)
grid on
xlabel('Time [s]');
ylabel('Reference aceleration [m/s^2]');

filename = 'speed_profile.jpg';
fullFileName = fullfile(directory, filename);
saveas(fig, fullFileName);


% electric motor efficiency map
fig = figure();
fig.Position = [100, 100, 1000, 600];
levels = [0:0.70:0.70 00.70:0.1:0.94 0.94:0.01:1]*100;
M = contourf(Speed,Shaft_Torque,Efficiency,levels);
hold on
plot(Speed_max,Torque_max)
grid on
title('EM efficiency map')
xlabel('EM speed [rpm]')
ylabel('Shaft Torque [Nm]')

filename = 'EM_eff_map.jpg';
fullFileName = fullfile(directory, filename);
saveas(fig, fullFileName);

% return

%% MPC

% PARAMETERS

t_sim = 100; % simulation time [s]
```

```matlab
Ts = 0.05; % Sampling time

par.nx        = 3; % number of states
par.nu        = 1; % control elements number
par.ny        = 3; % number of outputs
par.model = @prediction_longitudinal_model; %Modello di predizione
par.ub        = 250; % Upper bound saturazione input -> maximum value for control output
par.lb        = -250; % Lower bound saturazione input -> minimum value for control output
par.tol       = 1; % Reference tolerance
par.Nfev      = 150; % Interation number of fmincon in cost function (default 200)
par.Ts        = Ts;




%% SIMULATION - CONTROLLED AND REFERENCE MODELS
ref_mode = 3;
disp(ref_mode)

switch ref_mode
    case 1
        scenario = "Velocity Profile Reference";
        % Velocity reference parameters
        par.R = 0.05; % matrice diagonale definita positiva per cost function
        par.P = diag([0;0;10000]); % matrice diagonale definita positiva per cost function
        par.Q = diag([0;0;1]); % matrice diagonale definita positiva per cost function
        par.Tp = 10*Ts; % Prediction horizon (sempre multiplo intero del Ts)
        K = nmpc_design_4b(par); %Generazione parametri design NMPC

    case 2
        scenario = "Position Profile Reference";
        % Position reference parameters
        par.R = 0.05; % matrice diagonale definita positiva per cost function
        par.P = diag([0;50000;0]); % matrice diagonale definita positiva per cost function
        par.Q = diag([0;1;0]); % matrice diagonale definita positiva per cost function
        par.Tp = 10*Ts; % Prediction horizon (sempre multiplo intero del Ts)
        K = nmpc_design_4b(par); %Generazione parametri design NMPC
```

```
    case 3
        scenario = "ACC Scenario - simplified";
        % Position reference parameters
        par.R = 0.01; % matrice diagonale definita positiva per cost function
        par.P = diag([0;50000;0]); % matrice diagonale definita positiva per cost function
        par.Q = diag([0;1;0]); % matrice diagonale definita positiva per cost function
        par.Tp = 10*Ts; % Prediction horizon (sempre multiplo intero del Ts)
        K = nmpc_design_4b(par); %Generazione parametri design NMPC

    case 4
        scenario = "ACC Scenario";
        % Position reference parameters
        par.R = 0.01; % matrice diagonale definita positiva per cost function
        par.P = diag([0;50000;0]); % matrice diagonale definita positiva per cost function
        par.Q = diag([0;1;0]); % matrice diagonale definita positiva per cost function
        par.Tp = 10*Ts; % Prediction horizon (sempre multiplo intero del Ts)
        K = nmpc_design_4b(par); %Generazione parametri design NMPC
end

% open('Model_MPC.slx')
sim("Model_MPC.slx")
sim("tests\EV_BW_FW_reference_model")


%% SIMULATION RESULTS

% MPC model results
x1 = X.Data(:,1);
x2 = X.Data(:,2);
x3 = X.Data(:,3);
x3_dot = X_dot.Data(:,3);
sim_time = X.Time;

% reference model (BKD - FRD) reference
x1_ref = X_ref.Data(:,1);
x2_ref = X_ref.Data(:,2);
x3_ref = X_ref.Data(:,3);
sim_time_ref = X_ref.Time;


if ref_mode == 1
```

```

fig1 = figure();
fig1.Position = [100, 100, 1000, 600];
sgtitle("Model states evolution");
subplot(3,1,1);
plot(sim_time ,x1,'b', sim_time_ref ,x1_ref,'k','LineWidth',1.5);
ylabel('SOC [-]');
xlabel('Time [s]');
legend("Controlled", "Reference");
title('x1 (SOC) evolution')
grid on;

subplot(3,1,2);
plot(sim_time ,x2,'r', sim_time_ref ,x2_ref,'k', 'LineWidth',1.5)
;
ylabel('Position [m]');
xlabel('Time [s]');
legend("Controlled", "Reference");
title('x2 (position) evolution')
grid on;

subplot(3,1,3);
plot(sim_time ,x3, 'g',sim_time_ref,x3_ref,'k','LineWidth',1.5);
ylabel('Velocity [m/s]');
xlabel('Time [s]');
legend("Controlled", "Reference");
title('x3 (Velocity) evolution');
grid on;

% control torque and state derivative

fig2 = figure();
fig2.Position = [100, 100, 1000, 600];
sgtitle("MPC control torque output and plant acceleration");
subplot(2,1,1)
plot(sim_time, T_EM_MPC.Data, "b", sim_time_ref, T_EM_ref.Data, "
k", 'LineWidth',1.5);
grid on;
legend("Controlled", "Reference")
xlabel("Time [s]")
ylabel("Torque [Nm]")
title('MPC control torque output')

subplot(2,1,2)
plot(sim_time, x3_dot, "r", sim_time_ref, a_ref.Data, "k", '
LineWidth',1.5);
grid on;
xlabel("Time [s]")
ylabel("Acceleration [m/s^2]")
```

```
    legend("Controlled", "Reference")
    title('MPC model acceleration (x3_{dot})')

elseif ref_mode == 2

    fig1 = figure();
    fig1.Position = [100, 100, 1000, 600];
    sgtitle("Model states evolution - " + scenario);
    subplot(3,1,1);
    plot(sim_time ,x1,'b', sim_time_ref ,x1_ref,'k','LineWidth',1.5);
    ylabel('SOC [-]');
    xlabel('Time [s]');
    legend("Controlled", "Reference");
    title('x1 (SOC) evolution')
    grid on;

    subplot(3,1,2);
    plot(sim_time ,x2,'r', sim_time_ref ,x2_ref,'k', 'LineWidth',1.5)
    ;
    ylabel('Position [m]');
    xlabel('Time [s]');
    legend("Controlled", "Reference");
    title('x2 (position) evolution')
    grid on;

    subplot(3,1,3);
    plot(sim_time ,x3, 'g',sim_time_ref, x3_ref,'k','LineWidth',1.5);
    ylabel('Velocity [m/s]');
    xlabel('Time [s]');
    legend("Controlled", "Reference");
    title('x3 (Velocity) evolution');
    grid on;

    % control torque and state derivative

    fig2 = figure();
    fig2.Position = [100, 100, 1000, 600];
    sgtitle("MPC control torque output and plant acceleration - " +
    scenario);
    subplot(2,1,1)
    plot(sim_time, T_EM_MPC.Data, "b", sim_time_ref, T_EM_ref.Data, "
    k", 'LineWidth',1.5);
    grid on;
    legend("Controlled", "Reference")
    xlabel("Time [s]")
    ylabel("Torque [Nm]")
    title('MPC control torque output')

    subplot(2,1,2)
```

```
    plot(sim_time, x3_dot, "r", sim_time_ref, a_ref.Data, "k", 'LineWidth',1.5);
    grid on;
    xlabel("Time [s]")
    ylabel("Acceleration [m/s^2]")
    legend("Controlled", "Reference")
    title('MPC model acceleration (x3_{dot})')

    fig3 = figure();
    fig3.Position = [100, 100, 1000, 600];
    plot(sim_time, x2, "m", pos_leading.Time, pos_leading.Data, "b", 'LineWidth',1.5);
    grid on;
    xlabel("Time [s]")
    ylabel("Position [m]")
    legend("Controlled", "Reference")
    title('Position profile from controlled model vs reference - ' + scenario)

elseif ref_mode == 3
    fig1 = figure();
    fig1.Position = [100, 100, 1000, 600];
    sgtitle("Model states evolution - " + scenario);
    subplot(3,1,1);
    plot(sim_time ,x1, sim_time_ref ,x1_ref,'LineWidth',1.5);
    ylabel('SOC [-]');
    xlabel('Time [s]');
    legend("Controlled", "Reference");
    title('x1 (SOC) evolution')
    grid on;

    subplot(3,1,2);
    plot(sim_time ,x2, pos_leading.Time ,pos_leading.Data, 'LineWidth',1.5);
    ylabel('Position [m]');
    xlabel('Time [s]');
    legend("Controlled", "Reference");
    title('x2 (position) evolution')
    grid on;

    subplot(3,1,3);
    plot(sim_time ,x3,sim_time_ref,x3_ref,'LineWidth',1.5);
    ylabel('Velocity [m/s]');
    xlabel('Time [s]');
    legend("Controlled", "Reference");
    title('x3 (Velocity) evolution');
    grid on;

    % control torque and state derivative
```

```
    fig2 = figure();
    fig2.Position = [100, 100, 1000, 600];
    sgtitle("MPC control torque output and plant acceleration - " + scenario);
    subplot(2,1,1)
    plot(sim_time, T_EM_MPC.Data, sim_time_ref, T_EM_ref.Data, 'LineWidth',1.5);
    grid on;
    legend("Controlled", "Reference")
    xlabel("Time [s]")
    ylabel("Torque [Nm]")
    title('MPC control torque output')

    subplot(2,1,2)
    plot(sim_time, x3_dot, sim_time_ref, a_ref.Data, 'LineWidth',1.5);
    grid on;
    xlabel("Time [s]")
    ylabel("Acceleration [m/s^2]")
    legend("Controlled", "Reference")
    title('MPC model acceleration (x3_{dot})')

    fig3 = figure();
    fig3.Position = [100, 100, 1000, 600];
    subplot(2,1,1)
    plot(sim_time, x2,pos_leading.Time,pos_leading.Data,'LineWidth',1.5);
    grid on;
    xlabel("Time [s]")
    ylabel("Position [m]")
    legend("Controlled", "Reference")
    title('Leading and precesding vehicles positions - ' + scenario);

    subplot(2,1,2)
    plot(sim_time, pos_leading.Data - x2,'LineWidth',1.5);
    grid on;
    xlabel("Time [s]")
    ylabel("Position [m]")
    title('Relative distance - ' + scenario);
else
    fig1 = figure();
    fig1.Position = [100, 100, 1000, 600];
    sgtitle("Model states evolution - " + scenario);
    subplot(3,1,1);
    plot(sim_time ,x1, sim_time_ref ,x1_ref,'LineWidth',1.5);
    ylabel('SOC [-]');
    xlabel('Time [s]');
    legend("Controlled", "Reference");
```

```
    title('x1 (SOC) evolution')
    grid on;

    subplot(3,1,2);
    plot(sim_time ,x2, pos_leading.Time ,pos_leading.Data, 'LineWidth',1.5);
    ylabel('Position [m]');
    xlabel('Time [s]');
    legend("Controlled", "Reference");
    title('x2 (position) evolution')
    grid on;

    subplot(3,1,3);
    plot(sim_time ,x3,sim_time_ref,x3_ref,'LineWidth',1.5);
    ylabel('Velocity [m/s]');
    xlabel('Time [s]');
    legend("Controlled", "Reference");
    title('x3 (Velocity) evolution');
    grid on;

    % control torque and state derivative

    fig2 = figure();
    fig2.Position = [100, 100, 1000, 600];
    sgtitle("MPC control torque output and plant acceleration - " + scenario);
    subplot(2,1,1)
    plot(sim_time, T_EM_MPC.Data, sim_time_ref, T_EM_ref.Data, 'LineWidth',1.5);
    grid on;
    legend("Controlled", "Reference")
    xlabel("Time [s]")
    ylabel("Torque [Nm]")
    title('MPC control torque output')

    subplot(2,1,2)
    plot(sim_time, x3_dot, sim_time_ref, a_ref.Data, 'LineWidth',1.5);
    grid on;
    xlabel("Time [s]")
    ylabel("Acceleration [m/s^2]")
    legend("Controlled", "Reference")
    title('MPC model acceleration (x3_{dot})')

    fig3 = figure();
    fig3.Position = [100, 100, 1000, 600];
    subplot(2,1,1)
    plot(sim_time, x2,pos_leading.Time,pos_leading.Data,'LineWidth',1.5);
```

```
    grid on;
    xlabel("Time [s]")
    ylabel("Position [m]")
    legend("Controlled", "Reference")
    title('Leading and precesding vehicles positions - ' + scenario);

    subplot(2,1,2)
    plot(sim_time, pos_leading.Data - x2,'LineWidth',1.5);
    grid on;
    xlabel("Time [s]")
    ylabel("Position [m]")
    title('Relative distance - ' + scenario);

end
```

MIL initialization, simulation and plotting

```
    %% Model Parameters for Battery Electric Vehicle System Model
close all
clear all
clc

addpath("..\500e_Frugal_MIL- original\500e_Frugal_MIL");

%% Maneuver
load('DrivingCycles\WLTP.mat');              % Speed and Time vectors for desired Driving Cycle
% load('DrivingCycles\custom_cycle.mat');
T_z = T_z(1:1801);                                % [s]     Time
V_z = V_z(1:1801)/3.6;                                % [m/s] Uncomment for RDE simulations
% t_WLTC = max(T_z);                              % [s]     WLTC time
% t_WLTC_city = 1000;                               % [s]     WLTC-city time
% t_RDE_Urban = 2326;                             % [s]     RDE-Urban time
t_EDAS = max(T_z);
Time = 400;                                  % [s]     Simulation Time

%% Vehicle, MPC and simplified vehicle powertrain paramters
run("Init_FWD_Frugal_MPC.m")                        % 500e Frugal

%% Open Simulink Model
open('MIL_Frugal_500e_with_MPC.slx')
open("../500e_Frugal_MIL- original/500e_Frugal_MIL/MIL_Frugal_500e.slx")

%% ------------------------- EAD SCENARIO CREATION -------------------------------
```

```
% Vscenario(:,1) = T_z;
% Vscenario(:,2) = V_z;

%% General EAD Parameters
% TLpreview(:,1) = T_z;
% TLpreview(:,2) = TLpreview_z;
% TLstate(:,1) = T_z;
% TLstate(:,2) = TLstate_z;

%% ----------------------------- ACC SCENARIO CREATION
    ------------------------------------
Vscenario(:,1) = T_z;
Vscenario(:,2) = V_z;

%% General ACC Parameters
Ts = 0.2;                       % Simulation sample time                    (s)
v_set              = 20;        % ACC set speed URBAN                       (m/s
    )
default_spacing = 2;            % ACC default spacing URBAN                 (m)
max_spacing        = 50;        % ACC default spacing URBAN                 (m)
time_gap           = 3;         % ACC time gap                              (s)
max_acc            = 2;         % Maximum acceleration                      (m/s
    ^2)
min_acc            = -3;        % Minimum acceleration                      (m/s
    ^2)
% Classical ACC / CTG Parameters
verr_gain          = 0.1;       % ACC velocity error gain - CTG             (N/A
    )
xerr_gain          = 0.3;       % ACC spacing error gain - CTG              (N/A
    )
vx_gain            = 0.5;       % ACC relative velocity gain - CTG          (N/A
    )

%% Simulations
sim('MIL_Frugal_500e_with_MPC.slx')
sim("../500e_Frugal_MIL- original/500e_Frugal_MIL/MIL_Frugal_500e.slx
    ")


%% Parameters for Battery Electric Vehicle Forward Model

%% Vehicle
% 500e Frugal Parameters
vehicle.mass = 900 + 100 + 0.15*350;            % [kg]  WLTP test mass
vehicle.wheelbase = 2.322;                      % [m]   wheelbase - Dati
    500e LR
vehicle.aCG = 0.45*vehicle.wheelbase;           % [m]   front axle - CoG
    distance - Dati 500e LR
```

```
vehicle.bCG = 0.55*vehicle.wheelbase;          % [m]      rear axle - CoG
    distance - Dati 500e LR
vehicle.hCG = 0.3;                             % [m]      height CoG -
    Dati 500e LR
vehicle.Af = 2.15;                             % [m2]     Frontal area -
    NEW Coast Down
vehicle.Cd = 0.33;                             % [-]      Drag coefficient
     - NEW Coast Down
vehicle.tireRRcoeff = 0.006;                   % [-]      Rolling
    Resistance Coeff f0 - Dati 500e LR
vehicle.tireRollingRadius = 0.3;               % [m]      Wheel Radius -
    Dati 500e LR
% 500e LR Original Coast Down
vehicle.roadLoadA_N = 55.78;                   % [N]        F0 - NEW Coast
     Down a 1250kg
vehicle.roadLoadB_N_per_kph = 0;               % [N/kph]  F1 - NEW Coast
     Down
vehicle.roadLoadC_N_per_kph2 = 0.0335;         % [N/kph2] F2 - NEW Coast
     Down
% Other
vehicle.roadLoad_gravAccel_m_per_s2 = 9.81;
smoothing.vehicle_speedThreshold_kph = 1;
smoothing.vehicle_axleSpeedThreshold_rpm = 1;
initial.vehicle_speed_kph = 0;
road_grade = atan(0/100);

%% 52V Battery
battery52V.nominalVoltage_V = 51.8;
battery52V.internalResistance_Ohm = 0.0056*2;
battery52V.nominalCapacity_kWh = 51.8*315/1000;
battery52V.voltagePerCell_V = 3.7;  % Open Circuit Voltage. 3.5V to
    3.7V assuming Lithium-ion
battery52V.nominalCharge_Ahr = ...
  battery52V.nominalCapacity_kWh / battery52V.nominalVoltage_V *
    1000;
battery52V.mass_kg = battery52V.nominalCapacity_kWh / 0.172; % kWh /
    (kWh/kg)
% Initial conditions
initial.Battery_SOC_pct = 90;
initial.Battery_Charge_Ahr = battery52V.nominalCharge_Ahr * initial.
    Battery_SOC_pct/100;

%% Reduction Gear
bevGear.gearRatio = 13;
bevGear.efficiency = 0.97;

%% 52V Motor Drive Unit -
load("N42.mat");
Speed_max = N42.MAX_Speed(:,1);
```

```
Shaft_Torque_max = N42.MAX_Shaft_Torque;
Shaft_Power_max = N42.MAX_Shaft_Power;
Speed_cont = N42.CONT_Speed(:,1);
Shaft_Torque_cont = N42.CONT_Shaft_Torque;
Shaft_Power_cont = N42.CONT_Shaft_Power;

motorDrive.simplePmsmDrv_trqMax_Nm = max(N42.Shaft_Torque);
motorDrive.simplePmsmDrv_powMax_W = max(N42.Shaft_Power);
motorDrive.simplePmsmDrv_timeConst_s = 0.02;

motorDrive.simplePmsmDrv_rotorInertia_kg_m2 = 3.93*0.01^2;
motorDrive.simplePmsmDrv_rotorDamping_Nm_per_radps = 1e-5;
motorDrive.simplePmsmDrv_initialRotorSpd_rpm = 0;

motorDrive.spdCtl_trqMax_Nm = motorDrive.simplePmsmDrv_trqMax_Nm;
motorDrive.gearRatioCompensation = 3/bevGear.gearRatio;

%% Controller & Environment
bevControl.MotorSpdRef_tireRadius_m = vehicle.tireRollingRadius;
bevControl.MotorSpdRef_reductionGearRaio = bevGear.gearRatio;

bevControl.MotorSpdRef_Ki = 10; %15;
bevControl.MotorSpdRef_Kp = 0.2; %15;

%% Simplified EV powertrain and vehicle plant

addpath("..\Simplified EV model\model with MPC\polynomial_fits")
addpath("..\Simplified EV model\model with MPC\data\Eff_map")
addpath("..\Simplified EV model\model with MPC\data")

% simulation parameters
MPC.g = 9.81; % Gravity acceleration [m/s^2]
MPC.alpha = 0*pi/180; % road slope [rad]
MPC.rho = 1.25; % air density [kg/m^3] - from Onori HEV book

% battery parameters
MPC.N_s = 108; % Number of series
MPC.N_p = 1; % Number of parallels
MPC.N_b = MPC.N_s*MPC.N_p; % number of battery cells
MPC.Q_nom = 60*MPC.N_p; % nominal battery capacity [Ah]
MPC.eta_c = 0.95; % Coloumbic efficiency
load('bat_Ro_vs_SOC_data.mat'); % Ro variation with SOC - single
    battery cell
load('bat_Voc_vs_SOC_data.mat'); % Voc variation with SOC - single
    battery cell
MPC.F_Voc_s = griddedInterpolant(SOC_Voc_data(:,1),SOC_Voc_data(:,2))
    ; % interpolated Voc data for a single cell
MPC.F_Ro_s = griddedInterpolant(SOC_Ro_data(:,1),SOC_Ro_data(:,2)); %
    interpolated Ro data for a single cell
```

```
MPC.SOC_vec = linspace(0,1,500);
MPC.Voc_s = MPC.F_Voc_s(MPC.SOC_vec); % vector with interpolated
    values of Voc_s (single cell)
MPC.Ro_s = MPC.F_Ro_s(MPC.SOC_vec); % vector with interpolated values
     of Ro_s (single cell)

% vehicle parameters
MPC.M_veh = 1400; % vehicle mass [kg]
MPC.a = 1; % Front axle - CoG  Front axle - CoG  [m]
MPC.b = 1.3; % Rear axle - CoG  [m]
MPC.h = 0.3; % Height of CoG [m]
MPC.f_0 = 4.5*1e-3; % Static rolling coefficient [N/kN]
MPC.k = 0; % miscellaneous loss coeff [Ns/m]
MPC.r_w = 0.3; % Wheel radius [m]
MPC.C_d = 0.33; % Drag coeff
MPC.A_f = 2.15; % Frontal area [m^2]
MPC.tau_gb = 9.6; % Gear ratio
MPC.eta_gb = 0.97; % Gearbox efficiency
MPC.eta_inv = 1; % Inverter efficiency

% electric motor parameters
load("Eff_map\Efficiency.mat"); % loading the EM efficeincy map
load("Eff_map\Shaft_torque.mat"); % loading shaft torque map
load("Eff_map\Speed.mat"); % loading speed map
MPC.Speed_max = Speed(:,1);
MPC.Torque_max = Shaft_Torque(:,1);
MPC.Efficiency(Efficiency == 0) = 10;
MPC.eff = Efficiency/100;

%% MPC

% PARAMETERS
MPC.Ts = 0.05; % Sampling time

MPC.par.nx          = 3; % number of states
MPC.par.nu          = 1; % control elements number
MPC.par.ny          = 3; % number of outputs
MPC.par.model = @prediction_longitudinal_model; %Modello di
    predizione
MPC.par.ub          = 250; % Upper bound saturazione input -> maximum
    value for control output
MPC.par.lb          = -250; % Lower bound saturazione input -> minimum
     value for control output
MPC.par.tol         = 1; % Reference tolerance
MPC.par.Nfev        = 150; % Interation number of fmincon in cost
    function (default 200)
MPC.par.Ts          = MPC.Ts;
```

```
% initial conditions
MPC.SOC0 = 0.8;
MPC.x0 = 0;
MPC.v0 = 0;
MPC.xx0 = [MPC.SOC0; MPC.x0; MPC.v0];

% Position reference parameters
MPC.par.R = 0.01; % matrice diagonale definita positiva per cost
    function
MPC.par.P = diag([0;50000;0]); % matrice diagonale definita positiva
    per cost function
MPC.par.Q = diag([0;1;0]); % matrice diagonale definita positiva per
    cost function
MPC.par.Tp = 10*MPC.Ts; % Prediction horizon (sempre multiplo intero
    del Ts)
MPC.K = nmpc_design_4b(MPC.par); %Generazione parametri design NMPC


%% Eco-Driving Analysis - MPC controlled model

 directory = "C:\Users\gabri\Documents\TCC - EV powertrain control\
    Text\images";
 screen_size = get(0,"ScreenSize");
 fig_position = [0 0 screen_size(3) screen_size(4)];


% run("Init_MIL_Model_MPC.m")
 load("simulation_results.mat");

%% PLOTS
% POWERTRAIN SIGNALS
 fig1 = figure(1);
 fig1.Position = fig_position;

 sgtitle("MPC controlled and reference powertrain results")
 subplot(3,1,1)
 plot(veh_speed_MPC.Time,veh_speed_MPC.Data, 'b', veh_speed.Time,
    veh_speed.Data,'r','LineWidth', 1.5)
 title("Speed profile");
 xlabel("Time [s]");
 ylabel("Speed [m/s]");
 legend("Controlled", "Reference");
 grid on;

 subplot(3,1,2)
 plot(motor_speed_MPC.Time,motor_speed_MPC.Data, 'b', motor_speed.Time
    , motor_speed.Data,'r','LineWidth', 1.5)
 title("Motor speed profile");
 xlabel("Time [s]");
```

```
ylabel("Motor Speed [rpm]");
legend("Controlled", "Reference");
grid on;

subplot(3,1,3)
plot(T_EM_MPC.Time,T_EM_MPC.Data, 'b', T_EM.Time, T_EM.Data,'r','
    LineWidth', 1.5)
title("Electric Motor torque profile [Nm]");
xlabel("Time [s]");
ylabel("Motor Torque [Nm]");
legend("Controlled", "Reference");
grid on;

filename = sprintf('result_MIL_v_w_T_%d.jpg', Time);
fullFileName = fullfile(directory, filename);
saveas(fig1, fullFileName);


% BATTERY SIGNALS
fig2 = figure(2);
fig2.Position = fig_position;

sgtitle("MPC controlled and reference battery results")
subplot(2,1,1)
plot(battery_MPC.Time,battery_MPC.Data(:,1), 'b', battery.Time,
    battery.Data(:,1),'r','LineWidth', 1.5)
title("Battery Charge");
xlabel("Time [s]");
ylabel("Charge [Ah]");
legend("Controlled", "Reference");
grid on;

subplot(2,1,2)
plot(battery_MPC.Time,battery_MPC.Data(:,2), 'b', battery.Time,
    battery.Data(:,2),'r','LineWidth', 1.5)
title("Electrical Efficiency");
xlabel("Time [s]");
ylabel("Electrical Efficiency [kWh/100km]");
legend("Controlled", "Reference");
grid on;

filename = sprintf('result_MIL_battery_%d.jpg', Time);
fullFileName = fullfile(directory, filename);
saveas(fig2, fullFileName);

% RELATIVE DISTANCE
fig3 = figure(3);
fig3.Position = fig_position;

```

```
plot(Rel_distance_MPC.Time,Rel_distance_MPC.Data, 'b', Rel_distance.
    Time,Rel_distance.Data,'r','LineWidth', 1.5)
title("MPC controlled model vs uncontrolled model - relative distance
     comparison");
xlabel("Time [s]");
ylabel("Distance [m]");
legend("Controlled", "Reference");
grid on;

filename = sprintf('result_MIL_rel_distance_%d.jpg', Time);
fullFileName = fullfile(directory, filename);
saveas(fig3, fullFileName);
```

Appendix C

Python scripts

Python code for EM efficiency polynomial fit:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

# Step 1: Read the CSV files
angular_velocity = pd.read_csv('speed.csv',header=None, sep=';').
    values
torque = pd.read_csv('torque.csv', header=None, sep=';').values
efficiency = pd.read_csv('eff.csv', header=None, sep=';').values

# Step 2: Flatten the matrices and scale efficiency
angular_velocity_flat = angular_velocity.flatten()
torque_flat = torque.flatten()
efficiency_flat = efficiency.flatten() / 100.0  # Scale efficiency to
     be between 0 and 1

# Step 3: Create a meshgrid of the angular velocity and torque
X = np.column_stack((angular_velocity_flat, torque_flat))

# Step 4: Polynomial fitting
poly = PolynomialFeatures(degree=3)
X_poly = poly.fit_transform(X)

model = LinearRegression()
model.fit(X_poly, efficiency_flat)

# Step 5: Extract polynomial coefficients
coefficients = model.coef_
intercept = model.intercept_
```

```
feature_names = poly.get_feature_names_out(['w', 'T'])

# Function to predict efficiency using the polynomial model
def predict_efficiency(angular_velocity, torque):
    X_new = np.column_stack((angular_velocity.flatten(), torque.
    flatten()))
    X_poly_new = poly.transform(X_new)
    efficiency_pred = model.predict(X_poly_new)
    return efficiency_pred.reshape(angular_velocity.shape)

terms = [f'{coef:.14f}*{name}' for coef, name in zip(coefficients,
    feature_names)]
polynomial_expression = ' + '.join(terms)
polynomial_expression = f'{intercept:.14f} + ' +
    polynomial_expression
print("Polynomial Fit for MATLAB:")
print(polynomial_expression)
```