Politecnico di Milano
Department of Mechanical Engineering
Doctoral Programme In Mechanical Engineering

Optimal Powertrain Design and Control of an Electric Race Car

Doctoral Dissertation of:
HUILONG YU

Supervisor:
Prof. Francesco Castelli Dezza and Federico Cheli

Tutor:
Prof. Marco Giglio

The Chair of the Doctoral Program:
Prof. Maria Bianca Colosimo

2017 – XXIX

Acknowledgments

This PhD work has been carried out at the Department of Mechanical Engineering of Politecnico Di Milano, Italy. Looking back through my PhD, it has been a truly arduous but rewarding journey. Without the help and supports of many people who devoted their knowledge, time and patience, I am not able to goes so far in this journey.

First and foremost, I would like to express my sincere gratitude to Prof. Francesco Castelli Dezza and Prof. Federico Cheli for their guidance, encouragement and patience over the past three years. Besides the daily meeting, I occupied a lot of their personal time at weekends for the discussion and review of my PhD research activities. Their suggestions and comments are priceless for me to push the work to a new limit. Besides the academical help, they are also very kind and always provides me with timely assistance in daily life. Without their help and supports, it is impractical for me to complete this thesis work. The positive influences of not only their professional knowledge but also their excellent personality will resonate throughout my life.

I would like to thank Dr. Paolo Schito who helped me a lot in using the cluster for parallel computing. Prof. Enrico Bertolazzi from University of Trento, Dr. Yutao Chen from University of Padova, and Dr. Chen Zhang from Tsinghua University gave me a lot of valuable suggestions on optimal control and nonlinear programming, special thanks also to them. I would like to thank Prof. Stefano Melzi, Prof. Massimiliano Gobbi and Dr. Davide Tarsitano who provided me help and advices in vehicle modelling, optimal design and other research activities respectively. Thanks to Prof. Federico Cheli, I had the chances to communicate with the senior researcher Luca De Bernardi from Ferrari and Davide Almerico from NextEV who shared their precious industrial experiences in racing cars, I obtained the data of the vehicle model from VI-Grade and the researchers from VI-Grade helped me a lot in using their software, thanks also go to them for their time and suggestions.

During my PhD in Milano, it's my great honor to meet and make so many wonderful friends, the great memories here will be with me forever. Thanks to all of you, and I will always appreciate the rewards and favors from each of you.

Finally, I would like to thank my parents and sisters, who have been always supporting and encouraging me unconditionally. Their love is the source of energy and inspiration that keeps me going on. Last but not least, I owe a lot to my girlfriend Miss Lu, who left her stable life in BeiJing and joined my PhD journey in Milano. I greatly appreciated her love and accompany, she did much more than she could to support and understand me.

25.04.2017
Huilong Yu
Milano, Italy

Abstract

In order to face the growing challenges from air pollution, dependence on fossil oil and greenhouse gas emissions, electric vehicles have attracted unprecedented amount of global attentions from the governments, academia, industry, public and environmental organizations. Electric racing becomes more popular under this background: a new championship called Formula E has been held for three years since 2014. An upcoming championship named Roborace in 2017 will be the first global championship for autonomous electric race cars, which will open a new page of racing. Electric racing is undoubtedly a good platform to draw the attentions of the public on electric vehicles and also for testing and improving the most advanced design and control technologies. Compared with conventional internal combustion engine vehicles, electric vehicles can have very flexible powertrain topologies, which can be 1-motor, 2-motor and 4-motor driving with different mounted positions, and the transmissions can be single-speed or multi-speed. From the design point of view, the selection of the powertrain layouts, the motors and transmissions can affect the dynamic performance of the electric vehicles directly. As for control, different steering, accelerating and braking operations will result different trajectories, velocity profiles and lap time. Thus, the advanced design and control technologies are always expected.

Recognizing the limitations of the conventional powertrain design approaches, this work is dedicated to achieving further improvements by proposing innovative optimal design approaches of the electric powertrain, with an electric race car as the platform. In order to test the performance of a designed powertrain, a corresponding control strategy should be developed. However, there are various kinds of control approaches for the electric vehicles, and accordingly, different control strategies may result in different results with the same designed powertrain. Considering this, the optimal control of the electric race car is coupled into the optimal design problem. The final results includes both the optimal design and control solutions for different powertrain layouts.

In order to represent the vehicle behavior for cornering, braking, acceleration

and comfort performance studies for four wheel driving vehicles with independent suspensions more accurately, a 14-DOF vehicle model is necessary. In this work, a 14-DOF vehicle model together with a suspension model considering the details of toe angle, camber angle, anti-roll force and suspension forces, are developed based on Lagrangian dynamics. To accurately predict the behavior of a vehicle, it is also required to estimate the external forces acting on the vehicle as precisely as possible. An empirical tire model based on the well-known Magic formula equations is programmed to calculate the tire forces. In particular, the tire model developed in MATLAB supporting inputs of the standard '.tir' tire data file. In order to evaluate the effect of design parameters of the motor and the transmission to the lap time of the race car, the mass model of the motor and transmission mainly concerning the dependence of the mass and output torque of the powertrain on the design parameters are derived. A virtual driver model is also devised to track a given trajectory depicted in curvilinear coordinate system based on the proposed control logic, and the obtained results are served as the initial guess of the optimal powertrain design and control problem. Heavy computing workload is a common issue in large scale optimization and optimal control problem. In order to improve the computational efficiency, all of the mentioned models programmed supports matrices operations. The entire vehicle model is validated with a well-known vehicle dynamics simulator 'VI-CarRealTime' developed by VI-Grade.

After a detailed reviewing of the numerical approaches for optimal control problems, a MATLAB software package for General DYNamic OPTimal control problems (abbreviated as GDYNOPT) based on direct collocation methods is developed. GDYNOPT is implemented different transcription methods including both the local collocation and global collocation approaches, differential methods including forward, central, complex step and analytical differential methods. Moreover, it has the features of automatic scaling based on linear scaling and a proposed average gradients scaling approach, sparsity and supporting parallel computation.

The optimal powertrain design and control problems of the 1-motor driving, 2-motor and 4-motor driving topologies based on the developed entire vehicle model are formulated and solved with GDYNOPT based on an direct transcription method for the first time with reference to the existing literature. In addition, an innovative approach is proposed to smooth the control trajectories. The optimal powertrain design parameters, control arcs and the optimal racing lines are obtained and analyzed with different number of collocation nodes.

The obtained optimal design parameters in this work can be used as the reference for the motor and transmission design of electric race cars, while the optimal control results can serve as the benchmark to develop and evaluate the closed loop control strategy. In addition, the obtained racing line and steering wheel angles can be used to train the race car driver in a car simulator. The methodology proposed in this work can also be applied in the design and control of the common type ICE vehicles, EVs and HEVs with different driving profiles and objective functions.

Contents

CHAPTER 1

Introduction

1.1 Overview and background

Electric vehicles (EVs) first came into existence before the mid-19th century, which have longer history than internal combustion engine (ICE) vehicles, and later were manufactured by several companies of the USA, England, and France. EVs provided a level of comfort and ease of operation that could not be achieved by the ICE vehicles of the time when electricity was among the preferred methods for motor vehicle propulsion. However, due to the poor performance of their batteries contrasting to the rapidly developed ICE technology, EVs have almost disappeared from the global scene from 1930 [1].

The extremely high energy and power density of gasoline and petrol with their abundance and low price have made ICE the dominant propulsion solution of vehicles for almost 100 years. Modern ICE vehicles have the performances of top comfort, excellent dynamics and advanced safety with relatively low prices and have been the most attractive transportation products. However, despite approximately a century-long struggling to improve efficiency of ICE by the industry and academia, advanced designed ICEs still suffer from incredibly low efficiency in the range from 30% to 44%, which means that there is approximately 70% of the energy liberated by the combustion is lost. The typical energy path in gasoline fueled internal combustion engine vehicles is presented as Figure 1.1.The resulting situation is worse considering that the emissions formed mostly of CO_2, NO_x, C_xH_y [2], CO and soot, which are the principle culprits of air pollution and greenhouse effect.

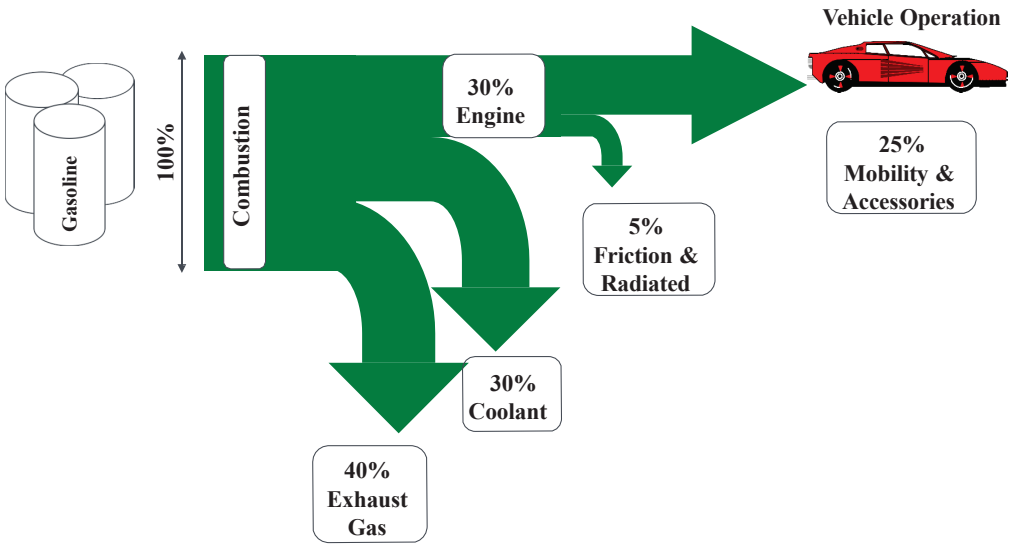

Figure 1.1: *Typical energy path in gasoline fueled internal combustion engine vehicles* [3]

Despite the dominance of ICE vehicles, electricity has remained its commonplace in trains and other vehicle types, and later in the early 1970s, triggered by the energy crisis, air pollution and greenhouse gas emissions, the world's greatest carmakers, governments, environmental organizations and academia started the rekindling of interests in EVs. After decades of research and development, EVs has already gained notable market share: by May 2015, more than 500,000 highway-capable all-electric passenger cars and light utility vehicles have been sold worldwide since 2008 [4]. Cumulative global sales of all-electric cars and vans passed the 1 million unit milestone in September 2016 [5]. However, high cost and short cycle life of batteries have always been the problem hindering their developing process and penetration, besides, complicated design and control of the flexible electric powertrain topologies have highlighted their limitations. As addressed in [1], advanced energy sources and intelligent energy management are key factors to enable EVs competing with ICE vehicles, moreover, novel design and control approaches are essential for EV engineering due to their fundamentally different characteristics compared with ICE vehicles. In recent years, due to the advances of technologies and to face the more severe challenges of energy and environmental crisis, EVs have aroused unprecedented amount of attentions from the researchers and manufacturers all over the world.

A typical electric powertrain consists of energy storage system, DC/AC inverter, electric motor, transmission and DC/DC converters will be introduced if hybrid energy storage systems are adopted. The electric motor outputs propulsion/braking torque by consuming/generating electrical energy from/to the energy storage system through a controlled DC/AC inverter, which regulates the amount of power according to the position of accelerator/brake pedal controlled by the driver. The propulsion system consists of electric motors and transmissions, plays an important role in both the dynamics and economic performances of an EV. The configurations of an electric

propulsion system can be more flexible compared with conventional ones driven by internal combustion engines, the possible configurations might be center drive, rear drive, front drive and all wheel drive with different numbers and mounting positions of the motors and transmissions. The possible topologies of the EVs are shown as Figure 1.2. The optimal propulsion system of an EV should be able to exhibit the best performance at aspects of dynamics, economic and handling with the least cost and maintenance. All of these evaluation indexes are significantly influenced by the configurations and related control strategies of the electric propulsion systems.

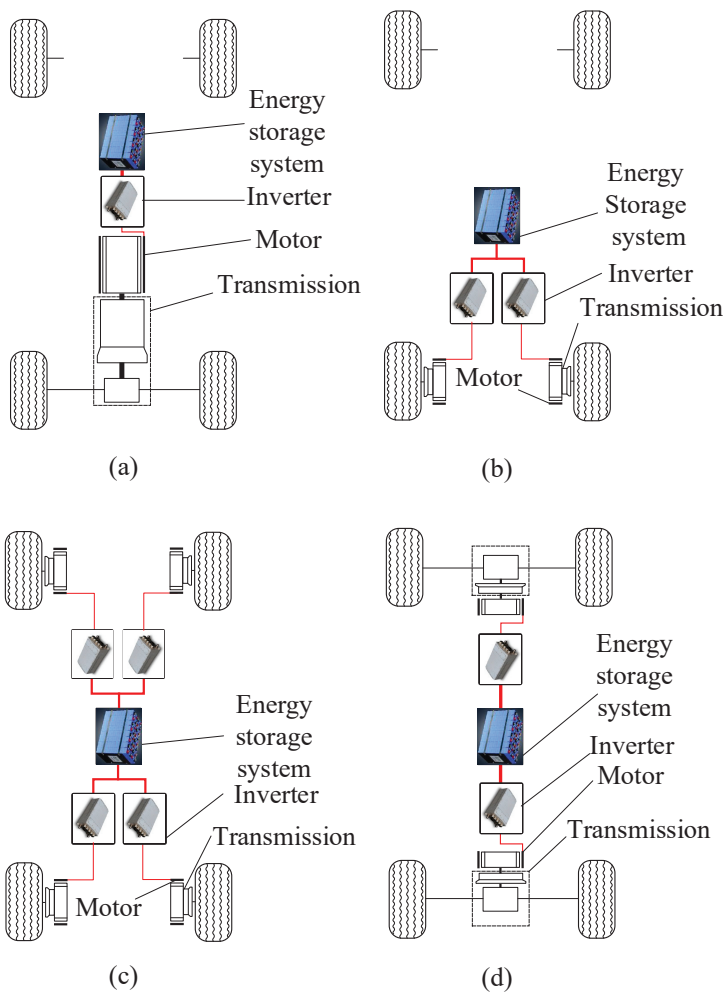

Figure 1.2: *Possible configurations of electric propulsion system. (a) Rear wheel central drive, (b) Rear wheel independent drive, (c) Four-wheel independent drive, (d) Four-wheel central drive*

Motor racing is a platform to test the feasibility and practicality of the most advanced technological innovations in vehicle industry. A lot of the most advanced technologies of the day were gradually applied on civilian vehicles after being developed, validated and improved on race vehicles, such as the turbocharging technology, the paddle shifter technology and the air spring technology, etc. Motor racing has been playing

an important role in the development of internal combustion engine powered ground vehicles since the early 20th century. Similarly, electric vehicle racing can also stimulate the academic organizations, manufacturers and customers to promote the development and application of the electric vehicles [6]. The electric vehicle racing events and categories has been increasing rapidly during the last decades. The inaugural championship Formula-E officially sanctioned by the Fédération Internationale de l'Automobile (FIA) started in Beijing September 2014 has successfully addressed its main objective to represent a vision for the future of motor industry, serving as a framework for R&D around the electric vehicle, accelerating general interest in these cars and promoting clean energy and sustainability [7]. Electric race car might be the most appropriate platform to realize and validate the most advanced technologies of electric powertrain.

Most of the research interests on EVs are focused on battery management strategy [8–10], electric motor control [11–13] and dynamics control [14–17]. Continuous research on these topics have improved the energy efficiency and dynamic performance of the EVs significantly. However, there are still some general recognized problems entrenched in the electric powertrain design and control, which hindering the developing process and penetration of EVs.

1.2 Motivations and aims

Normally, the proper parameters of the motor and transmission are the primary considerations to meet the performance requirements in electric vehicle design [18]. This is due to the fact that the power density, dynamic performance, energy efficiency and cost of an electric powertrain rely heavily on the matching of the motor and gearbox. Most of the electric powertrains are designed following a conventional method that can be summarized into three steps [19–27]. The first step is to define the motor power according to the requirements on dynamic performance with a simplified single point-mass vehicle model. Then the motor will be chosen from the available products of the manufacturers considering the power density and cost, etc. Generally, the last step is to select the gearbox according to the torque-speed characteristics of the motor, the required maximum speed and maximum torque on the wheels.

The limitations of the conventional approach are illustrated as follows. First, the searching space of the powertrain design is limited only within the one of transmissions since the motor is preselected. Second, relationship between the concerned performances of the EV, the control strategies and the driving conditions are not considered in detail throughout the design process. Recognizing the challenges in design of the EVs, this work is dedicated to achieving further improvements by proposing innovative optimal design approaches of the electric powertrain, with an electric race car as the platform.

In order to test the performance of a designed powertrain, a corresponding control strategy should be developed. However, there are various kinds of control approaches for the electric vehicles [28–33] and accordingly, different control strategies may result

in different results with the same designed powertrain. Considering this, the optimal control of the electric race car is coupled into the optimal design problem in this work. The final results will include both the optimal design and control solutions.

1.3 Contributions

The main contributions of this work are the formulating and solving of the optimal electric powertrain design and control problems with a developed optimal control software package in MATLAB . The details are elaborated as followings:

1) In the problem formulation part, in order to predict the overall vehicle behavior for cornering, braking, acceleration and comfort performance for different propulsion topologies with independent suspensions more accurately, a 14 DOF vehicle model together with a suspension model considering the details of toe angle, camber angle, anti-roll force, etc., is developed based on Lagrangian dynamics. To accurately predict the behavior of a vehicle, it is also required to estimate the external forces acting on the vehicle as precisely as possible. An empirical tire model supporting the inputs of standard '.tir' tire data file is programmed based on the full set Magic Formula equations in MATLAB to calculate the tire forces .

2) In order to evaluate the effect of design parameters of the motor and transmission to the dynamic performance of the race car, the mass model of the motor and transmission mainly concerning the dependence of the mass and output torque of the powertrain on the design parameters are derived.

3) A virtual driver model is devised to track a given trajectory depicted in curvilinear coordinate system based on the proposed control logic. The obtained path following results are served as the initial guess of the optimal design and control problems.

4) Heavy computational workload is a common issue in large scale optimization and optimal control problem, in order to improve the calculation efficiency, the mentioned 14-DOF vehicle model, suspension model, tire model and the powertrain mass model are all programmed in vector formats to support matrix operations. The entire vehicle model is finally validated with a well-known vehicle dynamics simulator 'VI-CarRealtime' developed by VI-Grade.

5) A General DYNamic OPTimal control software package (abbreviated as GDYNOPT) supporting parallel computing implemented with different kinds of direct transcription methods, differential methods and the proposed projected automatic scaling method, is developed to solve the formulated large scale optimal design and control problems based on the NLP solver 'IPOPT' [34].

6) The optimal design and control problems of the 1-motor driving, 2-motor independent and 4-motor independent driving electric powertrains based on the 14-DOF vehicle model are formulated and solved with GDYNOPT based on direct methods for the first time with reference to the existing literature. In addition, an innovative approach is proposed to smooth the control trajectories. The optimal powertrain design parameters, control arcs and the optimal racing lines are finally obtained and analyzed with different

number of collocation nodes [1].

1.4 Outline

Chapter 2 provides the literature review of the optimal control theory and its numerical approaches. The state-of-the-art vehicular optimal control is also reviewed and analyzed to introduce the further improvement of this work.
Chapter 3 elaborated the derivation of the 14-DOF vehicle model, the implement of the tire model, the development of the suspension model and powertrain mass model. A virtual driver model is also developed to track a given trajectory depicted in curvilinear coordinate system based on the proposed control logic. The validation results of the whole vehicle model are finally demonstrated in different maneuvers.
Chapter 4 presents the development of software package GDYNOPT. The framework of GDYNOPT, the detail implementation of the local collocation and global collocation methods [35], the different differential methods, the proposed scaling method together with the algorithm flowchart of GDYNOPT are depicted.
Chapter 5 gives the descriptions of the formulation and the solution of the optimal powertrain design and control problems. The variables and constraints, simulation settings, the proposed control smooth method are illustrated. Finally, the obtained optimal design and control results are presented, analyzed and compared.
Chapter 6 summarizes the main findings of this research and presents the recommendations for future work and developments.

[1] In mathematics, a collocation method is a numerical solution of differential and integral equations. The main idea is to choose a finite-dimensional space of polynomials up to a certain degree and a number of points in the domain (the chosen points are the so called collocation nodes or collocation points), and to select the solution that satisfies the given equation at the collocation nodes.

CHAPTER 2

Optimal Control Theory and its Application in Vehicles

2.1 Introduction

People are always trying to find a way to obtain the most they desired at the least cost intentionally or unintentionally when they are doing something. The process to explore and seek the best way to do things is indeed the optimization process. Similarly, optimal control deals with the problem of finding a control strategy for a given dynamic system such that a certain optimality criterion can be achieved. The way to evaluate the optimality is by calculating the cost function (or performance index) which is a function of state, control variables. The dynamic system is generally described by a set of differential equations. While the control law is usually composed of time-dependent and static variables, in particular, the static variables might be the design variables of the system. Optimal control theory is an outcome of the calculus of variations, with a history stretching back over 300 years. Leonhard Euler, the Bernoulli brothers and Lagrange are the mathematicians who gave the foundations of the calculus of variations. In the second half of the 19th century, other important mathematicians contributed to the theorems of existence are Hamilton, Jacobi and Weierstrass. However, interests on it actually mushroomed only with the advent of the computer in the early 1960s, launched by the successful applications of optimal trajectory optimization in aerospace [36].

The passage from calculus of variations to optimal control is attributed to the Russian mathematician L.S. Pontryagin and the American mathematician Richard

Bellman in the middle of last centenary [37]. The first mathematician is famous for his *Pontryagin's maximum (minimum) principle*, the proof of which is historically based on maximizing the Hamiltonian. The class of indirect methods using the necessary conditions of optimality of the infinite problem to derive a boundary value problem (BVP) in ordinary differential equations (ODE) are based on the *Pontryagin's maximum (minimum) principle*. The second mathematician introduced dynamic programming (DP) in 1953, which is another class of methods for optimal control problem. Later, direct methods based on the direct transcription of the optimal control problem into nonlinear programming (NLP) are introduced, different transcription approaches have been intensively studied over the past decades. Up to this point, the numerical solutions for the optimal control problem can be categorized as Figure 2.1.

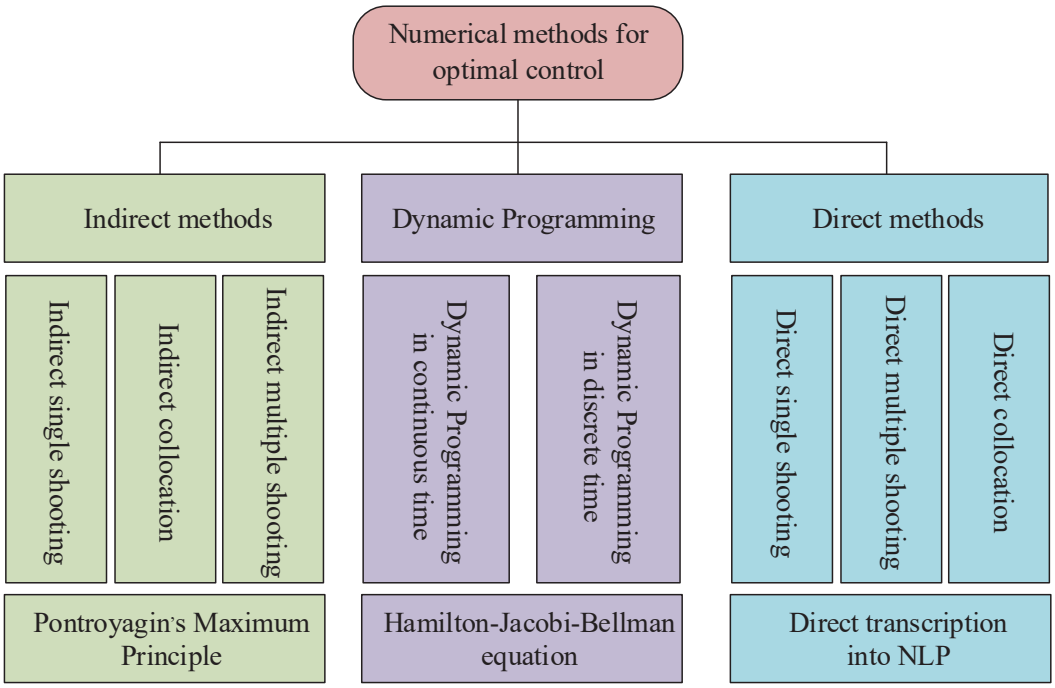

Figure 2.1: *Numerical methods for optimal control problems*

All of the three numerical approaches have their followers and have been improved based on the state-of-the-art development of nonlinear programming and integration of ordinary differential equations and differential-algebraic equations. As said by John T.Betts, who is the author of the well-known textbook "Practical methods for optimal control and estimation using nonlinear programming": *"Solving an optimal control is not easy. Pieces of the puzzle are found scattered throughout many different disciplines"* [38]. Studies on the optimal control problems in different applications and to solve them more quickly and more robustly are still a hot research topic. The following sections starting with a general formulation of the Optimal Control problems will give a more detail description of the three classes of methods. In addition, the

relevant theory of NLP which is the basis of many optimal control problems is also reviewed in this chapter.

2.2 General formulation

Without loss of generality, a constrained *Bolza* type optimal control problem can be formulated as,

$$J = \mathcal{M}(\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f, \mathbf{p}) + \int_{t_0}^{t_f} \mathcal{L}(\mathbf{x}(t), \mathbf{u}(t), t, \mathbf{p})dt \tag{2.1}$$

subject to the dynamic constraints

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{2.2}$$

the boundary constraints

$$\mathbf{b}_{min} \leqslant \mathbf{b}[\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f, \mathbf{p}] \leqslant \mathbf{b}_{max} \tag{2.3}$$

and the path constraints

$$\mathbf{g}_{min} \leqslant \mathbf{g}[\mathbf{x}(t), \mathbf{u}(t), t, \mathbf{p}] \leqslant \mathbf{g}_{max}. \tag{2.4}$$

The vector format state, control, and static variable can be written in a component form respectively as:

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ \vdots \\ x_{n_x}(t) \end{bmatrix}, \mathbf{u}(t) = \begin{bmatrix} u_1(t) \\ \vdots \\ u_{n_u}(t) \end{bmatrix}, \mathbf{p} = \begin{bmatrix} p_1 \\ \vdots \\ p_{n_p} \end{bmatrix} \tag{2.5}$$

The dimensions of the input and output variables in Equations (2.2), (2.3) and (2.4) are separately given as:

$$\begin{aligned} \boldsymbol{f} &: \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x} \\ \boldsymbol{g} &: \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_g} \\ \boldsymbol{b} &: \mathbb{R}^{n_x} \times \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_b} \end{aligned}$$

The terminal cost $\mathcal{M}$ in Equation (2.1) is usually called the *Mayer term* and the integral cost $\mathcal{L}$ is the so called *Lagrange term*, while the combination of the both terms is called a *Bolza* type optimal control problem. The cost function J is also commonly called the performance index which is used to evaluate the quality of the control or design solutions, Equation (2.2) is the differential equation that describes the system dynamics, Equation (2.3) is the constraint function of the terminal states, while Equation (2.4) is the constraint function of state and control trajectory at each time step. The objective of optimal control is to define the state $\mathbf{x}(t) \in \mathbb{R}^{n_x}$, control $\mathbf{u}(t) \in \mathbb{R}^{n_u}$, static parameter $\mathbf{p} \in \mathbb{R}^{n_p}$, initial time t_0, and final time t_f, to minimize the cost function J within the boundary constraints and path constraints, the schematic diagram is demonstrated in Figure 2.2.

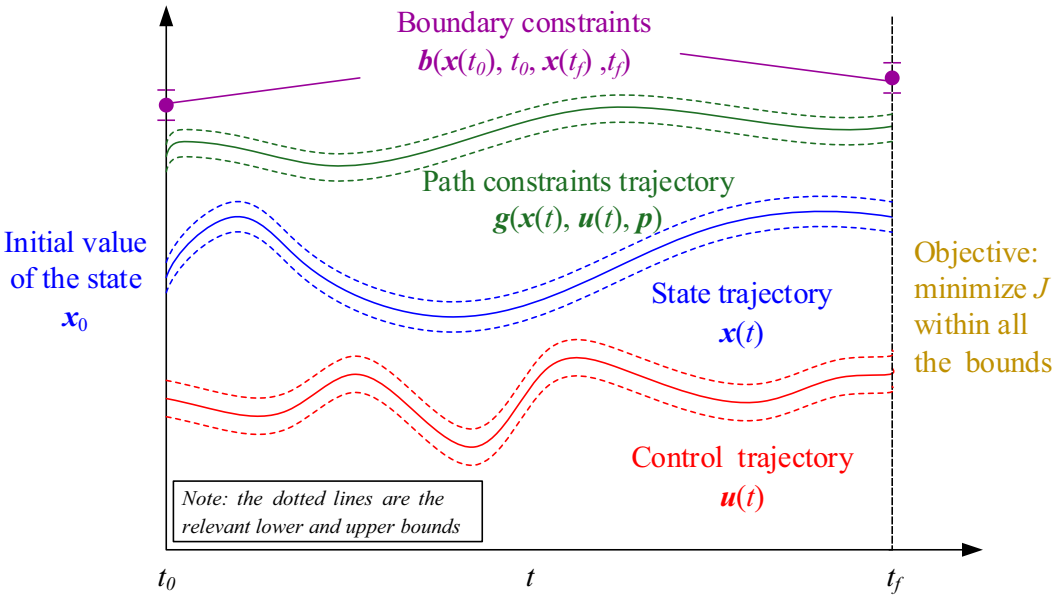

Figure 2.2: *The variables and constraints of a continuous optimal control problem*

Unless the system equations, along with the cost functional and the constraints, of the formulated problems are very simple, numerical methods must be employed to solve optimal control problems. With the development of economical, high speed computers over the last few decades, it has become possible to solve complicated problems in a reasonable amount of time. The followed sections will give a review of three common types of numerical approaches for the *Bolza* type optimal control problem.

2.3 Indirect methods

The indirect methods are based on the classic theory of calculus of variations and on the famous *Pontryagin's Maximum Principle*, whose full generality was developed in the middle of last centenary, starting with the work of Pontryagin and his coworkers [39]. One of the major achievements of their approach compared to the previous work is the possibility to treat inequality path constraints. In indirect methods, an optimal solution is found by satisfying first-order necessary optimality conditions instead of minimizing a cost criterion directly as in direct methods. Depending on the given optimal control problem, the necessary optimality conditions lead to a two-point or multi-point boundary value problem (BVP). Applying the calculus of variations to the optimal control problem given in Equations (2.1)-(2.4), the first-order necessary conditions for an extremal trajectory can be obtained. These conditions are generally derived with the augmented Hamiltonian function $\mathcal{H}$, which is defined as:

$$\mathcal{H}(\boldsymbol{x}, \boldsymbol{\lambda}_f, \boldsymbol{\lambda}_g, \boldsymbol{u}, t) = \mathcal{L} + \boldsymbol{\lambda}_f^T \mathbf{f} - \boldsymbol{\lambda}_g^T \mathbf{g} \tag{2.6}$$

where $\boldsymbol{\lambda}_f(t) \in \mathbb{R}^{n_x}$ is the so called costate or adjoint variable associated with the differential constraints, $\boldsymbol{\lambda}_g(t) \in \mathbb{R}^{n_g}$ is the Lagrange multiplier associated with the path

constraints. In a single phase optimal control problem without static parameters, the first-order necessary conditions of optimality are given in the following descriptions.

The Euler-Lagrange equations are denoted as:

$$\dot{\boldsymbol{x}} = \Big[\frac{\partial \mathcal{H}}{\partial \boldsymbol{\lambda}_f}\Big]^T, \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{2.7}$$

$$\dot{\boldsymbol{\lambda}}_f = -\Big[\frac{\partial \mathcal{H}}{\partial \boldsymbol{x}}\Big]^T,\ \boldsymbol{\lambda}_f(t_f) = \ \mathcal{M}(t_f,\ \boldsymbol{x}(t_f)) \tag{2.8}$$

$$\Big[\frac{\partial \mathcal{H}}{\partial \boldsymbol{u}}\Big]^T = 0 \tag{2.9}$$

where Equation (2.8) is often referred to as the *adjoint equation* (or the *costate equation*).

The boundary condition for optimality at the initial and final point is:

$$\boldsymbol{b}(\boldsymbol{x}(t_0),\ t_0,\ \boldsymbol{x}(t_f),\ t_f) = 0 \tag{2.10}$$

where $\nu \in \mathbb{R}^{n_b}$ is the Lagrange multiplier associated with the boundary condition **b**.

The transversality conditions at the initial and final point are:

$$\boldsymbol{\lambda}(t_0) = -\frac{\partial \mathcal{M}}{\partial \boldsymbol{x}(t_0)} + \boldsymbol{\nu}^T \frac{\partial \boldsymbol{b}}{\partial \boldsymbol{x}(t_0)} \tag{2.11}$$

$$\boldsymbol{\lambda}(t_f) = -\frac{\partial \mathcal{M}}{\partial \boldsymbol{x}(t_f)} + \boldsymbol{\nu}^T \frac{\partial \boldsymbol{b}}{\partial \boldsymbol{x}(t_f)} \tag{2.12}$$

$$\mathcal{H}(t_0) = \frac{\partial \mathcal{M}}{\partial t_o} - \boldsymbol{\nu}^T \frac{\partial \boldsymbol{b}}{\partial t_0} \tag{2.13}$$

$$\mathcal{H}(t_f) = \frac{\partial \mathcal{M}}{\partial t_f} - \boldsymbol{\nu}^T \frac{\partial \boldsymbol{b}}{\partial t_f} \tag{2.14}$$

The *complementary slackness conditions*, which are the conditions on the Lagrange multipliers of the path constraints are given as:

$$\boldsymbol{\lambda_g}(t) \leq 0, when\ \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, t) = 0,\ for\ each\ path\ constraint \tag{2.15}$$

$$\boldsymbol{\lambda_g}(t) = 0, when\ \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, t) > 0,\ for\ each\ path\ constraint \tag{2.16}$$

These formulated necessary conditions form a Hamiltonian boundary-value problem (HBVP), which is often numerically solved for extremal trajectories by iterative procedures. The optimal solution $\boldsymbol{u}^*$ is then obtained by choosing the extremal trajectory that minimizing the cost function within the feasible control set $\mathcal{U}$, according to the maximum (minimum) principle:

$$\boldsymbol{u}^*(\boldsymbol{x}, \boldsymbol{\lambda}_f, t) = \arg\min_{\mathrm{u}\in\mathcal{U}} \mathcal{H} \tag{2.17}$$

Now, we can understand that the words *indirect* means that, an indirect method does not attempts to find the minimum cost function *directly*, but by solving the necessary conditions of optimality firstly. There are three most common approaches, single shooting, multiple shooting, and collocation methods ban be used to solve the HBVP numerically. The following subsections will give a description of each.

2.3.1 Indirect Single Shooting

For simplicity and to set forth the basic procedures of indirect shooting methods, here regard only the simplified optimal control problem without inequality constraints:

$$minimize: \int_{t_0}^{t_f} \mathcal{L}(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt + \mathcal{M}(\boldsymbol{x}(t_f), t_f) \tag{2.18}$$

subject to:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t), \boldsymbol{x}(t_0) = \boldsymbol{x}(0) \ (fixed\ initial\ value) \tag{2.19}$$

$$\boldsymbol{b}(\boldsymbol{x}(t_f), t_f) = 0 \ (terminal\ constraint) \tag{2.20}$$

Suppose that $(\boldsymbol{u}^*, t_f)$ is an optimal solution, there must exist trajectories $(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{\lambda}^*, \boldsymbol{t}_f^*)$ that satisfies the following Euler-Lagrange equations,

$$\dot{\boldsymbol{x}}(t) = \Big[\frac{\partial \mathcal{H}}{\partial \boldsymbol{\lambda}}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{\lambda}^*(t), t)\Big]^T, \boldsymbol{x}^*(t_0) = \boldsymbol{x}_0 \tag{2.21}$$

$$\dot{\boldsymbol{\lambda}}_f^* = -\Big[\frac{\partial \mathcal{H}}{\partial \boldsymbol{x}}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{\lambda}^*(t), t)\Big]^T, \ \boldsymbol{\lambda}_f^*(t_f^*) = \ \mathcal{M}(t_f^*, \ \boldsymbol{x}^*(t_f^*)) \tag{2.22}$$

$$\Big[\frac{\partial \mathcal{H}}{\partial \boldsymbol{u}}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{\lambda}^*(t), t)\Big]^T = 0 \tag{2.23}$$

and the transversal conditions,

$$\mathcal{H}(t_f^*) = \frac{\partial \mathcal{M}}{\partial t_f^*}(t_f^*, \ \boldsymbol{x}^*(t_f^*)) - \boldsymbol{\nu}^{*T}\frac{\partial \boldsymbol{b}}{\partial t_f^*}(t_f^*, \ \boldsymbol{x}^*(t_f^*)) \tag{2.24}$$

$$\boldsymbol{b}(\boldsymbol{x}^*(t_f^*), \ t_f^*) = 0 \tag{2.25}$$

The general idea of an indirect method is to supply an initial guess of $\boldsymbol{\lambda}_*(t_0)$, ν^*, and t_f^* first, and then iteratively update the estimates to meet the transversal conditions:

$$F(\boldsymbol{\lambda}^*(t_0), \boldsymbol{\nu}^*, t_f^*) := \begin{pmatrix} \boldsymbol{\lambda}_f^* + \dfrac{\partial \mathcal{M}}{\partial \boldsymbol{x}} + \boldsymbol{\nu}^{*T}\dfrac{\partial \boldsymbol{b}}{\partial \boldsymbol{x}} \\ \boldsymbol{b} \\ \mathcal{H} + \dfrac{\partial \mathcal{M}}{\partial t} + \boldsymbol{\nu}^{*T}\dfrac{\partial \boldsymbol{b}}{\partial t} \end{pmatrix}_{t=t_f^*} = 0 \tag{2.26}$$

An algorithm based on Newton iteration implemented the indirect single shooting method for the formulated optimal control problem is summarized as Algorithm 1.

The above are optimal control problems without inequality constraints and the solutions of which consist of single arcs, the indirect shooting method proceeds by iteratively updating the estimates of the costate variables at initial time, the Lagrange multipliers and the terminal time. The situation becomes more complicated when either terminal or inequality path constraints are applied. In this case, a tentative sequence of constrained or unconstrained arcs together with the active terminal constraints should be provided at first, then the control, state, adjoint, and multiplier functions which satisfy the Euler-Lagrange equations are calculated. If the errors on the constraints are not satisfied, a new tentative sequence of arcs should be provided for a new iteration

Algorithm 1: Indirect multi-shooting method

function IndirectShooting $(\boldsymbol{\lambda}^0_{f,0}, \boldsymbol{\nu}^0, t^0_{\mathrm{f}})$

Input : initial guess $\boldsymbol{\lambda}^0_{f,0}, \boldsymbol{\nu}^0, t^0_{\mathrm{f}}$ and terminal tolerance ϵ

for *each* $i \in N$ **do**

  Calculate the defects: $F(\boldsymbol{\lambda}^i_{f,0}, \boldsymbol{\nu}^i, t^i_{\mathrm{f}})$

  if $\| F(\boldsymbol{\lambda}^i_{f,0}, \boldsymbol{\nu}^i, t^i_{\mathrm{f}}) \| < \epsilon$ **then**

    return

  end

  Calculate the gradient of the defects: $\begin{cases} \dfrac{\partial F(\boldsymbol{\lambda}^i_{f,0}, \boldsymbol{\nu}^i, t^i_{\mathrm{f}})}{\partial \boldsymbol{\lambda}_{f,0}} \\ \dfrac{\partial F(\boldsymbol{\lambda}^i_{f,0}, \boldsymbol{\nu}^i, t^i_{\mathrm{f}})}{\partial \boldsymbol{\lambda}_{f,0}} \\ \dfrac{\partial F(\boldsymbol{\lambda}^i_{f,0}, \boldsymbol{\nu}^i, t^i_{\mathrm{f}})}{\partial \boldsymbol{\lambda}_{f,0}} \end{cases}$

  Determine the search directions $\boldsymbol{d}^i_{\lambda_f}$, $\boldsymbol{d}^i_{\lambda_\nu}$ *and* $\boldsymbol{d}^i_{t_f}$ by solving the linear system:

$$\begin{pmatrix} \dfrac{\partial F(\boldsymbol{\lambda}^i_{f,0}, \boldsymbol{\nu}^i, t^i_{\mathrm{f}})}{\partial \boldsymbol{\lambda}_{f,0}} \\ \dfrac{\partial F(\boldsymbol{\lambda}^i_{f,0}, \boldsymbol{\nu}^i, t^i_{\mathrm{f}})}{\partial \boldsymbol{\lambda}_{f,0}} \\ \dfrac{\partial F(\boldsymbol{\lambda}^i_{f,0}, \boldsymbol{\nu}^i, t^i_{\mathrm{f}})}{\partial \boldsymbol{\lambda}_{f,0}} \end{pmatrix}^T \begin{pmatrix} \boldsymbol{d}^i_{\lambda_f} \\ \boldsymbol{d}^i_{\lambda_\nu} \\ \boldsymbol{d}^i_{t_f} \end{pmatrix} = -F(\boldsymbol{\lambda}^i_{f,0}, \boldsymbol{\nu}^i, t^i_{\mathrm{f}})$$

  Calculate the new estimates: $\begin{cases} \boldsymbol{\lambda}^{i+1}_{f,0} = \boldsymbol{\lambda}^i_{f,0} + \boldsymbol{d}^i_{\lambda_f} \\ \boldsymbol{\nu}^{i+1} + \boldsymbol{\nu}^i + \boldsymbol{d}^i_{\lambda_\nu} \\ t^{i+1}_{\mathrm{f}} = t^0_{\mathrm{f}} + \boldsymbol{d}^i_{t_f} \end{cases}$

  Update the iteration index: $k = k + 1$.

end

until all the errors are within the provided tolerance. For this approach, it is not difficult to understand that the errors on constraints are generally very sensitive to the initial values, and the convergence speed of the optimization also highly depends on the initial guess. Successful use of the indirect single shooting method requires a priori knowledge of the number of constrained arcs. The single shooting method has the feature of simplicity, however, in some cases, this method will present numerical difficulties when the forward simulation of the combined differential equitations are ill-conditioned. In order to overcome the numerical difficulties, the collocation method can be employed as an alternative.

2.3.2 Indirect Collocation

In an indirect collocation method, the differential equations composed of state and costate variables are discretized in a series of subintervals, the state and costate variables are approximated with piecewise polynomials. The Hamiltonian dynamics are rewritten as defect constraints, which are large scale, but sparse nonlinear equations. The obtained equation system can be solved with Newton's method, in particular, the

numerical accuracy and convergence speed highly depends on the position, the number of discretization nodes and the order collocation schemes. Applications and more discussions of the indirect collocation methods can be found in [40].

2.3.3 Indirect Multiple-Shooting

The multiple-shooting method is a kind of combination of the two approaches described above, the fundamental idea of this method is to subdivide the time interval $[t_0, t_f]$ into $M + 1$ shorter subintervals and employ the single shooting method within each subinterval.

Denoting the state and costate variables with a combined vector $z_i = [\boldsymbol{x}(t), \boldsymbol{\lambda}_f(t)]^T$, the basic procedures are summarized as Algorithm 2.

Algorithm 2: Indirect Multiple-shooting method

function IndirectMultipleShooting

Input : Provide artificial initial guess z_i^0 for each interval and the terminal tolerance ϵ, set $i = 0$

Divide the entire time intervals into $m - 1$ subintervals: $t_0 = t_1 < \cdots < t_f$

for *each* $i \in N$ **do**

  Solve the differential equations on each interval $[t_i, t_{i+1}]$ numerically with the provided initial guess of each interval, and get the integration $\hat{z}_i$;

  Calculate the linkage constraints: $\boldsymbol{F}_j(z_1, ..., z_{m-1}) = \hat{z}_i - z_{i+1}$

  Calculate the transversal conditions of the terminal interval and combine the likage

$$\text{conditions: } \boldsymbol{F} = \begin{bmatrix} \hat{z}_1 - z_2 \\ \vdots \\ \hat{z}_{m-2} - z_{m-1} \\ \boldsymbol{\lambda}_f + \dfrac{\partial \mathcal{M}}{\partial \boldsymbol{x}} + \boldsymbol{\nu}^T \dfrac{\partial \boldsymbol{b}}{\partial \boldsymbol{x}} \\ \boldsymbol{b} \\ \mathcal{H} + \dfrac{\partial \mathcal{M}}{\partial t} + \boldsymbol{\nu}^T \dfrac{\partial \boldsymbol{b}}{\partial t} \end{bmatrix}$$

  if $\| \boldsymbol{F} \| < \epsilon$ **then**

    return

  end

  Calculate the gradient of the defects:

  Determine the search directions Δz by solving the linear system: $\nabla \boldsymbol{F} \Delta z = -\boldsymbol{F}(z)$

  Calculate the new estimates: $z = z + \Delta z$

  Update the iteration index: $k = k + 1$.

end

Despite the increased size of the problem due to those introduced linkage variables and its relying on forward integration, the multiple-shooting method is still an improvement over the single shooting method and collocation method because it inherits the advantages of both. It can rely on the existing forward solvers with inbuilt adaptivity so that it can avoid numerical discretization errors, and it is able to deal better

with unstable and nonlinear systems.

The main drawbacks of the shooting method are that: besides the need of specifying very accurate estimates for the adjoint variables at initial time and entry times, another severe drawback of indirect multiple shooting method is that a detail priori knowledge of the structure of the optimal solution must be available.

2.4 Dynamic Programming

2.4.1 Dynamic Programming in continuous time

The fundamental idea of Dynamic Programming (DP) [41] is to introduce an optimal-cost-to-go function $J^*(x,\ t)$ and compute it recursively backwards, starting from a known value at its end and applying the Principle of Optimality. Considering the typical optimal control problem:

$$J^*(\boldsymbol{x}(t),\ t) = \min_{\boldsymbol{u},\boldsymbol{x}}\{\mathcal{M}(\boldsymbol{x}(t_f),\ t_f) + \int_t^{t_f} \mathcal{L}(\boldsymbol{x}(t),\ \boldsymbol{u}(t),\ t)dt\} \tag{2.27}$$

the optimal-cost-to-go function can be rewritten as:

$$\begin{aligned} J^*(\boldsymbol{x}(t),\ t) &= \min_{\boldsymbol{u}}\left\{\mathcal{M}(\boldsymbol{x}(t_f),\ t_f) + \int_t^{t+dt} \mathcal{L}(\boldsymbol{x}(t),\ \boldsymbol{u}(t),\ t)dt + \int_{t+dt}^{t_f} \mathcal{L}(\boldsymbol{x}(t),\ \boldsymbol{u}(t),\ t)dt\right\} \\ &= \min_{\boldsymbol{u}}\left\{\int_t^{t+dt} \mathcal{L}(\boldsymbol{x}(t),\ \boldsymbol{u}(t),\ t)dt + J^*(\boldsymbol{x}(t+dt),\ t+dt)\right\} \end{aligned} \tag{2.28}$$

taking the limit of the above equation, we can get:

$$\begin{aligned} J^*(\boldsymbol{x}(t),\ t) &= \lim_{dt\to 0}\min_{\boldsymbol{u}}\{\mathcal{L}(\boldsymbol{x}(t),\ \boldsymbol{u}(t),\ t)dt + J^*(\boldsymbol{x}(t+dt),\ t+dt)\} \\ &\approx \lim_{dt\to 0}\min_{\boldsymbol{u}}\left\{\mathcal{L}(\boldsymbol{x}(t),\ \boldsymbol{u}(t),\ t)dt + J^*(\boldsymbol{x}(t),\ t) + \frac{\partial J^*}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}}dt + \frac{\partial J^*}{\partial t}dt\right\} \end{aligned} \tag{2.29}$$

Simplify the above equation, we can obtain the so called *Hamilton-Jacobi-Bellman* (HJB) equation:

$$\min_{\boldsymbol{u}}\left\{\mathcal{L}(\boldsymbol{x}(t),\ \boldsymbol{u}(t),\ t) + J^*(\boldsymbol{x}(t),\ t) + \frac{\partial J^*}{\partial \boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x}(t),\ \boldsymbol{u}(t),\ t) + \frac{\partial J^*}{\partial t}\right\} = 0 \tag{2.30}$$

Solve this partial differential equation backwards from the end of the time interval on $t \in [0, t_f]$ with $J^*(\boldsymbol{x}(t_f),\ t_f) = \mathcal{M}(\boldsymbol{x}(t_f),\ t_f)$, the optimal control $\boldsymbol{u}(t)$ can be obtained from the Equation (2.31) subject to the relevant path constraints:

$$\boldsymbol{u}^*(\boldsymbol{x}(t), t) = \arg\min_{\boldsymbol{u}}\left\{\mathcal{L}(\boldsymbol{x}(t),\ \boldsymbol{u}(t),\ t) + \frac{\partial J^*}{\partial \boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x}(t),\ \boldsymbol{u}(t),\ t)\right\} \tag{2.31}$$

2.4.2 Dynamic Programming in discrete time

A typical optimal control problem in discrete time is denoted as:

$$J^*(\boldsymbol{x}_0) = \mathcal{M}(\boldsymbol{x}(N)) + \sum_{i=0}^{N-1} \mathcal{L}(\boldsymbol{x}(i),\ \boldsymbol{u}(i),\ i) \tag{2.32}$$

subject to:

$$\boldsymbol{x}(i+1) = \boldsymbol{f}(\boldsymbol{x}(n), \boldsymbol{u}(n), n) \tag{2.33}$$

The cost-to-go function in discrete form can be defined as:

$$J^*(\boldsymbol{x}_k, k) = \mathcal{M}(\boldsymbol{x}(N)) + \sum_{i=k}^{N-1} \mathcal{L}(\boldsymbol{x}(i),\ \boldsymbol{u}(i),\ i) \tag{2.34}$$

The optimal-cost-to-go function can be written recursively backwards from the terminal time N towards time 0:

$$\begin{cases} J^*(\boldsymbol{x}, N) & = \mathcal{M}(\boldsymbol{x}) \\ J^*(\boldsymbol{x}, N-1) & = \min\limits_{\boldsymbol{u}}\{\mathcal{L}(\boldsymbol{x}(N-1),\ \boldsymbol{u}(N-1),\ N-1) + \mathcal{M}(\boldsymbol{x}(N))\} \\ & \vdots \\ J^*(\boldsymbol{x}, i) & = \min\limits_{\boldsymbol{u}}\{\mathcal{L}(\boldsymbol{x}(i),\ \boldsymbol{u}(i),\ i) + J^*(\boldsymbol{x}(i+1), i+1)\} \end{cases} \tag{2.35}$$

The optimal control trajectory can be obtained by solving:

$$\boldsymbol{u}^*(\boldsymbol{x}, i) = \arg\min_{\boldsymbol{u}} \mathcal{L}(\boldsymbol{x}(i),\ \boldsymbol{u}(i),\ i) + J^*(\boldsymbol{x}(i+1), i+1) \tag{2.36}$$

Dynamic programming (DP) is greatly advantageous because the HJB presents both the necessary and sufficient conditions, the DP method is able to obtain the global optimum even for a non-convex problem. For the linear quadratic regulator, the HJB equations can be solved analytically or numerically by solving the Ricatti equation. For general nonlinear optimal control problems, the optimal control solutions can be obtained by numerically approximating the cost function, and solving the first-order partial differential equations. Despite the powerful advantages, the application of dynamic programming is only limited for small dimension problems because it need huge amount of storage space even for a problem with quite few variables. Computational complexity of the dynamic programming algorithm increases exponentially with dimensionality of the state (the so called *curse of dimensionality*), which makes it impractical in large-scale applications.

2.5 Direct methods

The direct methods have been studied extensively over the last 40 years, and have been proved as successful tools for solving even complicated practical optimal control problems. In a direct method, the original continuous optimal control problem is transcribed into a finite dimensional nonlinear programming problem (NLP) which can be then solved by some state-of-the-art NLP methods. The fundamental difference with respect to the indirect method is that it dose not need to derive and solve the necessary conditions but to evaluate the cost function directly. One of the important advantages of the direct methods is that the inequality path constraints can be easily treated. Another advantage is that they can be easily applied in problems described by DAE systems. On the other hand, an obvious drawback of direct methods is that they provide only local optimal solutions due to the discretization of the control problems and the NLP solvers.

The direct methods can be categorized into three groups which are the single shooting, multiple shooting, and collocation methods. The following sections will give a more detail description of them respectively.

2.5.1 Direct Single Shooting

In direct single shooting methods [40], the control variables $\boldsymbol{u}$ are parameterized by a finite set of parameters. Generally, the control variables are approximated by piecewise polynomials, and different orders of polynomials may be used for different control variables and/or for different control intervals, i.e., by linear, quadratic, B-spline or Lagrange polynomials. The control trajectories of different degrees and continuity orders are demonstrated in Figure 2.3.

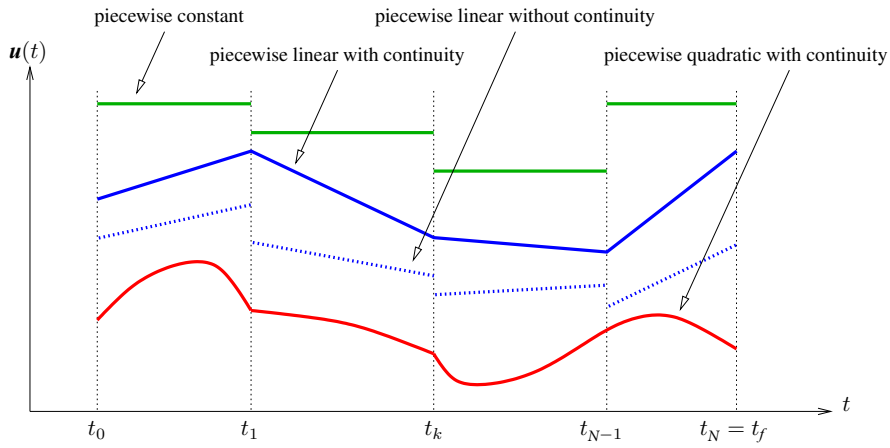

Figure 2.3: *Examples of control trajectories* [42]

The procedures of a direct single shooting method is summarized as Algorithm 3.

Algorithm 3: Direct Single-shooting method

function DirectSingleShooting

Input : Provide artificial initial guess of the parameters in the control trajectory parameterization

Divide the entire time intervals into N subintervals: $0 = t_0 = t_1 < \cdots < t_{N_n} = t_f$

Approximate the control in each time interval with finite parameters $\boldsymbol{q} = [\boldsymbol{q}_0, ..., \boldsymbol{q}_{N-1}]$ of the selected appropriate polynomials.

while *The cost function is not optimal or any constraints are not satisfied* **do**

  Use numerical integral methods to get the state variables as a function of the finite control paramters obtained in last step.

  Evaluate the obtained NLP: $\min_{\boldsymbol{q}} \left\{ \mathcal{M}(\boldsymbol{x}(t_f, \boldsymbol{q})) + \int_0^{t_f} \mathcal{L}(\boldsymbol{x}(t, \boldsymbol{q}),\ \boldsymbol{u}(t, \boldsymbol{q}))dt \right\}$

  Evaluate the discretized path constraints and terminal constraints.

  Update the approximation of $\boldsymbol{q}$ with NLP slovers

end

There are several advantages of the direct single shooting method. First, only very few degrees of freedom are used even for large system described by ordinary differential equation (ODE) or differential-algebraic equations (DAE). Second, it is easy to treat the active set changes. Third, it is possible to use the state-of-the-art ODE/DAE solvers; Besides, the only variables of the NLP are the control parameters $\boldsymbol{q}$. However, sometimes it is difficult to find an optimal, or even a feasible solution when the system is unstable or the response is undefined for some control parameters. Moreover, the states may depend nonlinearly on the control parameters, without the possibility of using knowledge of the state variables in initialization.

2.5.2 Direct Multiple Shooting

In a direct multiple shooting method which is originally developed in [43], the original optimal control problem is transcribed into a finite dimensional NLP problem by discretizing both the control and the state variables, which is also referred to *full discretization* in literature. The typical procedures of the direct multiple shooting method is summarized as Algorithm 4.

Algorithm 4: Direct Multiple-shooting method

function DirectMultipleShooting

Input : Provide artificial initial guess of the parameters in both the control and state trajectory parameterization

Divide the entire time intervals into N subintervals: $0 = t_0 = t_1 < \cdots < t_{N_n} = t_f$

Approximate the control variables in each time interval with finite parameters $\boldsymbol{q} = [\boldsymbol{q}_0, ..., \boldsymbol{q}_{N-1}]$ of the selected appropriate polynomials.

Approximate the state variables in each time interval with finite parameters $\boldsymbol{s} = [\boldsymbol{s}_0, ..., \boldsymbol{s}_{N-1}]$ of the selected appropriate polynomials.

while *The cost function is not optimal or any constraints are not satisfied* **do**

  Use numerical integral methods to get the state variables as a function of the finite control and state paramters obtained in last step.

  Evaluate the obtained NLP: $\min_{\boldsymbol{q},\boldsymbol{s}} \left\{ \mathcal{M}(\boldsymbol{x}(t_f, \boldsymbol{q}, \boldsymbol{s})) + \int_0^{t_f} \mathcal{L}(\boldsymbol{x}(t, \boldsymbol{q}, \boldsymbol{s}),\ \boldsymbol{u}(t, \boldsymbol{q}, \boldsymbol{s}))dt \right\}$

  Evaluate the discretized path constraints and terminal constraints.

  Evaluate the discretized linkage constraints of the state variabls.

  Update the approximation of $\boldsymbol{q}, \boldsymbol{s}$ with NLP slovers

end

As we can see that the direct multiple shooting method can use the knowledge of state variables in initialization since both state and control variables are parameterized. The direct multiple shooting method is able to handle both the path and terminal constraints more robustly with respect to the direct single shooting method, it is capable to find the optimal solution even for unstable systems. However, for the multiple shooting method, the size of the optimization problem is increased due to

the approximation of the state variables and the introducing of continuity linkage constraints of the state variables at each interval. The obtained NLP is not so sparse as direct collocation methods. Besides, the convergence performance also depends on the initial guess of the state parameters [40].

2.5.3 Direct Collocation

Direct collocation methods are another family of direct methods for the numerical solutions of optimal control problems, which approximate the state and control variables with specified functional forms. Direct methods of optimal control are known as discretize-then-optimize methods [44]. All of the direct collocation methods involve the discretization of the ODEs of the original problem, the dynamic differential constraints are transcribed into a set of algebraic equality defects constraints after discretization. The integrals associated with the problem can be computed with some well known quadrature formulas. Considering the general single phase optimal control problem formulated in Equations (2.1)-(2.3), the typical procedures of the direct collocation methods of it are given in Algorithm 5.

Algorithm 5: Direct Collocation method

1 function DirectCollocation

2 Divide the entire time intervals into N subintervals: $0 = t_0 = t_1 < \cdots < t_{N_n} = t_f$

Input : Construct and provide the initial value (optionally) of the NLP variables with the state, control, static parameters and the terminal time at each node:$\boldsymbol{y} = [\boldsymbol{x}_0, \boldsymbol{u}_0, \boldsymbol{x}_1, \boldsymbol{u}_1, ..., , \boldsymbol{x}_{N_n}, \boldsymbol{x}_{N_n}, t_f, \boldsymbol{p}]$

3 **while** *The cost function is not optimal or any constraints are not satisfied* **do**

4 Evaluate the cost function: $J = \mathcal{M}(\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f, \mathbf{p}) + \int_{t_0}^{t_f} \mathcal{L}(\mathbf{x}(t), \mathbf{u}(t), t, \mathbf{p})dt$

5 Evaluate the discretized constraints: $\boldsymbol{c}(\boldsymbol{y}) = [\zeta_1, ...\zeta_{N_n-1}, \boldsymbol{g}_1, ..., \boldsymbol{g}_{N_n-1}, \boldsymbol{b}_0, \boldsymbol{b}_f]$.

6 Calculate the gradient of the cost function: ∇J

7 Calculate the jacobian matrix of the constraints function: $\nabla \boldsymbol{c}$

8 Update the NLP variables $\boldsymbol{y}$

9 **end**

In particular, there are more options in the direct collocation methods for discretization, which can be categorized into local collocation methods and global collocation methods. The followed subsections will give some details of both.

2.5.3.1 Local Collocation

The mostly implemented local collocation methods are the Runge-Kutta (RK) methods which are a popular family of one-step methods, with the given value of a vector $\boldsymbol{x}_i$ at time t_i, a new estimates of $\boldsymbol{x}_{i+1}$ at time t_{i+1} can be obtained as Equation (2.37) within

the RK schemes.

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + h_i \sum_{j=1}^{K} \omega_j \boldsymbol{f}_{ij} \tag{2.37}$$

where

$$\boldsymbol{f}_{ij} = \boldsymbol{f}(\boldsymbol{x}_i + h_i \sum_{k=1}^{K} \alpha_{jk} \boldsymbol{f}_{ik},\ t_i + h_i \tau_i),\ 1 \leq j \leq K$$

with $0 \leq \tau_1 \leq ... \leq \tau_K \leq 1$, and $K \geq 1$ is referred to the number of stages, RK schemes differ in the choice of the parameters ω_j, τ_i, and α_{jk}, which are most conveniently denoted with the so called Butcher array:

$$\begin{array}{c|ccc} \tau_1 & \alpha_{11} & \cdots & \alpha_{1K} \\ \vdots & \vdots & \ddots & \vdots \\ \tau_K & \alpha_{K1} & \cdots & \alpha_{KK} \\ \hline & \omega_1 & \cdots & \omega_K \end{array}$$

The schemes are referred to *explicit* if $\alpha_{j\ell} = 0$ for $1 \geq j$ and *implicit* otherwise. Different method can be obtained by wise choosing of these coefficients in the Butcher array, the four common types of K stage Runge-Kutta schemes are represented in following paragraphs [38].

1) The explicit Euler method

The explicit Euler method is an one-stage RK scheme, the Butcher array of which is

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$$

its common expression is:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + h_i \boldsymbol{f}_i \tag{2.38}$$

2) The implicit Trapezoidal method

The implicit Trapezoidal method is a two-stage RK scheme, the Butcher array of which is

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 0 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array}$$

its common expression is:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \frac{h_i}{2}(\boldsymbol{f}_i + \boldsymbol{f}_{i+1}) \tag{2.39}$$

3) The implicit Hermite-Simpson method

The implicit Hermite-Simpson method is a three-stage RK scheme, the Butcher array of which is

0	0	0	0
1/2	5/24	1/3	-1/24
1	1/6	2/3	1/6
	1/6	2/3	1/6

its common expression is:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \frac{h_i}{6}(\boldsymbol{f}_i + 4\bar{\boldsymbol{f}} + \boldsymbol{f}_{i+1}) \tag{2.40}$$

where:

$$\begin{cases} \bar{\boldsymbol{x}} &= \frac{1}{2}(\boldsymbol{x}_i + \boldsymbol{x}_{i+1}) + \frac{h_i}{8}(\boldsymbol{f}_i - \boldsymbol{f}_{i+1}) \\ \bar{\boldsymbol{f}} &= \boldsymbol{f}(\bar{\boldsymbol{x}},\ t_i + \frac{h_i}{2}) \end{cases} \tag{2.41}$$

4) The explicit Runge-Kutta method

The explicit Runge-Kutta method is a four-stage RK scheme, the Butcher array of which is

0	0	0	0	0
1/2	1/2	0	0	0
1/2	0	1/2	0	0
1	0	0	1	0
	1/6	1/3	1/3	1/6

its common expression is:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \frac{h_i}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4) \tag{2.42}$$

where:

$$\begin{cases} \boldsymbol{k}_1 &= h_i\boldsymbol{f}_i \\ \boldsymbol{k}_2 &= h_i\boldsymbol{f}(\boldsymbol{x}_i + \frac{1}{2}\boldsymbol{k}_1, t_i + \frac{h_i}{2}) \\ \boldsymbol{k}_3 &= h_i\boldsymbol{f}(\boldsymbol{x}_i + \frac{1}{2}\boldsymbol{k}_2, t_i + \frac{h_i}{2}) \\ \boldsymbol{k}_4 &= h_i\boldsymbol{f}(\boldsymbol{x}_i + \boldsymbol{k}_3, t_i + h_i) \end{cases} \tag{2.43}$$

The widely used local Runge-Kutta methods are the trapezoidal method and the Hermite-Simpson method, and have been successfully implemented to solve many complicated problems [38, 45, 46].

2.5.3.2 Global Collocation

The global collocation methods mainly refer to the pseudospectral methods, which are a particular type of direct collocation methods using orthogonal collocation points. Pseudospectral methods were originally developed to solve the partial differential equations in the 1970s and have risen to prominence as important numerical methods

for solving optimal control problems over the last 20 years [47–52]. Compared with local collocation methods, the pseudospectral methods use information over samples of the entire spatial domain of the function to approximate its derivatives at the selected orthogonal collocation points. The most appealing feature of the pseudospectral method is that the approximation can converge at an exponential (or spectral) rate which is a function of the number of the collocation nodes. Both the theory and practice of approximation prove that the pseudospectral methods are very suitable to approximate the smooth functions, differentiations and integrations related in the optimal control problems. In pseudospectral methods, the state and control variables are approximated with smooth Lagrange polynomials, and the differential equations at the discretization nodes are easily obtained by multiplying a constant differentiation matrix with the right-hand dynamic equations, while the integration of the cost function is approximated by the Gauss quadrature method with spectral accuracy, which consists of a weighted (quadrature weights) sum of the function values at the all the discretization nodes. In cases of non-smooth functions, where global collocation is not suitable, the multi-interval pseudospectral techniques have been proposed to perform the global collocation within each subinterval [53].

There are three types of most commonly used collocation points which are the Legendre-Gauss (LG), the Legendre-Gauss-Radau (LGR), and the Legendre-Gauss-Lobatto (LGL) points defined on the domain $[-1,\ 1]$. The main differences among the three sets of collocation points are their locations and the including of endpoints as shown in Figure 2.4.

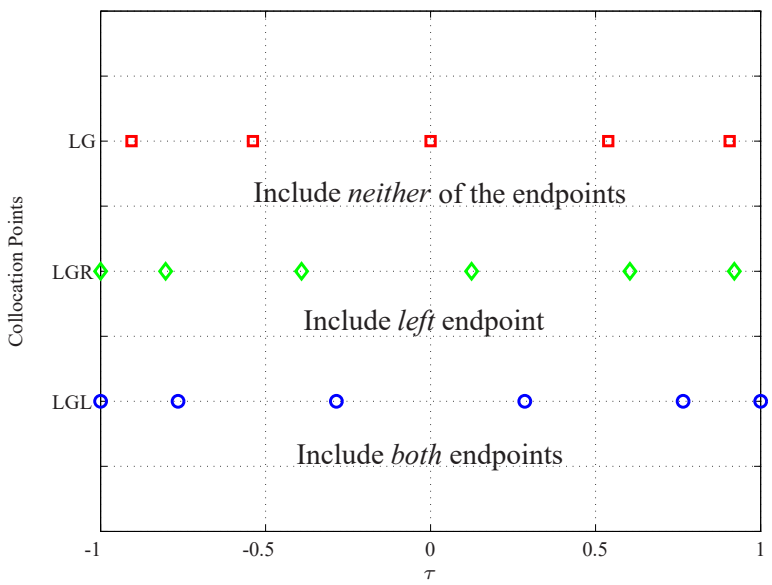

Figure 2.4: *Differences Between LGL, LGR, and LG Collocation Points*

The definition of the three types of collocation points (also called as nodes) are related with a particular class of orthogonal polynomials which are the Legendre polynomials, a N oder Legendre polynomial $L_N(\tau)$ can be denoted as:

$$L_N(\tau) = \frac{1}{2^N N!}\frac{d^N}{d\tau^N}(\tau^2 - 1)^N \tag{2.44}$$

The locations, weights of quadrature and the differential matrices of the LG nodes are given as followings.

• τ_k are defined as the zeros of $L_{N+1}(\tau),\ k = 0, ..., N$

• the weights of quadrature:

$$w_k = \frac{2}{(1-\tau_k^2)[\dot{L}_{N+1}(\tau_k)]^2},\ k = 0, ..., N \tag{2.45}$$

• the differential matrix:

$$D_{ki} = \begin{cases} \dfrac{\dot{L}_{N+1}(\tau_k)}{\dot{L}_{N+1}(\tau_i)}\dfrac{1}{\tau_k - \tau_i}, & if\ k \neq i \\ \dfrac{\tau_k}{1-\tau_k^2}, & if\ k = i \end{cases} \tag{2.46}$$

The location, weights of quadrature and the differential matrix of the LGR nodes are given as followings.

• τ_k are defined as the zeros of $L_N(\tau) + L_{N+1}(\tau),\ k = 0, ..., N$

• the weights of quadrature:

$$w_k = \begin{cases} \dfrac{2}{(N+1)^2}, & if\ k = 0 \\ \dfrac{1}{(N+1)^2}\dfrac{1-\tau_k}{[L_{N+1}(\tau_k)]^2}, & if\ k = 1, ...N \end{cases} \tag{2.47}$$

• the differential matrix:

$$D_{ki} = \begin{cases} -\dfrac{N(N+2)}{4}, & if\ k = i = 0 \\ \dfrac{\dot{Q}(\tau_k)}{\dot{Q}(\tau_i)}\dfrac{1}{\tau_k - \tau_i}, & if\ k \neq i \\ \dfrac{\tau_k}{1-\tau_k^2} + \dfrac{(N+1)L_N(\tau_k)}{(1-\tau_k^2)\dot{Q}(\tau_k)}, & if\ 1 \leq k = i \leq N \\ 0, & otherwise \end{cases} \tag{2.48}$$

where $Q(\tau) = L_N(\tau) + L_{N+1}(\tau)$.

The location, weights of quadrature and the differential matrix of the LGL nodes are given as followings.

• the location:

$$\tau_k = \begin{cases} -1, & if\ k = 0 \\ zeros\ of \dot{L}_N(\tau), & k = 1, ..., N-1 \\ 1, & k = N \end{cases} \tag{2.49}$$

• the weights of quadrature:

$$w_k = \frac{2}{N(N+1)}\frac{1}{[L_N(\tau)_k]^2},\ k = 0, ..., N \tag{2.50}$$

• the differential matrix:

$$D_{ki} = \begin{cases} -\dfrac{N(N+1)}{4}, & if\ k = i = 0 \\ \dfrac{L_N(\tau_k)}{L_N(\tau_i)} \dfrac{1}{\tau_k - \tau_i}, & if\ k \neq i \\ \dfrac{N(N+1)}{4}, & if\ k = i = N \\ 0, & otherwise \end{cases} \tag{2.51}$$

The above outlined only some conclusions of the three methods, more details of approximating continuous functions using Legendre polynomials are not given. Interested readers can refer to [54] for further detail derivation of the nodes, weights of quadrature and differential matrix.

With the obtained collocation nodes, weights of quadrature and differential matrix, we are ready to present the framework of transcription based on the pseudospectral methods, whose general procedures are given as:

1) Compute the collocation nodes τ.

2) Transformation of the original time domain $[t_0,\ t_f]$ with a new independent variable τ to $[-1,\ 1]$:

$$\tau = \frac{2}{t_N - t_0} t - \frac{t_N + t_0}{t_N - t_0},\ with\ t_N \equiv t_f \tag{2.52}$$

3) Approximate the state variables the Lagrange basis polynomials ϕ_i:

$$x(\tau) \approx X(\tau) = \sum_{i=0}^{N} \phi_i(\tau) X_i \tag{2.53}$$

with

$$\phi_i(\tau) = \prod_{j=0, j\neq i}^{N} \frac{\tau - \tau_j}{\tau_i - \tau_j} \tag{2.54}$$

4) Approximate the control variables:

$$u(\tau) \approx U(\tau) = \sum_{i=0}^{N} \phi_k(\tau) U_k \tag{2.55}$$

5) Approximate state differentiations:

$$\dot{x}(\tau_k) \approx \dot{X}_k = \sum_{i=0}^{N} \dot{\phi}_i(\tau_k) X_i = \sum_{i=0}^{N} D_{ki} X_i \tag{2.56}$$

6) Approximate the system dynamics:

$$\zeta = \sum_{i=0}^{N} D_{ki} X_i - \frac{t_f - t_0}{2} f(X_k, U_k, \tau_k) = 0 \tag{2.57}$$

7) Approximate the cost function with the Gauss-Lobatto quadrature:

$$J = \mathcal{M} + \frac{t_f - t_0}{2} \sum_{i=0}^{N} w_i \mathcal{L}(X_i, U_i, \tau_i) \tag{2.58}$$

After transcription, the formulated problem can be solved with Algorithm 5. The most important advantage of using pseudospectral discretization method is the spectral accuracy in the discretization of the differential constraints and Gauss type integration. However, the resulting Jacobian and Hessian matrices are denser than those obtained by local collocation methods.

2.6 NLP

From the above section we can understand that the nonlinear programming algorithms are the basis of many optimal control problems, thus, it is necessary to give a brief review of the NLP methods. Considering a NLP aiming to find a n dimensional decision vector $\boldsymbol{y}$ to minimize the given objective function:

$$\min\ f(\boldsymbol{y}) \tag{2.59}$$

subject to the constraints:

$$\begin{cases} \boldsymbol{g}(\boldsymbol{y}) & = 0, \quad \boldsymbol{g}(\boldsymbol{y}) \in \mathbb{R}^{n_g} \\ \boldsymbol{h}(\boldsymbol{y}) & \leq 0, \quad \boldsymbol{h}(\boldsymbol{y}) \in \mathbb{R}^{n_h} \end{cases} \tag{2.60}$$

2.6.1 Karush-Kuhn-Tucker (KKT) conditions

The Karush-Kuhn-Tucker (KKT) conditions [55] which are the necessary and sufficient conditions for the optimality of a solution to a constrained nonlinear program are given as follows:

- First-order optimality conditions

$$\begin{cases} \nabla f(\boldsymbol{y}) + \nabla \boldsymbol{g}(\boldsymbol{y})^T \lambda + \nabla h(\boldsymbol{y})^T \mu & = 0 \\ \mu_i h_i(\boldsymbol{y}) & = 0,\ i = 1, ..., n_h \\ \mu & \geq 0 \\ \boldsymbol{h}(\boldsymbol{y}) & \leq 0 \\ \boldsymbol{g}(\boldsymbol{y}) & = 0 \end{cases} \tag{2.61}$$

where $\nabla \boldsymbol{f}$ is a column vector, $\nabla \boldsymbol{g}$ collect the gradient vectors of all output components in the Jacobian matrix, $\nabla \boldsymbol{g} = [\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{y}}]^T$.

- Second Order Necessary Condition

$$w^T \left(\nabla^2 \boldsymbol{f}(\boldsymbol{y}) + \nabla^2 \boldsymbol{g}(\boldsymbol{y})^T \lambda + \nabla^2 h(\boldsymbol{y})^T \mu \right) w \geq 0 \tag{2.62}$$

with $\nabla \boldsymbol{g}_i(\boldsymbol{y})^T w$ and $\nabla h(\boldsymbol{y})^T w$.

- Second Order Sufficient Condition

$$w^T \left(\nabla^2 \boldsymbol{f}(\boldsymbol{y}) + \nabla^2 \boldsymbol{g}(\boldsymbol{y})^T \lambda + \nabla^2 h(\boldsymbol{y})^T \mu \right) w > 0 \tag{2.63}$$

with :

$$\begin{cases} \nabla \boldsymbol{g}_i(\boldsymbol{y})^T w & = 0 \quad \textit{with } \lambda_i > 0 \\ \nabla \boldsymbol{g}_i(\boldsymbol{y})^T w & \leq 0 \quad \textit{with } \lambda_i = 0 \\ \nabla \boldsymbol{h}(\boldsymbol{y})^T w & = 0 \end{cases} \tag{2.64}$$

2.6.2 Newton type methods for constrained NLP

The fundamental idea of the Newton type methods is to solve the nonlinear KKT conditions with non-smoothness in different ways. There are two big classes of commonly used algorithms for solving the NLP problems with inequality constraints, which are known as the interior point (IP) methods and sequential quadratic programming (SQP) methods. The main difference between the two kinds of methods is that they deal with the non-smoothness of the KKT conditions in different manners.

2.6.2.1 Interior point (IP) method

1) Interior point (IP) method

In an IP method, a positive penalty factor ρ is introduced to penalize the solutions that approaching the boundaries of the feasible region and to make sure the solution remains within the feasible region. The value of ρ varies with the distance of constraint function to the boundaries. For an interior point method, the iterations usually start with a large value of ρ and the Newton's method is often used to solve the resulting nonlinear equation system, the penalty factor ρ is then iteratively decreased and estimated for each iteration. The previously obtained solution will be the initial one for the next iteration.

The logarithmic barrier function based on the logarithmic interior function is given as:

$$B(\boldsymbol{y},\rho) = f(\boldsymbol{y}) - \rho \sum_{i=1}^{n_h} \log(h_i(y_i)) \tag{2.65}$$

To solve the constrained NLP, a positive penalty factor μ is introduced in addition to the typical KKT conditions. By adding convergence properties with relaxation factor s the modified KKT conditions are given as:

$$\begin{cases} \nabla f(\boldsymbol{y}) + \nabla \boldsymbol{g}(\boldsymbol{y})^T \lambda + \nabla h(\boldsymbol{y})^T \mu &= 0 \\ g(\boldsymbol{y}) &= 0 \\ h(\boldsymbol{y}) + s &= 0 \\ \mu s - \rho &= 0 \\ s &\geq 0 \\ \mu &\geq 0 \\ \rho &\geq 0 \end{cases} \tag{2.66}$$

The obtained nonlinear equations can be iteratively solved by Newton's methods [56]. First, the variables Δy and $\Delta\lambda$ can be obtained by solving the reduced linear system,

$$\begin{bmatrix} H & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{y} \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} \tag{2.67}$$

where

$$H = \nabla_x^2 \mathcal{L} + \nabla h(\boldsymbol{y})^T \begin{bmatrix} \mu \\ s \end{bmatrix} \nabla h(\boldsymbol{y}) \tag{2.68}$$

$$G = \nabla g(\boldsymbol{y}) \tag{2.69}$$

$$T_1 = -\nabla_x \mathcal{L} - \nabla h(\boldsymbol{y})^T s^{-1}(\rho + \mu \nabla h(\boldsymbol{y})) \tag{2.70}$$

$$T_2 = -g(\boldsymbol{y}) \tag{2.71}$$

$$\mathcal{L}(\boldsymbol{y}, \lambda, \mu) = f(\boldsymbol{y}) + \lambda^T g(\boldsymbol{y}) + \mu^T h(\boldsymbol{y}) \tag{2.72}$$

then, the slack variables s and corresponding multipliers μ can be computed with:

$$\begin{cases} \Delta s &= -g(\boldsymbol{y}) - s - \nabla g(\boldsymbol{y}) \Delta \boldsymbol{y} \\ \Delta \mu &= -\mu + s^{-1}\rho - \mu \Delta s \end{cases} \tag{2.73}$$

The penalty factor ρ must converge to zero during the iterations, since the the optimal point should satisfy the original KKT conditions. The penalty factor ρ can be obtained with the primal-dual distance which can be expressed as a function of the relaxation factor s,

$$\rho = \sigma d_p = \sigma \frac{\mu^t s}{n_{iq}} \tag{2.74}$$

where σ is the so called parameter of the direction combination which defines the trajectory of the optimal solution, d_p is the primal-dual average distance, and n_{iq} accounts for the inequality constraints.

The suggested range of σ is between 0 and 1, which represents a linear combination of the affine scaling and centralization directions [57]. For the extreme conditions:

- $\sigma = 0$ corresponds to the affine-scaling direction where the optimal point is obtained through solution of original KKT conditions without the penalty factor.
- $\sigma = 1$ corresponds to centralization direction where the non-optimal solution is found with a primal-dual distance equal to the initial value of ρ

In a conventional primal-dual IP approach, σ is assumed to be a constant value (generally close to 0.1) during the iterations. This results in a search direction where 90% is defined towards the optimal point and 10% towards the trajectory of centralization [58].

The widespread open source software IPOPT is well known for its successful implementation of the IP method [34]. IPOPT is able to achieve the global solution for convex problems, for non-convex problems it can provide a local solution but the convergence to KKT conditions can still be proven and it performs very well in practice.

2.6.2.2 Sequential quadratic programming method

Sequential quadratic programming (SQP) is another iterative method for nonlinear optimization. SQP methods are suitable for problems with twice continuously differentiable objective function and the constraints. It solves an inequality constrained quadratic programming problem obtained by linearizing the objective function and the constraints in each iteration:

$$\begin{aligned} \min_{d} \quad & \nabla f(\boldsymbol{y})^T \boldsymbol{d} + \tfrac{1}{2} \boldsymbol{d}^T \mathcal{H} \boldsymbol{d} \\ \text{s.t.} \quad & g(\boldsymbol{y}) + \nabla g(\boldsymbol{y})^T \boldsymbol{d} \geq 0 \\ & h(\boldsymbol{y}) + \nabla h(\boldsymbol{y})^T \boldsymbol{d} = 0. \end{aligned} \tag{2.75}$$

where $\boldsymbol{d}$ is the search direction, $\mathcal{H}_k$ is the Hessian matrix.

For NLP with equality constraints, the Hessian $\mathcal{H}$ can be obtained in several different manners. First, the exact Hessian can be calculated with

$$\mathcal{H} = \nabla_x^2 \mathcal{L}(\boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \nabla_x^2, \tag{2.76}$$

in this case, the second order sufficient conditions in Equation (2.63) are satisfied and a local quadratic convergence can be obtained when the solutions are approaching the optimal one. Second, for the optimization problems with the objective function form $f(\boldsymbol{y}) = \| r(\boldsymbol{y}) \|^2$, the Hessian is approximated by $H = 2\nabla r(\boldsymbol{y})^T \nabla r(\boldsymbol{y})$ in a Gauss-Newton method, The advantage over the Newton method is that the second-order derivatives $\nabla^2 r(\boldsymbol{y})$ are not calculated. However, if any residual component $r(\boldsymbol{y})$ and/or the corresponding curvature $\nabla^2 r(\boldsymbol{y})$ close to the solution is large, the Gauss-Newton method will converge slower than the Newton method may or not even be locally convergent. The third method is the widely used Quasi-Newton method with Broyden-Fletcher-Goldfarb-Shanno (BFGS) approximation of the Hessian:

$$\mathcal{H}_{k+1} = \mathcal{H}_k - \frac{(\mathcal{H}_k d_k)(\mathcal{H}_k d_k)^T}{d_k^T \mathcal{H}_k d_k} + \frac{z_k z_k^T}{z_k^T d_k} \tag{2.77}$$

where $d_k = y_{k+1} - y_k$, $z_k = \nabla f(y_{k+1}) - \nabla f(y_k)$. BFGS has proven to have good performance even for non-smooth optimizations and it is one of the most popular members of Quasi-Newton methods, however, the BFGS method are not guaranteed to converge unless the function has a quadratic Taylor expansion close to an optimum.

A significant advantage of SQP is that feasible points are not required at any stage of the process. It is more efficient than general NLP approaches and has been successfully applied in a number of research and commercial algorithms [59]. However, SQP can suffer from oscillations when approaching the optimal solution. A widely used software based on SQP is SNOPT [59], which has been successfully applied in solving many large scale NLP, moreover, it is also the implemented in some optimal control softwares, such as GPOPS [60], TOMLAB [61], ASTOS [62].

2.7 Optimal control in vehicles

It is recognized that the development of optimal control theories and the relevant numerical methods has been driven by its widespread applications in aerospace engineering, i.e., low-thrust orbit transfer, launch vehicles, and supersonic aircrafts, since early 1960s. In comparison, its application in vehicle is a little late, according to the searchable literature, the earliest application of optimal control in vehicles stretch back to the work of Ref. [63] in 1989, in which the authors used optimal control theory with indirect methods to find the minimum manoeuvring time for a transient vehicle model on a given path. Although it was not yet a breakthrough for more accurate lap simulation due to its simplification of the vehicle model and tire model, it pointed out the true nature of minimum lap time problem. In 1992, Ref. [64] solved a minimum manoeuvring problem of a simple vehicle model on a hairpin conner with

the sequential conjugate gradient-restoration algorithm proposed in [65, 66]. Later in 1996, Ref. [67] employed a 3-DOF vehicle model with simplified Magic Formula tire model and formulated a minimum manoeuvring problem in a lane-change manoeuvre. The normal loads acted on the tires were modeled in a quasi-static way considering the longitudinal and lateral inertia forces, roll stiffness distribution and longitudinal aerodynamic forces. The formulated optimal control problem was solved by with a gradient algorithm which improves the control actions based on the linearization of the expanded cost function around the current control and state variables. Later continuous research on the vehicular optimal control problems can be reviewed according to the applied approaches.

2.7.1 Vehicular optimal control with indirect methods

A research group in University of Padova has investigated the application of indirect methods continuously since 1999. In [68], the optimal control problem of a motorcycle on a s-shaped path of the circuit Mugello in Italy is formulated, the employed dynamic model involves the longitudinal, lateral, yaw and roll movements. The formulated optimal control problem is solved with an indirect method that converting the original work into TVB problem. This work is really worthy of attention not only because of its early application of the indirect methods but also the introducing of the curvilinear coordinates system, which has been recognized as a very convenient way to describe the position of a motorcycle or a vehicle on a given track. Latter, a series of work related with optimal control problems of motorcycles [69–74] were published and solved with indirect methods. In particular, the optimal solution in work [71] is obtained with an optimal control toolbox developed by a research group from University of Trento based on indirect method [72, 75]. According to the published literature, optimal control of vehicles of University of Padova starts from the work [76], which investigated the minimum time manoeuvre handling at the physical limits of the car. Minimum maneouvring problems on different road surfaces with different drivetrain layouts were formulated and solved with the indirect method [72, 75]. The work [76] is based on a single track model with a simplified tire model and only longitudinal load transfer was taken into account. Another work [77] published in the same year solved the minimum lap time problem of a race series hybrid electric vehicle on the Mugello circuit based on a 3-DOF vehicle model with the same toolbox mentioned above. Specially, this work introduced the friction coefficients together with the normal load to constraint the tire adherence inside their traction ellipse and there was no tire model employed. In 2014, a methodology which combines numerically efficient modelling technique and a 3D curvilinear coordinates technique for the road modelling was presented [32]. The minimum lap time problem of a GT car was formulated and solved with indirect method [72, 75] as a case study, in which, a 3-DOF vehicle model with simplified Magic Formula tire model was implemented. In work [78], they developed three vehicle models: a 14-DOF vehicle model including longitudinal, lateral, vertical, roll, pitch, yaw movements of the vehicle body, and rotational, vertical motions of the four

wheels, however, the lateral and longitudinal motions of the wheels are neglected and the details of suspension model are not presented. A 10-DOF vehicle model neglecting the four vertical motions of the wheels and a 7-DOF vehicle model considering only the longitudinal, lateral, yaw motions of the vehicle body and the four rotation motions of the wheels. The minimum lap time problem of the three different vehicle models on the Adria circuit are formulated and solved with the same optimal control toolbox as they used before [72, 75]. The results they obtained showed that the 10-DOF and 14-DOF vehicle model produced almost the same results, but the former one reduced the computing time by 43%, however, the optimal lap time achieved with the 7-DOF vehicle model highlighted remarkable differences with the those obtained by the 14-DOF and 10-DOF vehicle model, although the 7-DOF can reduced the computing time by 63%. A latest work of this university is [79], in which, the minimum lap time optimal control problem on Pista Azzurra circuit for a developed go-kart model was solved with the same indirect method, the simulation results were compared with those obtained by a real professional driver to validate the model. What's more, the peculiar dynamics of go-karts and focus to tire slippage dynamics were analyzed with the obtained optimal control results.

2.7.2 Vehicular optimal control with direct methods

An early work using direct methods in a standard form in this area can stretch back to the year 2000 [80–83], where direct multiple shooting methods were employed to solve the minimum lap time problem of a 7-DOF vehicle model on a full lap. In particular, the work in [82] studied the influence of the yaw moment of inertia on cornering performance in a double lane change manoeuvre, and the resulting nonlinear programming problem was solved with a SQP algorithm implemented in Matlab Optimisation Toolbox in about 2 hours. In the thesis work [80], entire-lap optimal control of a Formula one race car with the 7-DOF vehicle model was studied, as for the tire forces, the Magic Formula tire model in the version of that time [84] was employed. The computing time on the Barcelona track required between 28 and 60 hours to converge on a Sun Spark workstation, while the simulation time on the Suzuka circuit was ahout 7 minutes for one iteration and between 7 to 10 hours for the full lap. One thing should be noticed that this work implemented a SQP algorithm for large-scale constrained optimization software SNOPT [85]. Subsequently, in work [86], the above framework was applied to investigate the influence of vehicle mass on the performance potential of a Formula One race car. The actual sensitivity on the Suzuka circuit and Barcelcna circuit are obtained and analyzed. In addition, in the above work, the initial guess of the states and controls was generated by a path following algorithm, and the continuous optimal controls problem were discretized with the traveled distance as variables. later related work from the same research group is described in [87], where they used a similar vehicle model but a simplified tire model that handle the combined slip condition by a simple 'friction circle' procedure. The influence of mass, yaw inertia, roll moment distribution, longitudinal center of gravity location on the

manoeuvre time, driven line and stability were investigated on two short manoeuvres. As for the computing time, a full lap of the Jerez circuit with 10 m control spacing and 250 m preview took approximately 8 hours with a Intel T7700 2.4 GHz processor. Most recently, the work of a research group from University of Oxford made important contributions to the research work in this area with direct methods [88–92]. The vehicle model they used is a 7-DOF vehicle model similar with the one in [87], and tire model is the same as [87]. In [88], the minimum lap time optimal control problem of a Formula One race car on the 'Circuit de Catalunya' (Barcelona) track was formulated and solved with a MATLAB-based optimal control transcription toolbox ICLOCS [46] and a well-known NLP solver IPOPT [34], in which, the vehicular optimal control problem was formulated in a more standard format taking into account also several car set-up parameters. Moreover, the computing time was reduced within 1 hour with acceptable accuracy by using a combination of a track description based on curvilinear coordinates, analytical derivatives and a manual scaling method. In [89], kinetic energy recovery systems (ERSs) were investigated and compared with hybrid kinetic and thermal/heat ERSs respectively. In particular, a pseudospectral method based on optimal control transcription toolbox GPOPS-II [93, 94] was implemented to solve the formulated optimal control problem. The obtained results showed that the ERSs can produce contemporary lap time with a two-thirds of the fuel reduced compared with the earlier generation race cars. In [91], the optimal control of a Formula One car on a developed three-dimensional (3D) track described by its geodesic and normal curvatures, and its relative torsion [90] was studied. The comparative minimum-lap-time results on the two-dimensional and 3D track 'Circuit de Catalunya' of Barcelona were presented and compared.

2.7.3 Summery on vehicular optimal control

Applications of optimal control theory in vehicles haven obtained a lot of valuable achievements, which can be used as the theoretical benchmarks for both the vehicle (especially, the race car) control and design. According to the reviewed literature above, the indirect methods solved the optimal control problem using 3-DOF, 7-DOF, and 14-DOF vehicle models with tire models simplified in different levels. Indirect methods are appealing because when a good initial guess is provided, the computation can be very effective. However, as mentioned, the optimal solutions and computing time are too sensitive to the initial guess and it is not easy to handle complicated inequality constraints with the indirect methods. Besides, it is also not suitable for problems that are not easy to obtain the analytical derivatives. In the case of direct methods, the benefits incorporates the easy handling of inequality constraints and the less sensitiveness to the initial guess. However, currently, the most complicated optimal control problem using direct methods are based on the 7-DOF vehicle models with different kinds of simplifications on the load transfer and tire models, which limited the number of parameters that can be optimized and a more accurate representation of the vehicle dynamics. Moreover, studies on electric vehicles with optimal control

approaches are found in much less literature, only [95] and [30] presented some results of the optimal torque control of a 4 independent-wheel-driving electric vehicle. The implemented vehicle model in both work is the same 7-DOF vehicle model with simplified tire model and load transfer model.

CHAPTER 3

Vehicle Modelling

3.1 Introduction

Vehicle modelling involves idealizing the real vehicle system into a set of equations of motion. These equations are then solved by integrating methods to test the vehicle's performance with corresponding design or control solutions. The simplest vehicle model is a point mass model with a single wheel and has two degrees of freedom (DOF) which represent the lateral and longitudinal acceleration. Ref. [96] illustrates that the yaw DOF can be added by extending the model to have two wheels, which is commonly known as a single track model. The yaw moment are introduced to calculate the yaw accelerations with a suitable value of yaw inertia. To improve its accuracy and give a better estimation of the slip angles of the tyres, the single track model can be extended to a two track model with four wheels. In order to evaluate the normal load of each wheel, the load transfer in lateral and longitudinal directions is then approximated by a quasi-static approximation [97]. The tire slip of each wheel can be obtained with the velocity of the wheel center and angular velocity of each wheel. The two track model is also called the 7 DOF vehicle model. In order to predict overall vehicle behavior for cornering, braking, acceleration and comfort performance studies for four wheel driving vehicles with independent suspensions more accurately, a 14-DOF vehicle model is essential to be developed [98].

This chapter serves as the base of this PhD project and provides a detail description of the developed models for the optimal powertrain design and control. A 14 DOF vehicle model supporting vectorized computation together with a suspension model

considering the details of toe angle, camber angle, anti-roll force, etc., were derived based on Lagrangian dynamics. To accurately predict the behavior of a vehicle, it is also required to estimate the external forces acting on the vehicle as precisely as possible. An empirical tire model based on the well-known Magic formula equations [99] was programmed to calculate the tire forces, in particular, the tire model developed in MATLAB supporting inputs of the standard '.tir' tire data file. In order to evaluate the effect of design parameters of the motor and the transmission to the lap time of the race car, the mass model of the motor and transmission mainly concerning the dependence of the mass and output torque of the powertrain on the design parameters were derived. At last, a virtual driver model is devised to track a given trajectory depicted in curvilinear coordinate system based on the proposed control logic. Tremendous computing workload is a common issue in large scale optimization and optimal control problem, in order to improve the calculation efficiency, the mentioned models are all programmed in vectorized formats. The entire vehicle model developed in MATLAB is finally validated with a well-known vehicle dynamics simulator 'VI-CarRealtime' developed by VI-Grade.

3.2 14-DoF vehicle model

The configuration of the entire vehicle model is presented in Figure 3.1, the interactions between the vehicle body, suspension, unsprung mass and tire model are demonstrated.

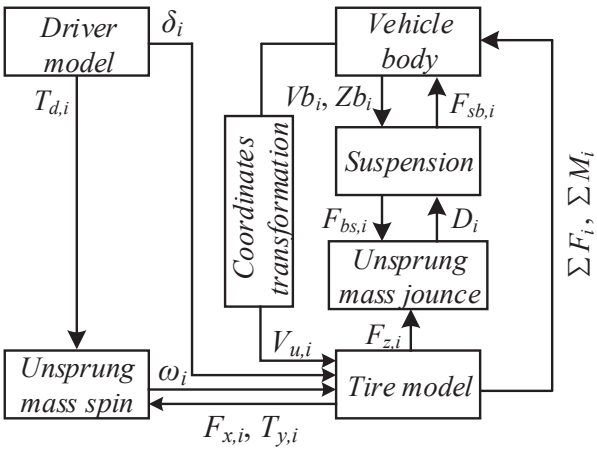

Figure 3.1: *Configuration of the 14 DOF vehicle model*

The inputs of the developed 14-DOF vehicle model are the steer wheel angle δ_i and wheel torque $T_{d,i}$, which will be different for different maneuvers. The spin motion of each wheel is driven by the torque and longitudinal force acted on each. The inputs of each tire model are the angular velocity of the wheel ω_i , wheel center velocity $V_{u,i}$,, camber angle and normal load $F_{z,i}$, while the outputs are the tire forces and moments.

The vertical motion of the each unsprung mass is driven by the normal force of the tire road interaction and the vertical suspension force. While the 6 DOF of vehicle body are driven by the longitudinal, lateral tire forces, aerodynamic forces and the suspension forces acted on it.

3.2.1 Degrees of freedom

The 14-DOF vehicle model in this work represents the dynamic behavior of a simplified vehicle consists of five rigid parts, one of which is the vehicle body (sprung mass), and the left four are the connected 4 wheel parts (unsprung mass). As it is shown in Figure 3.2, there are 6 DOF of the vehicle body allows it to displace in the longitudinal, lateral and vertical direction as weel as to roll, pitch and yaw. In this work, the four wheels are supposed to be fixed with the chassis to move in the longitudinal and lateral direction except their independent vertical and rotational displacement. Thus, the 4 wheel parts have 2 DOF each, one allows the wheel to move in vertical direction with regard to the vehicle body, and the other allows the wheel to rotate around the axle.

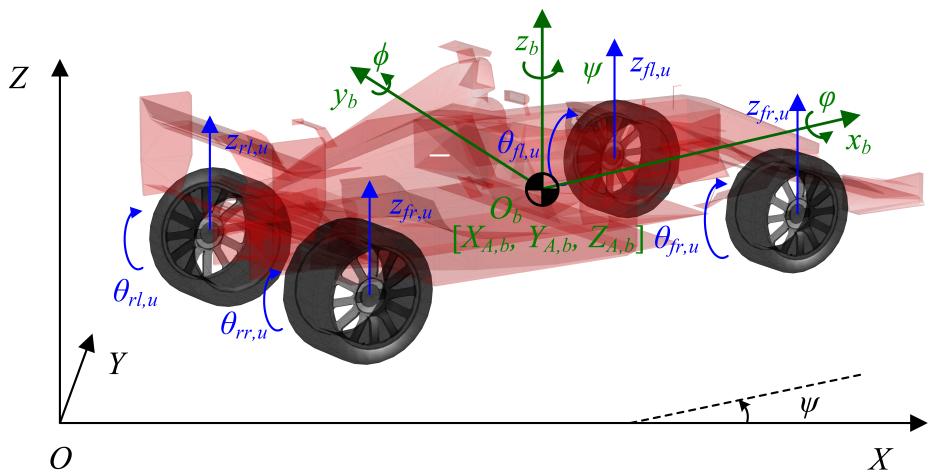

Figure 3.2: *Degree of freedoms*

There are two reference systems used to describe the 14 DOF, $O-XYZ$ is the global reference system to describe the absolute position of the mass center of the vehicle, while $O_b - x_b y_b z_b$ is the moving frame fixed on the mass center of the sprung mass to describe the relative displacements.

The generalized coordinates are chosen and denoted with vector $\boldsymbol{q}$:

$$\boldsymbol{q} = \left\{ \begin{array}{c} \boldsymbol{q}_b \\ \boldsymbol{q}_u \end{array} \right\} = \left\{ \begin{array}{c} [X_{A,b}, Y_{A,b}, Z_{A,b}, \varphi, \phi, \psi]^T \\ [\theta_{ufr}, \theta_{ufl}, \theta_{urr}, \theta_{url}, z_{ufr}, z_{ufl}, z_{urr}, z_{url}]^T \end{array} \right\} \tag{3.1}$$

where the vector in the first row $\boldsymbol{q}_b$ is composed by the six independent variables of the rigid chassis, which are the absolute displacements of the center of gravity (CoG) in X_A , Y_A and Z_A directions of the horizontal plane, and the rotations φ, ϕ, and ψ around the

X_A , Y_A, Z_A axes respectively. The vector $\boldsymbol{q}_u$ is composed by the vertical displacement z_{ui} and rotation $\theta_{u,i}$ of each wheel.

The corresponding velocity vector is presented as:

$$\dot{\boldsymbol{q}} = \begin{Bmatrix} \dot{\boldsymbol{q}}_b \\ \dot{\boldsymbol{q}}_u \end{Bmatrix} = \begin{Bmatrix} [\dot{X}_{A,b}, \dot{Y}_{A,b}, \dot{z}, \dot{\varphi}, \dot{\phi}, \dot{\psi}]^T \\ [\dot{z}_{ufr}, \dot{z}_{ufl}, \dot{z}_{urr}, \dot{z}_{url}, \dot{\theta}_{ufr}, \dot{\theta}_{ufl}, \dot{\theta}_{urr}, \dot{\theta}_{url}]^T \end{Bmatrix} \tag{3.2}$$

The dimensions of the race car in XY plane are presented as Figure 3.3. The relative position of the wheel center in XY plane of the vehicle reference frame can be denoted as:

$$\begin{Bmatrix} \boldsymbol{x}_w \\ \boldsymbol{y}_w \end{Bmatrix} = \begin{Bmatrix} [l_f, l_f, -l_r, -l_r] \\ \frac{1}{2}[-w_f, w_f, -w_r, w_r] \end{Bmatrix} \tag{3.3}$$

The vertical relative position of the unsprung mass z_w in the vehicle reference frame can be denoted with its absolute vertical position z_u , xy position in the vehicle reference frame and the roll angle, pitch angle, absolute vertical position:

$$z_w = z_u - (\boldsymbol{y}_w \psi - \boldsymbol{x}_w \phi + Z_{A,b}) \tag{3.4}$$

the corresponding velocity of the unsprung mass in the vehicle reference frame can be denoted as:

$$\dot{z}_w = \dot{z}_u - (\boldsymbol{y}_w \dot{\psi} - \boldsymbol{x}_w \dot{\phi} + \dot{Z}_{A,b}). \tag{3.5}$$

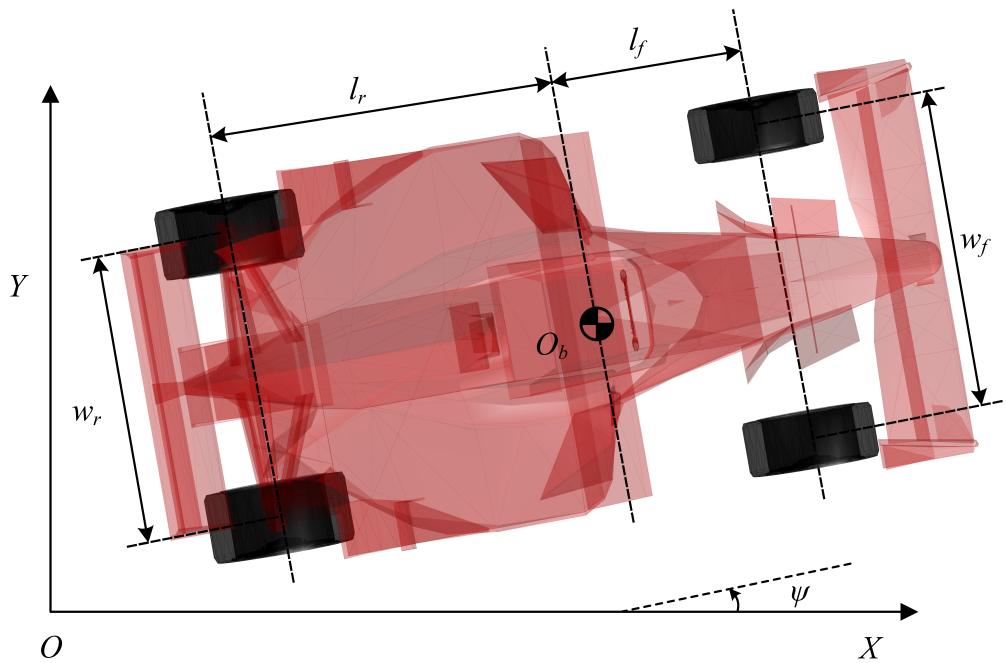

Figure 3.3: *Dimension of the race car in XY plane*

The motion equations of the 14-DOF vehicle model can be derived based on the Lagrangian dynamics:

$$\begin{cases} \dfrac{d}{dt}\left(\dfrac{\partial T}{\partial \dot{\boldsymbol{q}}_b}\right) - \dfrac{\partial T}{\partial \boldsymbol{q}_b} = \boldsymbol{Q}_b \\ \dfrac{d}{dt}\left(\dfrac{\partial T}{\partial \dot{\boldsymbol{q}}_u}\right) - \dfrac{\partial T}{\partial \boldsymbol{q}_u} = \boldsymbol{Q}_u \end{cases} \tag{3.6}$$

where T is the kinetic energy of the system, $\boldsymbol{Q}_b$ and $\boldsymbol{Q}_u$ are the generalized forces applied on the sprung mass and unsprung mass respectively.

3.2.2 Kinetic energy of the sprung mass

As aforementioned, the wheels are fixed with the vehicle body in $x_b - y_b$ directions. Based on this consideration, the kinetic energy of the sprung mass can be denoted as:

$$T_b = \frac{1}{2}\boldsymbol{V}_b^T[M_b]\boldsymbol{V}_b + \sum \frac{1}{2}\boldsymbol{V}_{u,i}^T[M_{u,i}]\boldsymbol{V}_{u,i} \tag{3.7}$$

where the symbols in the above equation will be described in the following paragraphs.

As shown in Figure 3.2, there are two reference systems used in this work: the global (inertia) reference system fixed with the ground and the moving reference system fixed on the vehicle body. The origin of the moving frame is located in the CoG of the sprung mass, while the $\boldsymbol{x}_b$, $\boldsymbol{y}_b$ and $\boldsymbol{z}_b$ axes point forward the longitudinal, lateral and vertical direction of motion. The two reference frame are connected with a transformation matrix $[h_{A,b}]$:

$$[h_{A,b}] = \begin{bmatrix} \cos\psi & \sin\psi & 0 & 0 & 0 & 0 \\ -\sin\psi & \cos\psi & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

The velocity of the vehicle body $\boldsymbol{V}_b$ in the moving frame can thus be denoted as:

$$\boldsymbol{V}_b = \begin{bmatrix} V_{xb} \\ V_{yb} \\ V_{zb} \\ \omega_{xb} \\ \omega_{yb} \\ \omega_{zb} \end{bmatrix} = [h_{A,b}] \begin{bmatrix} \dot{X}_{A,b} \\ \dot{Y}_{A,b} \\ \dot{z} \\ \dot{\varphi} \\ \dot{\phi} \\ \dot{\psi} \end{bmatrix} = [h_{A,b}]\dot{q}_b \tag{3.9}$$

The velocity $\boldsymbol{V}_{u,i}$ of each unsprung mass is calculated with its relative position in the

moving frame and the velocity vector of the vehicle body, which can be denoted as:

$$\boldsymbol{V}_{u,i} = \begin{bmatrix} v_{ux,i} \\ v_{uy,i} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & z_{w,i} & -y_{w,i} \\ 0 & 1 & 0 & -z_{w,i} & 0 & x_{w,i} \end{bmatrix} \begin{bmatrix} V_{xb} \\ V_{yb} \\ V_{zb} \\ \omega_{xb} \\ \omega_{yb} \\ \omega_{zb} \end{bmatrix} = [h_{b,u,i}][h_{A,b}]\dot{q}_b \tag{3.10}$$

where $\boldsymbol{v}_{ux,i}$ and $\boldsymbol{v}_{uy,i}$ are the longitudinal and lateral velocities of the unsprung mass in the moving frame, while $\boldsymbol{x}_{w,i}$, $\boldsymbol{y}_{w,i}$ and $\boldsymbol{z}_{u,i}$ are the position coordinates of the unsprung mass in the moving frame denoted by Equation (3.3). $i = \{fr, fl, rr, rl\}$ means front right, front left, rear right and rear left.

The mass matrix of the sprung mass $[M_b]$ and each unsprung mass $[M_{u,i}]$ are denoted as Equation (3.11) and Equation (3.12) respectively.

$$[M_b] = \begin{bmatrix} m_b & 0 & 0 & 0 & 0 & 0 \\ 0 & m_b & 0 & 0 & 0 & 0 \\ 0 & 0 & m_b & 0 & 0 & 0 \\ 0 & 0 & 0 & J_{xxb} & 0 & 0 \\ 0 & 0 & 0 & 0 & J_{yyb} & 0 \\ 0 & 0 & 0 & 0 & 0 & J_{zzb} \end{bmatrix} \tag{3.11}$$

$$[M_{u,i}] = \begin{bmatrix} m_{u,i} & 0 \\ 0 & m_{u,i} \end{bmatrix} \tag{3.12}$$

where m_b is the mass of the sprung mass, J_{xxb}, J_{yyb} and J_{zzb} are respectively the inertias of the sprung mass around the $O_b - x_b$, $O_b - y_b$ and $O_b - z_b$ axles.

With the above items, the kinetic energy of the sprung mass can be written in a more compact form:

$$\begin{aligned} T_b &= \frac{1}{2}\boldsymbol{V}_b^T[M_b]\boldsymbol{V}_b + \sum \frac{1}{2}\boldsymbol{V}_{u,i}^T[M_{u,i}]\boldsymbol{V}_{u,i} \\ &= \frac{1}{2}\dot{\boldsymbol{q}}_b^T[h_{A,b}]^T[M_b][h_{A,b}]\dot{\boldsymbol{q}}_b + \sum \frac{1}{2}\dot{\boldsymbol{q}}_b^T[h_{A,b}]^T[h_{b,u,i}]^T[M_{u,i}][h_{b,u,i}][h_{A,b}]\dot{\boldsymbol{q}}_b \\ &= \frac{1}{2}\dot{\boldsymbol{q}}_b^T[M_{gb}]\dot{\boldsymbol{q}}_b \end{aligned} \tag{3.13}$$

The generalized mass matrix $[M_{gb}]$ of the sprung mass is a function of the

generalized coordinates $Z_{A,b}$, ψ and z_u , and can be derived as:

$$\left[M_{gb}\right] = \begin{bmatrix} m_b + \sum m_{u,i} & 0 & 0 & \sum m_{u,i}z_{u,i}\sin\psi & \sum m_{u,i}z_{u,i}\cos\psi & -\sum(m_{u,i}x_{u,i}\sin\psi + m_{u,i}y_{u,i}\cos\psi) \\ 0 & m_b + \sum m_{u,i} & 0 & -\sum m_{u,i}z_{u,i}\cos\psi & \sum m_{u,i}z_{u,i}\sin\psi & \sum(m_{u,i}x_{u,i}\cos\psi - m_{u,i}y_{u,i}\sin\psi) \\ 0 & 0 & m_b & 0 & 0 & 0 \\ \sum m_{u,i}z_{u,i}\sin\psi & -\sum m_{u,i}z_{u,i}\cos\psi & 0 & \sum m_{u,i}z_{u,i}^2 + J_{xxb} & 0 & -\sum m_{u,i}x_{u,i}z_{u,i} \\ \sum m_{u,i}z_{u,i}\cos\psi & \sum m_{u,i}z_{u,i}\sin\psi & 0 & 0 & \sum m_{u,i}z_{u,i}^2 + J_{yyb} & -\sum m_{u,i}y_{u,i}z_{u,i} \\ -\sum(m_{u,i}x_{u,i}\sin\psi + m_{u,i}y_{u,i}\cos\psi) & \sum(m_{u,i}x_{u,i}\cos\psi - m_{u,i}y_{u,i}\sin\psi) & 0 & -\sum m_{u,i}x_{u,i}z_{u,i} & -\sum m_{u,i}y_{u,i}z_{u,i} & \sum(m_{u,i}x_{u,i}^2 + m_{u,i}y_{u,i}^2) + J_{zzb} \end{bmatrix} \tag{3.14}$$

3.2.3 Kinetic energy of the unsprung mass

The kinetic energy of the unsprung mass is composed by the vertical and rotational motion parts, which can be denoted as:

$$T_u = \frac{1}{2}\boldsymbol{\omega}_u^T[J_u]\boldsymbol{\omega}_u + \frac{1}{2}\boldsymbol{V}_{uz}^T[M_u]\boldsymbol{V}_{uz} \tag{3.15}$$

In this work, the inertia of the propulsion system is taken into account as a part of the unsprung mass, the angular velocity $\boldsymbol{\omega}_u$ of the propulsion system and the unsprung mass for a independent drive topology can be denoted as Equation (3.16).

$$\boldsymbol{\omega}_u = [h_u]\begin{bmatrix} \dot{\theta}_{ufr} \\ \dot{\theta}_{ufl} \\ \dot{\theta}_{urr} \\ \dot{\theta}_{url} \end{bmatrix} \tag{3.16}$$

The inertia matrix J_u of the one-motor rear driving (1M-RWD), two-motor rear driving (2M-RWD) and four-motor independent wheel driving (4M-IWD) topologies can be denoted with a diagonal matrix respectively as Equation (3.17).

$$[J_u] = \begin{cases} diag\{J_{u,fr}, J_{u,fl}, J_{u,rr}, J_{u,rl}, J_d\}, & \text{1M-RWD} \\ diag\{J_{u,fr}, J_{u,fl}, J_{u,rr}, J_{u,rl}, J_{d,rr}, J_{d,rl}\}, & \text{2M-RWD} \\ diag\{J_{u,fr}, J_{u,fl}, J_{u,rr}, J_{u,rl}, J_{d,fr}, J_{d,fl}, J_{d,rr}, J_{d,rl}\}, & \text{4M-IWD} \end{cases} \tag{3.17}$$

where $J_{u,i}$ is the rotational inertia of the unsprung mass, $J_{d,i}$ is the rotational inertia of each motor.

The transformation matrix $[h_u]$ used to calculate the angular velocities of different propulsion systems is presented as Equation (3.18).

$$[h_u] = \begin{cases} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & i/2 & i/2 \end{bmatrix}, & \text{1M-RWD} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & i_{rr} & 0 \\ 0 & 0 & 0 & i_{rl} \end{bmatrix}, & \text{2M-RWD} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ i_{fr} & 0 & 0 & 0 \\ 0 & i_{fl} & 0 & 0 \\ 0 & 0 & i_{rr} & 0 \\ 0 & 0 & 0 & i_{rl} \end{bmatrix}, & \text{4M-IWD} \end{cases} \tag{3.18}$$

where i_d, i_{fr}, i_{fl}, i_{rr} and i_{rl} are the related transmission ratios.

The vertical velocity matrix $\boldsymbol{V}_{uz}$, and the mass matrix $[M_u]$ of the unsprung mass can be denoted as:

$$\boldsymbol{V}_{uz} = \begin{bmatrix} \dot{z}_{ufr} \\ \dot{z}_{ufl} \\ \dot{z}_{urr} \\ \dot{z}_{url} \end{bmatrix} \tag{3.19}$$

$$[M_u] = \begin{bmatrix} m_{ufr} & 0 & 0 & 0 \\ 0 & m_{ufl} & 0 & 0 \\ 0 & 0 & m_{urr} & 0 \\ 0 & 0 & 0 & m_{url} \end{bmatrix} \tag{3.20}$$

With the above terms, the kinetic energy of the unsprung mass is denoted as Equation (3.21), and it can be derived as a function of the generalized coordinates

and generalized mass matrix.

$$
\begin{aligned}
T_u &= \frac{1}{2}\boldsymbol{\omega}_u^T[J_u]\boldsymbol{\omega}_u + \frac{1}{2}\boldsymbol{V}_{uz}^T[M_u]\boldsymbol{V}_{uz} \\
&= \frac{1}{2}\begin{bmatrix}\dot{\theta}_{ufr}\\ \dot{\theta}_{ufl}\\ \dot{\theta}_{urr}\\ \dot{\theta}_{url}\end{bmatrix}^T [h_u]^T[J_u][h_u]\begin{bmatrix}\dot{\theta}_{ufr}\\ \dot{\theta}_{ufl}\\ \dot{\theta}_{urr}\\ \dot{\theta}_{url}\end{bmatrix} + \frac{1}{2}\begin{bmatrix}\dot{z}_{ufr}\\ \dot{z}_{ufl}\\ \dot{z}_{urr}\\ \dot{z}_{url}\end{bmatrix}^T [I][M_u][I]\begin{bmatrix}\dot{z}_{ufr}\\ \dot{z}_{ufl}\\ \dot{z}_{urr}\\ \dot{z}_{url}\end{bmatrix} \\
&= \frac{1}{2}\begin{bmatrix}\dot{\theta}_{ufr}\\ \dot{\theta}_{ufl}\\ \dot{\theta}_{urr}\\ \dot{\theta}_{url}\\ \dot{z}_{ufr}\\ \dot{z}_{ufl}\\ \dot{z}_{urr}\\ \dot{z}_{url}\end{bmatrix}^T \begin{bmatrix}[h_u]^T[J_u][h_u] & \\ & [M_u]\end{bmatrix} \begin{bmatrix}\dot{\theta}_{ufr}\\ \dot{\theta}_{ufl}\\ \dot{\theta}_{urr}\\ \dot{\theta}_{url}\\ \dot{z}_{ufr}\\ \dot{z}_{ufl}\\ \dot{z}_{urr}\\ \dot{z}_{url}\end{bmatrix} \\
&= \frac{1}{2}\dot{\boldsymbol{q}}_u^T[M_{gu}]\dot{\boldsymbol{q}}_u
\end{aligned}
\tag{3.21}
$$

where $[M_{gu}]$ is the generalized unsprung mass collecting only static values, it is denoted as:

$$
[M_{gu}] = \begin{bmatrix}[h_u]^T[J_u][h_u] & \\ & [M_u]\end{bmatrix} \tag{3.22}
$$

3.2.4 Generalized forces

The forces and torques acted on the sprung mass is illustrated as Figure 3.4. The vertical forces includes the gravity of the sprung mass G_s, the four suspension forces $F_{bs,i}$, the front and rear aerodynamic down forces $F_{down,f}$ and $F_{down,r}$, and the anti-roll force $F_{atr,i}$. Forces applied in the longitudinal direction are the four longitudinal tire forces $F_{x,i}$ and the aerodynamic drag force F_w. In lateral direction, there are only the four lateral tire forces $F_{y,i}$. The torques applied on the unsprung mass are the four driving torques $T_{d,i}$ transmitted by the shafts that connecting the wheels.

The aerodynamic drag and down forces are presented as :

$$
F_{drag} = \frac{1}{2}C_d\rho A V_x^2 \tag{3.23}
$$

$$
F_{down,i} = \frac{1}{2}C_{l,i}\rho A V_x^2 \tag{3.24}
$$

where C_d is the drag coefficient, ρ is the air density, A is the vehicle effective area, V_x is the vehicle longitudinal velocity, $F_{down,i}$ is the aerodynamic downforce, $C_{l,i}$ is the lift Coefficient, $i = \{front, rear\}$.

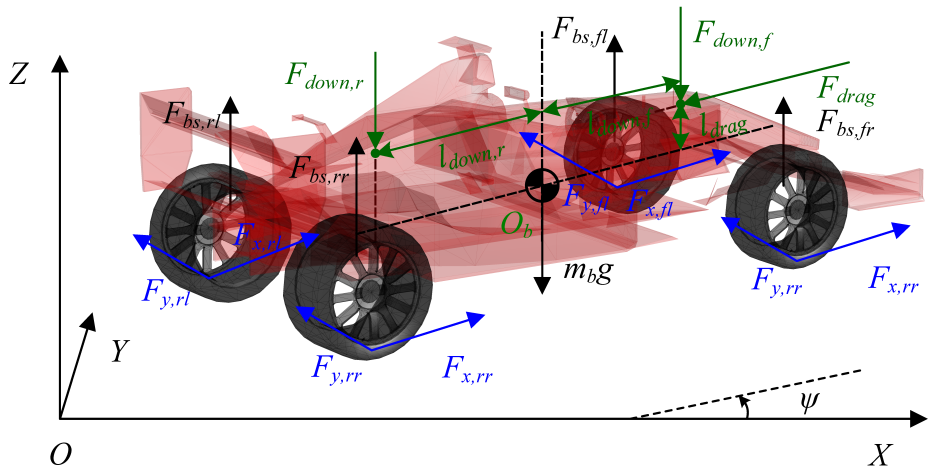

Figure 3.4: *Forces and torques applied on the sprung mass*

The force matrix of including the forces and torques acted on the sprung mass in each direction is denoted as:

$$\boldsymbol{F}_b = \begin{bmatrix} \sum (F_{x,i}\cos\delta_i - F_{y,i}\sin\delta_i)\cos\psi - \sum (F_{x,i}\sin\delta_i + F_{y,i}\cos\delta_i)\sin\psi - F_w\cos\psi \\ \sum (F_{x,i}\cos\delta_i - F_{y,i}\sin\delta_i)\sin\psi + \sum (F_{x,i}\sin\delta_i + F_{y,i}\cos\delta_i)\cos\psi - F_w\sin\psi \\ \sum F_{bs,i} - mg + F_{down,f} + F_{down,r} \\ -\sum F_{atr,i}y_{u,i} + \sum F_{bs,i}y_{u,i} + \sum (F_{x,i}\sin\delta_i + F_{y,i}\cos\delta_i)Z_{A,b} + \sum T_{d,i}\sin\delta_i \\ \sum F_{down,i}x_{u,i} - \sum F_{bs,i}x_{u,i} - \sum (F_{x,i}\cos\delta_i - F_{y,i}\sin\delta_i)Z_{A,b} - \sum T_{d,i}cos\delta_i \\ \sum M_{z,i} + \sum (F_{x,i}\sin\delta_i + F_{y,i}\cos\delta_i)x_{w,i} - \sum (F_{x,i}\cos\delta_i - F_{y,i}\sin\delta_i)y_{w,i} \end{bmatrix} \tag{3.25}$$

The torques and forces applied on the unsprung mass is presented in Figure 3.5.

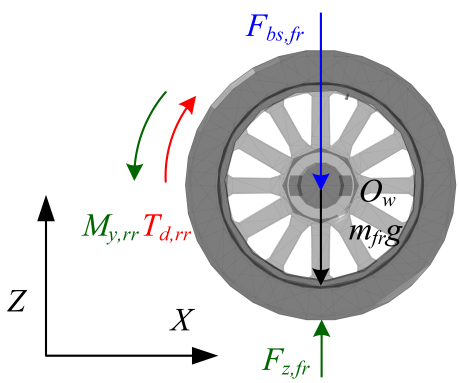

Figure 3.5: *Forces and torque applied on the unsprung mass*

The force matrix including the torques and forces applied on the unsprung mass can

be denoted as

$$\boldsymbol{F}_u = \begin{bmatrix} T_{d,fr} + M_{y,fr} - F_{x,fr}z_{fr} \\ T_{d,fl} + M_{y,fl} - F_{x,fl}z_{fl} \\ T_{d,rr} + M_{y,rr} - F_{x,rr}z_{rr} \\ T_{d,rl} + M_{y,rl} - F_{x,fr}z_{rl} \\ F_{atr,fr} - F_{bs,fr} + F_{z,fr} - m_{fr}g \\ -F_{atr,fl} - F_{bs,fl} + F_{z,fl} - m_{fl}g \\ F_{atr,rr} - F_{bs,rr} + F_{z,rr} - m_{rr}g \\ -F_{atr,rl} - F_{bs,rl} + F_{z,rl} - m_{rl}g \end{bmatrix} \tag{3.26}$$

where $M_{y,i}$ is the rolling resistance moment, $F_{atr,i}$ is the anti-roll force, $F_{z,i}$ is the vertical force applied by the ground, m_i is the mass of each wheel.

The generalized forces can be derived based on the force analysis and virtual work principle, which are presented in the following equations. The similar detail derivation process can be referred to [98].

$$\boldsymbol{Q}_b = \sum \boldsymbol{F}\frac{\partial \boldsymbol{r}_b}{\partial \boldsymbol{q}_b} = \boldsymbol{F}_b \tag{3.27}$$

$$\boldsymbol{Q}_u = \sum \boldsymbol{F}\frac{\partial \boldsymbol{r}_u}{\partial \boldsymbol{q}_u} = \boldsymbol{F}_u \tag{3.28}$$

3.2.5 Lagrange's equations

The generalized motion equations of the rigid sprung and unsprung mass are derived in the form of Equation (3.29) according to the Lagrangian mechanics and D'Alembert's principle. In this work, we differentiate the generalized mass matrix directly instead of calculating the partial differentials of the system kinetic energy to the time and generalized coordinates separately, which is more efficient for derivation.

$$\begin{cases} \dfrac{d}{dt}\left(\dfrac{\partial T}{\partial \dot{\boldsymbol{q}}_b}\right) - \dfrac{\partial T}{\partial \boldsymbol{q}_b} = [M_{gb}]\ddot{\boldsymbol{q}}_b + [\dot{M}_{gb}]\dot{\boldsymbol{q}}_b - \dfrac{1}{2}\dot{\boldsymbol{q}}_b^T[\dfrac{\partial M_{gb}}{\partial \boldsymbol{q}_b}]\dot{\boldsymbol{q}}_b = \boldsymbol{Q}_b \\ \dfrac{d}{dt}\left(\dfrac{\partial T}{\partial \boldsymbol{q}_u}\right) - \dfrac{\partial T}{\partial \boldsymbol{q}_u} = [M_{gu}]\ddot{\boldsymbol{q}}_u + [\dot{M}_{gu}]\dot{\boldsymbol{q}}_u - \dfrac{1}{2}\dot{\boldsymbol{q}}_b^T[\dfrac{\partial M_{gb}}{\partial \boldsymbol{q}_u}]\dot{\boldsymbol{q}}_b = \boldsymbol{Q}_u \end{cases} \tag{3.29}$$

3.3 Suspension model

The suspension model in this section involves the calculation of spring forces, damping forces, anti-roll forces, toe angles, camber angles and the steering angles. The suspension model elaborated below is capable to described the behavior of both the dependent and independent suspensions.

3.3.1 Spring and damping forces

The suspension force is composed by the spring force and damping force. When the stiffness and damping ratio are constant values, the suspension force can be denoted as

Equation (3.30), the spring force on the spring is denoted as a function of the stiffness $k_{bs,i}$ and the deformation $l_{bs,i}$ of the spring, while the damping force is a function of damping $c_{d,i}$ and velocity $\dot{l}_{d,i}$ of the damper.

$$F_{bss,i}=k_{bs,i}\Delta l_{s,i} + c_{d,i}\dot{l}_{d,i} \tag{3.30}$$

The spring stiffness can be a constant value or a nonlinear function of the deformation, while the damping ratio can be a constant or a nonlinear function of the damper velocity as shown in Equation (3.31), in which case the spring force and damping force are respectively nonlinear functions of the deformation and damping velocity.

$$\begin{cases} k_{bs,i} &= f(l_{s,i}) \\ c_{d,i} &= f(\dot{l}_{s,i}) \end{cases} \tag{3.31}$$

The nonlinear functions can also be described by tables and 1-D interpolations, for instance, the damping force can be given as Figure 3.6.

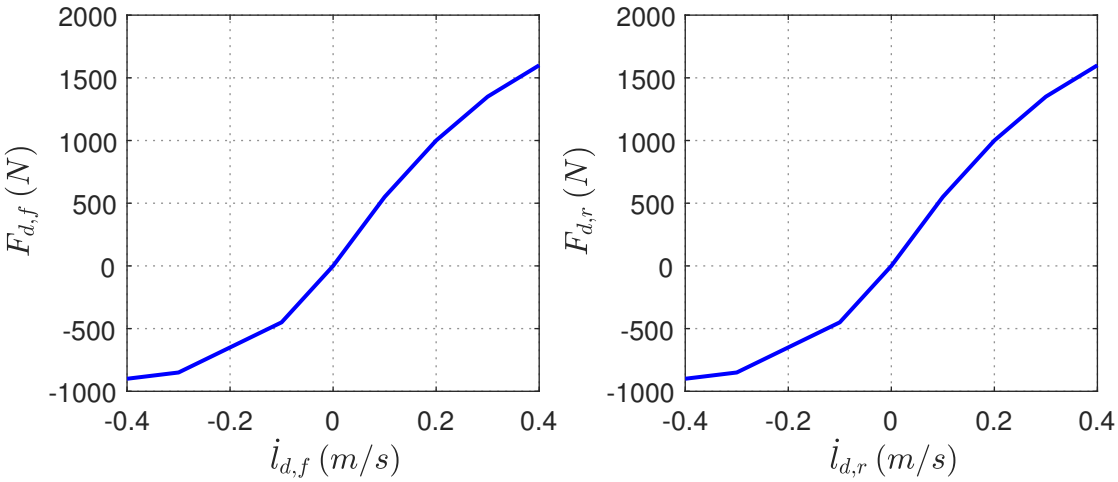

Figure 3.6: *Damping forces of the front and rear damper*

The deformation of the spring travel $\Delta l_{s,i}$ is a function of wheel jounce ΔD_i, which can be calculated by a transmission ratio $\lambda_{s,i}$,

$$\Delta l_{s,i} = \lambda_{s,i}\Delta D_i \tag{3.32}$$

where the wheel jounce ΔD_i is the vertical movement of wheel or axle relative to the vehicle reference frame, which can be defined as:

$$\Delta D_i = z_{w,i} - z_{w0,i} \tag{3.33}$$

while the deformation velocity of the damper can be denoted as:

$$\dot{l}_{s,i} = \lambda_{s,i}\dot{z}_{w,i} \tag{3.34}$$

where $z_{w,i}$ is the vertical position of the unsprung mass in the moving frame, $z_{w0,i}$ is its initial value.

Finally, the suspension force acted on each wheel can be denoted as:

$$F_{bs,i} = \lambda_{s,i} F_{bss,i} \tag{3.35}$$

3.3.2 Anti-roll forces

In this work, the anti-roll bars are implemented to reduce the roll displacement of the race car during fast cornering or over road irregularities. The anti-roll bar connects opposite left and right wheels together through a short lever arm linked by a torsion spring. Tacking the front axle as an example, the anti-roll force is a function of the average jounce $\bar{D}_{l,f}$ and delta jounce $\Delta D_{l,f}$ of the left and right wheels as denoted in Equation (3.36), the anti-roll forces $F_{atr,f}$, $F_{atr,r}$ can also be calculated with the given parameter tables and a 2-D interpretation method such as in Figure 3.7.

$$F_{atr,f} = f(\bar{D}_{l,f}, \Delta D_{l,f}) \tag{3.36}$$

where $\bar{D}_{l,f} = \dfrac{D_{r,f} + D_{l,f}}{2}$, $\Delta D_{l,f} = D_{r,f} - D_{l,f}$, $D_{r,f}$ and $D_{l,f}$ are respectively the front wheel jounce of the right and left side.

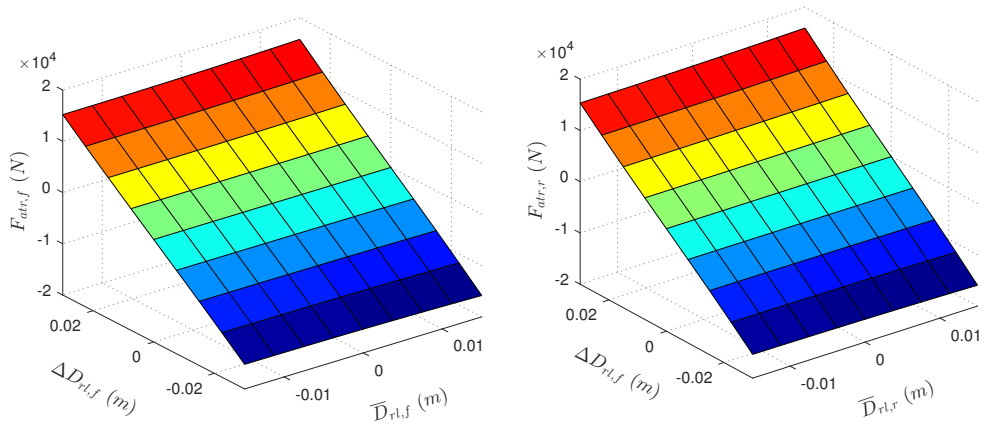

Figure 3.7: *Anti-roll forces of the race car*

3.3.3 Camber angles

The camber angle γ_i can be denoted as a function of the wheel jounce and steering wheel angle input by the driver,

$$\gamma_i = f(\Delta D_i, \delta_{driver}) \tag{3.37}$$

Similarly, the lookup table method based on the 2-D interpolation can be utilized to calculated the camber angle as it is shown in Figure 3.8 and Figure 3.9.

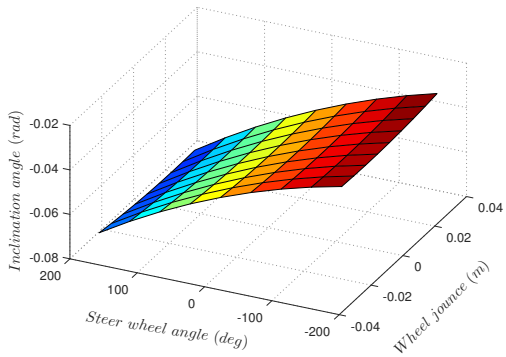

Figure 3.8: *Inclination angle of right side*

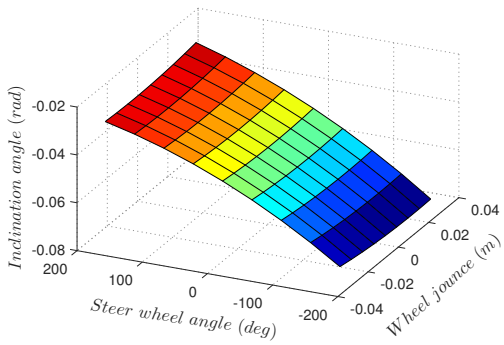

Figure 3.9: *Inclination angle of left side*

3.3.4 Toe angles

Toe angle of each wheel is also considered in this suspension model, which is denoted as a function of the wheel jounce,

$$\xi_i = f(\Delta D_i) \tag{3.38}$$

The toe angles can be calculated with the aforementioned 1-D interpolation method as it is shown in Figure 3.10.

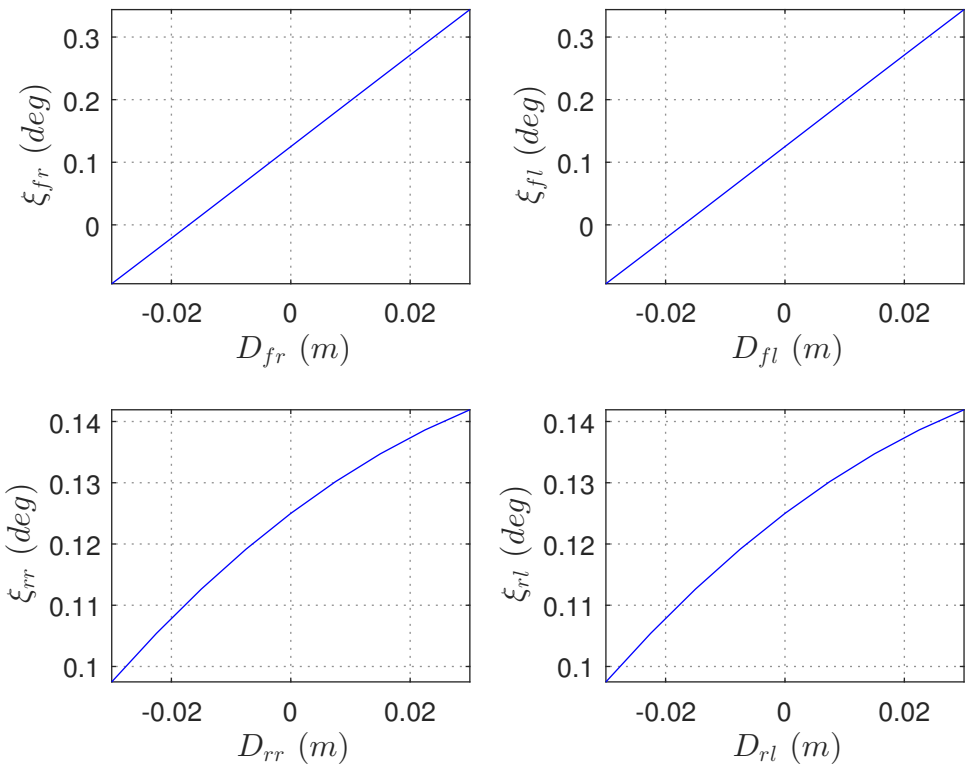

Figure 3.10: *Toe angle of the race car*

3.3.5 Steering angles

The steer angle of each wheel on the ground introduced by the driver is expressed as a function of the wheel jounce and the steering-wheel angle input by the driver,

$$\delta_{d,i} = f(\Delta D_i, \delta_{driver}) \tag{3.39}$$

Again, a 2D interpolation method can be used in this model to obtain the steering angle on the ground δ_i with the presented data in Figure 3.11. The final steering angle

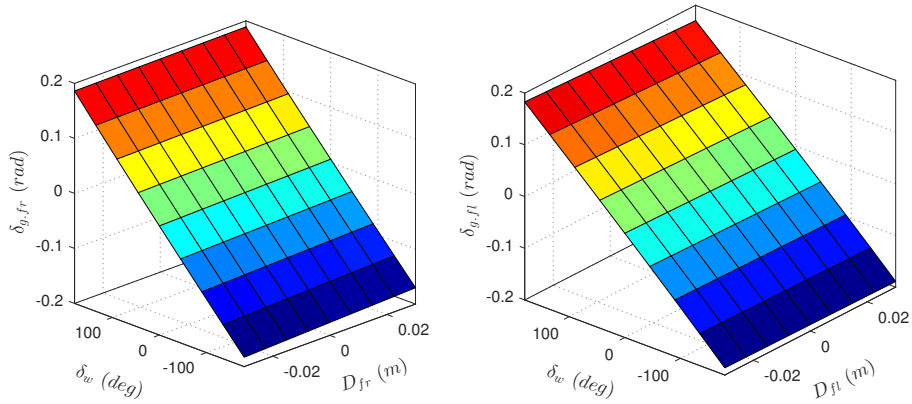

Figure 3.11: *Steering angle at ground*

on the ground in the vehicle reference system is composed by the wheel spindle angle and the toe angle:

$$\delta_i = \delta_{d,i} + \xi_i \tag{3.40}$$

The tables describing the all of the above relationships can be obtained via experiments in bench test.

3.4 Tire model

One of the important aspects in vehicle dynamics simulation is accurate modeling of the tire-road interaction forces because the movement of the vehicle depends on the forces and moments applied to the tires. Many reliable commercial vehicle simulation software integrated the Magic Formula (MF) tire model developed by Pacejka [99]. The Magic Formula is a set of mathematical formula equations that are capable of describing the basic tire characteristics for the interaction forces between the tire and the road under several steady-state operating conditions.

In this section a semi-empirical tire model has been developed based on the full set of magic formula (MF) equations [100], the forces and moments calculated from the MF is shown in Figure 3.12. The longitudinal force $F_{x,i}$, lateral force $F_{y,i}$, overturning couple $M_{x,i}$, rolling resistance moment $M_{y,i}$ and aligning moment $M_{z,i}$ of the tire are calculated with the vertical force $F_{z,i}$, longitudinal slip κ_i , side slip angle α_i and inclination angle γ_i, wheel velocities v_i as inputs, in this section $i = \{FR, FL, RR, RL\}$, which means front right, front left, rear right and rear left.

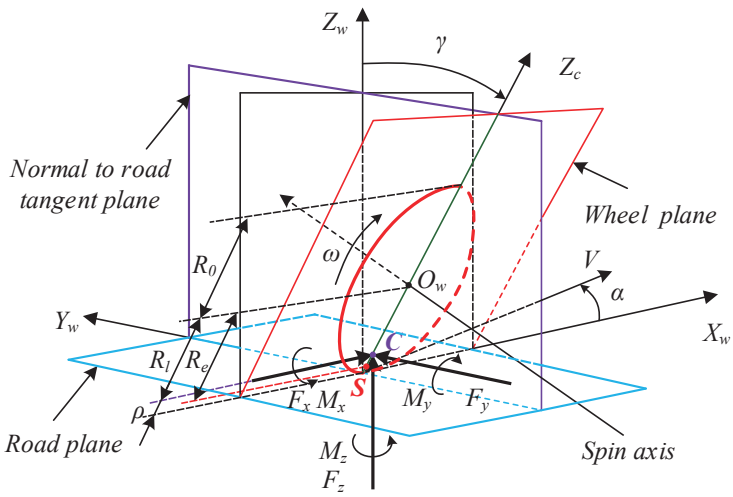

Figure 3.12: *Forces, moments and kinematics variables of the tyre road contact*

The contact point of the road and tire is the intersection of the wheel center plane, road tangent plane and the plane through the wheel spin axis.

In this work, the effective radius $R_{e,i}$ of each tire in the above figure is denoted as:

$$R_{e,i} = R_{0,i} - \frac{F_{z0,i}}{K_{t,i}}(D_{reff,i} arctan(B_{reff,i}\frac{F_{z,i}}{F_{z0,i}}) + F_{reff,i}\frac{F_{z,i}}{F_{z0,i}}) \tag{3.41}$$

where $R_{0,i}$ is the radius of the unloaded tire, $F_{z0,i}$ is the nominal wheel load, $K_{t,i}$ is the vertical stiffness of the tire, $D_{reff,i}$ is the peak value of effective rolling radius, $B_{reff,i}$ is the low load stiffness effective rolling radius, $F_{z,i}$ is the normal load of each tire, $F_{reff,i}$ is the high load stiffness effective rolling radius.

3.4.1 Normal loads

The normal load acted on each tire is the summation of the suspension force, gravity of the unsprung mass and the anti-roll force,

$$F_{z,i} = F_{bs,i} + m_{u,i}g - F_{zar,i} \tag{3.42}$$

3.4.2 Tire slip

The longitudinal slip κ_i in the contact point is denoted with the wheel center velocity $v_{wx,i}$, the wheel rotational velocity ω_i in the wheel reference system, and the effective rolling radius $R_{e,i}$,

$$\kappa_i = \frac{\omega_i R_{e,i} - v_{wx,i}}{v_{wx,i}} \tag{3.43}$$

where $v_{wx,i}$ is the wheel center velocity in the wheel frame which can be denoted as:

$$\begin{aligned} v_{wx,i} &= v_{ux,i} cos\delta_i + v_{uy,i} sin\delta_i \\ v_{wy,i} &= -v_{ux,i} sin\delta_i + v_{uy,i} cos\delta_i \end{aligned} \tag{3.44}$$

where $v_{ux,i}$, $v_{uy,i}$ are the velocity of the wheel center in the vehicle reference frame and are derived in Equation (3.10).

The lateral slip angle α_i generates due to the tire carcass's flexibility when there is cornering operation, and it is defined as the angular difference between the wheel traveling direction and the direction of the tread as shown in Figure 3.13. If the slip angles of the rear tires are greater than the slip angles of the front tires, the vehicle is said to be oversteering, while if the slip angles of the front tires are greater than the slip angles of the rear tires, the vehicle is said to be understeering.

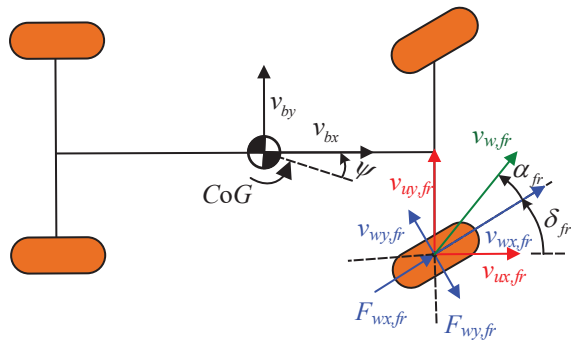

Figure 3.13: *Tire slip angle and the steering angle*

There are two ways to calculate the lateral tire slip which leads to the same results, the first one is to denote it with the wheel center velocity in the vehicle reference frame and the steering angle, their relationship can be denoted as:

$$\tan(\delta_i + \alpha_i) = \frac{v_{uy,i}}{v_{ux,i}}, \tag{3.45}$$

where the slip angle can be derive as:

$$\alpha_i = -\delta_i + arctan(\frac{v_{uy,i}}{v_{ux,i}}) \tag{3.46}$$

The other method is to denote it with the longitudinal and lateral velocity of the wheel center in the wheel frame directly,

$$\alpha_i = arctan(\frac{v_{wy,i}}{v_{wx,i}}) \tag{3.47}$$

3.4.3 Tire forces

With the parameters calculated above and the tire data file which collects all the parameters needed by the MF tire model, it is ready to calculate the tire forces and moments of the each tire now.

The longitudinal tire force in the contact point is:

$$F_{x,i} = (D_{x,i}\sin(C_{x,i}\arctan(B_{x,i}\kappa_i - E_{x,i}(B_{x,i}\kappa_{xi} - \arctan(B_{x,i}\kappa_{x,i})))) + S_{Vx,i})G_{x\alpha,i} \tag{3.48}$$

The lateral force in the contact point is:

$$F_{y,i} = (D_{y,i}\sin(C_{y,i}\arctan(B_{y,i}\alpha_i - E_{y,i}(B_{y,i}\alpha_i - \arctan(B_{y,i}\alpha_i))))) + S_{Vy,i})G_{y\kappa,i} + S_{Vy\kappa,i} \tag{3.49}$$

The vertical tire force $F_{zt,i}$ between the ground and the tire is calculated with the tire vertical stiffness $K_{t,i}$ and damping ratio $C_{t,i}$,

$$F_{zt,i} = K_{t,i}\Delta R_{,i} + C_{t,i}\Delta\dot{R}_i \tag{3.50}$$

where ΔR_i is the vertical deformation of each tire and can be calculated with the original radius of the tire and vertical position of the wheel center.

The overturing moment M_x is denoted as

$$M_x = -R_0 F_Z \lambda_{Mx}(Q_{Sx1}\lambda_{Mx} - Q_{Sx2}\gamma + \frac{Q_{Sx3}F_y}{F_{z0}}) \tag{3.51}$$

For tire data where FITTYP is equal to 5, the rolling resistance M_y is denoted as:

$$M_y = R_0(S_{Vx} + K_x S_{Hx}) \tag{3.52}$$

Otherwise:

$$M_y = -R_0 F_Z \lambda_{My}\{Q_{sy1} + Q_{sy2}\frac{F_x}{F_{z0}} + Q_{sy3}|\frac{V_x}{V_{ref}}| + Q_{sy4}(\frac{V_x}{V_{ref}})^4\}. \tag{3.53}$$

The self aligning moment M_z is denoted as

$$M_{z0} = -tF_{y0} + M_{zr} \tag{3.54}$$

where the meaning of all the above symbols are presented in Appendix A.

To improve the computation efficiency, the MF tire model is programmed in vector format in MATLAB. The tire fores of the front and rear tires using in combined slip conditions are demonstrated in Figure 3.14.

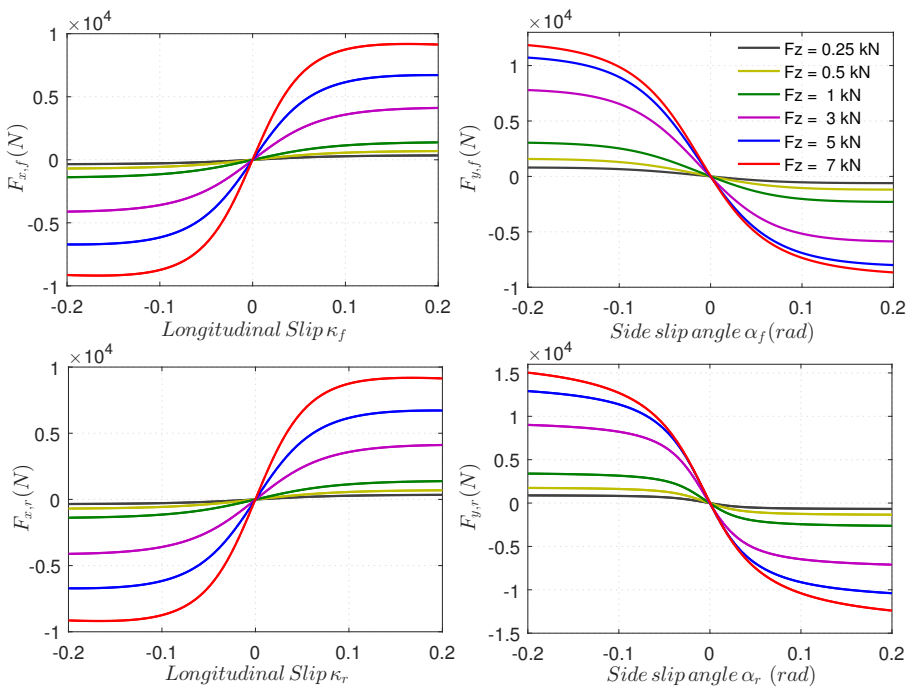

Figure 3.14: *Tire Forces of the front and rear tires*

3.5 Mass model of the powertrain

The powertrain modeling in this paragraph mainly concerns the dependence of the mass of the powertrain on the design parameters.

3.5.1 Mass model of the Electric motor

The propulsion unit selected in this paper is the alternating current (AC) electric motor due to its high power output and light weight. The relationship between the mass and the maximum power $P_{\max}$ and the base speed n_{base} is derived based on the fundamental theory of AC motor design. Equation (3.55) to (3.60) are the basic reference equations when design an AC motor [101],

$$P' = N_p EI \tag{3.55}$$

$$E = \sqrt{2}\pi f N K_w \Phi_m \tag{3.56}$$

$$\Phi_m = BA_{Fe} \tag{3.57}$$

$$I = JA_{Cu}/2NN_p \tag{3.58}$$

$$P' = \sqrt{2}N_p \pi f J K_w B A_{Fe} A_{Cu} \tag{3.59}$$

$$P' = \sqrt{2}N_p \pi n_b p J K_w B A_{Fe} A_{Cu}/120 \tag{3.60}$$

where P' is the apparent power, N_p is the number of phases, E is armature electromotive force, I is armature current, f is frequency of the current, N is the number of series-connected turns of the stator, K_w is the winding factor, Φ_m is the maximum flux of one pole, B is the flux density through the iron, A_{Fe} is the cross-sectional area of the iron, J is current density, A_{Cu} is the total cross-sectional areas of the winding coil, p is the number of pole pairs, n_b is the base speed.

The maximum mechanical power $P_{\max}$ of the motor can be obtained with the power factor $cos\phi$, the efficiency η_m and the apparent power P',

$$P_{\max}=P'\eta_m \cos\phi_N/K_E= \sqrt{2}m\pi n_b p J K_w B A_{Fe} A_{Cu} \eta_m \cos\phi_N/(120K_E) \tag{3.61}$$

where K_E is the ratio of electromotive force to the terminal voltage, $cos\phi$ is the power factor, which is typically given as 0.85 for induction motors.

The relationship between the power and the mass of the electric motor can be derived based on the above equations. The cross-sectional area of the iron A_{Fe} and the total cross-sectional areas of the winding coil A_{Cu} are proportional to the square of the fundamental unit 'Length' respectively,

$$\frac{P_{\max}}{n_b} \propto A_{Fe}A_{Cu} \propto l^2 l^2 \tag{3.62}$$

The volume is proportional to the third power of the fundamental unit Length,

$$V_d \propto l^3 \tag{3.63}$$

based on Equations (3.61), (3.62) and (3.63), the mass of the electric motor can be derived as

$$m_d = \rho_m (\frac{P_{\max}}{n_b})^{3/4} \tag{3.64}$$

where m_d is the mass of the electric motor, ρ_m is the mass factor, l is the fundamental unit 'Length'.

3.5.2 Mass model of the Transmission

In general, the gearbox is one of the heaviest components of the powertrain, hence, weight reduction of the gearbox is a top consideration of the vehicle design. Ref. [102] proposed a method to reveal the relationships among the gear dimensions, transmitted power, gear speed of the input shaft and gear ratio of a helical/spur gearset, the basic equation is given as

$$d_s^2 w = \frac{4774650P}{Kn} \frac{(i+1)^3}{i} \tag{3.65}$$

where d_s is center distance between the two shafts (mm), w is width of the gear (mm), all the gears are assumed to have the same width in this research, P is the transmitted power (kW), n is the rotational speed of input gear (rpm), K is the surface durability factor (N/mm2), i is the gear ratio.

When the physical units of power and rotation speed are respectively kW and Rpm, the torque on the input shaft T_{in} can be expressed as

$$T_{in} = 9549.3 \frac{P}{n} \tag{3.66}$$

by substituting into Equation (3.66), (3.65) becomes [102]:

$$d_s^2 w = \frac{500T_{in}}{K} \frac{(i+1)^3}{i} \tag{3.67}$$

The gear ratio and center distance can be expressed as Equations (3.68), (3.69) respectively,

$$i = \frac{d_o}{d_{in}} \tag{3.68}$$

$$d_s = \frac{1}{2}(d_{in} + d_o) \tag{3.69}$$

by combining Equations (3.68), (3.69) can be denoted in two forms,

$$d_s = \frac{1}{2}(1+i)d_{in} \tag{3.70}$$

$$d_s = \frac{1}{2}(1+\frac{1}{i})d_o \tag{3.71}$$

where d_{in} is the diameter of the input gear (mm), d_o is the diameter of the output gear (mm).

Substitute Equations (3.70) (3.71) into (3.67), we can get:

$$d_{in}^2 w = \frac{2000T_{in}}{K} \frac{(i+1)}{i} \tag{3.72}$$

$$d_o^2 w = \frac{2000T_{in}}{K} i(i+1) \tag{3.73}$$

the weight of one gear set can be derived as:

$$m_g = \frac{1}{4}\psi\rho\pi w(d_{in}^2 + d_o^2). \tag{3.74}$$

Combine Equations (3.72), (3.73) and (3.74), the weight of one gear set can be expressed as

$$m_g(i) = \frac{500T_{in}}{K}\psi\rho\pi(1 + \frac{1}{i} + i + i^2), \tag{3.75}$$

and the weight of all the gear set in a gearbox can be estimated as

$$m_{gt} = \sum_{j=1}^{N_g} m_{g,j} \tag{3.76}$$

where m_g is the mass of one pair of gear set, ψ is the gear volume fill factor, ρ is the mass density of the gear, m_{gt} is the total mass of all the gear sets.

3.6 Track model

3.6.1 Curvature of the track

The XY coordinates of the Nurburgring circuit is presented in Figure 3.15. These data can be obtained with GPS, or can be extracted and converted from the commercial or open source map.

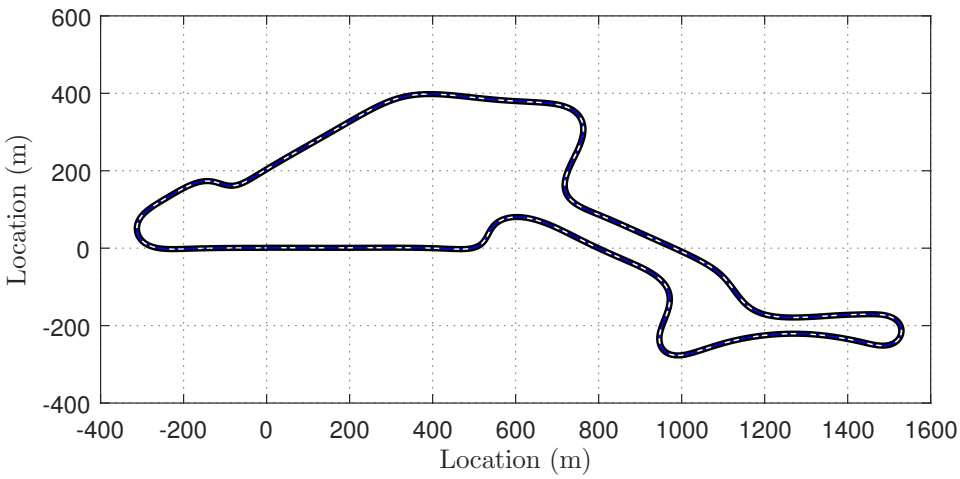

Figure 3.15: *The XY coordinates of the Nurburgring circuit*

The curvature of the track can be calculated with the given X-Y coordinates with Equation (3.77). The track can be described by its curvature and arc length in a curvilinear coordinates system as it is presented in Figure 3.16, the origin position is [0, 0].

$$C = \frac{dx \cdot ddy - ddx \cdot dy}{(\sqrt{dx^2 + dy^2})^{\frac{3}{2}}} \tag{3.77}$$

where dx, ddx, dy, ddy are the first and second order gradients of the X-Y coordinates respectively.

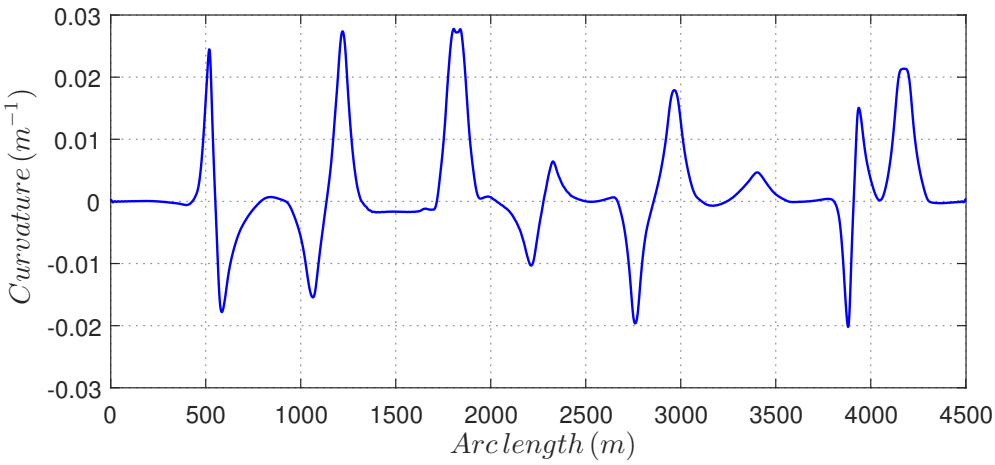

Figure 3.16: *Curvature of the track*

3.6.2 Path following model

In curvilinear coordinates system, the vehicle's position on the track can be described by its traveled distance s, its normal distance to the reference trajectory n, and its orientation angle θ at the current traveled distance. As it is shown in Figure 3.17. The orientation angle is denoted as:

$$\theta = \psi - \chi. \tag{3.78}$$

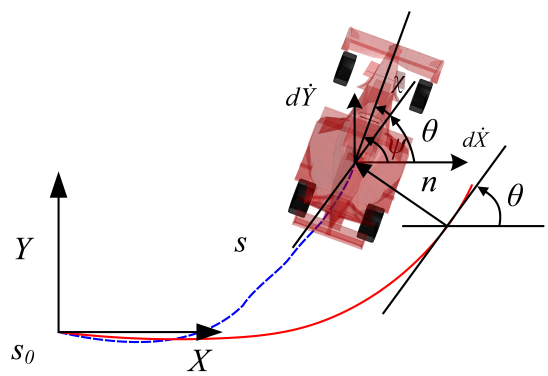

Figure 3.17: *Vehicle position in curvilinear coordinates system*

With Equation (3.80) and Equation (3.79), the derivative of the traveled distance $\dot{s}$ can be denoted as Equation (3.81) with the absolute longitudinal and lateral velocity of

the vehicle in global reference frame.

$$\dot{s} - n\dot{\theta} = d\dot{X}\cos\theta + d\dot{Y}\sin\theta \tag{3.79}$$

where $\dot{\theta}$ can be expressed as:

$$\dot{\theta} = C\dot{s} \tag{3.80}$$

the derivative of s is derived as:

$$\dot{s} = \frac{d\dot{X}\cos\theta + d\dot{Y}\sin\theta}{1 - nC} \tag{3.81}$$

The derivative of the normal distance $\dot{n}$ can be denoted as Equation (3.82).

$$\dot{n} = d\dot{X}sin\theta - d\dot{Y}cos\theta \tag{3.82}$$

The derivative of χ can be denoted as:

$$\dot{\chi} = \dot{\psi} - C\dot{s} \tag{3.83}$$

Finally, there are 31 states variables of the 14-DOF model and trajectory tracking information, the states vector is given as:

$$\begin{aligned} X_{in} = \{&\dot{X}_{A,b}, \dot{Y}_{A,b}, \dot{z}, \dot{\varphi}, \dot{\phi}, \dot{\psi}, \dot{z}_{ufr}, \dot{z}_{ufl}, \dot{z}_{urr}, \dot{z}_{url}, \dot{\theta}_{ufr}, \dot{\theta}_{ufl}, \dot{\theta}_{urr}, \dot{\theta}_{url}, \\ &X_{A,b}, Y_{A,b}, z, \varphi, \phi, \psi, z_{ufr}, z_{ufl}, z_{urr}, z_{url}, \theta_{ufr}, \theta_{ufl}, \theta_{urr}, \theta_{url}, s, n, \chi\} \end{aligned} \tag{3.84}$$

3.7 Driver model

The optimal design and control problems are solved by transcribing the original continuous problem into finite dimensional NLP problem. For the NLP problem, all of the variables, and constraints should have reasonable and valid boundaries, besides, the initial guess also is very important to achieve a feasible solution or a fast converge speed. In order to estimate the values of these boundaries and obtain a initial guess of the transcribed NLP problem, it is necessary to develop a diver model to control the race car move on the given track following the defined trajectory line. The driver model developed in this section is composed by the longitudinal control and lateral control.

3.7.1 longitudinal control

The purpose of the longitudinal control is to let the race car following a given reference longitudinal velocity which is based on the PID control logic and is presented in Equation (3.85).

$$T_d(n) = K_p\left\{e(n) + \frac{T}{T_i}\sum_{i=0}^{n} e(i) + \frac{T_d}{T}[e(n) - e(n-1)]\right\} \tag{3.85}$$

where T_d is the desired driving torque, n is the time step, e is the tracking error of longitudinal velocity , K_p is the proportional gain, T_i is the integral time, and T_d is the derivative time.

3.7.2 Lateral control

The lateral control aims at generating the appropriate steering angle, in order to control the vehicle following the reference track. Considering the delayed response of the vehicle, instead of using only proportional control, a lateral controller based on PD control presented as is developed to track the given reference line accurately.

$$\delta(n) = K_{p1}e_1(n) + K_{d1}(e_1(n) - e_1(n-1)) + K_{p2}e_2(n) + K_{d2}(e_2(n) - e_2(n-1)) \quad (3.86)$$

where K_{p1}, K_{p2} are the proportional gains, K_{d1}, K_{d2} are the derivative terms, e_1, e_1 are the normal distance and yaw angle tracking errors denoted as:

$$e_1 = n,\ e_2 = \psi_{ref} - \psi. \quad (3.87)$$

Figure 3.18 presents the path following results of lateral preview control with constant longitudinal speed $V_x = 75km/h$.

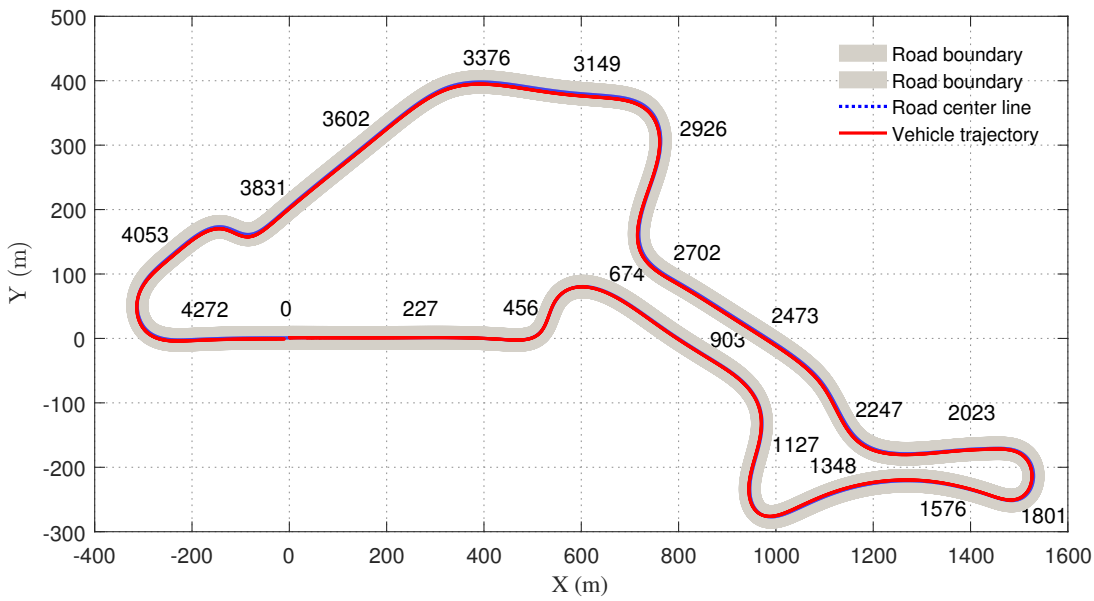

Figure 3.18: *Path following at speed $V_x = 75km/h$*

3.8 Validation

The 14-DOF model developed in MATLAB is validated with the swept sine steering and step steering manoeuvres based on a Formula 3 chassis. The simulation results are compared with the commercial software VI-CarRealTime.

3.8.1 Step steer

The step steer input of the step steer manoeuvre starts from the 2s second and ends at 3s with a steering amplitude of 30°. The comparison of the wheel steering angles on the ground, normal loads acted on tires, side slip angles, camber angles, spring

forces, damping forces, tire lateral forces, yaw rate, lateral acceleration and roll angle are demonstrated in Figure 3.19 to Figure 3.28, respectively.

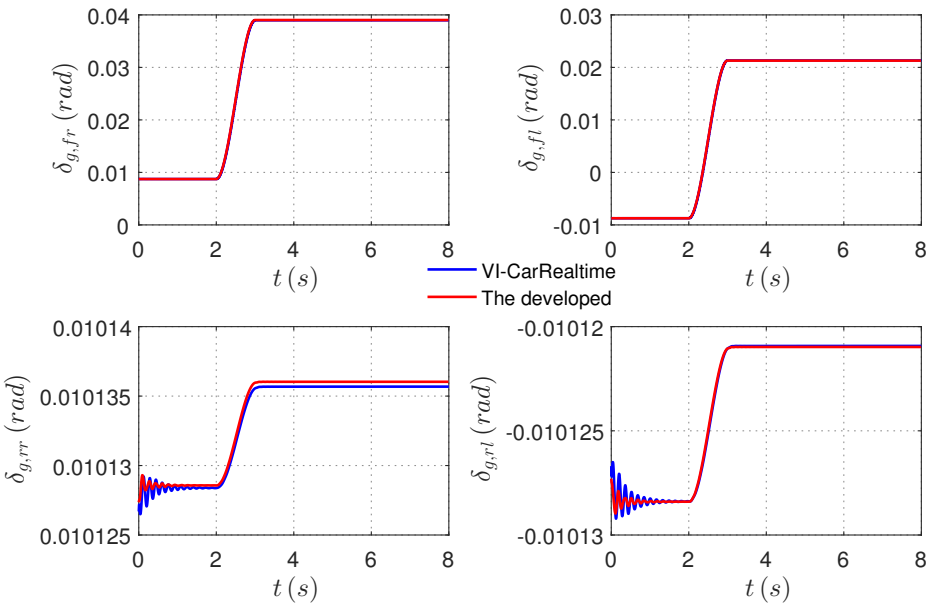

Figure 3.19: *Step steer manoeuvre: Comparison of wheel steering angles on the ground*

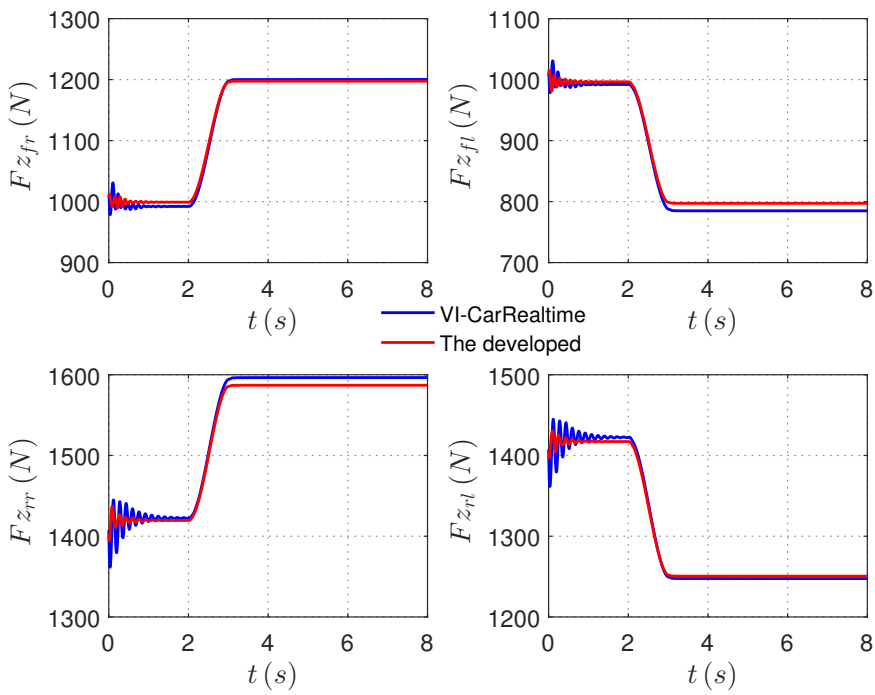

Figure 3.20: *Step steer manoeuvre: Comparison of normal loads acted on tires*

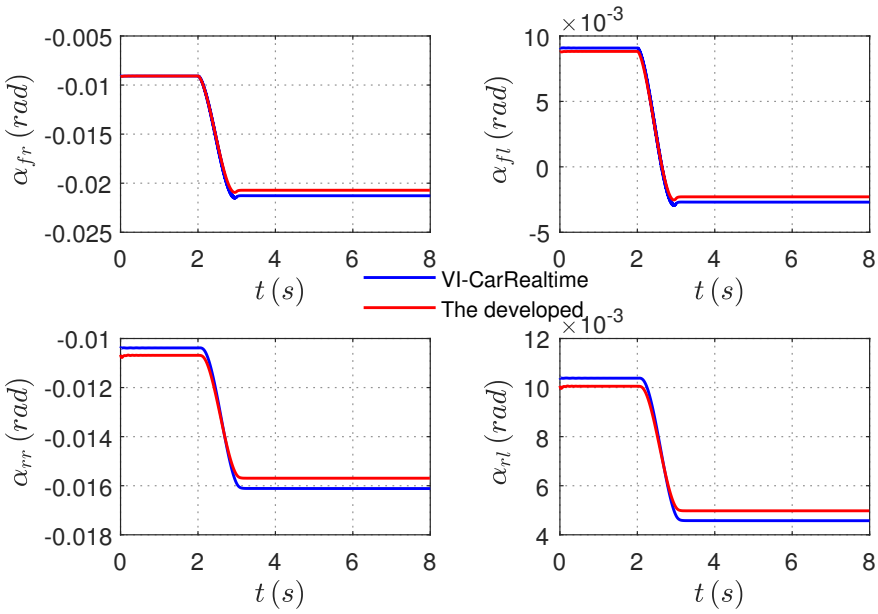

Figure 3.21: *Step steer manoeuvre: Comparison of side slip angles*

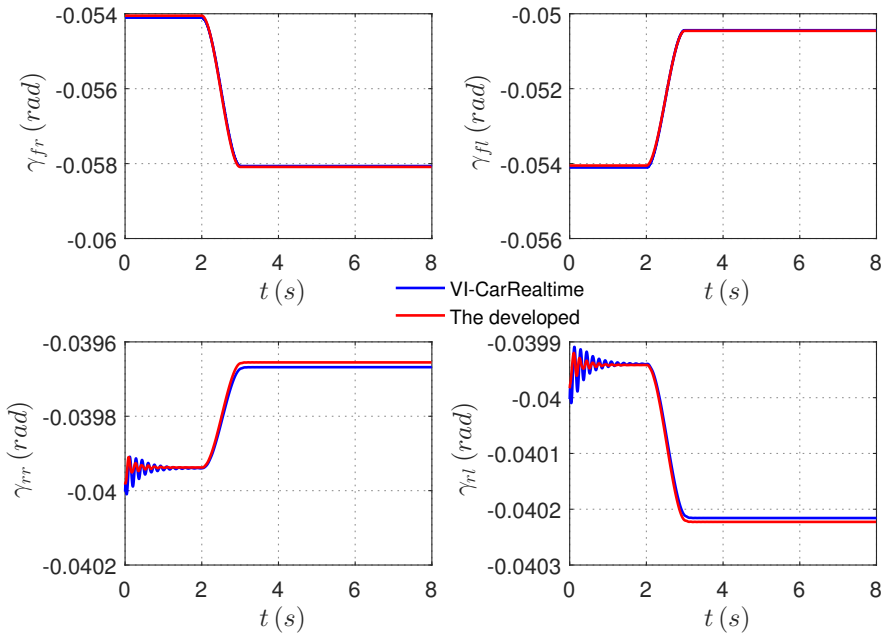

Figure 3.22: *Step steer manoeuvre: Comparison of camber angles*

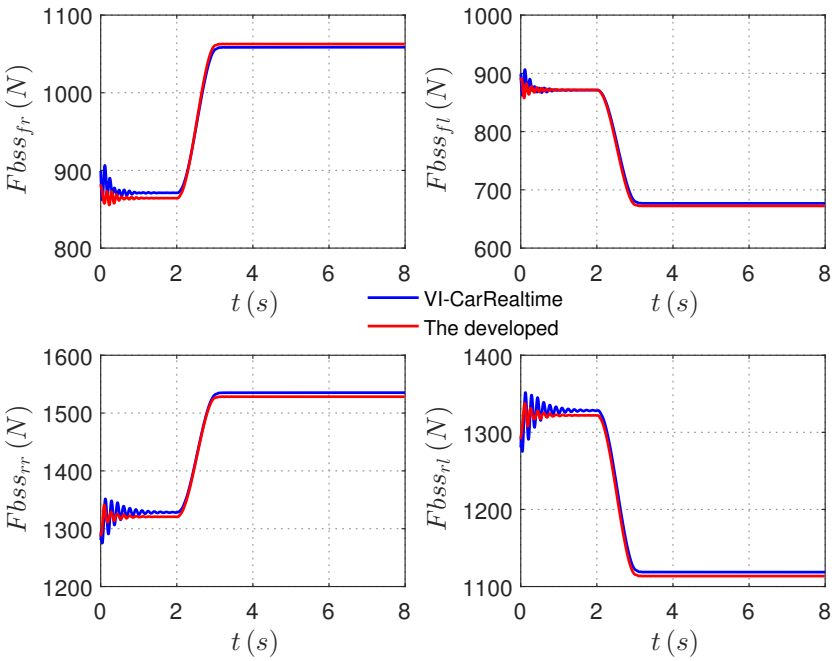

Figure 3.23: *Step steer manoeuvre: Comparison of spring forces*

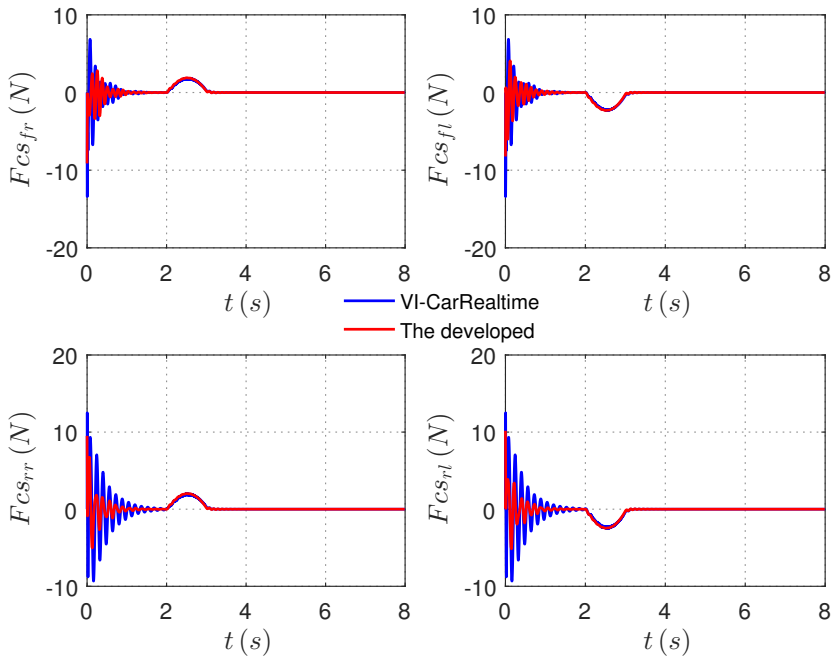

Figure 3.24: *Step steer manoeuvre: Comparison of damping forces*

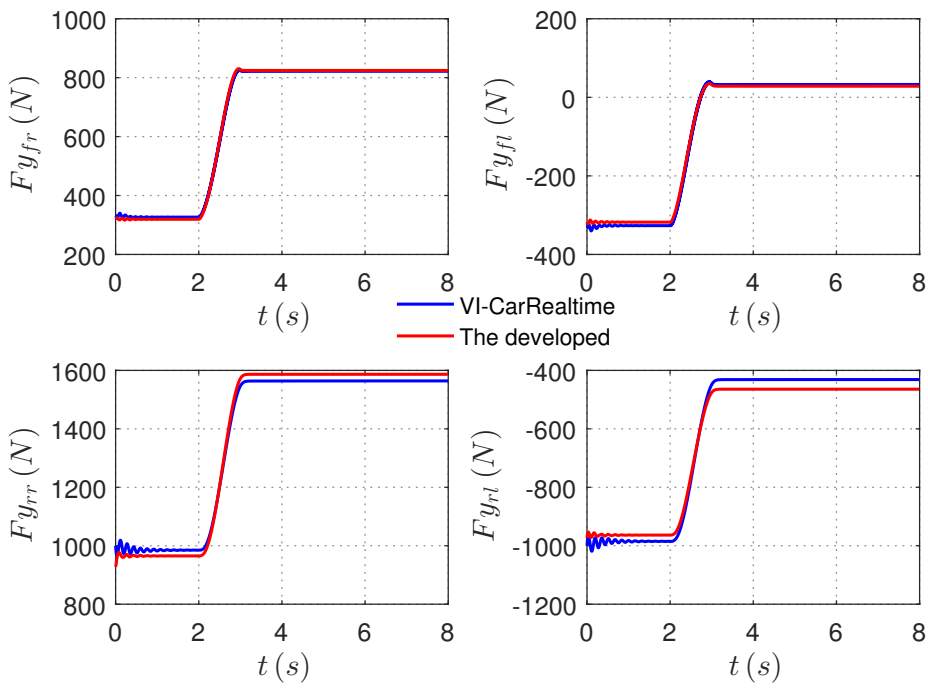

Figure 3.25: *Step steer manoeuvre: Comparison of tire lateral forces*

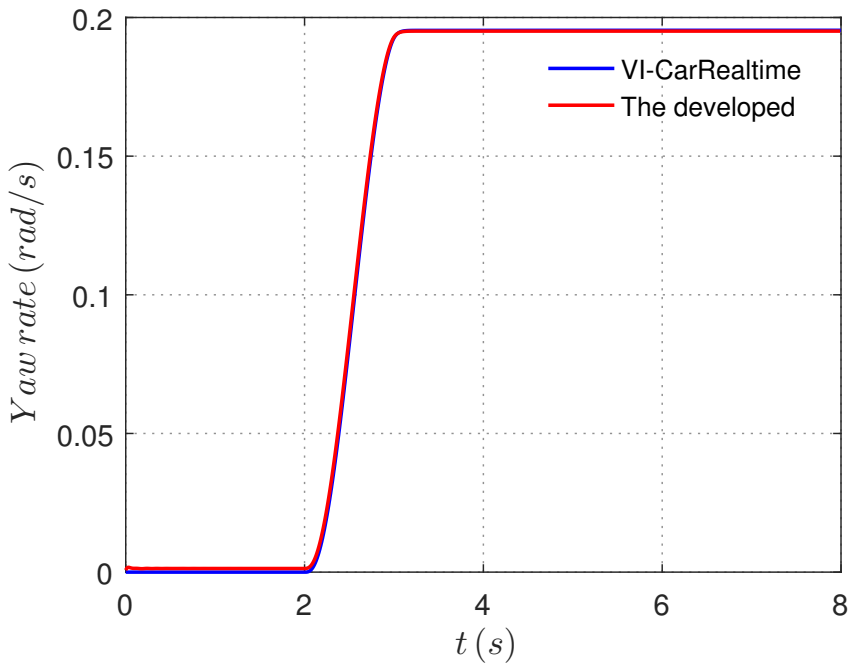

Figure 3.26: *Step steer manoeuvre: Comparison of yaw rate*

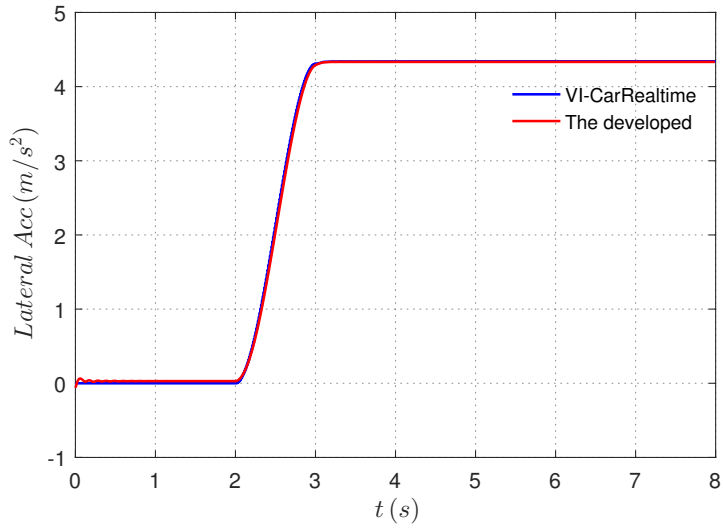

Figure 3.27: *Step steer manoeuvre: Comparison of lateral acceleration*

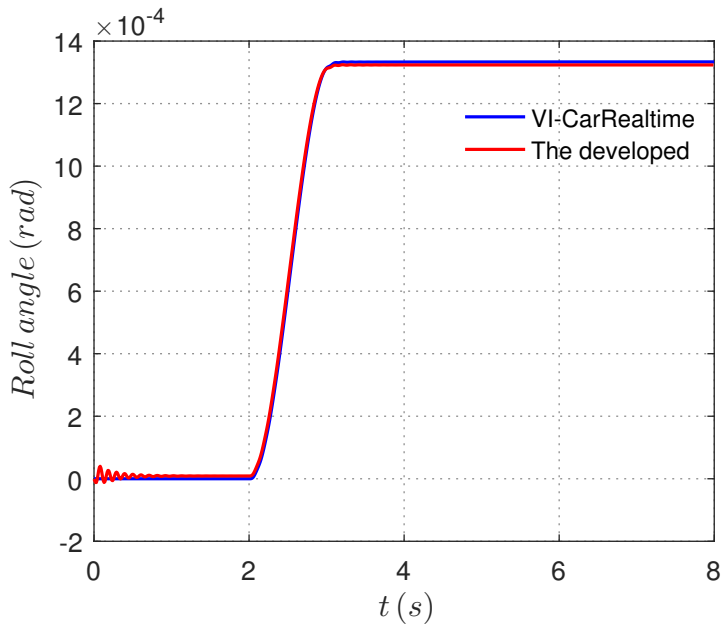

Figure 3.28: *Step steer manoeuvre: Comparison of roll angle*

As demonstrated in Figure 3.19 to Figure 3.28, the simulation results in the step steer manoeuvre of the developed 14-DOF vehicle model are highly consistent with VI-CarRealTime.

3.8.2 Swept sine steer

The swept sine steer manoeuvre is started at 2s with a longitudinal speed 80 km/h, a steering amplitude is 30° and a steering frequency from 1 Hz to 8 Hz. The comparison of the wheel steering angles on the ground, normal loads acted on tires, side slip angles, camber angles, spring forces, damping forces, tire lateral forces, yaw rate, lateral acceleration and roll angle in swept sine steer manoeuvre are demonstrated in Figure 3.29 to Figure 3.38, respectively. As demonstrated in the Figure 3.29 to Figure 3.38, the simulation results in the swept sine steer manoeuvre of the developed 14-DOF vehicle model are highly consistent with the ones obtained in VI-CarRealTime.

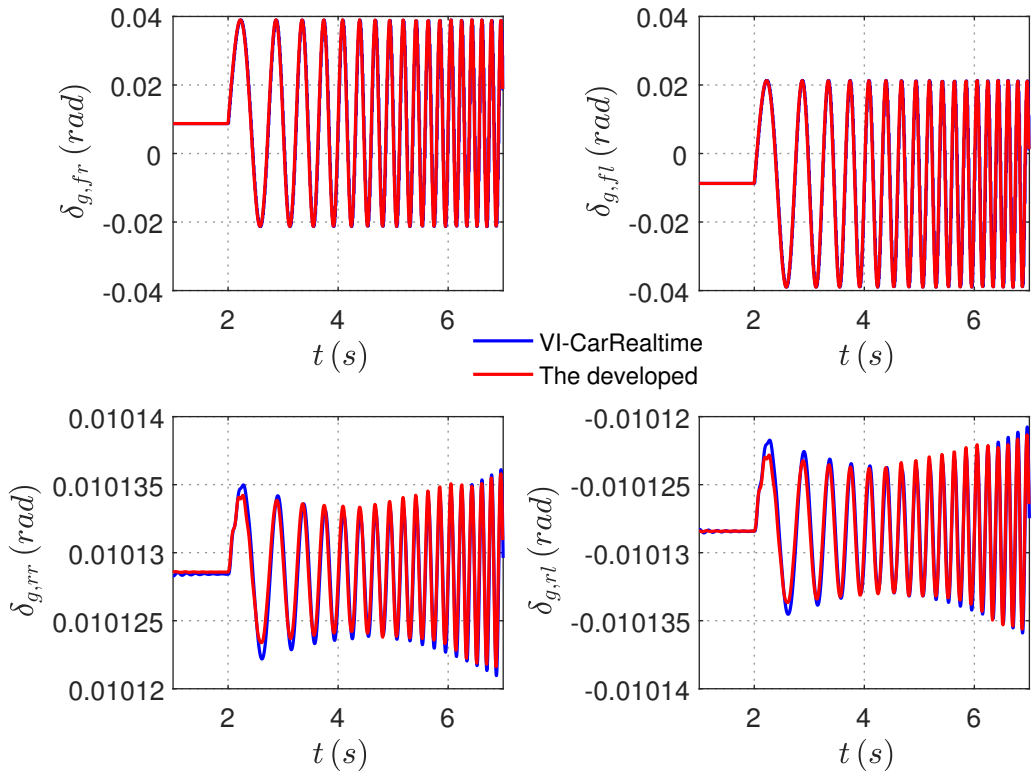

Figure 3.29: *Swept sine steer manoeuvre: Comparison of wheel steering angles on the ground*

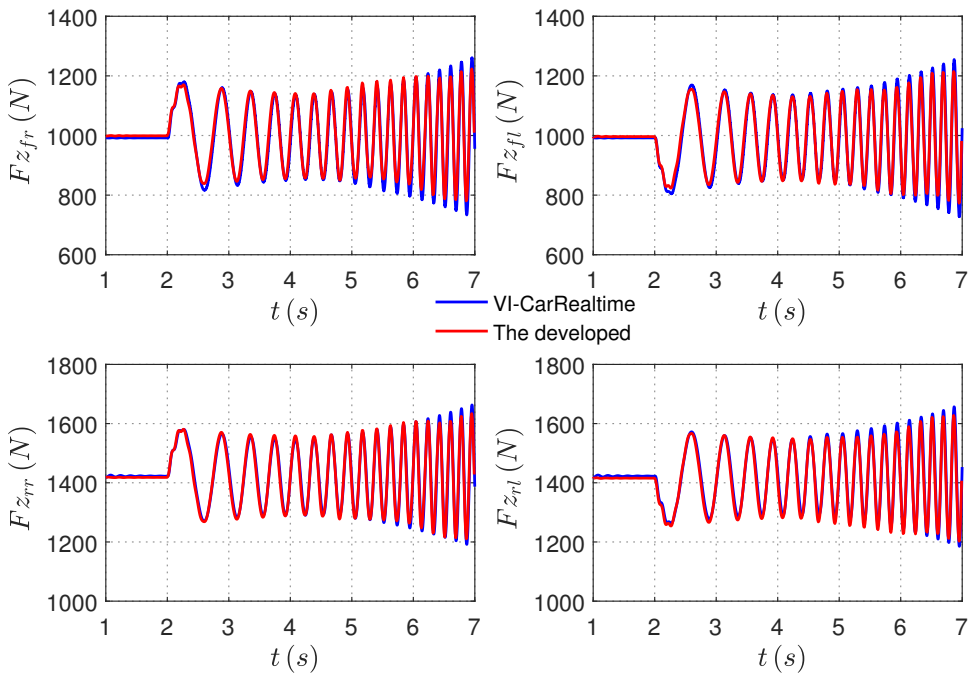

Figure 3.30: *Swept sine steer manoeuvre: Comparison of normal loads acted on tires*

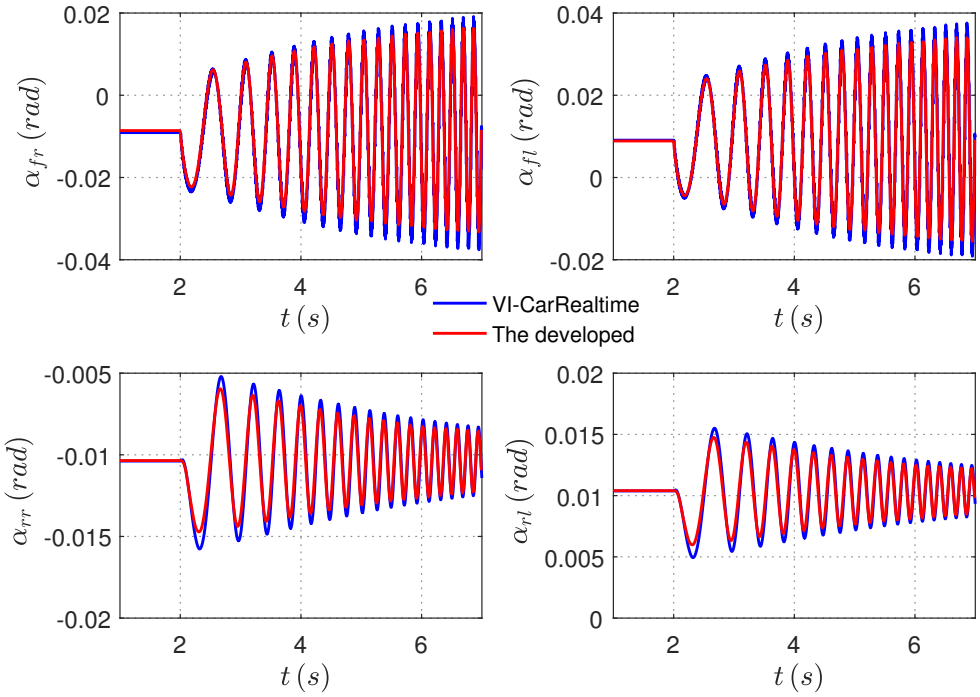

Figure 3.31: *Swept sine steer manoeuvre: Comparison of side slip angles*

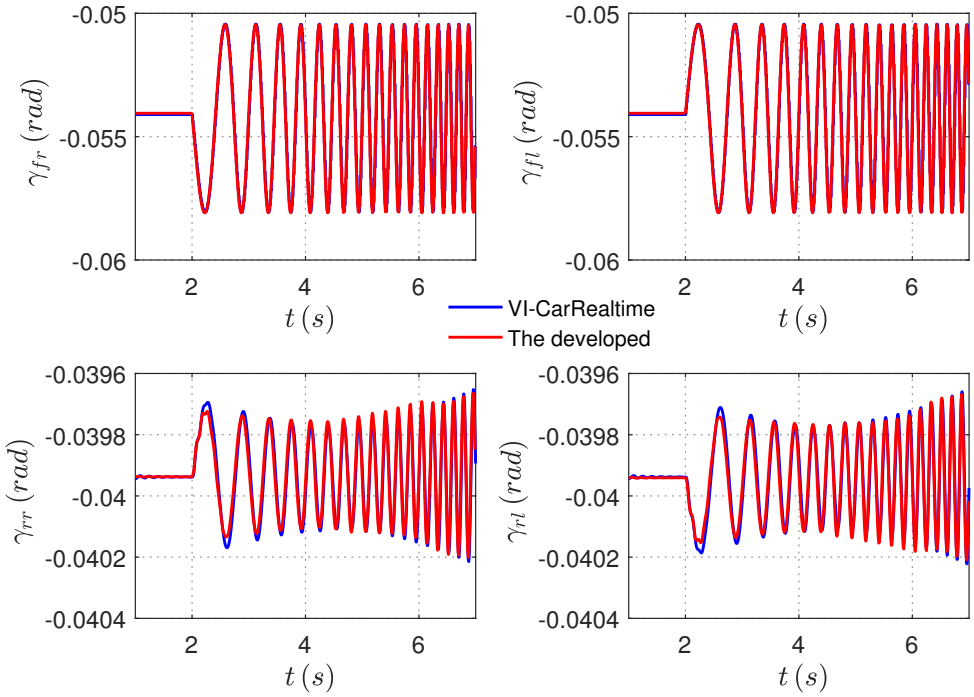

Figure 3.32: *Swept sine steer manoeuvre: Comparison of camber angles*

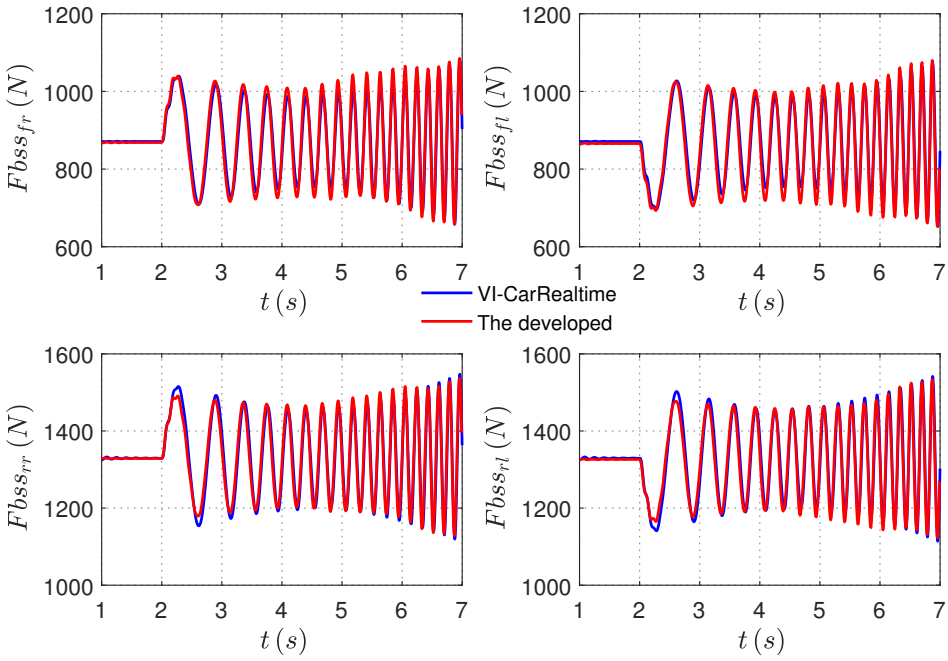

Figure 3.33: *Swept sine steer manoeuvre: Comparison of spring forces*

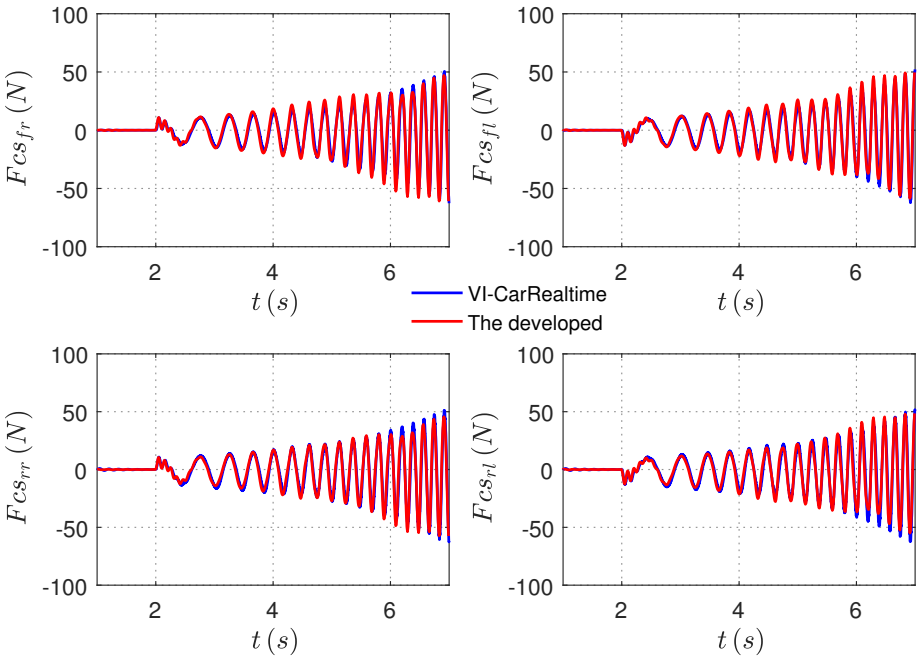

Figure 3.34: *Swept sine steer manoeuvre: Comparison of damping forces*

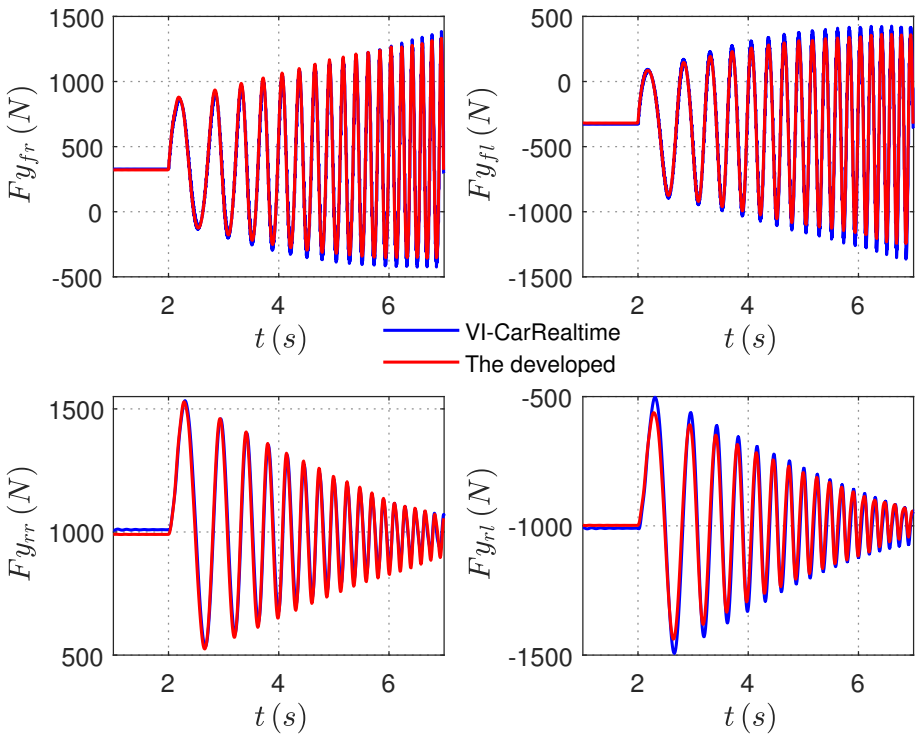

Figure 3.35: *Swept sine steer manoeuvre: Comparison of tire lateral forces*

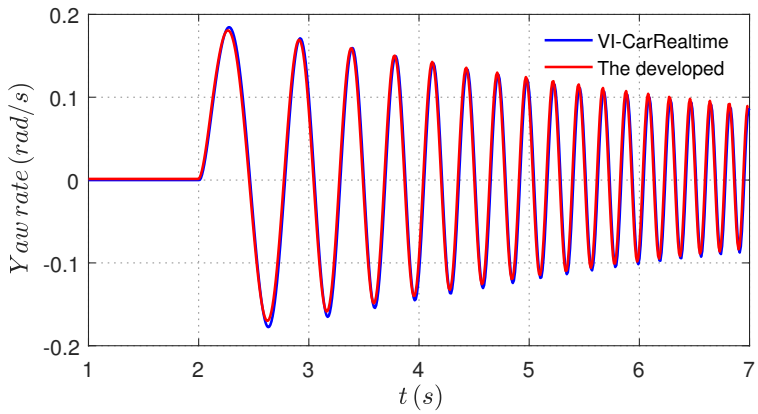

Figure 3.36: *Swept sine steer manoeuvre: Comparison of yaw rate*

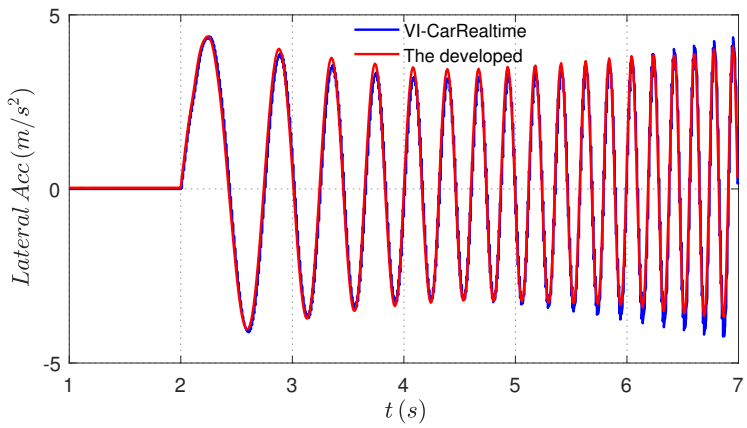

Figure 3.37: *Swept sine steer manoeuvre: Comparison of lateral acceleration*

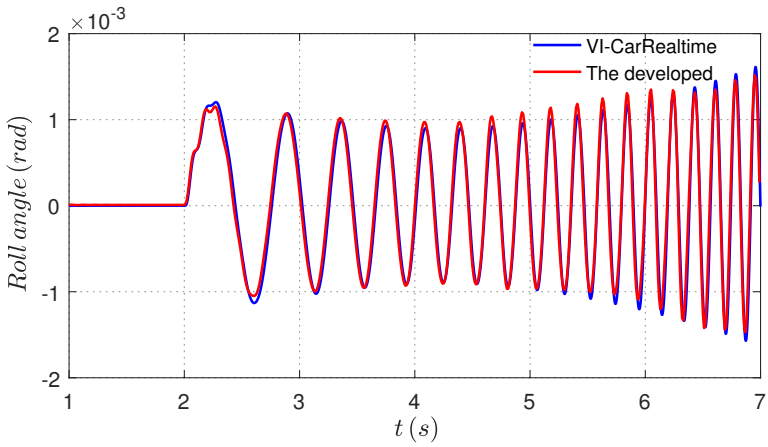

Figure 3.38: *Swept sine steer manoeuvre: Comparison of roll angle*

CHAPTER 4

Development of a Software Package for General Dynamic Optimal Control Problems

4.1 Introduction

A MATLAB software package for General DYNamic OPTimal control problems abbreviated as GDYNOPT is developed in this work to solve the formulated optimal powertrain design and control problems in Chapter 5. Figure 4.1 presents the sketch of the developed MATLAB software package. GDYNOPT is implemented and flexible with interface to different transcription methods including both the local collocation and global collocation approaches, differential methods including forward, central, complex step and analytical differential method based on MATLAB symbolic toolbox and with features of automatic scaling based on the proposed average norm of gradient and linear scaling, sparsity and parallel computing. GDYNOPT is mainly composed of three parts which are the user defined, transcription and NLP solver part. In the user defined part, the user should provide the initial guess, the bounds of the state, control, design variables and constraints. The Lagrangian term $\mathcal{L}$, Meyer term $\mathcal{M}$ of the cost function, the dynamic equation $\boldsymbol{f}$, the path constraints $\boldsymbol{g}$, the linkage constraints $\boldsymbol{\psi}$ and the boundary constraints $\boldsymbol{b}$ should also be provided in the developed function framework. Besides, the user needs to set some parameters related with the optimization, i.e., the transcription method m_{trans}, the differential method m_{diff}, the number of collocation nodes N_n, scaling setting *scal*, parameters to be plotted in the iterations *plot* , weights of the to be smoothed control parameters w_u, the meanings of other symbols are described in the following sections. In the transcription part, the user requested settings will be

realized and the formats of variables, functions will be transformed into the required ones by the NLP solver. The different transcription methods, differential methods, automatic scaling method, plot functions are all developed in the transcription part. The last part of the software package is the NLP solver and the employed solver in this work is IPOPT [34, 103].

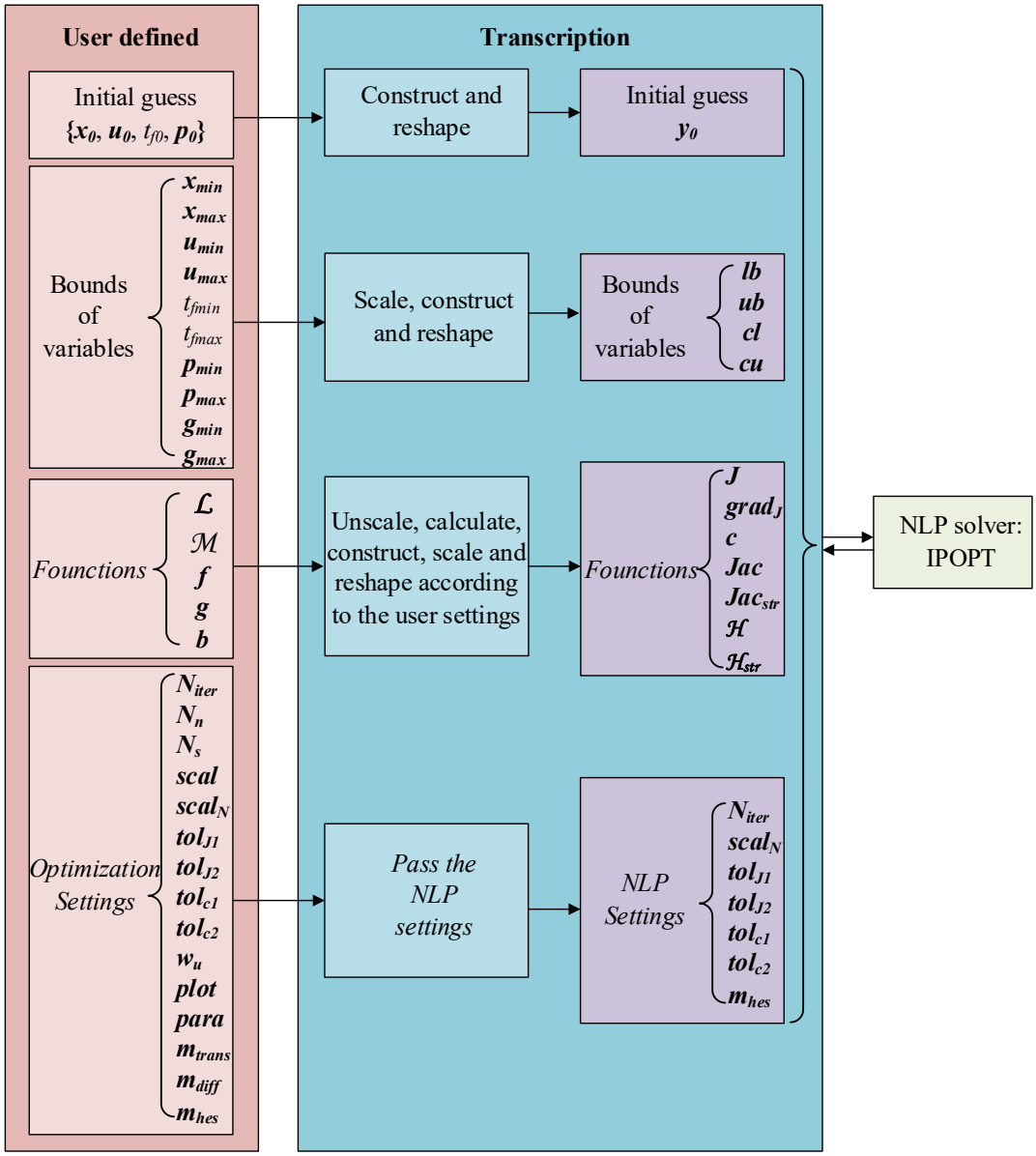

Figure 4.1: *Sketch of the developed MATLAB software GDYNOPT*

4.2 User settings of GDYNOPT

To use this software package, the user need to supply the initial guess of the state $\boldsymbol{x}_0$, control $\boldsymbol{u}_0$, terminal time t_{f0} and the design variables $\boldsymbol{p}_0$ with the following dimensions:

$$\begin{cases} \boldsymbol{x}_0 & \in \mathbb{R}^{N_{x,usr} \times n_x} \\ \boldsymbol{u}_0 & \in \mathbb{R}^{N_{u,usr} \times n_u} \\ \boldsymbol{p}_0 & \in \mathbb{R}^{1 \times n_p} \\ \boldsymbol{t} & \in \mathbb{R}^{N_{x,usr} \times 1} \\ t_{f0} & \in \mathbb{R}^{n_{t_f}} \end{cases} \tag{4.1}$$

where $N_{x,usr}$ is the number of nodes of the user provided state variables, $N_{u,usr}$ is the number of nodes of the user provided control variables, n_x is the number of the state variables of the dynamic equations, n_u is the number of control variables, n_{t_f} is 1 if the terminal time is also the parameters to be optimized otherwise it is 0, n_p is the number of design parameters, $\mathbb{R}^{i \times j}$ means a 2-dimensional matrix with i rows and j columns. For the global transcription methods, the initial value of the starting and terminal time of each phase will be generated by the GDYNOPT.

The dimension of the lower and upper bounds of the state $\boldsymbol{x}$, control $\boldsymbol{u}$, terminal time t_f , design variables $\boldsymbol{p}$ and the inequality constraints $\boldsymbol{g}$ are given as:

$$\begin{cases} \boldsymbol{x}_{min} & \in \mathbb{R}^{1 \times n_x} \\ \boldsymbol{x}_{max} & \in \mathbb{R}^{1 \times n_x} \\ \boldsymbol{u}_{min} & \in \mathbb{R}^{1 \times n_u} \\ \boldsymbol{u}_{max} & \in \mathbb{R}^{1 \times n_u} \\ \boldsymbol{p}_{min} & \in \mathbb{R}^{1 \times n_p} \\ \boldsymbol{p}_{max} & \in \mathbb{R}^{1 \times n_p} \\ t_{fmin} & \in \mathbb{R}^{n_{t_f}} \\ \boldsymbol{g}_{min} & \in \mathbb{R}^{1 \times n_g} \\ \boldsymbol{g}_{max} & \in \mathbb{R}^{1 \times n_g} \end{cases} \tag{4.2}$$

where n_g is the number of path constraint variables.

The user defined functions are the Lagrangian term $\mathcal{L}$, Meyer term $\mathcal{M}$ of the objective function, the first order dynamic constraint function $\boldsymbol{f}$, the path constraint function $\boldsymbol{g}$ and the boundary constraint function $\boldsymbol{b}$, with the following input and output dimensions:

$$\begin{cases} \mathcal{L} & = Lagrange(x, u, t, p), & \{\mathbb{R}^{N_n} \times \mathbb{R}^{n_x}, \mathbb{R}^{N_n} \times \mathbb{R}^{n_u}, \mathbb{R}^{N_n} \times \mathbb{R}, \mathbb{R} \times \mathbb{R}^{n_p}\} \longrightarrow \mathbb{R}^{N_n \times 1} \\ \mathcal{M} & = Meyer(x_0, t_0, x_f, t_f, p), & \{\mathbb{R} \times \mathbb{R}^{n_x}, \mathbb{R}, \mathbb{R} \times \mathbb{R}^{n_x}, \mathbb{R}, \mathbb{R} \times \mathbb{R}^{n_p}\} \longrightarrow \mathbb{R} \\ \boldsymbol{f} & = dyn(x, u, t, p), & \{\mathbb{R}^{N_n} \times \mathbb{R}^{n_x}, \mathbb{R}^{N_n} \times \mathbb{R}^{n_u}, \mathbb{R}^{N_n} \times \mathbb{R}, \mathbb{R} \times \mathbb{R}^{n_p}\} \longrightarrow \mathbb{R}^{N_n \times n_x} \\ \boldsymbol{g} & = path(x, u, t, p), & \{\mathbb{R}^{N_n} \times \mathbb{R}^{n_x}, \mathbb{R}^{N_n} \times \mathbb{R}^{n_u}, \mathbb{R}^{N_n} \times \mathbb{R}, \mathbb{R} \times \mathbb{R}^{n_p}\} \longrightarrow \mathbb{R}^{N_n \times n_g} \\ \boldsymbol{b} & = bound(x_0, t_0, x_f, t_f, p), & \{\mathbb{R} \times \mathbb{R}^{n_x}, \mathbb{R}, \mathbb{R} \times \mathbb{R}^{n_x}, \mathbb{R}, \mathbb{R} \times \mathbb{R}^{n_p}\} \longrightarrow \mathbb{R}^{2 \times n_b} \end{cases} \tag{4.3}$$

where N_n is the number of the user set nodes. For the global transcription methods, the linkage constraint ψ of each neighboring phases will be generated by the GDYNOPT.

In oder to improve the computing efficiency the author provided functions should support vector or matrix operations and strictly obey the above requirements on the dimensions of the functions' inputs and outputs.

The other parameters that the user should set are list as Table 4.1:

Table 4.1: *Optimization settings defined by the user*

Parameters	Meanings and settings
N_{iter}	The maximum iterations, the NLP solver will terminate when reach N_{iter}
N_n	The total number of collocation nodes determining the discretized step
N_s	The total number of intervals when multiple phase control activated
$scal$	Scale settings of the implemented automatic scaling, active: $scal = 1$, inactive: $scal = 0$
$scal_N$	Scale settings of the provided by the NLP solver, active: $scal_N = 1$, inactive: $scal_N = 0$
tol_{J1}	The absolute tolerance of the objective function
tol_{J2}	The acceptable tolerance of the objective function
tol_{c1}	The absolute tolerance of the constraint function
tol_{c2}	The acceptable tolerance of the constraint function
w_u	The weights of the controls to be smoothed, $w_u \in \mathbb{R}^{1\times n_u}$
$plot$	The debug index, 1: means to plot and print the desired variables in command window, 0: inactive
$para$	1: means to use parallel computing, 0: means inactive
m_{trans}	The transformation methods, choose one from { Hermite-Simpson, Trapezoidal, LGL and CGL }
m_{diff}	The differential methods, choose one from { Forward, Central, ComplexStep and Analytical }
m_{hes}	The methods to obtain Hessian, choose one from $\{Forward,\ Central,\ ComplexStep\ and\ Analytical\}$

With the above user defined information, the transformation software need to provide the initial guess, bounds of the variables and functions in the required formats of the NLP solvers, besides, pass some user defined setting to the NLP solvers directly. The manners to provide these variables and functions vary with the transformation methods defined by the users, which are presented in the following sections.

4.3 Local collocation approach

4.3.1 NLP variables

In Trapezoidal approach and the Hermite-Simpson approach when the state and control variables locating at the middle of the each sample interval are not parameters to be optimized, the continuous state and control variables can be discretized over the whole time interval in to N_n nodes with the linear interpretation method. The new independent time samples are obtained by generating a linearly spaced vector on $[t_0,\ t_f]$ with length Nn. The obtained variables are given in Equation (4.4).

$$\begin{cases} \boldsymbol{x}_0 & \in \mathbb{R}^{N_{x,usr}\times n_x} \\ \boldsymbol{u}_0 & \in \mathbb{R}^{N_{u,usr}\times n_u} \\ \boldsymbol{p}_0 & \in \mathbb{R}^{1\times n_p} \\ \boldsymbol{t} & \in \mathbb{R}^{N_{x,usr}\times 1} \\ t_{f0} & \in \mathbb{R}^{n_{t_f}} \end{cases} \longrightarrow \begin{cases} \boldsymbol{x}'_0 & \in \mathbb{R}^{N_n\times n_x} \\ \boldsymbol{u}'_0 & \in \mathbb{R}^{N_n\times n_u} \\ \boldsymbol{p}'_0 & \in \mathbb{R}^{1\times n_p} \\ \boldsymbol{t}' & \in \mathbb{R}^{N_n\times 1} \\ t'_{f0} & \in \mathbb{R}^{n_{t_f}} \end{cases} \tag{4.4}$$

Subsequently, the discretized state, control variables at each node together with the static parameters of the continuous optimal design and control problem can be reconstructed as a column vector of NLP variables:

$$\boldsymbol{y} = \left[\ \boldsymbol{x}_1, \boldsymbol{u}, \boldsymbol{x}_2, \boldsymbol{u}_2, \ldots, \boldsymbol{x}_{N_n}, \boldsymbol{u}_{N_n}, \boldsymbol{t}_f, \boldsymbol{p} \ \right]^T \tag{4.5}$$

where $\boldsymbol{x}_i$ is the row vector of the state variables at node i with dimension $1 \times n_x$, $\boldsymbol{u}_i$ is the row vector of the control variables at node i with dimension $1 \times n_u$, t_f is the final time, $\boldsymbol{p}$ is the row vector of the design variables with dimension $1 \times n_p$. The dimension of $\boldsymbol{y}$ is $N_n(n_u + n_x) + n_{t_f} + n_p$.The initial guess of the NLP variables $\boldsymbol{y}_0$ is obtained by reconstructing the resulted variables in Equation (4.4).

For the Hermite-Simpson approach, when the control variables $\bar{\boldsymbol{u}}$ between the center of each sample interval are chosen as the parameters to be optimized, the NLP variables are given as:

$$\boldsymbol{y} = \left[\ \boldsymbol{x}_1, \boldsymbol{u}, \bar{\boldsymbol{u}}_1, \boldsymbol{x}_2, \boldsymbol{u}_2, \bar{\boldsymbol{u}}_2, \ldots, \bar{\boldsymbol{u}}_{N_n-1}, \boldsymbol{x}_{N_n}, \boldsymbol{u}_{N_n}, \boldsymbol{t}_f, \boldsymbol{p} \ \right]^T \tag{4.6}$$

In this case, the user should supply an initial guess of $\bar{\boldsymbol{u}}_{N_i}^T \in \mathbb{R}^{(N_n-1)\times n_u}$. The dimension of the NLP variables will be $N_n(2n_u + n_x) - n_u + n_{t_f} + n_p$.

When both the state $\bar{\boldsymbol{x}}$ and control $\bar{\boldsymbol{u}}$ variables between the center of each sample interval are selected as the to be optimized parameters, the NLP variables will become into:

$$\boldsymbol{y} = \left[\ \boldsymbol{x}_1, \boldsymbol{u}, \bar{\boldsymbol{x}}_1, \bar{\boldsymbol{u}}_1, \boldsymbol{x}_2, \boldsymbol{u}_2, \bar{\boldsymbol{x}}_2, \bar{\boldsymbol{u}}_2, \ldots, \bar{\boldsymbol{x}}_{N_n-1}, \bar{\boldsymbol{u}}_{N_n-1}, \boldsymbol{x}_{N_n}, \boldsymbol{u}_{N_n}, \boldsymbol{t}_f, \boldsymbol{p} \ \right]^T \tag{4.7}$$

In this case, the user should supply both the initial guess of $\bar{\boldsymbol{x}}_{N_i}^T \in \mathbb{R}^{(N_n-1)\times n_x}$ and $\bar{\boldsymbol{u}}_{N_i}^T \in \mathbb{R}^{(N_n-1)\times n_u}$. The dimension of the NLP variables will be $2N_n(n_u + n_x) - n_u - n_x + n_{t_f} + n_p$.

The lower bounds and upper bounds of the NLP variables given in Equation (4.2) should be reconstructed and consistent with the constructed NLP variables.

4.3.2 Constraints

1) Defect constraints of the Hermite-Simpson approach

Based on the 3rd order Hermite interpolation, the state variable $\bar{\boldsymbol{x}}$ locating at the middle of $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k+1}$, and its derivative $\dot{\bar{\boldsymbol{x}}}$ can be separately denoted as:

$$\bar{\boldsymbol{x}} = \frac{1}{2}(\boldsymbol{x}_k + \boldsymbol{x}_{k+1}) + \frac{1}{8}h(\boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{p}) - \boldsymbol{f}(\boldsymbol{x}_{k+1}, \boldsymbol{u}_{k+1}, \boldsymbol{p})) \tag{4.8}$$

$$\dot{\bar{\boldsymbol{x}}} = \frac{3(\boldsymbol{x}_k + \boldsymbol{x}_{k+1})}{2h} - \frac{\boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{p}) + \boldsymbol{f}(\boldsymbol{x}_{k+1}, \boldsymbol{u}_{k+1}, \boldsymbol{p})}{4} \tag{4.9}$$

The defects constraint $\boldsymbol{\zeta}_k = \dot{\bar{\boldsymbol{x}}} - \boldsymbol{f}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \boldsymbol{p})$ at node $\bar{\boldsymbol{x}}$ based on the Hermite-Simpson approach is finally derived as:

$$\boldsymbol{\zeta}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k - \frac{1}{6}h(\boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{p}) + 4\boldsymbol{f}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \boldsymbol{p}) + \boldsymbol{f}(\boldsymbol{x}_{k+1}, \boldsymbol{u}_{k+1}, \boldsymbol{p})), \boldsymbol{\zeta} \in \mathbb{R}^{(N_n-1)\times n_x} \tag{4.10}$$

In this work, the control variable $\bar{\boldsymbol{u}}$ located at the middle of the interval $[\boldsymbol{x}_k, \ \boldsymbol{x}_{k+1}]$ is simply implemented with the linear interpolation method, which is given by:

$$\bar{\boldsymbol{u}} = \frac{\boldsymbol{u}_k + \boldsymbol{u}_{k+1}}{2} \tag{4.11}$$

There are two advantages of this approach, first, the dimension of the NLP variables are significantly reduced compared with Equation (4.6) and (4.7); second, it is more convenient to smooth the control variables by introducing penalty functions in the cost function and supplying the relevant gradients.

The NLP solver will determine the variables $[\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}_{k+1}, \boldsymbol{u}_{k+1}, t_f, \boldsymbol{p}]$ to drive $\boldsymbol{\zeta}_k$ towards zero based on the supplied Jacobian and Hessian information during the iteration process. Finally, the interpolate polynomials will be enforced to approximate the true dynamics accurately.

For the numerical computation, of course one can follow the Equations (4.8)-(4.10) step by step, however, in this case, it will introduce some repetitive computing which will slow down the optimization iterations. A better solution is to take the advantage of matrix operations in MATLAB. The state variable $\bar{\boldsymbol{x}}$, and control variable $\bar{\boldsymbol{u}}$ can be obtained with:

$$\bar{\boldsymbol{x}} = T_{s1}\boldsymbol{x} + \frac{h}{8}T_{s2}\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) \tag{4.12}$$

$$\bar{\boldsymbol{u}} = T_{s1}\boldsymbol{u} \tag{4.13}$$

where $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) \in \mathbb{R}^{N_n \times n_x}$ is the system dynamics denoted as

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{x}_1, \boldsymbol{u}_1, \boldsymbol{p}) \\ \vdots \\ \boldsymbol{f}(\boldsymbol{x}_{N_n}, \boldsymbol{u}_{N_n}, \boldsymbol{p}) \end{bmatrix} = \begin{bmatrix} f_1(\boldsymbol{x}_1, \boldsymbol{u}_1, \boldsymbol{p}) & \dots & f_{n_x}(\boldsymbol{x}_1, \boldsymbol{u}_1, \boldsymbol{p}) \\ \vdots & \ddots & \vdots \\ f_1(\boldsymbol{x}_{N_n}, \boldsymbol{u}_{N_n}, \boldsymbol{p}) & \dots & f_{n_x}(\boldsymbol{x}_{N_n}, \boldsymbol{u}_{N_n}, \boldsymbol{p}) \end{bmatrix}, \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) \in \mathbb{R}^{N_n \times n_x}, \tag{4.14}$$

the transformation matrix T_{s1} and T_{s2} are given as:

$$T_{s1} = \begin{bmatrix} 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 1 \end{bmatrix}, \ T_{s1} \in \mathbb{R}^{(N_n-1)\times N_n} \tag{4.15}$$

$$T_{s2} = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix}, \ T_{s2} \in \mathbb{R}^{(N_n-1)\times N_n} \tag{4.16}$$

The defect constraints $\boldsymbol{\zeta}$ in Equation (4.10) can be computed with the matrix operation:

$$\boldsymbol{\zeta} = T_{s2}\boldsymbol{x} - \frac{h}{6}(T_{s1}\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) + 4\boldsymbol{f}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \boldsymbol{p})), \ \boldsymbol{\zeta} \in \mathbb{R}^{(N_n-1)\times n_x} \tag{4.17}$$

2) Defect constraints of the Trapezoidal approach

The defect constraints $\boldsymbol{\zeta}_k$ of the Trapezoidal method are denoted as:

$$\boldsymbol{\zeta}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k - \frac{1}{2}h(\boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{p}) + \boldsymbol{f}(\boldsymbol{x}_{k+1}, \boldsymbol{u}_{k+1}, \boldsymbol{p})), \boldsymbol{\zeta} \in \mathbb{R}^{(N_n-1)\times n_x} \tag{4.18}$$

Again, the defect constraints matrix $\boldsymbol{\zeta} \in \mathbb{R}^{(N_n-1)\times n_x}$ can be calculated with a more compact way:

$$\boldsymbol{\zeta} = T_{s2}\boldsymbol{x} - \frac{h}{2}T_{s1}\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p}) \tag{4.19}$$

where the state variable $\boldsymbol{x} \in \mathbb{R}^{N_n\times n_x}$, the control variable $\boldsymbol{u} \in \mathbb{R}^{N_n\times n_u}$, T_{s1} and T_{s2} are the transformation matrix defined in the above paragraphs.

3) Path and boundary constraints

The path constraints $\boldsymbol{g} \in \mathbb{R}^{N_n\times n_g}$ are the functions of state, control, design and terminal time variables denoted ad Equation (4.20), while the boundary constraints $\boldsymbol{b} \in \mathbb{R}^{1\times n_b}$ are the functions of initial and final state variables.

$$\boldsymbol{g}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p}) = \begin{bmatrix} \boldsymbol{g}(\boldsymbol{x}_1,\boldsymbol{u}_1,\boldsymbol{p}) \\ \vdots \\ \boldsymbol{g}(\boldsymbol{x}_{N_n},\boldsymbol{u}_{N_n},\boldsymbol{p}) \end{bmatrix} = \begin{bmatrix} g_1(\boldsymbol{x}_1,\boldsymbol{u}_1,\boldsymbol{p}) & \dots & g_{n_g}(\boldsymbol{x}_1,\boldsymbol{u}_1,\boldsymbol{p}) \\ \vdots & \ddots & \vdots \\ g_1(\boldsymbol{x}_{N_n},\boldsymbol{u}_{N_n},\boldsymbol{p}) & \dots & g_{n_g}(\boldsymbol{x}_{N_n},\boldsymbol{u}_{N_n},\boldsymbol{p}) \end{bmatrix}, \boldsymbol{g}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p}) \in \mathbb{R}^{N_n\times n_g} \tag{4.20}$$

After the presented constraints are obtained, the NLP constraints are ready to be constructed as:

$$\begin{aligned} \boldsymbol{c}(\boldsymbol{y}) = \; & [\boldsymbol{\zeta}_{1,1},\dots,\boldsymbol{\zeta}_{1,n_x},\dots,\boldsymbol{\zeta}_{(N_n-1),1},\dots,\boldsymbol{\zeta}_{(N_n-1),n_x}, \\ & \boldsymbol{g}_{1,1},\dots,\boldsymbol{g}_{1,n_g},\dots,\boldsymbol{g}_{N_n,1},\dots,\boldsymbol{g}_{N_n,n_g}, \\ & \boldsymbol{b}_{1,1},\dots,\boldsymbol{b}_{1,n_b}]^T, \; \boldsymbol{c}(\boldsymbol{y}) \in \mathbb{R}^{((N_n-1)n_x+N_nn_g+n_b)\times 1} \end{aligned} \tag{4.21}$$

Both the lower and upper bounds of the defect constraints are zeros, while the bounds of path and boundary constraints are set by the user in Equation (4.2). The bounds of the NLP constraints are constructed consistently with Equation (4.21).

4.3.3 Jacobians

When the state and control located at the middle of each interval are not the to be optimized parameters in the Hermite-Simpson approach, the Jacobian patterns of the constraints are the same as the Trapezoidal approach, which are demonstrated in Figure 4.2.

As we can see, the Jacobian matrix $Jac \in \mathbb{R}^{((N_n-1)n_x+N_nn_g+n_b)\times(N_n(nx+n_u)+n_{tf}+np)}$ is composed by the derivatives of the defect constraints, path constraints and boundary constraints. The dimension of the full Jacobian matrix is huge, however, it is very sparse and we can take advantage of this feature to save a lot of memory.

1) Jacobians of the defect constraints

Considering the fact that the integral step $h = \dfrac{t_f - t_0}{N_n - 1}$ and $\boldsymbol{\zeta}$ is a function of $[\boldsymbol{x}_a,\ \boldsymbol{x}_b, \boldsymbol{u}_a,\ \boldsymbol{u}_b, t_f\ \boldsymbol{p}]$ in both the Hermite-Simpson and Trapezoidal approaches, the 3-dimensional Jacobian matrix $J_\zeta \in \mathbb{R}^{n_x\times(2(n_x+n_u)+n_{t_f}+n_p)\times(N_n-1)}$ of $\boldsymbol{\zeta} \in \mathbb{R}^{(N_n-1)\times n_x}$ to the

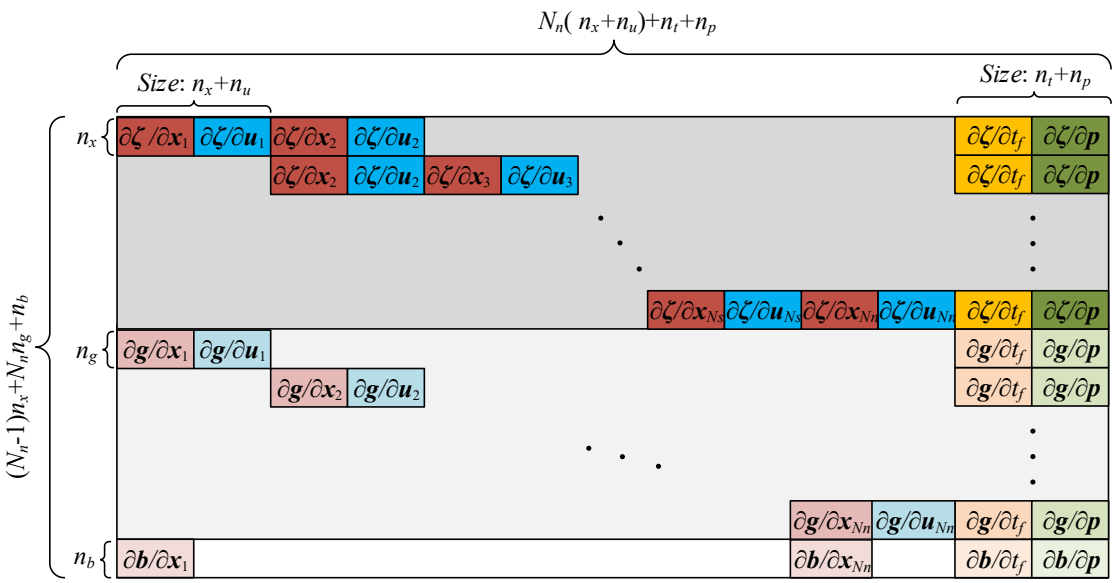

Figure 4.2: *Jacobian patterns of the constraints*

inputs can be given as:

$$J_\zeta = \begin{bmatrix} \frac{\partial \zeta_1}{\partial x_{a,1}}, \ldots, \frac{\partial \zeta_1}{\partial x_{a,n_x}}, \frac{\partial \zeta_1}{\partial u_{a,1}}, \ldots, \frac{\partial \zeta_1}{\partial u_{a,n_u}}, \frac{\partial \zeta_1}{\partial x_{b,1}}, \ldots, \frac{\partial \zeta_1}{\partial x_{b,n_x}}, \frac{\partial \zeta_1}{\partial u_{b,1}}, \ldots, \frac{\partial \zeta_1}{\partial u_{b,n_u}}, \frac{\partial \zeta_1}{\partial t_f}, \frac{\partial \zeta_1}{\partial b_1}, \ldots, \frac{\partial \zeta_1}{\partial b_{n_b}} \\ \frac{\partial \zeta_2}{\partial x_{a,1}}, \ldots, \frac{\partial \zeta_2}{\partial x_{a,n_x}}, \frac{\partial \zeta_2}{\partial u_{a,1}}, \ldots, \frac{\partial \zeta_2}{\partial u_{a,n_u}}, \frac{\partial \zeta_2}{\partial x_{b,1}}, \ldots, \frac{\partial \zeta_2}{\partial x_{b,n_x}}, \frac{\partial \zeta_2}{\partial u_{b,1}}, \ldots, \frac{\partial \zeta_2}{\partial u_{b,n_u}}, \frac{\partial \zeta_2}{\partial t_f}, \frac{\partial \zeta_2}{\partial b_1}, \ldots, \frac{\partial \zeta_2}{\partial b_{n_b}} \\ \vdots, \ldots, \vdots, \vdots, \ldots, \vdots, \vdots, \ldots, \vdots, \vdots, \ldots, \vdots, \vdots, \vdots, \ldots, \vdots \\ \frac{\partial \zeta_{n_x}}{\partial x_{a,1}}, \ldots, \frac{\partial \zeta_{n_x}}{\partial x_{a,n_x}}, \frac{\partial \zeta_{n_x}}{\partial u_{a,1}}, \ldots, \frac{\partial \zeta_{n_x}}{\partial u_{a,n_u}}, \frac{\partial \zeta_{n_x}}{\partial x_{b,1}}, \ldots, \frac{\partial \zeta_{n_x}}{\partial x_{b,n_x}}, \frac{\partial \zeta_{n_x}}{\partial u_{b,1}}, \ldots, \frac{\partial \zeta_{n_x}}{\partial u_{b,n_u}}, \frac{\partial \zeta_{n_x}}{\partial t_f}, \frac{\partial \zeta_{n_x}}{\partial b_1}, \ldots, \frac{\partial \zeta_{n_x}}{\partial b_{n_b}} \end{bmatrix} \tag{4.22}$$

Here, we can again take advantage of the matrix operation, in this case, the 3-dimensional Jacobian matrix $J_\zeta \in \mathbb{R}^{n_x \times (2(n_x+n_u)+n_{t_f}+n_p)\times(N_n-1)}$ can be obtained with $2(n_x+n_u)+n_{t_f}+n_p$ matrix operations by computing Equation (4.23).

$$J_\zeta = \left[\frac{\partial \zeta}{\partial x_{a,1}}, \ldots, \frac{\partial \zeta}{\partial x_{a,n_x}}, \frac{\partial \zeta}{\partial u_{a,1}}, \ldots, \frac{\partial \zeta}{\partial u_{a,n_u}}, \frac{\partial \zeta}{\partial x_{b,1}}, \ldots, \frac{\partial \zeta}{\partial x_{b,n_x}}, \frac{\partial \zeta}{\partial u_{b,1}}, \ldots, \frac{\partial \zeta}{\partial u_{b,n_u}}, \frac{\partial \zeta}{\partial t_f}, \frac{\partial \zeta}{\partial b_1}, \ldots, \frac{\partial \zeta}{\partial b_{n_b}} \right] \tag{4.23}$$

where

$$\begin{cases} x_a &= x(1:(N_n-1),:), \quad x_a \in \mathbb{R}^{(N_n-1)\times n_x} \\ x_b &= x(2:N_n,:), \quad x_b \in \mathbb{R}^{(N_n-1)\times n_x} \\ u_a &= u(1:(N_n-1),:), \quad u_a \in \mathbb{R}^{(N_n-1)\times n_u} \\ u_b &= u(2:N_n,:), \quad u_b \in \mathbb{R}^{(N_n-1)\times n_u} \end{cases} \tag{4.24}$$

We can see that the defect constraints $\zeta \in \mathbb{R}^{(N_n-1)\times n_x}$ is a function of $[x_a, x_b, u_a, u_b, t_f\ p]$, the 3-dimensional Jacobian matrix $J_\zeta \in \mathbb{R}^{n_x \times (2(n_x+n_u)+n_{t_f}+n_p)\times(N_n-1)}$ can be obtained by $2(n_x+n_u)+n_{t_f}+n_p$ times of matrix differential operations. However, in this case, there will introduce repeatedly calling of the dynamic equations, which some times take long time for computation.

Taking the Trapezoidal approach for example, the following procedures can help to reduce the computation time further. First, calculate the derivatives of $\boldsymbol{\zeta}$ to $\boldsymbol{x}$, $\boldsymbol{u}$, and $\boldsymbol{p}$, let's denote:

$$\boldsymbol{f}'_{\boldsymbol{x}_i} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_i} = \begin{bmatrix} \frac{\partial \boldsymbol{f}_{1,1}}{\partial \boldsymbol{x}_{1,i}} & \cdots & \frac{\partial \boldsymbol{f}_{1,n_x}}{\partial \boldsymbol{x}_{1,i}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{f}_{N_n,1}}{\partial \boldsymbol{x}_{N_n,i}} & \cdots & \frac{\partial \boldsymbol{f}_{N_n,n_x}}{\partial \boldsymbol{x}_{N_n,i}} \end{bmatrix}, \ i = 1, \ldots, n_x \tag{4.25}$$

$$\boldsymbol{f}'_{\boldsymbol{u}_i} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}_i} = \begin{bmatrix} \frac{\partial \boldsymbol{f}_{1,1}}{\partial \boldsymbol{u}_{1,i}} & \cdots & \frac{\partial \boldsymbol{f}_{1,n_x}}{\partial \boldsymbol{u}_{1,i}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{f}_{N_n,1}}{\partial \boldsymbol{u}_{N_n,i}} & \cdots & \frac{\partial \boldsymbol{f}_{N_n,n_x}}{\partial \boldsymbol{u}_{N_n,i}} \end{bmatrix}, \ i = 1, \ldots, n_u \tag{4.26}$$

$$\boldsymbol{f}'_{\boldsymbol{p}_i} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}_i} = \begin{bmatrix} \frac{\partial \boldsymbol{f}_{1,1}}{\partial \boldsymbol{p}_i} & \cdots & \frac{\partial \boldsymbol{f}_{1,n_x}}{\partial \boldsymbol{p}_i} \\ \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{f}_{N_n,1}}{\partial \boldsymbol{p}_i} & \cdots & \frac{\partial \boldsymbol{f}_{N_n,n_x}}{\partial \boldsymbol{p}_i} \end{bmatrix}, \ i = 1, \ldots, n_p \tag{4.27}$$

The above differential matrix can be obtained with n_x, n_u and n_p times of matrix operations respectively, subsequently, the following derivatives of $\boldsymbol{\zeta}$ can be obtained:

$$\frac{\partial \boldsymbol{\zeta}}{\partial \boldsymbol{x}_{a_i}} = -Z(i) + hT_a \boldsymbol{f}'_{\boldsymbol{x}_i}, \ i = 1, \ldots, n_x \tag{4.28}$$

$$\frac{\partial \boldsymbol{\zeta}}{\partial \boldsymbol{x}_{b_i}} = Z(i) + hT_b \boldsymbol{f}'_{\boldsymbol{x}_i}, \ i = 1, \ldots, n_x \tag{4.29}$$

$$\frac{\partial \boldsymbol{\zeta}}{\partial \boldsymbol{u}_{a_i}} = hT_a \boldsymbol{f}'_{\boldsymbol{u}_i}, \ i = 1, \ldots, n_u \tag{4.30}$$

$$\frac{\partial \boldsymbol{\zeta}}{\partial \boldsymbol{u}_{b_i}} = hT_b \boldsymbol{f}'_{\boldsymbol{u}_i}, \ i = 1, \ldots, n_u \tag{4.31}$$

$$\frac{\partial \boldsymbol{\zeta}}{\partial \boldsymbol{p}_i} = hT_{s2} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}_i}, \ i = 1, \ldots, n_p \tag{4.32}$$

$$\frac{\partial \boldsymbol{\zeta}}{\partial t_f} = \frac{1}{N_n - 1} T_{s2} \boldsymbol{f}(x, u, p) \tag{4.33}$$

where $Z(i) \in \mathbb{R}^{(N_n-1)\times n_x}$ is a zero matrix with the *ith* column a one vector, T_a and T_b are transformation matrix given as:

$$T_a = \begin{bmatrix} -\frac{1}{2} & 0 & 0 & \cdots & 0 & 0 \\ 0 & -\frac{1}{2} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -\frac{1}{2} & 0 \end{bmatrix}, \ T_a \in \mathbb{R}^{(N_n-1)\times N_n} \tag{4.34}$$

$$T_b = \begin{bmatrix} 0 & -\frac{1}{2} & 0 & \cdots & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & \frac{1}{2} \end{bmatrix}, \; T_b \in \mathbb{R}^{(N_n-1)\times N_n} \tag{4.35}$$

With the obtained derivative matrix, the entire Jacobian matrix of the defect constraints presented in Equation (4.23) can be constructed.

2) Jacobians of the path and boundary constraints

The Jacobian matrix of the path constraints can be obtained with $n_x + n_u + n_{tf} + n_p$ times of matrix differential operations.

The derivatives to the state variables $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}} \in \mathbb{R}^{N_n \times n_g \times n_x}$ are:

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_i} = \begin{bmatrix} \frac{\partial \boldsymbol{g}_{1,1}}{\partial \boldsymbol{x}_{1,i}} & \cdots & \frac{\partial \boldsymbol{g}_{1,n_g}}{\partial \boldsymbol{x}_{1,i}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{g}_{N_n,1}}{\partial \boldsymbol{x}_{N_n,i}} & \cdots & \frac{\partial \boldsymbol{g}_{N_n,n_g}}{\partial \boldsymbol{x}_{N_n,i}} \end{bmatrix}, \; i = 1, \ldots, n_x \tag{4.36}$$

The derivatives of the path constraints to the control variables $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}} \in \mathbb{R}^{N_n \times n_g \times n_u}$ are:

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}_i} = \begin{bmatrix} \frac{\partial \boldsymbol{g}_{1,1}}{\partial \boldsymbol{u}_{1,i}} & \cdots & \frac{\partial \boldsymbol{g}_{1,n_g}}{\partial \boldsymbol{u}_{1,i}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{g}_{N_n,1}}{\partial \boldsymbol{u}_{N_n,i}} & \cdots & \frac{\partial \boldsymbol{g}_{N_n,n_g}}{\partial \boldsymbol{u}_{N_n,i}} \end{bmatrix}, \; i = 1, \ldots, n_u \tag{4.37}$$

The derivatives of the path constraints to the design variables $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{p}} \in \mathbb{R}^{N_n \times n_g \times n_p}$ are:

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{p}_i} = \begin{bmatrix} \frac{\partial \boldsymbol{g}_{1,1}}{\partial \boldsymbol{p}_i} & \cdots & \frac{\partial \boldsymbol{f}_{1,n_g}}{\partial \boldsymbol{p}_i} \\ \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{g}_{N_n,1}}{\partial \boldsymbol{p}_i} & \cdots & \frac{\partial \boldsymbol{g}_{N_n,n_g}}{\partial \boldsymbol{p}_i} \end{bmatrix}, \; i = 1, \ldots, n_p \tag{4.38}$$

The derivatives of the path constraints to the terminal time $\frac{\partial \boldsymbol{g}}{\partial t_f} \in \mathbb{R}^{N_n \times n_g}$ are:

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{t}_f} = \begin{bmatrix} \frac{\partial \boldsymbol{g}_{1,1}}{\partial t_f} & \cdots & \frac{\partial \boldsymbol{g}_{1,n_g}}{\partial t_f} \\ \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{g}_{N_n,1}}{\partial t_f} & \cdots & \frac{\partial \boldsymbol{g}_{N_n,n_g}}{\partial t_f} \end{bmatrix} \tag{4.39}$$

The derivatives of the boundary constraints to the initial state variables $\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{x}_0} \in \mathbb{R}^{1 \times n_b \times n_x}$ are given as:

$$\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{x}_{0,i}} = \begin{bmatrix} \frac{\partial \boldsymbol{b}_{1,1}}{\partial x_{0,i}} & \cdots & \frac{\partial \boldsymbol{b}_{1,n_b}}{\partial \boldsymbol{x}_{0,i}} \end{bmatrix}, \; i = 1, \ldots, n_x \tag{4.40}$$

The derivatives of the boundary constraints to the final state variables $\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{x}_f} \in \mathbb{R}^{1 \times n_b \times n_x}$ are given as:

$$\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{x}_{f,i}} = \begin{bmatrix} \frac{\partial \boldsymbol{b}_{1,1}}{\partial \boldsymbol{x}_{f,i}} & \cdots & \frac{\partial \boldsymbol{b}_{1,n_g}}{\partial \boldsymbol{x}_{f,i}} \end{bmatrix}, \; i = 1, \ldots, n_x \tag{4.41}$$

The derivatives of the boundary constraints to the design variables $\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{p}} \in \mathbb{R}^{1 \times n_b \times n_p}$ are given as:

$$\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{p}_i} = \begin{bmatrix} \frac{\partial \boldsymbol{b}_{1,1}}{\partial \boldsymbol{p}_i} & \cdots & \frac{\partial \boldsymbol{b}_{1,n_b}}{\partial \boldsymbol{p}_i} \end{bmatrix}, \; i = 1, \ldots, n_p \tag{4.42}$$

The derivatives of the boundary constraints to the terminal time $\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{t}_f} \in \mathbb{R}^{1 \times n_b \times 1}$ are given as:

$$\frac{\partial \boldsymbol{b}}{\partial t_f} = \begin{bmatrix} \frac{\partial \boldsymbol{b}_{1,1}}{\partial t_f} & \cdots & \frac{\partial \boldsymbol{b}_{1,n_b}}{\partial t_f} \end{bmatrix} \tag{4.43}$$

Moreover, we can also introduce the parallel computing by using *parfor* loops and GPU computing support in MATLAB to accelerate the calculation of the above derivatives. The obtained Jacobian matrices of the defect constraints, path constraints and boundary constraints are reconstructed into the sparse patterns demonstrated in Figure 4.2.

4.3.4 Cost function

The user defined cost function $J \in \mathbb{R}$ is composed by two parts which are the Meyer term $\mathcal{M} \in \mathbb{R}$ and the Lagrangian term $\mathcal{L} \in \mathbb{R}^{N_n \times 1}$, and it can be denoted as Equation (4.44), where the trapezoidal integration method is employed to deal with the Lagrangian term.

$$J = \mathcal{M} + \frac{h}{2} T_{s1} \mathcal{L} \tag{4.44}$$

where T_{s1} is the transformation matrix introduced in Equation (4.15).

4.3.5 Gradients

The user defined cost function $J \in \mathbb{R}$ is a function of the input variables $[\boldsymbol{x}, \boldsymbol{u}, t_f, \boldsymbol{p}]$, the gradient of the state and control variables $gradJ_{xu}$ can be denoted as:

$$gradJ_{xu} = \left[\frac{\partial J}{\partial \boldsymbol{x}_1}, \ldots, \frac{\partial J}{\partial \boldsymbol{x}_{n_x}}, \frac{\partial J}{\partial \boldsymbol{u}_1}, \ldots, \frac{\partial J}{\partial \boldsymbol{u}_{n_u}}\right], \; gradJ_{xu} \in \mathbb{R}^{N_n \times (n_x + n_u)} \tag{4.45}$$

After the calculation of the cost function gradient to the terminal time and design variables, the final NLP gradients should be constructed in following format:

$$\begin{aligned} gradJ = \; & \left[\frac{\partial J}{\partial \boldsymbol{x}_{1,1}}, \ldots, \frac{\partial J}{\partial \boldsymbol{x}_{1,n_x}}, \frac{\partial J}{\partial \boldsymbol{u}_{1,1}}, \ldots, \frac{\partial J}{\partial \boldsymbol{u}_{1,n_u}}, \ldots, \frac{\partial J}{\partial \boldsymbol{x}_{N_n,1}}, \ldots, \frac{\partial J}{\partial \boldsymbol{u}_{N_n,n_u}}, \frac{\partial J}{\partial \boldsymbol{x}_{1,1}}, \ldots, \frac{\partial J}{\partial \boldsymbol{u}_{N_n,n_u}}, \right. \\ & \left. \frac{\partial J}{\partial t_f}, \frac{\partial J}{\partial \boldsymbol{b}_{1,1}}, \ldots, \frac{\partial J}{\partial \boldsymbol{b}_{1,n_b}}\right]^T, \; gradJ \in \mathbb{R}^{(N_n(n_x+n_u)+n_{t_f}+n_p)\times 1} \end{aligned} \tag{4.46}$$

4.4 Global collocation approach

4.4.1 Multi-phase optimal control

For some complicated optimal control problems with features of non-smooth and highly nonlinear, it might be inaccurate to approximate the solution with only one polynomial [35]. In this case, a multiple phase approximation with more than one polynomials may be beneficial. A general N_p phases optimal control problem can be formulated as:

$$J = \sum_{k=1}^{N_p} \left[\mathcal{M}^{(k)}(\boldsymbol{x}^{(k)}(t_0^{(k)}), t_0^{(k)}, \boldsymbol{x}^{(k)}(t_f^{(k)}), t_f^{(k)}, \boldsymbol{p}^{(k)}) + \int_{t_0^{(k)}}^{t_f^{(k)}} \mathcal{L}^{(k)}(\boldsymbol{x}^{(k)}(t), \boldsymbol{u}^{(k)}(t), t, \boldsymbol{p}^{(k)}))dt\right] \tag{4.47}$$

s.t.: the first order dynamic constraints:

$$\dot{\boldsymbol{x}}^{(k)}(t) = \boldsymbol{f}^{(k)}[\boldsymbol{x}^{(k)}(t), \boldsymbol{u}^{(k)}(t), t, \boldsymbol{p}^{(k)}] \tag{4.48}$$

the algebraic path constraints:

$$\boldsymbol{g}_{min}^{(k)} \leqslant \boldsymbol{g}^{(k)}[\boldsymbol{x}^{(k)}(t), \boldsymbol{u}^{(k)}(t), t, \boldsymbol{p}^{(k)}] \leqslant \boldsymbol{g}_{max}^{(k)} \tag{4.49}$$

the linkage constraints :

$$\boldsymbol{\psi}_{min} \leqslant \boldsymbol{\psi}\begin{bmatrix} \boldsymbol{x}^{(1)}(t_0^{(1)}), \boldsymbol{u}^{(1)}(t_0^{(1)}), t_0^{(1)}, \boldsymbol{x}^{(1)}(t_f^{(1)}), \boldsymbol{u}^{(1)}(t_f^{(1)}), t_f^{(1)}, \boldsymbol{p}^{(1)} \\ \boldsymbol{x}^{(2)}(t_0^{(2)}), \boldsymbol{u}^{(2)}(t_0^{(2)}), t_0^{(2)}, \boldsymbol{x}^{(2)}(t_f^{(2)}), \boldsymbol{u}^{(2)}(t_f^{(2)}), t_f^{(2)}, \boldsymbol{p}^{(2)} \\ \vdots \\ \boldsymbol{x}^{(N_p)}(t_0^{(N_p)}), \boldsymbol{u}^{(N_p)}(t_0^{(N_p)}), t_0^{(N_p)}, \boldsymbol{x}^{(N_p)}(t_f^{(N_p)}), \boldsymbol{u}^{(N_p)}(t_f^{(N_p)}), t_f^{(N_p)}, \boldsymbol{p}^{(N_p)} \end{bmatrix} \leqslant \boldsymbol{\psi}_{max} \tag{4.50}$$

and the boundary constraints:

$$\boldsymbol{b}_{min} \leqslant \boldsymbol{b}(\boldsymbol{x}_0^1, t_0^1, \boldsymbol{x}_0^{N_p}, t_0^{N_p}, \boldsymbol{p}) \leqslant \boldsymbol{b}_{max} \tag{4.51}$$

4.4.2 NLP variables

In a multi-phase optimal control problem, the time domain is discretized into N_p subintervals:

$$t_0 = t_0^{(1)} < t_f^{(1)} = t_0^{(2)} < t_f^{(2)} = \ldots = t_0^{(N_p)} < t_f^{(N_p)} = t_f \tag{4.52}$$

The NLP variables are given as:

$$\boldsymbol{y} = \left[\boldsymbol{x}_0^{(1)}, \boldsymbol{u}_0^{(1)}, \ldots, \boldsymbol{x}_{N_n^{(1)}}^{(1)}, \boldsymbol{u}_{N_n^{(1)}}^{(1)}, t_0^{(1)}, t_f^{(1)}, \ldots, \boldsymbol{x}_0^{(N_p)}, \boldsymbol{u}_0^{(N_p)}, \ldots, \boldsymbol{x}_{N_n^{(N_p)}}^{(N_p)}, \boldsymbol{u}_{N_n^{(N_p)}}^{(N_p)}, t_0^{(N_p)}, t_f^{(N_p)}, \boldsymbol{p} \right]^T \tag{4.53}$$

where $N_n^{(p)}$ with $p = 1, \ldots, N_p$ denotes the number of collocation nodes at phase p. $\boldsymbol{x}_i^{(p)}$ with $p = 1, \ldots, N_p, i = 1, \ldots, N_n^{(p)}$ means the ith state node at phase p, and $\boldsymbol{u}_i^{(p)}$ with $p = 1, \ldots, N_p, i = 1, \ldots, N_n^{(p)}$ means the ith control node at phase p.

The continuous time samples are denoted as a function of $t_0^{(p)}, t_f^{(p)}$ and $\tau \in [-1,\ 1]$:

$$t = \frac{(t_f^{(p)} - t_0^{(p)})}{2}\tau + \frac{(t_f^{(p)} + t_0^{(p)})}{2} \tag{4.54}$$

The polynomial that approximating the state and control variables at phase p are denoted as Equation (4.55) and Equation (4.56) respectively:

$$x^{(p)}(\tau) \approx P_x^{(p)}(\tau) = \sum_{i=0}^{N_n^{(p)}} L_i^{(p)}(\tau) x_i^{(p)}, p = 1,\ \ldots, N_p \tag{4.55}$$

$$u^{(p)}(\tau) \approx P_u^{(p)}(\tau) = \sum_{i=0}^{N_n^{(p)}} L_i^{(p)}(\tau) u_i^{(p)}, p = 1,\ \ldots, N_p \tag{4.56}$$

where $L_i^{(p)}$ is the Lagrange basis polynomial at phase p, given as:

$$L_i^{(p)}(\tau) = \prod_{j=0, j\neq i}^{N} \frac{\tau - \tau_j^{(p)}}{\tau_i^{(p)} - \tau_j^{(p)}} \tag{4.57}$$

The details of the calculation of the collocation nodes have been presented in Chapter 2.

4.4.3 Constraints

1) Defect constraints

The derivative of the state variable at sample position $\tau_k^{(p)}$ in phase p is given as:

$$\dot{x}^{(p)}(\tau_k^{(p)}) \approx \dot{P}_x^{(p)}(\tau_k^{(p)}) = \sum_{i=0}^{N_n^{(p)}} \dot{L}_i^{(p)}(\tau_k^{(p)}) x_i^{(p)} = \sum_{i=0}^{N_n^{(p)}} D_{ki}^{(p)} x_i^{(p)} \tag{4.58}$$

where $D_{ki}^{(p)}$ is the differential matrix in phase p. For the Legendre-Gauss-Lobatto (LGL)

collocation method [54] $D_{ki}^{(p)}$ is denoted as:

$$D_{ki}^{(p)} = \begin{cases} \dfrac{P_{N_n^{(p)}}^{(p)}(\tau_k^{(p)})}{P_{N_n^{(p)}}(\tau_i^{(p)})(\tau_k^{(p)} - \tau_i^{(p)})}, & i \neq k \\ -N_n^{(p)}(N_n^{(p)} + 1)/4, & i = k = 0 \\ N_n^{(p)}(N_n^{(p)} + 1/4), & i = k = N_n^{(p)} \\ 0, & otherwise \end{cases} \tag{4.59}$$

The details of the differential matrix of other pseudospectral methods are given in Chapter 2.

The defect constraints $\boldsymbol{\zeta}_i^{(p)} \in \mathbb{R}^{(N_n^{(p)}+1)\times n_x}$ of LGL collocation method in phase p can be denoted as:

$$\boldsymbol{\zeta}_i^{(p)} = \sum_{i=0}^{N_n^{(p)}} D_{ki}^{(p)} x_i^{(p)} - \frac{t_f^{(p)} - t_0^{(p)}}{2} f\left(x_k^{(p)}, u_k^{(p)}, \tau_k^{(p)}\right), \tag{4.60}$$

2) Linkage constraints

The linkage constraint $\boldsymbol{\psi} \in \mathbb{R}^{N_p\times(n_x+n_u+1)}$ is denoted as:

$$\boldsymbol{\psi}^{(p)} = \left[\boldsymbol{x}_{N_n^{(p)}}^{(p)} - \boldsymbol{x}_0^{(p+1)}, \boldsymbol{u}_{N_n^{(p)}}^{(p)} - \boldsymbol{u}_0^{(p+1)}, t_f^{(p)} - t_0^{(p+1)}\right],\ p = 1, \ldots, N_p - 1 \tag{4.61}$$

The path constraints are the same as the ones presented in Section 4.3.4, while the boundary constraints are incorporated into the path constraints.

The final constraints are

$$\begin{aligned} \boldsymbol{c}(\boldsymbol{y}) = \ & [\boldsymbol{\zeta}_{0,1}, \ldots, \boldsymbol{\zeta}_{0,n_x}, \ldots, \boldsymbol{\zeta}_{(\sum_{p=1}^{N_p} N_n^{(p)}-1),1}, \ldots, \boldsymbol{\zeta}_{(\sum_{p=1}^{N_p} N_n^{(p)}-1),n_x}, \\ & \boldsymbol{g}_{1,1}, \ldots, \boldsymbol{g}_{1,n_g}, \ldots, \boldsymbol{g}_{N_n,1}, \ldots, \boldsymbol{g}_{N_n,n_g}, \\ & \boldsymbol{\psi}_{1,1}, \ldots, \boldsymbol{\psi}_{1,n_x+n_u+1}, \ldots, \boldsymbol{\psi}_{N_p,1}, \ldots, \boldsymbol{\psi}_{N_p,n_x+n_u+1} \\ & \boldsymbol{b}_{1,1}, \ldots, \boldsymbol{b}_{1,n_b}]^T,\ \boldsymbol{c}(\boldsymbol{y}) \in \mathbb{R}^{((N_n-1)n_x+N_n n_g+n_b)\times 1} \end{aligned} \tag{4.62}$$

4.4.4 Jacobians

The sparse patterns of the Jacobian matrix with pseudospectral methods are demonstrated in Figure 4.3.

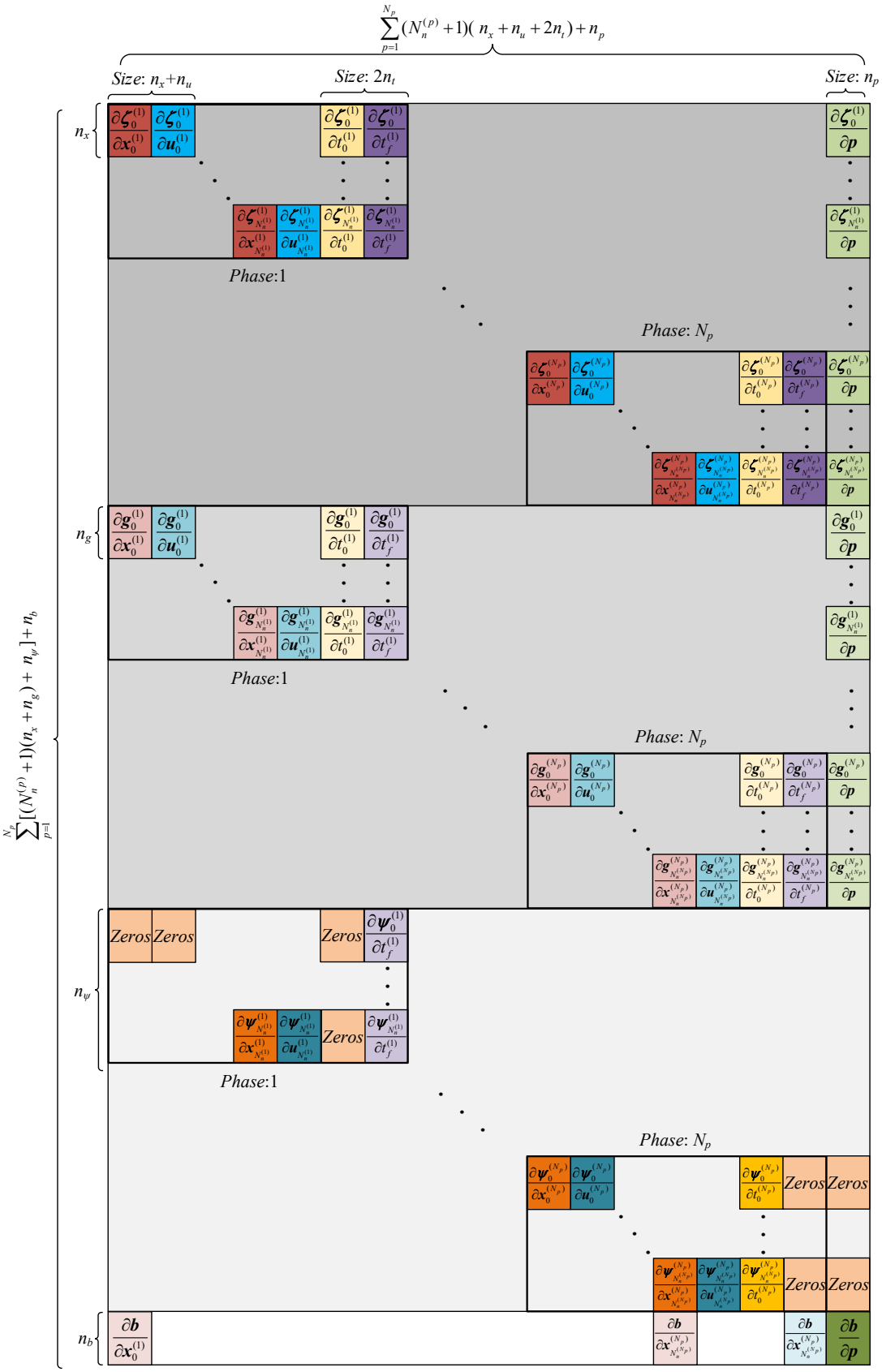

Figure 4.3: *The Jacobian patterns of the multi-phase optimal control problem*

4.4.5 Cost function

The objective function of the multi-phase optimal control problem with LGL collocation method is given as:

$$J = \mathcal{M}(\boldsymbol{x}_0^{(1)}, t_0^{(1)}, \boldsymbol{x}_{N_n^{(N_p)}}^{(N_p)}, t_f^{(N_p)}, \boldsymbol{p}) + \sum_{p=1}^{N_p} \left[\frac{t_f^{(p)} - t_0^{(p)}}{2} \sum_{i=0}^{N_n^{(p)}} w_i^{(p)} \mathcal{L}\left(x_i^{(p)}, u_i^{(p)}, \tau_i^{(p)}, \boldsymbol{p}\right) \right] \tag{4.63}$$

$$w_i^{(p)} = \int_{-1}^{1} L_i^{(p)}(\tau) d\tau = \frac{2}{N_n^{(p)}\left(N_n^{(p)} + 1\right) L_{N_n^{(p)}}^2\left(\tau_i^{(p)}\right)} \tag{4.64}$$

4.4.6 Gradients

The gradients of the cost function $J \in \mathbb{R}$ to the state and control variables can be calculated with the matrix operation:

$$gradJ_{xu} = \left[\frac{\partial J}{\partial \boldsymbol{x}_1}, \ldots, \frac{\partial J}{\partial \boldsymbol{x}_{n_x}}, \frac{\partial J}{\partial \boldsymbol{u}_1}, \ldots, \frac{\partial J}{\partial \boldsymbol{u}_{n_u}} \right], \; gradJ_{xu} \in \mathbb{R}^{N_n \times (n_x + n_u)} \tag{4.65}$$

After the calculation of the cost function gradient to the initial time, terminal time of each phase and the gradient to the design variables, the final NLP gradients should be constructed in following format:

$$\begin{aligned} gradJ = \Bigg[& \frac{\partial J}{\partial \boldsymbol{x}_0^{(1)}}, \frac{\partial J}{\partial \boldsymbol{u}_0^{(1)}}, \ldots, \frac{\partial J}{\partial \boldsymbol{x}_{N_n^{(1)}}^{(1)}}, \frac{\partial J}{\partial \boldsymbol{u}_{N_n^{(1)}}^{(1)}}, \ldots, \frac{\partial J}{\partial t_0^{(1)}}, \frac{\partial J}{\partial t_f^{(1)}}, \ldots, \\ & \frac{\partial J}{\partial \boldsymbol{x}_0^{(N_p)}}, \frac{\partial J}{\partial \boldsymbol{u}_0^{(N_p)}}, \ldots, \frac{\partial J}{\partial \boldsymbol{x}_{N_n^{(N_p)}}^{(N_p)}}, \frac{\partial J}{\partial \boldsymbol{u}_{N_n^{(N_p)}}^{(N_p)}}, \frac{\partial J}{\partial t_0^{(N_p)}}, \frac{\partial J}{\partial t_f^{(N_p)}}, \frac{\partial J}{\partial \boldsymbol{p}} \Bigg]^T \end{aligned} \tag{4.66}$$

4.5 Differential methods

The first order derivatives of the constraints $c(\boldsymbol{y})$ are required by most of the NLP solvers, and precise calculation of the derivatives will assist to obtain a faster convergence. The differential methods implemented in GDYNOPT are the forward, central, complex step and analytical methods.

The forward finite difference approximation is denoted as:

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_i} = \frac{\boldsymbol{f}(\boldsymbol{x} + \boldsymbol{h}_i) - \boldsymbol{f}(\boldsymbol{x})}{h_i} + O(h) \tag{4.67}$$

where $\boldsymbol{h}_i$ is the perturbation vector of the i_{th} column state variables $\boldsymbol{x}_i$, $O(h)$ is the truncation error. The forward finite difference approximation benefits from this reduced number of calling the dynamic equations, however, it has a first-order precision. Of course we can reduce the perturbation h to have a more precise approximation, however, in this case, the round-off error might be introduced.

The central finite difference approximation denoted as

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_i} = \frac{\boldsymbol{f}(\boldsymbol{x} + \boldsymbol{h}_i) - \boldsymbol{f}(\boldsymbol{x} - \boldsymbol{h}_i)}{2h_i} + O(h^2) \tag{4.68}$$

has the second-order precision, however it will introduce doubled numbers of calling the dynamic function. In some cases, the benefit of the central finite difference approximation might not be so significant.

The complex-step derivative approximation methods can compute the first derivatives in relatively easy way, which has been verified to be very powerful to provide accuracy closed to the analytical one [104] [105].

The complex-step derivative approximation is based on the Taylor's series expansion of an analytic function [104]:

$$f(x + ih) = f(x) + ihf'(x) - h^2\frac{f''(x)}{2!} - ih^3\frac{f'''(x)}{3!} + h^4\frac{f^{(4)}(x)}{4!} + ... \tag{4.69}$$

Taking the imaginary parts of both sides of Equation (4.69) and divide by h, we can obtain:

$$f'(x) = Im[f(x + ih)]/h + h^2\frac{f'''(x)}{3!} + ... \tag{4.70}$$

Neglecting the terms with order h^2 or higher, the first-order the complex-step derivative approximation with approximation error $O(h^2)$ can be denoted as:

$$f'(x) = Im[f(x + ih)]/h \tag{4.71}$$

The complex-step derivative approximation method can reduce the computation time significantly with the in-built image function in MATLAB. Besides, when the perturbation is very small, i.e., $h = 1e - 20$ it can work quite well in nearly all cases with very small round-off errors. However, sometimes, it need some tricks to handle some special functions.

Another option of the derivative approximation method implemented in GDYNOPT is the analytical method which derives the symbolic equations of the derivatives directly and calls it as a function. This approach can give the exact values of derivatives, however, it might be difficult or impossible to obtain the analytical expressions of the derivative for some complicated problems.

4.6 Scaling

In many cases, the variables of the formulated optimal design and control problem are of significantly different orders of magnitude. The transcribed NLP without any scaling are often badly scaled and may cost a long time for convergence or even fail to achieve a solution. To overcome this problem, the manual scaling method by normalizing the basic physical units of length, mass, and time is an optional approach [35, 38, 91, 106–108], however, this approach is usually time consuming. Another solution is the automatic scaling method [38, 53, 109], where the state, control, and static variables of the optimal control problem are normalized to a specific range with the user supplied lower and upper bounds of there variables. The defect constraints are

scaled using the same scaling factors used to scale the state variables. While the path constraints, linkage constraints and cost function are scaled with their average norm of the gradients respectively. The followed subsections will give a detail description of the proposed scaling approach in this work, taking the local collocation method for instance.

4.6.1 Scaling of the variables

In this work, a linear scaling method is used to scale the state, control, final time and design variables to let them lie in [0, 1], the scaled $\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}}, \hat{t}_f$, and $\hat{\boldsymbol{p}}$ are,

$$\hat{\boldsymbol{x}} = \boldsymbol{a}_x \boldsymbol{x} + \boldsymbol{b}_x \tag{4.72}$$

$$\hat{\boldsymbol{u}} = \boldsymbol{a}_u \boldsymbol{u} + \boldsymbol{b}_u \tag{4.73}$$

$$\hat{t}_f = a_{t_f} t_f + b_{t_f} \tag{4.74}$$

$$\hat{\boldsymbol{p}} = \boldsymbol{a}_p \boldsymbol{p} + \boldsymbol{b}_p \tag{4.75}$$

where the scaling factors are given as followings:

$$\begin{cases} \boldsymbol{a}_x &= \dfrac{1}{\boldsymbol{x}_{\max} - \boldsymbol{x}_{\min}} \\ \boldsymbol{b}_x &= -\dfrac{\boldsymbol{x}_{\min}}{\boldsymbol{x}_{\max} - \boldsymbol{x}_{\min}} \\ \boldsymbol{a}_u &= \dfrac{1}{\boldsymbol{u}_{\max} - \boldsymbol{u}_{\min}} \\ \boldsymbol{b}_u &= -\dfrac{\boldsymbol{u}_{\min}}{\boldsymbol{u}_{\max} - \boldsymbol{u}_{\min}} \\ \boldsymbol{a}_{t_f} &= \dfrac{1}{t_{f,\max} - t_{f,\min}} \\ \boldsymbol{b}_{t_f} &= -\dfrac{t_{f,\min}}{t_{f,\max} - t_{f,\min}} \\ \boldsymbol{a}_p &= \dfrac{1}{\boldsymbol{p}_{\max} - \boldsymbol{p}_{\min}} \\ \boldsymbol{b}_p &= -\dfrac{\boldsymbol{p}_{\min}}{\boldsymbol{p}_{\max} - \boldsymbol{p}_{\min}} \end{cases} \tag{4.76}$$

4.6.2 Scaling of the constraints

The constraints should also be scaled to a uniform scale for a faster convergence. In this work, an average norm of gradient based method is proposed to scale the constraints and cost functions. The final scale of the constraints and cost functions are approximately within [0, 1].

The scaled defects constraints $\hat{\boldsymbol{\zeta}}_i \in \mathbb{R}^{(N_n-1)}$ can be denoted as:

$$\hat{\boldsymbol{\zeta}}_i = \hat{\boldsymbol{k}}_{\zeta,i} \boldsymbol{\zeta}_i, i = 1, 2, ...n_x, \ \boldsymbol{\zeta}_i \in \mathbb{R}^{(N_n-1)} \tag{4.77}$$

The scaling factor $\hat{\boldsymbol{k}}_{\zeta,i} \in \mathbb{R}$ is predefined in the optimization iterations without scaling. The scaling factor $\boldsymbol{k}_{\zeta,i}^{(k)}$ of the defect constraint $\boldsymbol{\zeta}_i$ at iteration k is defined as:

$$\boldsymbol{k}_{\zeta,i}^{(k)} = \frac{1}{N_n - 1} \sum_{j=1}^{N_n-1} \left(\frac{1}{\| \nabla \boldsymbol{\zeta}_{j,i} \|^2} \right) = \frac{1}{N_n - 1} \sum_{j=1}^{N_n-1} \left(\frac{1}{\sqrt{J_{\zeta,temp}}} \right) \tag{4.78}$$

$$\begin{aligned} J_{\zeta,temp} &= (\frac{\partial \boldsymbol{\zeta}_{j,i}}{\partial \boldsymbol{x}_{aj,1}})^2 + ... + (\frac{\partial \boldsymbol{\zeta}_{j,i}}{\partial \boldsymbol{x}_{aj,n_x}})^2 + (\frac{\partial \boldsymbol{\zeta}_{j,i}}{\partial \boldsymbol{u}_{aj,1}})^2 + ... + (\frac{\partial \boldsymbol{\zeta}_{j,i}}{\partial \boldsymbol{u}_{aj,n_u}})^2 \\ &+ (\frac{\partial \boldsymbol{\zeta}_{j,i}}{\partial \boldsymbol{x}_{bj,1}})^2 + ... + (\frac{\partial \boldsymbol{\zeta}_{j,i}}{\partial \boldsymbol{x}_{bj,n_x}})^2 + (\frac{\partial \boldsymbol{\zeta}_{j,i}}{\partial \boldsymbol{u}_{bj,1}})^2 + ... + (\frac{\partial \boldsymbol{\zeta}_{j,i}}{\partial \boldsymbol{u}_{bj,n_u}})^2 \\ &+ (\frac{\partial \boldsymbol{\zeta}_{j,i}}{\partial \boldsymbol{t}_f})^2 + (\frac{\partial \boldsymbol{\zeta}_{j,i}}{\partial \boldsymbol{p}_{j,1}})^2 + ... + (\frac{\partial \boldsymbol{\zeta}_{j,i}}{\partial \boldsymbol{p}_{j,n_p}})^2 \end{aligned} \tag{4.79}$$

The final scaling factor $\hat{\boldsymbol{k}}_{\zeta,i} \in \mathbb{R}$ is the average value of $\boldsymbol{k}_{\zeta,i}^{(k)}$ in the optimization iterations without scaling:

$$\hat{\boldsymbol{k}}_{\zeta,i} = \frac{1}{N_{scal}} \sum_{k=1}^{N_{scal}} \boldsymbol{k}_{\zeta,i}^{(k)} \tag{4.80}$$

where $i = 1, \ldots, n_x$, N_{scal} is the user defined maximum iterations to obtain the scaling factor.

Similarly, the scaled path constraints is given as:

$$\hat{\boldsymbol{g}}_i = \hat{\boldsymbol{k}}_{g,i} \boldsymbol{g}_i, i = 1, 2, ...n_g, \boldsymbol{g}_i \in \mathbb{R}^{N_n} \tag{4.81}$$

$$\boldsymbol{k}_{\boldsymbol{g},i}^{(k)} = \frac{1}{N_n} \sum_{j=1}^{N_n} \left(\frac{1}{\| \nabla \boldsymbol{g}_{j,i} \|^2} \right) = \frac{1}{N_n} \sum_{j=1}^{N_n} \left(\frac{1}{\sqrt{J_{\boldsymbol{g},temp}}} \right) \tag{4.82}$$

$$\begin{aligned} J_{\boldsymbol{g},temp} &= (\frac{\partial \boldsymbol{g}_{j,i}}{\partial \boldsymbol{x}_{aj,1}})^2 + ... + (\frac{\partial \boldsymbol{g}_{j,i}}{\partial \boldsymbol{x}_{aj,n_x}})^2 + (\frac{\partial \boldsymbol{g}_{j,i}}{\partial \boldsymbol{u}_{aj,1}})^2 + ... + (\frac{\partial \boldsymbol{g}_{j,i}}{\partial \boldsymbol{u}_{aj,n_u}})^2 \\ &+ (\frac{\partial \boldsymbol{g}_{j,i}}{\partial \boldsymbol{x}_{bj,1}})^2 + ... + (\frac{\partial \boldsymbol{g}_{j,i}}{\partial \boldsymbol{x}_{bj,n_x}})^2 + (\frac{\partial \boldsymbol{g}_{j,i}}{\partial \boldsymbol{u}_{bj,1}})^2 + ... + (\frac{\partial \boldsymbol{g}_{j,i}}{\partial \boldsymbol{u}_{bj,n_u}})^2 \\ &+ (\frac{\partial \boldsymbol{g}_{j,i}}{\partial \boldsymbol{t}_f})^2 + (\frac{\partial \boldsymbol{g}_{j,i}}{\partial \boldsymbol{p}_{j,1}})^2 + ... + (\frac{\partial \boldsymbol{g}_{j,i}}{\partial \boldsymbol{p}_{j,n_p}})^2 \end{aligned} \tag{4.83}$$

The final scaling factor of the path constraints $\hat{\boldsymbol{k}}_{\boldsymbol{g},i} \in \mathbb{R}$ is the average value of $\boldsymbol{k}_{\boldsymbol{g},i}^{(k)}$ in the optimization iterations without scaling:

$$\hat{\boldsymbol{k}}_{\boldsymbol{g},i} = \frac{1}{N_{scal}} \sum_{k=1}^{N_{scal}} \boldsymbol{k}_{\boldsymbol{g},i}^{(k)} \tag{4.84}$$

In this work, the same approach is implemented to scale the boundary constraints and cost function.

4.6.3 Scaling of the Jacobians

The Scaling of the Jacobians takes into account the relationship between the scaled variables and the scaled constraints. The scaled defect constraint is defined as:

$$\hat{J}_{\zeta,i} = [\boldsymbol{k}_f] \circ J_{\zeta,i}\boldsymbol{\xi}_i^{-1}, i = 1, \ldots, N_v \tag{4.85}$$

where $\circ$ means the Hadamard product [110], $\boldsymbol{\xi} = [\boldsymbol{a}_x, \boldsymbol{a}_u, \boldsymbol{a}_x, \boldsymbol{a}_u, a_{t_f}, \boldsymbol{a}_p]$ is a combination of the variable scaling factors defined in Section 4.6.1 with length $N_v = 2(n_x + n_u) + n_{t_f} + n_p$, $[\hat{\boldsymbol{k}}_f]$ is the scaling matrix defined in Equation (4.86).

$$[\boldsymbol{k}_f] = \begin{bmatrix} \hat{\boldsymbol{k}}_{f,1} & \ldots & \hat{\boldsymbol{k}}_{f,n_x} \\ \vdots & \ddots & \vdots \\ \hat{\boldsymbol{k}}_{f,1} & \ldots & \hat{\boldsymbol{k}}_{f,n_x} \end{bmatrix}, [\boldsymbol{k}_f] \in \mathbb{R}^{(N_n-1)\times n_x} \tag{4.86}$$

The scaled path constraint is defined as:

$$\hat{J}_{\boldsymbol{g},i} = [\boldsymbol{k}_g] \circ J_{\boldsymbol{g},i}\boldsymbol{\xi}_i^{-1}, i = 1, \ldots, N_v \tag{4.87}$$

where $[\hat{\boldsymbol{k}}_g]$ is the scaling matrix defined in Equation (4.88).

$$[\boldsymbol{k}_g] = \begin{bmatrix} \hat{\boldsymbol{k}}_{g,1} & \ldots & \hat{\boldsymbol{k}}_{g,n_g} \\ \vdots & \ddots & \vdots \\ \hat{\boldsymbol{k}}_{g,1} & \ldots & \hat{\boldsymbol{k}}_{g,n_g} \end{bmatrix}, [\boldsymbol{k}_g] \in \mathbb{R}^{N_n\times n_g} \tag{4.88}$$

The scaling of the Jacobians of the boundary constraints and gradient of the cost function can be obtained with the same manner.

4.7 Algorithm flowchart of GDYNOPT

The algorithm flowchart of GDYNOPT is presented in Figure 4.4. GDYNOPT will reshape the user provided initial guess, bounds of variables and constraints into the required format of the NLP solver first. If the user set the scaling parameter to 1, GDYNOPT will call the main body of the optimization function and iterate without any scaling, during which, the scaling factors of the defect, path, boundary constraints and the cost function will be calculated and saved for each iteration. When the user set maximum number of iterations is reached, the first round of optimization will be terminated. Subsequently, the mean values of the saved scaling factors in the first round of iterations are calculated. The bounds of the variables and constraints will be modified with the obtained scaling factors in the second round of iterations. In the second round of iteration, there will be scale and unscale operations of the state, control, and static variables, in addition, the constraints, Jacobians, cost functions and gradients will be scaled with the scaling factors with the approach proposed in last section. In particular, there are plot functions implemented in the constraints function which can draw the curves of the user set variables, this feature can help the user to monitor the optimization process and it is specially useful during the debugging stage of the implementation.

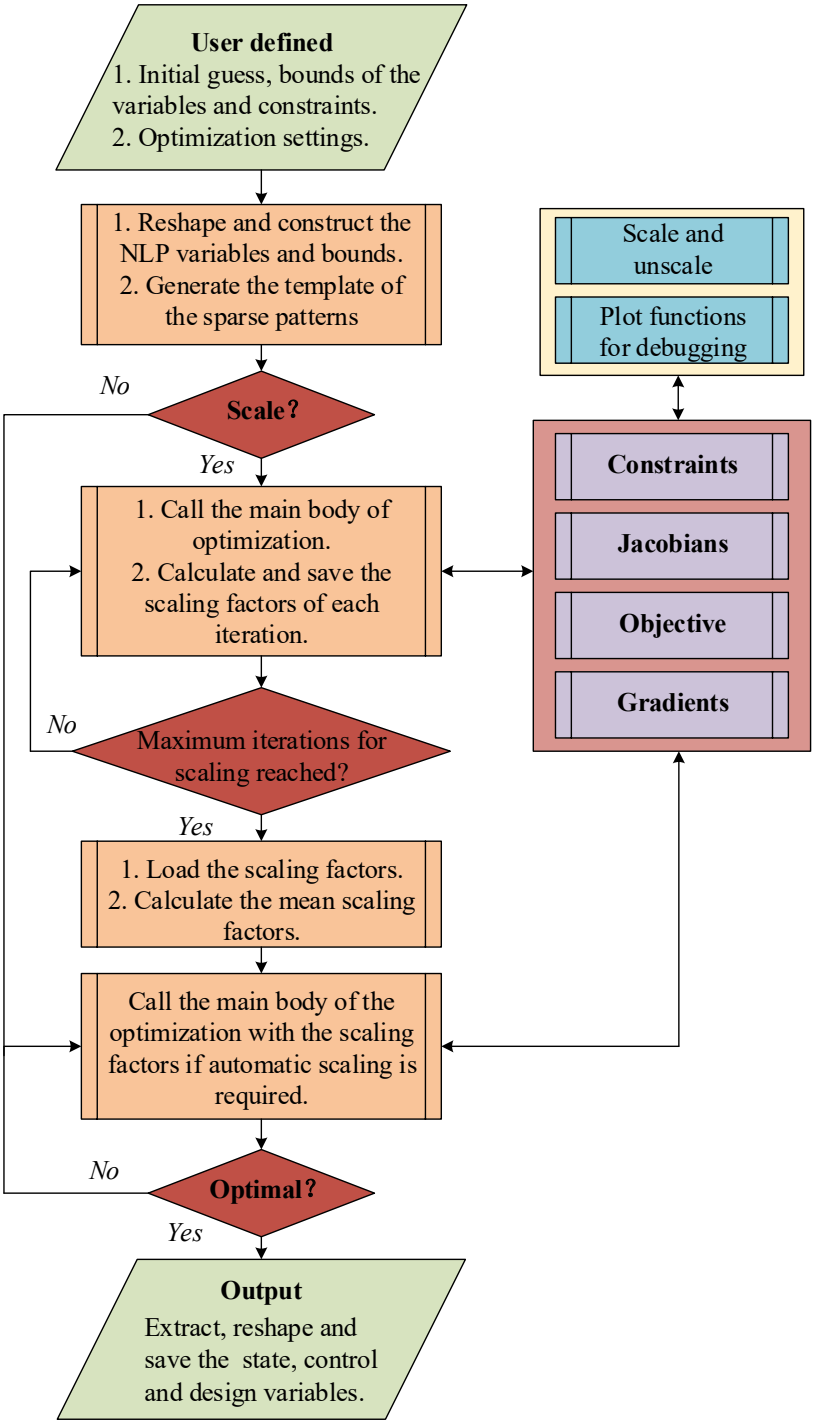

Figure 4.4: *The algorithm flowchart of GDYNOPT*

CHAPTER 5

Optimal Powertrain Design and Control

5.1 Introduction

This Chapter is dedicated to formulate and solve the optimal powertrain design and control problem of the electric rave car with the developed 14-DOF vehicle model in Chapter 3 and the software package GDYNOPT developed in Chapter 4. In particular, the Trapezoidal collocation approach is employed in this work for its robustness and low cost of computation time. The objective is to minimize the lap time:

$$J = \min t_f \tag{5.1}$$

subject to:

• the first order dynamic constraints:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}[\boldsymbol{x}(t), \boldsymbol{u}(t), t, \boldsymbol{p}] \tag{5.2}$$

• the boundaries of the state, control and design variables:

$$\begin{aligned} \boldsymbol{x}_{min} \leqslant \boldsymbol{x}(t) \leqslant \boldsymbol{x}_{max} \\ \boldsymbol{u}_{min} \leqslant \boldsymbol{u}(t) \leqslant \boldsymbol{u}_{max} \\ \boldsymbol{p}_{min} \leqslant \boldsymbol{p} \leqslant \boldsymbol{p}_{max} \end{aligned} \tag{5.3}$$

• the algebraic path constraints:

$$\boldsymbol{g}_{min} \leqslant \boldsymbol{g}[\boldsymbol{x}(t), \boldsymbol{u}(t), t, \boldsymbol{p}] \leqslant \boldsymbol{g}_{max} \tag{5.4}$$

• and the boundary conditions:

$$\boldsymbol{b}_{min} \leqslant \boldsymbol{b}[\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f, \boldsymbol{p}] \leqslant \boldsymbol{b}_{max}, \tag{5.5}$$

where $\dot{\boldsymbol{x}}$ is the first order derivative of the state variables, $\boldsymbol{f}$ is the dynamic model developed in Chapter 3, $\boldsymbol{x}$, $\boldsymbol{u}$, $\boldsymbol{p}$ are respectively the state, control and design vector with their lower and upper bounds: $\boldsymbol{x}_{min}$, $\boldsymbol{u}_{min}$, $\boldsymbol{p}_{min}$ and $\boldsymbol{x}_{max}$, $\boldsymbol{u}_{max}$, $\boldsymbol{p}_{max}$. While $\boldsymbol{g}$ and $\boldsymbol{b}$ are the path and boundary equations respectively with their lower and upper bounds $\boldsymbol{g}_{min}$, $\boldsymbol{b}_{min}$ and $\boldsymbol{g}_{max}$, $\boldsymbol{b}_{max}$. The dimensions of the input and output variables in Equations (5.2), (5.4) and (5.5) are separately given as:

$$\begin{aligned}
\boldsymbol{f} &: \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R} \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_x} \\
\boldsymbol{g} &: \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R} \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_g} \\
\boldsymbol{b} &: \mathbb{R}^{n_x} \times \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R} \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_b}
\end{aligned}$$

The state variables $\boldsymbol{x}$ in this work are represented in Equation (5.6), the details of which are given in Section 3.6.

$$\begin{aligned}
\boldsymbol{x} = \Big[& \dot{X}_{A,b}, \dot{Y}_{A,b}, \dot{z}, \dot{\varphi}, \dot{\phi}, \dot{\psi}, \dot{z}_{ufr}, \dot{z}_{ufl}, \dot{z}_{urr}, \dot{z}_{url}, \dot{\theta}_{ufr}, \dot{\theta}_{ufl}, \dot{\theta}_{urr}, \dot{\theta}_{url}, \\
& X_{A,b}, Y_{A,b}, z, \varphi, \phi, \psi, z_{ufr}, z_{ufl}, z_{urr}, z_{url}, \theta_{ufr}, \theta_{ufl}, \theta_{urr}, \theta_{url}, s, n, \chi \Big]
\end{aligned} \tag{5.6}$$

The control variables $\boldsymbol{u}$, design parameters $\boldsymbol{p}$ and the path constraints $\boldsymbol{g}$ vary with the powertrain layouts and will be presented in the following sections.

The initial guess and bounds of the variables and constraints of the formulated problems are obtained via the path following control of the race car on the Nuburgring circuit described in Section 3.7.

5.2 Variables and constraints

5.2.1 Design variables

The base speed n_b and the constant power speed ratio (CPSR) β are selected as the design parameters of the motor, the sensitivity of the motor mass to these two parameters are presented in Section 3.5.1. The envelope curve of the torque-speed characteristics of the motor can be obtained as Figure 5.1 with the two design parameters and the maximum power of the motor P_{max}, which is normally given by the racing event. In this work, the single-speed transmission is investigated for different topologies, and the gear speed ratio i_g is selected as the design variable of the gearbox.

For the 1-motor and 2-motor driving topologies, a static braking force distribution factor λ is introduced as a design parameter to allocate the braking forces between the front and rear wheels. The design vector of the electric powertrain is finally given as:

$$\boldsymbol{p} = [n_b, \beta, i_g, \lambda],\ n_p = 4 \tag{5.7}$$

The design vector of a 4-motor driving electric powertrain is given as:

$$\boldsymbol{p} = [n_b, \beta, i_g],\ n_p = 3 \tag{5.8}$$

The powertrain mass will change with different design parameters and will affect the dynamic response of the race car, and the detail relationship between them have been investigated in detail in Section 3.5. Besides, the available torque and rotational speed should be limited within the working zone of the electric powertrain, whereas,

both parameters vary with the design parameters. Section 5.2.3 will introduce the path constraints to enforce the powertrain operating in the appropriate working zone.

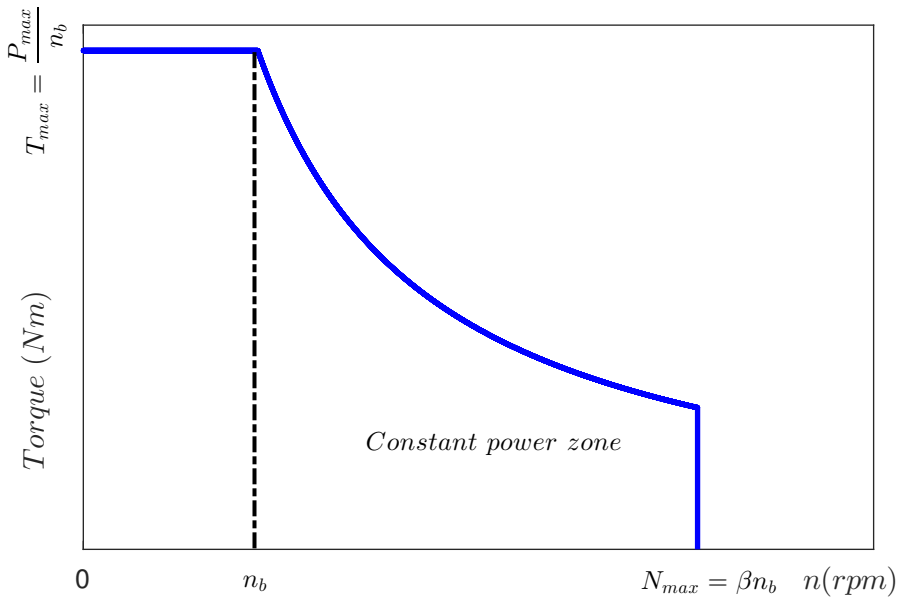

Figure 5.1: *Design parameters of the motor*

5.2.2 Control variables

In order to evaluate the lap time performance of the electric race car, the angle of the steering wheel and the torques applied on different wheels should be controlled.

In this study, for the 1-motor and 2-motor driving topologies, the race car is assumed to be controlled using front steering and two rear wheel driving. Neglecting the dynamic response of the steering system and the motors, the control variables of the 1-motor driving electric race car are given as:

$$\mathbf{u} = [\delta, T_b, T_m],\ n_u = 3 \tag{5.9}$$

The 1-motor driving electric powertrain investigated in this work is assumed to be implemented with the central drive with a differential, thus, the torques acted on the four wheels are denoted as:

$$T = \left[\frac{\lambda T_b}{2}, \frac{\lambda T_b}{2}, \frac{(1-\lambda)T_b + T_m i_g}{2}, \frac{(1-\lambda)T_b + T_m i_g}{2}\right] \tag{5.10}$$

where δ is the steering angle on the steering wheel, T_b is the total braking torque, T_m are the driving torques output by the motor, respectively.

For this topology, the braking and accelerating operations are not allowed to be activated at the same time, and this is realized by introducing a penalty function $J_p \geq 0$. The ideal relationship between the negative braking torque T_b and the positive driving torque T_m should be:

$$J_p = -T_b T_m = 0 \tag{5.11}$$

which means that when the braking torque T_b is applied, the driving torque should be zero, in the contrary, when the driving torque is acted, the braking torque should be zero.

Since the penalty function $J_p \geqslant 0$, the objective function in Equation (5.1) can be rewritten as:

$$J = \min(t_f + wJ_p) \tag{5.12}$$

where w is the weight of J_p, the analytical approach is employed to derive the relevant derivative of the objective function.

For the 2-motor driving topology, the two rear independent wheel driving propulsion layout is adopted in this work, the corresponding control variables and torque acted on each wheel are respectively:

$$\mathbf{u} = [\delta, T_b, T_{m1}, T_{m2}],\ n_u = 4 \tag{5.13}$$

where T_{m1} and T_{m2} denote the motor driving torque acted on the two rear wheel respectively.

$$T = \left[\frac{\lambda T_b}{2}, \frac{\lambda T_b}{2}, \frac{(1-\lambda)T_b}{2} + T_{m1}i_g, \frac{(1-\lambda)T_b}{2} + T_{m2}i_g\right] \tag{5.14}$$

Similar with the 1-motor driving topology, the braking and accelerating torque are forbidden to be applied at the same time by introducing the penalty function:

$$J_p = -T_b(T_{m1} + T_{m2}) = 0 \tag{5.15}$$

For the 4-motor driving topology, the four independent wheel driving propulsion solution is adopted, the corresponding control variables and torque acted on each wheel are respectively:

$$\mathbf{u} = [\delta, T_{m1}, T_{m2}, T_{m3}, T_{m4}],\ n_u = 4 \tag{5.16}$$

$$T = \left[T_{m1}i_g, T_{m2}i_g, T_{m3}i_g, T_{m4}i_g\right] \tag{5.17}$$

where T_{m1}, T_{m2}, T_{m3} and T_{m4} are respectively the torques acted on the four wheels.

5.2.3 Path constraints

The algebraic path constraints are a set of functions of the state, control, final time and design parameters. For the motor design, the maximum rotational speed $N_{max,i}$ should be constrained to a user set range $[\boldsymbol{cl}_{N_{max}}, \boldsymbol{cu}_{N_{max}}]$:

$$\boldsymbol{cl}_{N_{max}} \leq \boldsymbol{N}_{max} = \boldsymbol{N}_b\boldsymbol{\beta} \leq \boldsymbol{cu}_{N_{max}} \tag{5.18}$$

In order to let the motors work within their available operation zone, the motor speed $N_{m,i}$ and output torque $T_{m,i}$ should be constrained within the user set ranges $[\boldsymbol{cl}_{N_{cm}},\ \boldsymbol{cu}_{N_{cm}}]$ and $[\boldsymbol{cl}_{T_{cm}},\ \boldsymbol{cu}_{T_{cm}}]$ respectively:

$$\boldsymbol{cl}_{N_{cm}} \leq \boldsymbol{N}_{cm} = \boldsymbol{N}_m - \boldsymbol{N}_{max} \leq \boldsymbol{cu}_{N_{cm}} \tag{5.19}$$

$$\boldsymbol{cl}_{T_{cm}} \leq \boldsymbol{T}_{cm} = \boldsymbol{T}_m - \boldsymbol{T}_{max} \leq \boldsymbol{cu}_{T_{cm}} \tag{5.20}$$

where $\boldsymbol{N}_m$ is the motor speed which can be denoted as a function of the angular velocity of rear right wheel ω_{rr}, the angular velocity of rear left wheel ω_{rl} and the gear speed ratio i_g,

$$\boldsymbol{N}_m = \frac{30}{\pi}\frac{\omega_{rl} + \omega_{rr}}{2}\boldsymbol{i}_g \tag{5.21}$$

When the units of torque, rotation speed, power are respectively Nm, rpm and kW, the available maximum torque of the motor is given as:

$$\boldsymbol{T}_{max} = \begin{cases} \dfrac{9550\boldsymbol{P}_{max}}{\boldsymbol{N}_b}, & \boldsymbol{N}_m \leq \boldsymbol{N}_b \\ \\ \dfrac{9550\boldsymbol{P}_{max}}{\boldsymbol{N}_m}, & \boldsymbol{N}_m > \boldsymbol{N}_b \end{cases} \tag{5.22}$$

The normal load $\boldsymbol{F}z$ also should be constrained within a reasonable range.

$$\boldsymbol{cl}_{F_z} \leq \boldsymbol{F}_z \leq \boldsymbol{cu}_{F_z} \tag{5.23}$$

where $\boldsymbol{cl}_i$ and $\boldsymbol{cu}_i$ means the minimum and maximum value of the mentioned variables, respectively. The final path constraints vector are given as $\boldsymbol{g} = [\boldsymbol{T}_{cm}, \boldsymbol{N}_{cm}, \boldsymbol{N}_{max}, \boldsymbol{F}z]$, the dimension of which varies with the powertrain layouts. The detail values of there bounds of the constraints are provided based on the simulation results of the closed loop path following control in Section 3.7.

5.3 Simulation parameters

The optimal design and control of the electric race car is based on the chassis of a Formula 3 race car, the simulation parameters of the which are presented in Table 5.1. In order to compare different topologies, the sprung mass m_v of the three topologies are assumed to be the same which is 490 kg, while the powertrain mass (in this work, the powertrain means the motor and the transmission) of all the topologies are controlled within 80kg with one more path constraint.

The bounds of the variables and constraints, problem dimensions will elaborated in the following subsections.

5.3.1 1-Motor driving topology

In order to let the race car operate at the reasonable conditions and reduce the optimization time, the bounds of the variables and constraints should be provided. The detail values of these bounds are obtained based on engineering experiences and the closed loop path following control in Section 3.7. For the 1-motor driving race car, the bounds are illustrated as Table 5.2.

The problem dimensions depend on the number of collocation node N_n, the corresponding number of the variables, constraints and non-zero Jacobians are given as Table 5.3.

Table 5.1: *Simulation parameters of the powertrain optimization*

Prameters	Values
Mass factor of the motor ρ_m	0.2845
Gear volume fill factor ψ	0.7
Mass density of the gear ρ_g	78550 kg/m^3
Sprung mass with powertrain m_v	490 kg
Wheelbase l	2.73 m
CoG to front axle distance l_f	1.595 m
CoG to rear axle distance l_f	1.135 m
Front wheel track w_f	1.585 m
Rear wheel track w_r	1.535 m
Maximum power P_{max}	165 kW
Aerodynamic drag force location h_g	0.18 m
Surface durability factor K	8920000 N/mm^2
Mass factor of the gearbox ρ_{gc}	3.1136

Table 5.2: *Bounds of the optimal powertrain design and control for the 1-motor driving topology*

Prameters	Values
Lower bound of the design vector $\boldsymbol{p}_{min}$	$[500, 1.2, 1, 0.1]$
Upper bound of the design vector $\boldsymbol{p}_{max}$	$[15000, 6.5, 30, 0.9]$
Lower bound of the control vector $\boldsymbol{u}_{min}$	$[-180, -8000, 0]$
Upper bound of the control vector $\boldsymbol{u}_{max}$	$[180, 1e{-}6, 7600]$
Lower bound of the path constraints $\boldsymbol{g}_{min}$	$[-7600, -2e4, 500, 0, 120, 120, 120, 120]$
Upper bound of the path constraints $\boldsymbol{g}_{max}$	$[0, 0, 2e4, 80, 3000, 3000, 3000, 3000]$

Table 5.3: *Problem dimensions of the 1-motor driving topology*

N_n	500	1000	1500	2000	2500	3000	3500	4000	4500
State variables N_x	15500	31000	46500	62000	77500	93000	108500	124000	139500
Control variables N_u	1500	3000	4500	6000	7500	9000	10500	12000	13500
Total NLP variables N_{NLP}	17005	34005	51005	68005	85005	12005	119005	136005	153005
Defects constraints N_{cd}	15469	30969	46469	61969	162469	92969	108469	123969	139469
Path constraints N_{cp}	3500	7000	10500	14000	17500	21000	24500	28000	31500
Non-zero Jacobians N_{def}	1265737	2533737	3801737	5069737	6337737	7605737	8873737	10141737	11409737

5.3.2 2-Motor driving topology

For the 2-motor driving race car, the bounds of the variables and constraints are elaborated as Table 5.4.

The investigated number of the collocation nodes of the 2-motor driving powertrain are also 1000, 2000 and 4500, the relevant number of the variables, constraints and non-zero Jacobians are given as Table 5.5.

5.3.3 4-Motor driving topology

For the 4-motor driving race car, the bounds of the variables and constraints are listed as Table 5.6. The studied number of the collocation nodes of 4-motor driving race car

Table 5.4: *Bounds of the optimal powertrain design and control for the 2-motor driving topology*

Prameters	Values
Lower bound of the design vector $\boldsymbol{p}_{min}$	$[500, 1.2, 1, 0.1]$
Upper bound of the design vector $\boldsymbol{p}_{max}$	$[15000, 6.5, 30, 0.9]$
Lower bound of the control vector $\boldsymbol{u}_{min}$	$[-180, -8000, 0, 0]$
Upper bound of the control vector $\boldsymbol{u}_{max}$	$[180, 1e{-}6, 3600, 3600]$
Lower bound of the path constraints $\boldsymbol{g}_{min}$	$[-3600, -3600, -2e4, 500, 0, 120, 120, 120, 120]$
Upper bound of the path constraints $\boldsymbol{g}_{max}$	$[0, 0, 0, 2e4, 80, 3000, 3000, 3000, 3000]$

Table 5.5: *Problem dimensions of the 2-motor driving topology*

N_n	500	1000	1500	2000	2500	3000	3500	4000	4500
State variables N_x	15500	31000	46500	62000	77500	93000	108500	124000	139500
Control variables N_u	2000	4000	6000	8000	10000	12000	14000	16000	22500
Total NLP variables N_{NLP}	17505	35005	52005	70005	87505	12005	122505	140005	162005
Defects constraints N_{cd}	15469	30969	46469	61969	162469	92969	108469	123969	139469
Path constraints N_{cp}	4500	9000	13500	18000	22500	27000	31500	36000	49500
Non-zero Jacobians N_{def}	1340175	2682675	4025175	5367675	6710175	8052675	9395175	10737675	12080175

Table 5.6: *Bounds of the optimal powertrain design and control for the 4-motor driving topology*

Prameters	Values
Lower bound of the design vector $\boldsymbol{p}_{min}$	$[500, 1.2, 1]$
Upper bound of the design vector $\boldsymbol{p}_{max}$	$[15000, 6.5, 30]$
Lower bound of the control vector $\boldsymbol{u}_{min}$	$[-180, -5500, -5500, -5500, -5500]$
Upper bound of the control vector $\boldsymbol{u}_{max}$	$[180, 5500, 5500, 5500, 5500]$
Lower bound of the path constraints $\boldsymbol{g}_{min}$	$[-5500, -5500, -5500, -5500, -5e4, -5e4, -5e4, -5e4, 500, 0, 120, 120, 120, 120]$
Upper bound of the path constraints $\boldsymbol{g}_{max}$	$[0, 0, 0, 0, 0, 0, 0, 0, 2e4, 80, 3000, 3000, 3000, 3000]$

are still 1000, 2000 and 4500, the corresponding number of the variables, constraints and non-zero Jacobians are given as Table 5.7.

Table 5.7: *Problem dimensions of the 4-motor driving topology*

N_n	500	1000	1500	2000	2500	3000	3500	4000	4500
State variables N_x	15500	31000	46500	62000	77500	93000	108500	124000	139500
Control variables N_u	2500	5000	7500	10000	12500	15000	17500	20000	22500
Total NLP variables N_{NLP}	18004	36004	54004	72004	90004	108004	126004	144004	162004
Defects constraints N_{cd}	15469	30969	46469	61969	162469	92969	108469	123969	139469
Path constraints N_{cp}	6500	13000	19500	26000	32500	39000	45500	52000	58500
Non-zero Jacobians N_{def}	1435644	2873644	4311644	5749644	7187644	8625644	10063644	11501644	12939644

5.3.4 Optimization settings

As for the NLP solver, the constraints violation is set as $1e{-}7$ which is acceptable for all the constraints, the desired converge tolerance of the formulated problem is set as $1e{-}3$. The Jacobians of the defect and path constraints are evaluated with the central differential method, while the Hessian matrices are approximated with the inbuilt limited-memory BFGS approach of the NLP solver [103].

5.4 Smooth the control

The control variables might be extremely non-smooth when only bounds of the control variables are provided, for instance, the obtained steering wheel angles without any smooth operations are demonstrated as Figure 5.2.

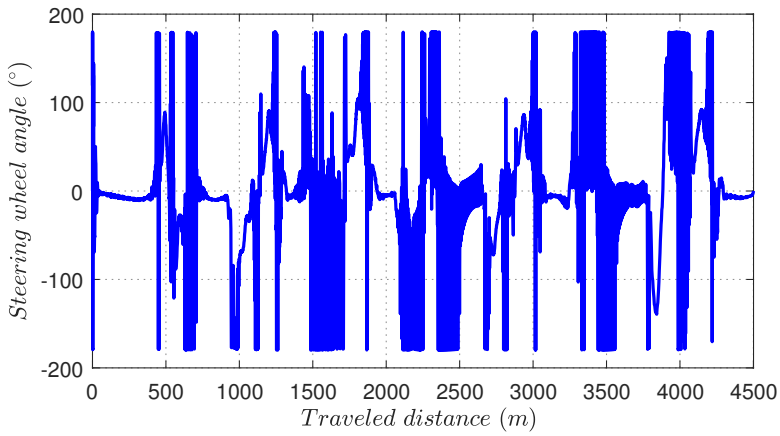

Figure 5.2: *Steering wheel angle without smooth operations*

In real case, it might be impossible for a driver to handle the steering wheel in such high frequency. In this work, a control smooth approach is proposed to obtain a reasonably smooth solution. The fundamental idea is to introduce a penalty function into the objective function which is denoted as:

$$J_{u,i} = \sum_{j=1}^{N_n-1} (\boldsymbol{u}_{j+1} - \boldsymbol{u}_j)^2, \; i = 1, \ldots, n_u \tag{5.24}$$

The analytical gradient of the control smooth function is derived as:

$$\frac{\partial J_{u,i}}{\partial \boldsymbol{u}_j} = \begin{cases} 2(\boldsymbol{u}_j - \boldsymbol{u}_{j+1}) & j = 1 \\ 2(2\boldsymbol{u}_j - \boldsymbol{u}_{j+1} - \boldsymbol{u}_{j-1}) & j = 2, \ldots, N_n - 1 \\ 2(\boldsymbol{u}_j - \boldsymbol{u}_{j-1}) & j = N_n \end{cases} \tag{5.25}$$

The final augmented objective function is denoted as:

$$J = \min(t_f + wJ_p + \sum_{i=1}^{n_u} w_{u,i} J_{u,i}) \tag{5.26}$$

where w is the penalty weight of J_p, $w_{u,i}$ is the penalty weight of the ith control smooth function. The improvements of the proposed control smooth approach are demonstrated in Appendix B.

5.5 Results

The computation of the optimal design and control problem was performed with MATLAB 2015a on a Linux OS based cluster with the Intel Xeon CPU X5355

@2.66GHz. The computation efficiency is approximately improved by 3 times with parallel computation using 8 CPUs on each node.

5.5.1 1-Motor driving topology

The obtained optimal design parameters $[\beta, n_b, \lambda, i_g]$, the maximum torque of the motor $T_{\max}$, the mass of the motor m_d and transmission m_g, the total mass of the powertrain m_p, the minimum lap time t_f, the number of iterations converging to the optimal solutions N_{iter}, the CPU time in IPOPT t_{IP} and the total CPU time t_c of the 1-motor driving topology with different collocation nodes N_n are presented in Table 5.8.

Table 5.8: *Parameters of the optimized one-motor powertrain*

N_n	β	n_b(Rpm)	$T_{\max}$(Nm)	λ	i_g	m_d(kg)	m_g(kg)	m_p(kg)	t_f(s)	N_{iter}	t_{IP}(h)	t_c(h)
500	6.43	1810	871	0.47	2.20	45.61	8.59	54.20	90.603	2250	0.22	1.86
1000	6.46	1790	880	0.47	2.10	45.96	8.16	54.13	90.459	2044	0.42	2.93
1500	6.48	1790	880	0.47	2.14	45.98	8.37	54.36	90.377	1721	0.48	3.31
2000	6.48	1786	882	0.47	2.12	46.06	8.29	54.35	90.324	1921	0.74	4.84
2500	6.49	1787	881	0.47	2.14	46.03	8.36	54.39	90.312	1265	0.61	3.88
3000	6.49	1786	882	0.47	2.14	46.05	8.36	54.41	90.293	1535	0.92	5.65
3500	6.49	1786	882	0.47	2.14	46.06	8.38	54.44	90.285	1382	0.99	5.96
4000	6.49	1785	882	0.47	2.14	46.07	8.38	54.45	90.281	1082	0.89	5.24
4500	6.49	1785	882	0.47	2.14	46.07	8.38	54.45	90.273	1350	1.25	7.31
4500NS	6.49	1784	883	0.46	2.11	46.10	8.29	54.39	90.282	3502	3.27	19.06

When the number of collocation nodes is more than 2000, it can be observed from Table 5.8 that the design parameters $[\beta, n_b, \lambda, i_g]$ stabilize around $[6.49, 1785, 0.47, 2.14]$, while the mass of the motor and transmission shift slightly around 46 kg and 8.38 kg respectively. The minimum lap time of each simulation can accurate to 0.1s which is around 90.28s. The CPU time increases approximately linearly with the increasing of the number of collocation nodes.

In addition, the simulation results without any scaling when $N_n = 4500$ are listed at the last row of Table 5.8, we can see that the saved computation time is about 12 hours compared with the simulation implemented with the proposed automatic scaling method. In fact, the simulations even failed to converge to a solution when there was no scaling method employed in some cases.

The optimal racing line of the case when $N_n = 4500$ is demonstrated in Figure 5.3, we can see that the electric race car entries in a corner the longitudinal velocity decreases and increases when it runs out the corner, and the minimum velocity at each corner increases with turning radius. From Figure 5.3, it is observed that the normal distance of the race car to the center line approaches to the supplied maximum/minimum bound at each corner in order to achieve a larger turning radius. The basic conclusions above are consistent with the real case.

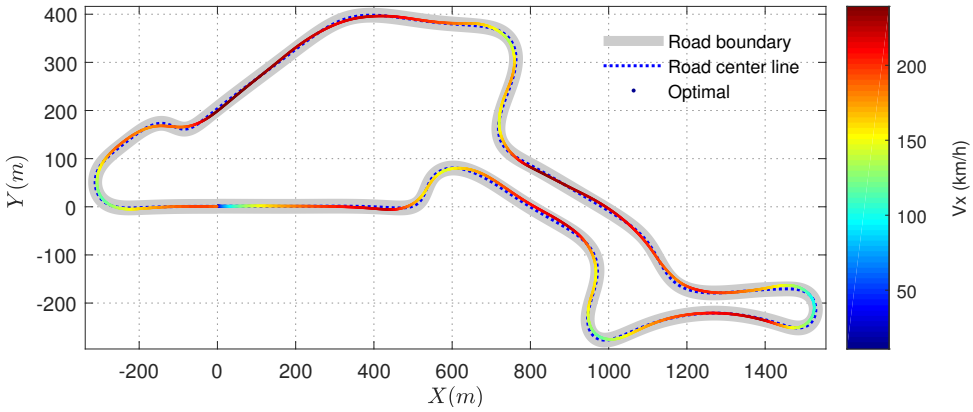

Figure 5.3: *The optimal racing line of 1-motor driving topology*

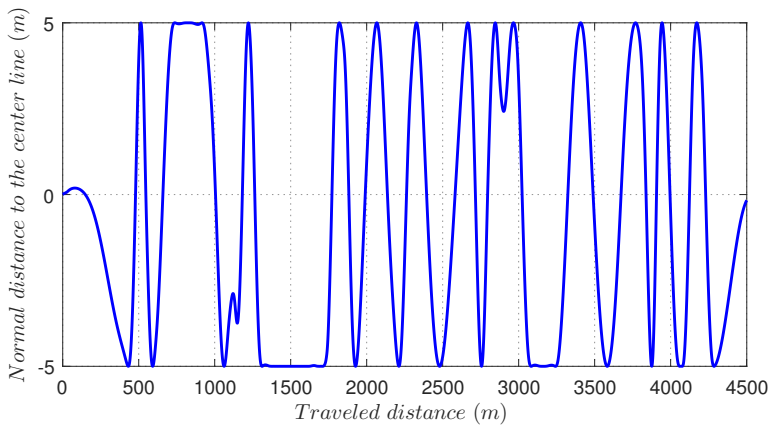

Figure 5.4: *The normal distance of the race car to the center line of the 1-motor driving topology*

The longitudinal velocity profile of the 1-motor driving electric race car is presented in Figure 5.5.

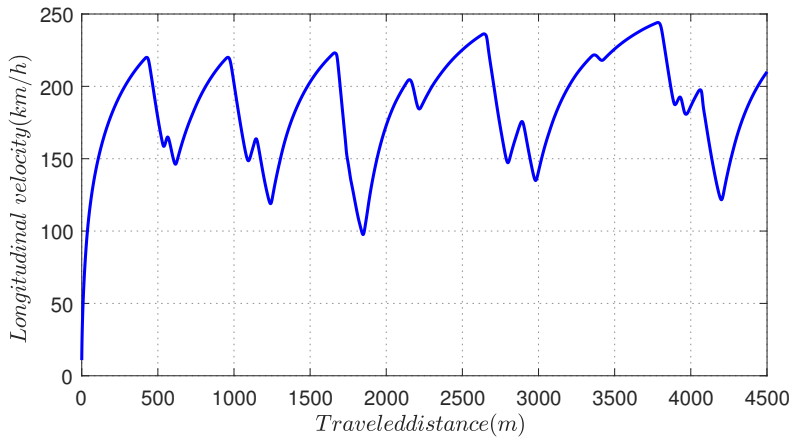

Figure 5.5: *The longitudinal velocity profile 1-motor driving topology*

For the 1-motor driving electric race car, the zoomed out trajectory at each corner,

the steering wheel angle, braking torque, propulsion torque, orientation angle, tire forces are also demonstrated separately in detail, the interested reader can refer to the figures in Appendix B, Section B.1.

5.5.2 2-Motor driving topology

The obtained optimal design parameters $[\beta, n_b, \lambda, i_g]$, the maximum torque of the motor $T_{\max}$, the mass of the motor m_d and transmission m_g, the total mass of the powertrain m_p, the minimum lap time t_f, the number of iterations converging to the optimal solutions N_{iter}, the CPU time in IPOPT t_{IP} and the total CPU time t_c of the 2-motor driving topology with different collocation nodes N_n are presented in Table 5.9.

Table 5.9: *Parameters of the optimized 2-motor powertrain*

N_n	β	n_b(Rpm)	$T_{\max}$(Nm)	λ	i_g	m_d(kg)	m_g(kg)	m_p(kg)	t_f(s)	N_{iter}	t_{IP}(h)	t_c(h)
500	6.47	2002	394	0.46	2.82	25.14	5.56	61.41	90.431	1771	0.17	1.46
1000	6.48	1895	415	0.47	2.31	26.20	4.39	61.18	90.348	2196	0.43	3.10
1500	6.48	1790	880	0.47	2.14	45.98	8.37	54.36	90.377	1721	0.48	3.31
2000	6.49	1883	418	0.46	2.27	26.32	4.30	61.24	90.255	1928	0.79	5.03
2500	6.49	1888	417	0.47	2.29	26.27	4.34	61.21	90.238	1832	0.93	5.94
3000	6.49	1889	417	0.46	2.30	26.26	4.36	61.24	90.225	2343	1.43	8.83
3500	6.49	1887	418	0.46	2.30	26.28	4.37	61.28	90.219	2124	1.57	9.31
4000	6.49	1887	418	0.47	2.30	26.28	4.35	61.27	90.216	1282	1.06	6.32
4500	6.49	1888	417	0.47	2.30	26.26	4.40	61.32	90.208	1788	1.71	10.13

When the number of collocation nodes is more than 2000, the design parameters $[\beta, n_b, \lambda, i_g]$ of the 2-motor driving topology stabilize around $[6.49, 1888, 0.47, 2.30]$. The mass of the motor, transmission and the race car are about 26.26 kg, 4.36 kg and 551.2 kg respectively. The minimum lap time of each simulation can accurate to 0.1s and shifts slightly around 90.21s.

The optimal racing line of the case when $N_n = 4500$ is demonstrated in Figure 5.6, while Figure 5.7 presents the longitudinal velocity profile.

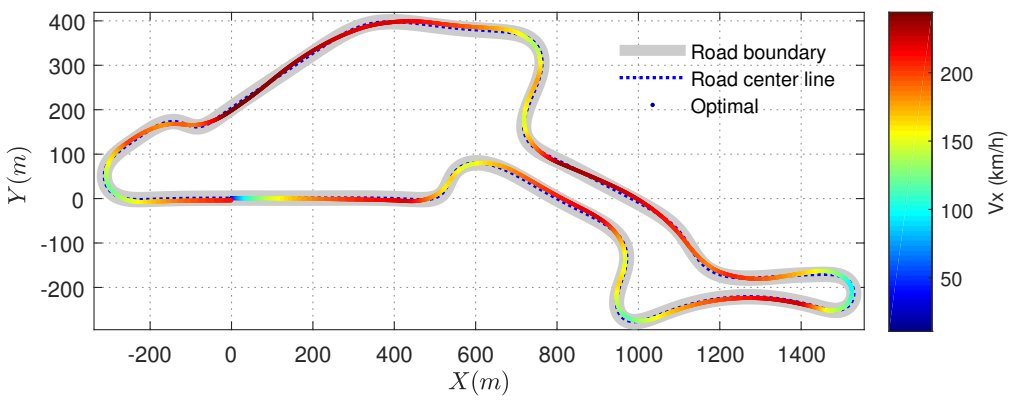

Figure 5.6: *The optimal racing line of 2-motor driving topology*

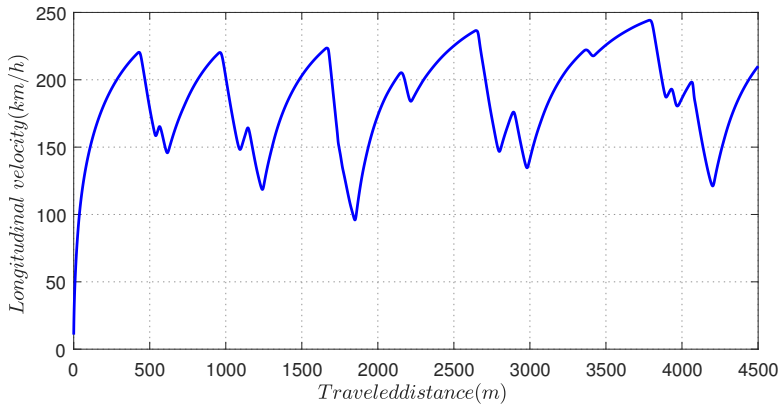

Figure 5.7: *The longitudinal velocity profile 2-motor driving topology*

The zoomed out trajectory at each corner, the steering wheel angles, braking torque, propulsion torque, normal distance to the center line, orientation angle, tire forces of the 2-motor driving electric race car are demonstrated separately in Appendix B, Section B.2.

5.5.3 4-Motor driving topology

The obtained optimal design parameters $[\beta, n_b, i_g]$, the maximum torque of each motor $T_{\max}$, the mass of each motor m_d and transmission m_g, the total mass of the race car m_t, the minimum lap time t_f, the number of iterations converging to the optimal solutions N_{iter}, the CPU time in IPOPT t_{IP} and the total CPU time t_c of the 4-motor driving topology are presented in Table 5.10. The optimal racing line of the case when N_n =

Table 5.10: *Parameters of the optimized 4-motor powertrain*

N_n	β	n_b(Rpm)	$T_{\max}$(Nm)	i_g	m_d(kg)	m_g(kg)	m_t(kg)	t_f(s)	N_{iter}	t_{IP}(h)	t_c(h)
500	6.49	2767	142	5.00	11.73	5.16	67.54	90.570	727	0.076	0.61
1000	6.50	2920	134	6.54	11.26	7.91	76.07	90.404	2768	0.59	4.11
1500	6.50	2783	142	5.05	11.68	5.21	67.55	90.383	1150	0.38	2.38
2000	6.50	2793	141	5.17	11.65	5.42	68.24	90.307	2118	0.94	5.73
2500	6.50	2782	142	5.03	11.68	5.19	67.47	90.288	1952	1.11	6.59
3000	6.50	2775	142	5.08	11.70	5.29	67.97	90.266	1625	1.12	6.44
3500	6.50	2774	142	5.06	11.70	5.20	67.60	90.256	1576	1.30	7.36
4000	6.50	2777	142	5.02	11.70	5.18	67.49	90.246	1834	1.71	9.63
4500	6.50	2777	142	5.01	11.70	5.15	67.39	90.241	1945	2.07	11.65

4500 is demonstrated in Figure 5.8, while Figure 5.9 presents the longitudinal velocity profile.

The zoomed out trajectory at each corner, the steering wheel angles, braking torque, propulsion torque, normal distance to the center line, orientation angle, tire forces of the

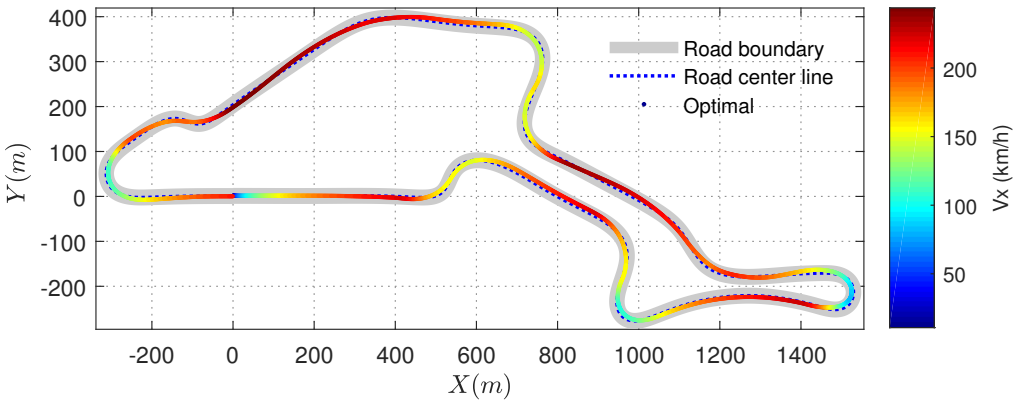

Figure 5.8: *The optimal racing line of 4-motor driving topology*

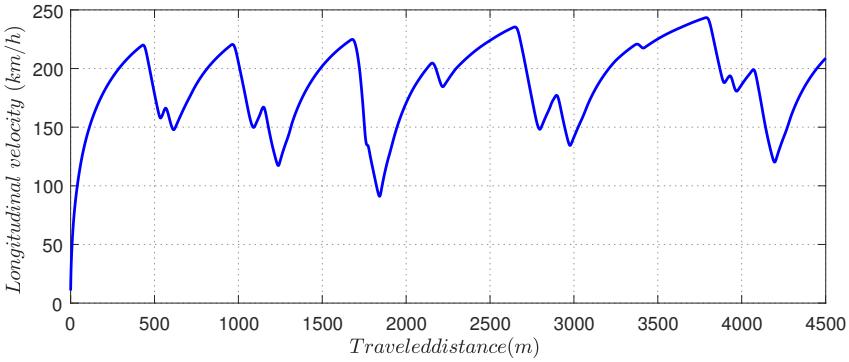

Figure 5.9: *The longitudinal velocity profile 4-motor driving topology*

4-motor driving electric race car are demonstrated separately in Appendix B, Section B.3.

5.6 Comparison of different propulsion systems

The obtained optimal design parameters of the electric powertrain are respectively summarized as $[\beta, n_b, \lambda, i_g] = [6.49, 1785, 0.47, 2.14]$, $[\beta, n_b, \lambda, i_g] = [6.49, 1888, 0.47, 2.30]$ and $[\beta, n_b, i_g] = [6.50, 2777, 5.02]$, the lap time are respectively 90.28s, 90.21s and 90.24s for the 1-motor, 2-motor and 4-motor driving topologies. As we can see, the 4-motor independent driving topology dose not achieve the best lap time performance as expected. One possible reason is that the mass center of the chassis is fixed and the average normal loads on the front wheels are smaller than the ones acted on the rear wheels. Thus, the distance from the mass center to front axle of the chassis l_f is then added into the design parameters, the design parameters of the 4-motor driving electric race car become into $[\beta, n_b, i_g, l_f]$. The newly obtained optimal design parameters are $[6.5, 3137, 8.33, 1.343]$, compared with the data in Table 5.1, the mass center of the race car is move forward by 0.252 m. Figure 5.10 demonstrates that the normal loads of the race car with the optimized mass center distribute more reasonably, the normal loads acted on the front tires increases while ones acted on the rear wheels decreases.

Before optimizing the location of the mass center, the average normal load acted on the front axles is 2443 N, while the one on the rear axle is 4574 N. The average positive longitudinal tire forces of the front axle and rear axle are respectively 1026 N and 1284 N.

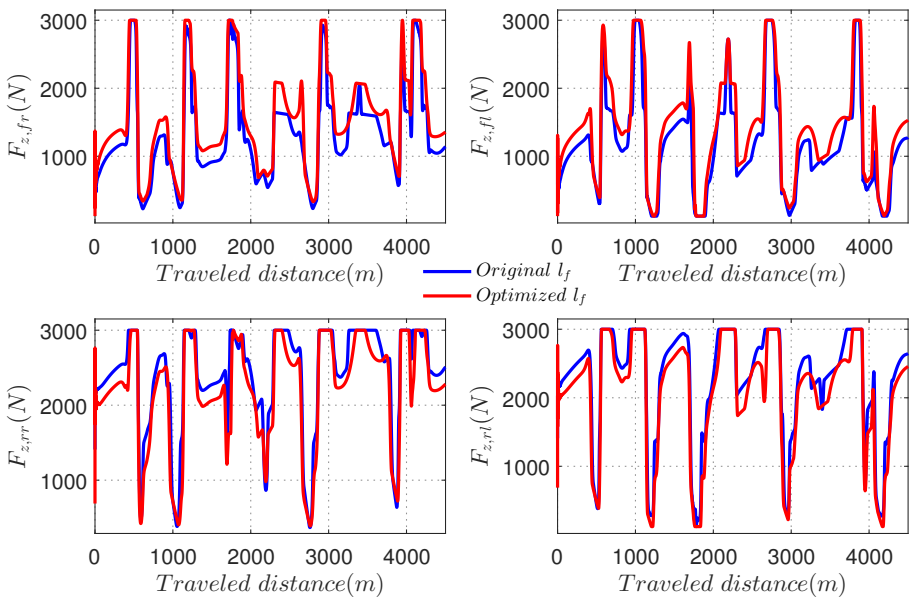

Figure 5.10: *The normal loads acted on the four tires*

After optimization of the mass center location, the average normal loads acted on the front and rear axles become into 2912 N and 4207 N. The average positive longitudinal tire forces of the front axle and rear axle are respectively 1016 N and 1295 N. The obtained lap time is $t_f = 87.623$, which is improved about 2.6s compared with the obtained results in Table 5.10.

As we can see, the average normal loads distributed on the front and rear tires are not equal in the test maneuver. In fact, in almost all the maneuvers the normal loads do not distribute equally on the four wheels. However, currently, in almost all the developed four wheel driving electric race cars, the four motors and transmissions are implemented with the same physical parameters, which is also the assumption of this work. In this case, considering that the maximum power of the race car is limited by the race event, the power of the motors mounted on the front axle is a kind of surplus while the one of the rear axle is deficit.

Figure 5.11 presents the comparison of the propulsion power of the 4-motor driving topologies, we can see that the total propulsion power output by the motors of the two solutions are actually very similar. The only possible reason for the lap time improvement is that the dissipated friction power of the tires is reduced and the propulsion power for accelerating the race car is thus increased after optimizing the position of the mass center.

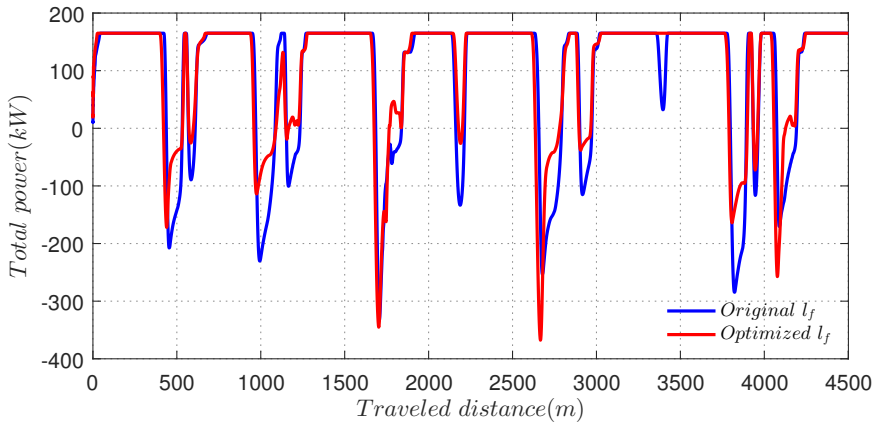

Figure 5.11: *The propulsion power of the 4-motor driving topology*

Based on the above analysis, a possible 4-motor driving topology can have two pairs of motors and transmissions mounted on the front and rear axles respectively. Each pair has two same motors and transmissions but different with the other pair. The updated design parameters of the 4-motor driving electric powertrain are given as:

$$\boldsymbol{p} = [\beta_f, n_{bf}, i_{gf}, \beta_r, n_{br}, i_{gr}, P_r, l_f] \tag{5.27}$$

where β_f, n_{bf}, i_{gf} and β_r, n_{br}, i_{gr} are respectively the CPSR ratio, base speed of the motors and the speed ratio of the transmissions mounted on the front and rear axles, P_r is the maximum power of the rear motors.

The final obtained design parameters of the 4-motor driving electric powertrain when $N_n = 3000$ are demonstrated in Table 5.11. The mass of each motor, transmission on the front and rear axles are respectively 7.47 kg, 1.94 kg, 17.33 kg and 4.95kg. The total mass of the powertrain is 63.6 kg.

Table 5.11: *Parameters of the optimized 4-motor driving electric powertrain with two different pairs of motors*

Paramters	β_f	n_{bf}	i_{gf}	β_r	n_{br}	i_{gr}	P_r (kW)	l_f(m)	t_f(s)	N_{iter}	t_{IP}(h)	t_c(h)
Values	6.5	2501	4.02	6.5	2473	3.59	125	1.343	87.278	1759	1.35	9.06

We can see that the lap time is further improved compared with the previous design by 0.34s. The optimal powertrain of the 4-motor driving topology is with two small power motors (20 kW) driving the front wheels and two bigger motors (62.5 kW) driving the rear wheels. The distance from the mass center to the front axle is 1.343m. The average positive longitudinal tire forces of the front axle and rear axle are respectively 502 N and 1881 N.

The normal loads acted on each wheel are constrained within $[120\ N, 3000\ N]$, when it is enlarged into $[120\ N, 5000\ N]$, the obtained lap time of the 1-motor, 2-motor, 4-motor driving topologies are respectively 87.24s, 87.11s and 87.18s. The lap time of the 4-motor driving topology is 85.47s after the optimization of the location of the mass center, using two different pairs of motors and transmissions the lap time can be further

reduced to 85.13s. The conclusion is consistent with the one obtained with normal loads constraint $[120\ N, 3000\ N]$.

CHAPTER 6

Conclusions and future work

The preceding chapters developed a 14-DOF vehicle model including the suspension model, tire model and path following model, then formulated and solved the complicated highly non-linear optimal design and control problems for different electric powertrain topologies with the self-developed software package GDYNOPT. At last, the obtained results are demonstrated, compared and analyzed. In this chapter, the main findings, strengths and weaknesses will be reviewed. The recommendations for future work and developments in vehicle design and control and the developed software package GDYNOPT will also be presented respectively.

6.1 Conclusions

6.1.1 The optimal design and control of the race car

By analyzing the data obtained in last section, we can conclude that when the mass center of the race car is fixed, the achieved minimum lap time of the three topologies are similar, and the 4-motor driving topology does not guarantee the best lap time performance. When two different pairs of motors and transmissions are implemented on the front and rear axles, the lap time performance of the 4-motor driving topology can be significantly improved with also the location of the mass center as design parameters.

The obtained design parameters can serve as the reference for the motor and transmission design of the electric race car, while the optimal control results can be used as the benchmark to develop and to evaluate the closed loop control strategies.

Moreover, the obtained racing line and steering wheel angle can be used to train the race car driver in a car simulator.

In addition, the methodology proposed in this work can also be applied in the design and control of the common type ICE vehicles, EVs and HEVs with different driving profiles and objective functions.

6.1.2 The developed GDYNOPT

The most important features of a optimal control software are the robustness and converging time. The robustness of the developed GDYNOPT is validated by using different number of collocation nodes for different powertrain topologies. The converging performance of GDYNOPT is significantly improved by introducing the proposed automatic scaling method.

6.2 Future work

6.2.1 The optimal design and control of the race car

The computation time of the optimal design and control of the race car is still relatively long. The main reason is that the for loops are introduced to calculate the derivatives (Equation (3.29)) in the vehicle dynamic model since MATLAB does not support inverse operation for 3-D matrices. There are two approaches for further improvements, the first is to take advantage GPU computing while the other is to derive the expressions of each derivative with the symbolic toolbox of MATLAB directly, however, both approaches are limited by the graphics card or memory of the accessible computers currently.

This work is focused on the powertrain design and control to improve the lap time performance of the electric race car. In fact, the lap time of a race car is influenced by also other parameters, such as the parameters of the anti-roll bar, the spring and damper of the suspension, the location of the mass center, the mounted positions of the motors, the location of aerodynamic downforce and drag force, the toe angles and the camber angles. The future work can introduce these parameters as design parameters for further improvement.

6.2.2 The developed GDYNOPT

In this work, the local collocation method is implemented with automatic scaling approach to solve the formulated optimal design and control problem. Future work may explore possible improvement in the converging time and accuracy with the global collocation methods. More case studies will be implemented to test the robustness and convergence performance of GDYNOPT.

Moreover, the parallel computing currently is only accessible to the CPUs on one node of the cluster currently, further improvement maybe achieved by using of more CPUs across different nodes in the future work.

APPENDIX $\mathcal{A}$

Tire model

A.1 Longitudinal tire force

The longitudinal tire force in the contact point is:

$$F_{x,i} = (D_{x,i}\sin(C_{x,i}\arctan(B_{x,i}\kappa_i - E_{x,i}(B_{x,i}\kappa_{xi} - \arctan(B_{x,i}\kappa_{x,i})))) + SV_{x,i})G_{x\alpha,i} \quad \text{(A.1)}$$

with the following coefficients:

$$\begin{cases} D_{x,i} & = \mu_{x,i}F_{z,i} \\ \mu_{x,i} & = (P_{Dx1,i} + P_{Dx2,i}df_{z,i})(1 - P_{Dx3,i}\gamma_i^2)\lambda_{\mu x,i} \\ C_{x,i} & = P_{cx1,i}\lambda_{cx,i} \\ B_{x,i} & = \frac{K_{x\kappa,i}}{C_{x,i}D_{x,i}} \\ K_{x\kappa,i} & = (P_{Kx1,i} + P_{Kx2,i}df_{z,i})exp(P_{Kx3,i}df_{z,i})\lambda_{Kx,i}F_{z,i} \\ \kappa_{x,i} & = \kappa_i + S_{Hx,i} \\ E_{x,i} & = (P_{Ex1,i} + P_{Ex2,i}df_{z,i} + P_{Ex3,i}df_{z,i}^2)(1 - P_{Ex4,i}\mathrm{sgn}(\kappa_{x,i}))\lambda_{Ex,i} \\ S_{Hx,i} & = (P_{Hx1,i} + P_{Hx2,i}df_{z,i})\lambda_{Hx,i} \\ SV_{x,i} & = (P_{Vx1,i} + P_{Vx2,i}df_{z,i})\lambda_{Vx,i}\lambda_{\mu x,i}F_{z,i} \end{cases} \quad \text{(A.2)}$$

where the meaning of all the symbols can be found in Ref. [100].

When the tire is used in pure slip mode $G_{x\alpha,i} = 1$, while in the combined slip mode, it is denoted as:

$$G_{x\alpha,i} = \frac{\cos[C_{x\alpha,i}\arctan\{B_{x\alpha,i}\alpha_{s,i} - E_{x\alpha,i}(B_{x\alpha,i}\alpha_{s,i} - \arctan(B_{x\alpha,i}\alpha_{s,i}))\}]}{\cos[C_{x\alpha,i}\arctan\{B_{x\alpha,i}S_{Hx\alpha,i} - E_{x\alpha,i}(B_{x\alpha}S_{Hx\alpha,i} - \arctan(B_{x\alpha,i}S_{Hx\alpha,i}))\}]} \tag{A.3}$$

with:

$$\begin{cases} B_{x\alpha,i} & = (r_{Bx1,i} + r_{Bx3,i}\gamma_i^2)cos[arctan\{r_{Bx2,i}\}]\lambda_{x\alpha,i} \\ C_{x\alpha,i} & = r_{Cx1,i} \\ \alpha_{s,i} & = \alpha_{F,i} + S_{Hx\alpha,i} \\ E_{x\alpha,i} & = r_{Ex1,i} + r_{Ex2,i}df_{z,i} \\ S_{Hx\alpha,i} & = r_{Hx1,i}. \end{cases} \tag{A.4}$$

where the meaning of all the symbols can be found in Ref. [100].

A.2 Lateral tire force

The lateral force in the contact point is:

$$F_{y,i} = (D_{y,i}\sin(C_{y,i}\arctan(B_{y,i}\alpha_i - E_{y,i}(B_{y,i}\alpha_i - \arctan(B_{y,i}\alpha_i))))) + S_{Vy,i})G_{y\kappa,i} + S_{Vy\kappa,i} \tag{A.5}$$

with the following coefficients:

$$\begin{cases} D_{y,i} & = \mu_{y,i}F_{z,i} \\ \mu_{y,i} & = (P_{Dy1,i} + P_{Dy2,i}df_{z,i})(1 - P_{Dy3,i}\gamma_i^2)\lambda_{\mu y,i} \\ C_{y,i} & = P_{cy1,i}\lambda_{cy,i} \\ B_{y,i} & = \frac{K_{y\alpha,i}}{C_{y,i}D_{y,i}} \\ K_{y\alpha,i} & = P_{Ky1,i}F_{z0,i}\sin\left[P_{Ky4,i}\arctan\left\{\frac{F_{z,i}}{(P_{Ky2,i}+P_{Ky5,i}\gamma_i^2)F_{z0,i}}\right\}\right](1 - P_{Ky3,i}|\gamma_i|)\lambda_{Ky\alpha,i} \\ \alpha_{y,i} & = \alpha_i + S_{Hy,i} \\ S_{Hy,i} & = S_{Hy0,i} + S_{Hy\gamma,i} \\ S_{Hy0,i} & = (P_{Hy1,i} + P_{Hy2,i}df_{z,i})\lambda_{Hy,i} \\ S_{Hy\gamma,i} & = \frac{K_{y\gamma,i}\gamma_i - S_{Vy\gamma,i}}{K_{y\alpha,i}} \\ K_{y\gamma,i} & = (P_{Vy3,i} + P_{Vy4,i}df_{z,i})F_{z,i} \\ E_{y,i} & = (P_{Ey1,i} + P_{Ey2,i}df_{z,i})(1 + P_{Ey5,i}\gamma_i^2 - (P_{Ey3,i} + P_{Ey4,i}\gamma_i)sgn(\alpha_{y,i}))\lambda_{Ey,i} \\ S_{Vy,i} & = S_{Vy0,i} + S_{Vy\gamma,i} \\ S_{Vy0,i} & = (P_{Vy1,i} + P_{Vy2,i}df_{z,i})\lambda_{Vy,i}\lambda_{\mu y,i}F_{z,i} \\ S_{Vy\gamma,i} & = (P_{Vy3,i} + P_{Vy4,i}df_{z,i})\lambda_{K\gamma y,i}\lambda_{\mu y,i}F_{z,i}\gamma_i. \end{cases} \tag{A.6}$$

where the meaning of all the symbols can be found in Ref. [100].

When there is pure slip between the road and tires, $S_{Vy\kappa,i} = 0$, $G_{y\kappa,i} = 0$, while there

is combined slip, these parameters are:

$$\begin{cases} S_{V y\kappa,i} &= D_{V y\kappa,i} \sin\left(r_{Vy5,i} \arctan(r_{Vy6,i}\kappa_{x,i})\right) \lambda_{V y\kappa,i} \\ D_{V y\kappa,i} &= \mu_{y,i}(r_{Vy1,i} + r_{Vy2,i} df_{z,i} + r_{Vy3,i}\gamma_i) \cos\left(\arctan(r_{Vy4,i}, \alpha_{y,i})\right) \\ G_{y\kappa,i} &= \dfrac{\cos[C_{y\kappa,i} \arctan\{B_{y\kappa,i}\alpha_{s,i} - E_{y\kappa,i}(B_{y\kappa,i}\alpha_{s,i} - \arctan(B_{y\kappa,i}\alpha_{s,i}))\}]}{\cos[C_{y\kappa,i} \arctan\{B_{y\kappa,i}S_{Hy\kappa,i} - E_{y\kappa,i}(B_{y\kappa}S_{Hx\alpha,i} - \arctan(B_{y\kappa,i}S_{Hx\alpha,i}))\}]} \\ B_{y\kappa,i} &= (r_{By1,i} + r_{By4,i}\gamma_i^2) \cos[\arctan\{r_{By2,i}(\alpha_i - r_{By3,i})\}]\lambda_{y\kappa,i} \\ C_{y\kappa,i} &= r_{Cy1,i} \\ E_{y\kappa,i} &= r_{Ey1,i} + r_{Ey2,i} df_{z,i} \\ S_{Hy\kappa,i} &= r_{Hy1,i} + r_{Hy2,i} df_{z,i} \end{cases} \tag{A.7}$$

where the meaning of all the symbols can be found in Ref. [100].

A.3 Vertical tire force

The vertical tire force $F_{zt,i}$ between the ground and the tire is calculated with the tire vertical stiffness $K_{t,i}$ and damping ratio $C_{t,i}$,

$$F_{zt,i} = K_{t,i}\Delta R_{,i} + C_{t,i}\Delta\dot{R}_i \tag{A.8}$$

where ΔR_i is the vertical deformation of each tire and can be calculated with the original radius of the tire and vertical position of the wheel center.

$$\Delta R_i = R_{0,i} - z_{u,i} \tag{A.9}$$

A.4 Overturning moment

The overturing moment M_x is denoted as

$$M_x = -R_0 F_Z \lambda_{Mx}(Q_{Sx1}\lambda_{Mx} - Q_{Sx2}\gamma + \frac{Q_{Sx3}F_y}{F_{z0}}) \tag{A.10}$$

where the meaning of all the symbols can be found in Ref. [100].

A.5 Rolling resistance moment

For tire data where FITTYP is equal to 5, the rolling resistance M_y is denoted as

$$M_y = R_0(S_{Vx} + K_x S_{Hx}) \tag{A.11}$$

Otherwise:

$$M_y = -R_0 F_Z \lambda_{My}\{Q_{sy1} + Q_{sy2}\frac{F_x}{F_{z0}} + Q_{sy3}|\frac{V_x}{V_{ref}}| + Q_{sy4}(\frac{V_x}{V_{ref}})^4\}. \tag{A.12}$$

where the meaning of all the symbols can be found in Ref. [100].

A.6 Self aligning moment

The self aligning momeent M_z is denoted as

$$M_{z0} = -tF_{y0} + M_{zr} \tag{A.13}$$

with

$$\begin{cases} S_{Ht} & = Q_{Hz1} + Q_{Hz2}df_z + (Q_{Hz3} + Q_{Hz4}df_z)\lambda_z \\ \alpha_t & = \alpha + S_{Ht} \\ t & = D_t cos[C_t arctan\{B_t\alpha_t - E_t(B_t\alpha_t - arctan(B_t\alpha_t))\}]cos(\alpha) \\ M_{zr} & = D_r cos[C_r arctan(B_r\alpha_r)]cos(\alpha) \\ \alpha_r & = \alpha + S_{Hf} \\ B_t & = (Q_{Bz1} + Q_{Bz2}df_z + Q_{Bz3}df_z^2)(1 + Q_{Bz4}\gamma^2 + Q_{Bz5}|\gamma_z|)\lambda_{Ky}/\lambda_{\mu y} \\ \gamma_z & = \gamma\lambda_{\gamma_z} \\ C_t & = Q_{Cz1} \\ D_t & = F_z(Q_{Dz1} + Q_{Dz2}df_z)(1 + Q_{Dz3}\gamma_z^2 + Q_{Dz4}\gamma_z^2)\frac{R_0}{F_{z0}}\lambda_t \\ E_t & = (Q_{Ez1} + Q_{Ez2}df_z + Q_{Ez3}df_z)\{1 + (Q_{Ez4} + Q_{Ez5}\gamma_z)((\frac{2}{\pi}arctan(B_tC_t\alpha_t)) \\ B_r & = (Q_{Bz9}\frac{\lambda_{Ky}}{\lambda_{\mu z}} + Q_{Bz10}B_y c_y) \\ C_r & = 1 \\ D_r & = F_z[(Q_{Dz6} + Q_{Dz7}df_z)\lambda_r + (Q_{Dz8} + Q_{Dz9}df_z)\lambda_z]R_0\lambda_{\mu y} \end{cases} \tag{A.14}$$

where the meaning of all the symbols can be found in Ref. [100].

APPENDIX B

Simulation results

B.1 1-Motor driving topology

For the 1-motor driving electric race car, the zoomed out trajectory at each corner of Figure 5.3 is respectively demonstrated in Figure B.1-Figure B.5.

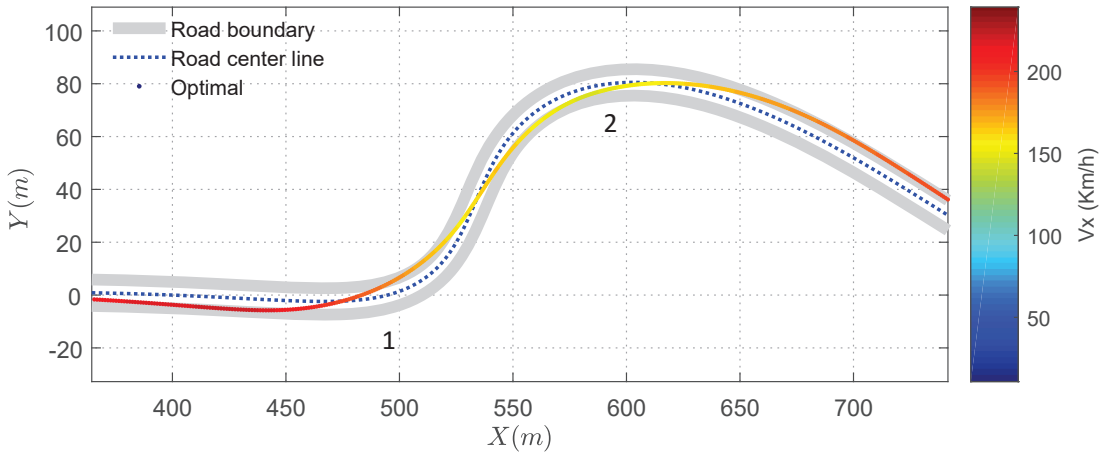

Figure B.1: *The optimal racing line of 1-motor driving topology: corner 1,2*

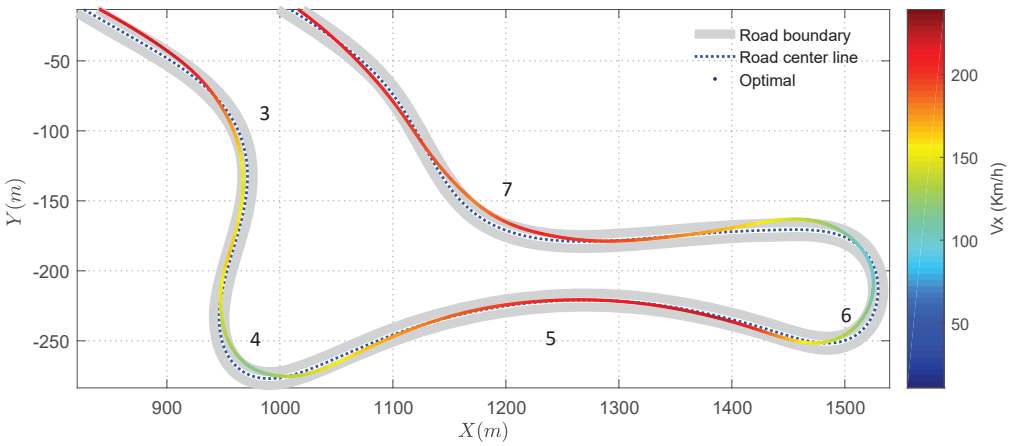

Figure B.2: *The optimal racing line of 1-motor driving topology: corner 3-7*

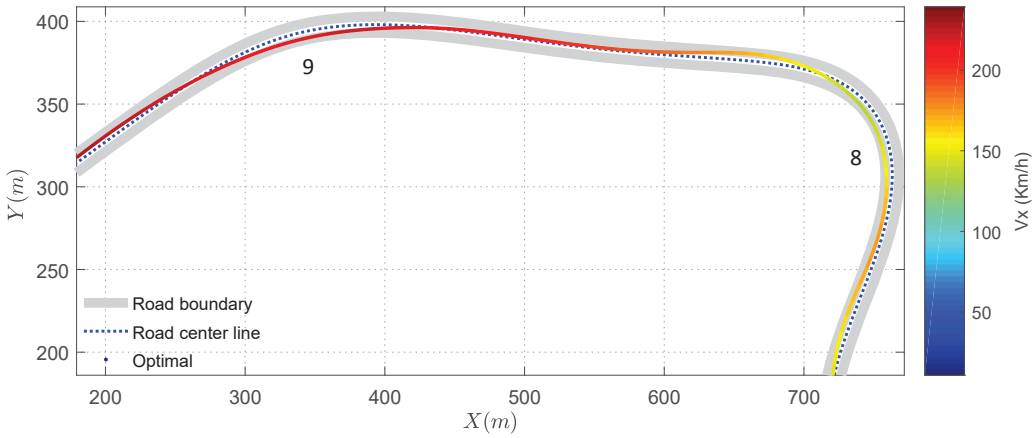

Figure B.3: *The optimal racing line of 1-motor driving topology: corner 8,9*

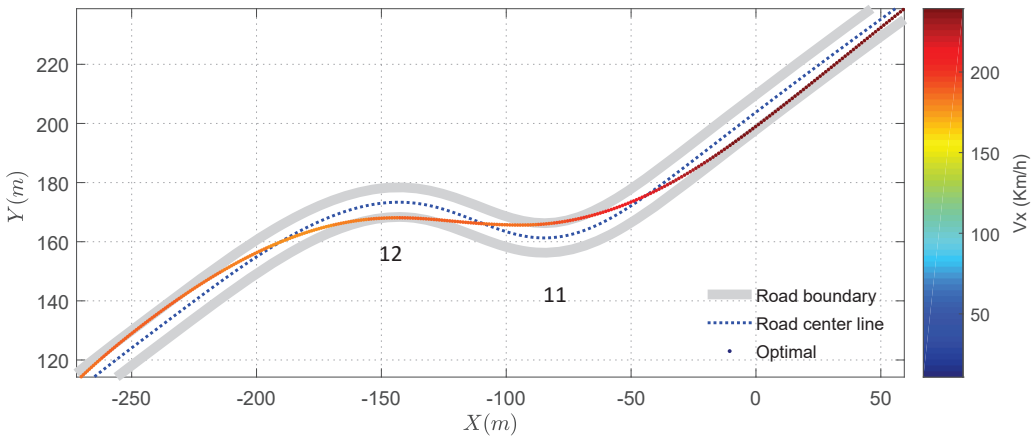

Figure B.4: *The optimal racing line of 1-motor driving topology: corner 10,11*

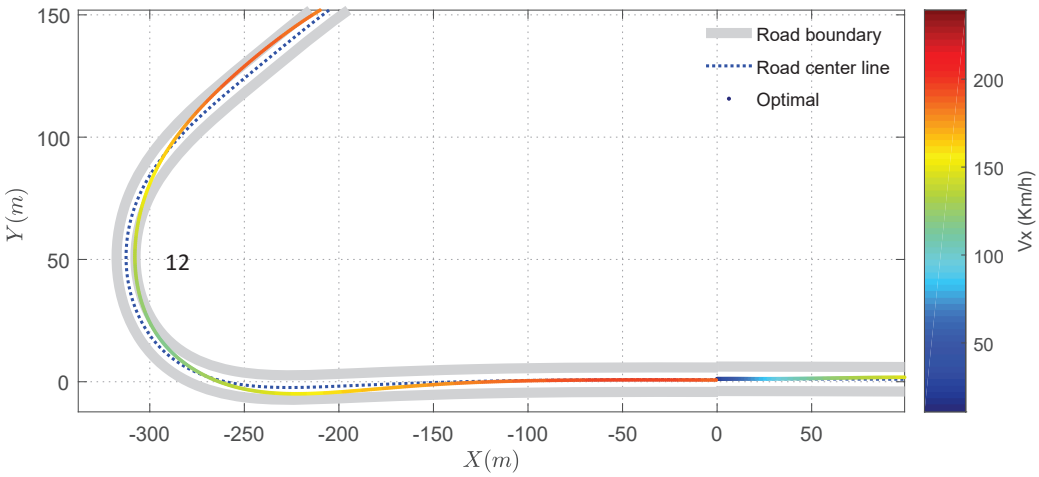

Figure B.5: *The optimal racing line of 1-motor driving topology: corner 12*

The steering wheel angle,orientation angle, braking torque and propulsion torque are demonstrated separately as Figure B.6, Figure B.7, Figure B.8 and Figure B.9.

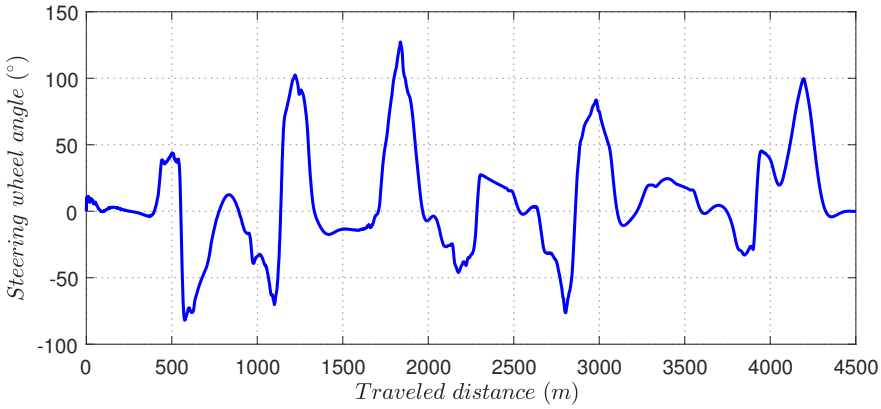

Figure B.6: *Steering wheel angle of the 1-motor driving topology*

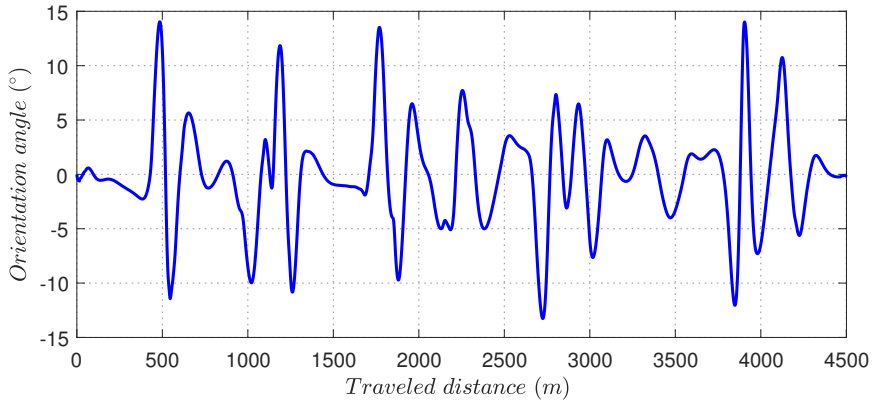

Figure B.7: *The orientation angle of the 1-motor driving topology*

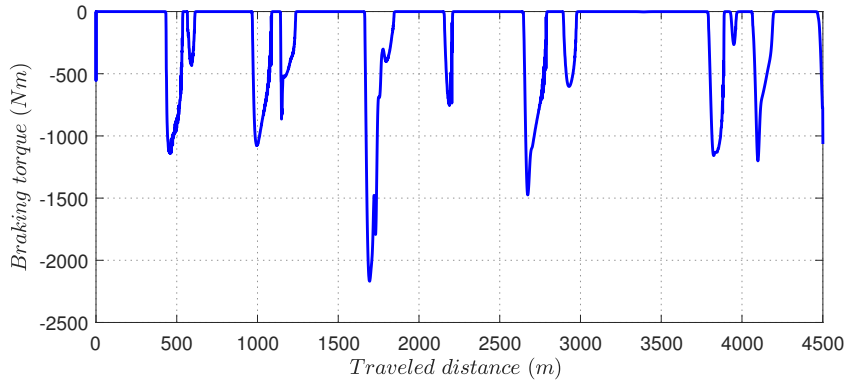

Figure B.8: *Braking torque of the 1-motor driving topology*

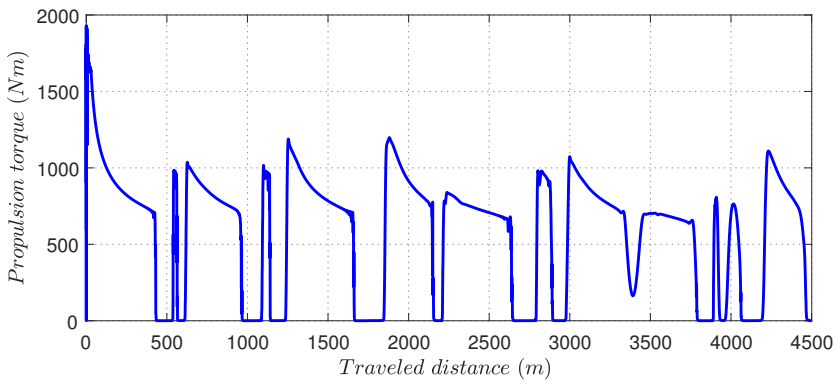

Figure B.9: *Propulsion torque of the 1-motor driving topology*

Longitudinal tire forces and lateral tire forces of the 1-motor driving topology are demonstrated as Figure B.10 and Figure B.11 respectively.

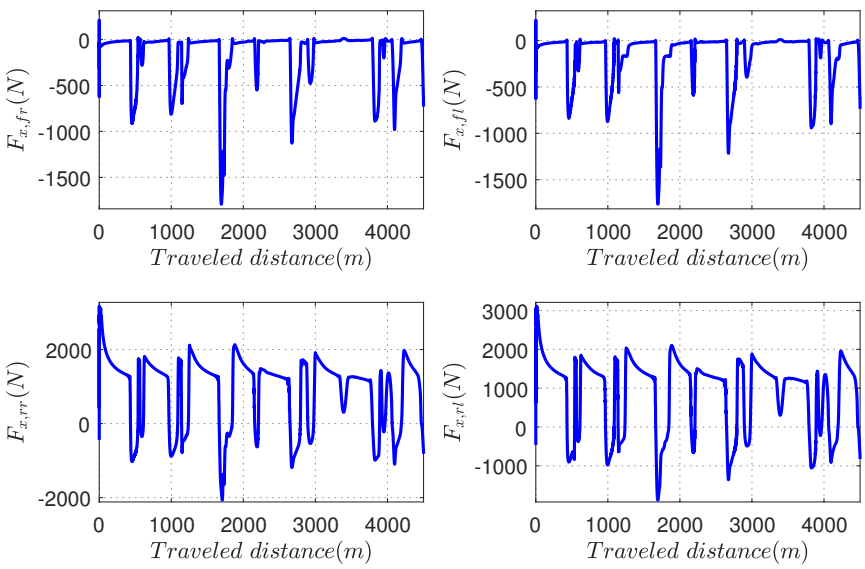

Figure B.10: *Longitudinal tire forces of the 1-motor driving topology*

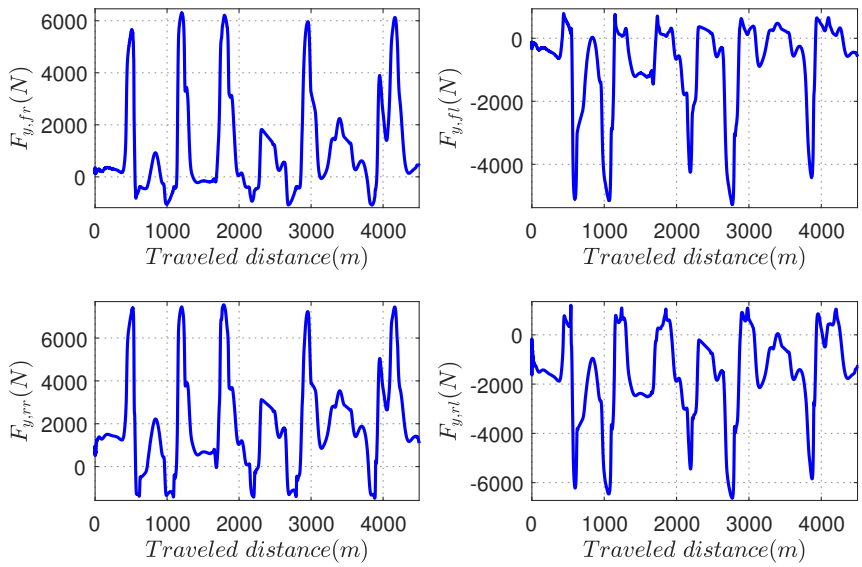

Figure B.11: *Lateral tire forces of the 1-motor driving topology*

B.2 2-Motor driving topology

For the 2-motor driving electric race car, the zoomed out trajectory at each corner of Figure 5.6 are demonstrated in Figure B.12-Figure B.16.

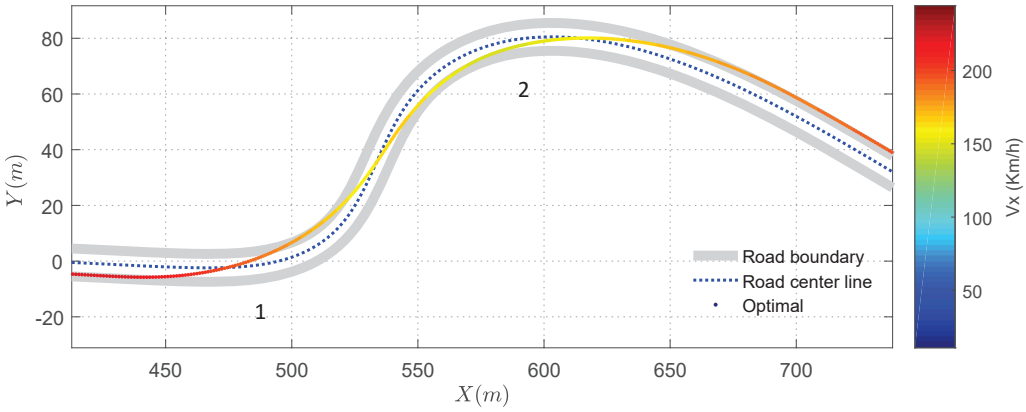

Figure B.12: *The optimal racing line of 2-motor driving topology: corner 1,2*

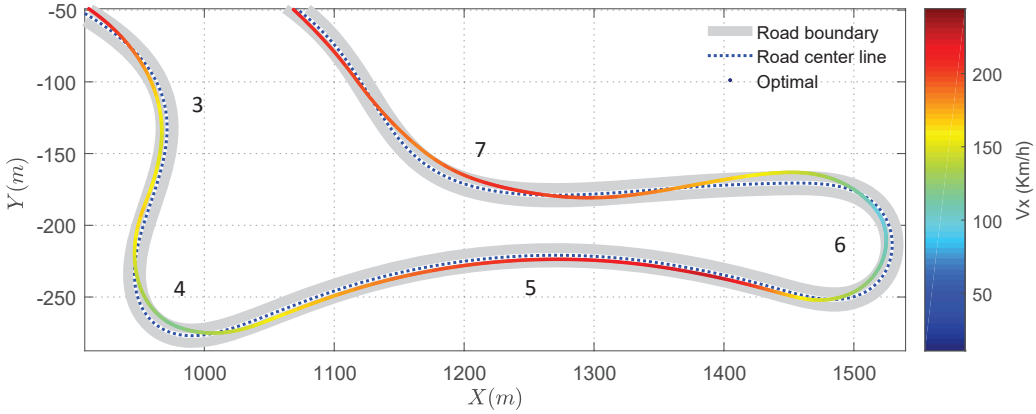

Figure B.13: *The optimal racing line of 2-motor driving topology: corner 3:7*

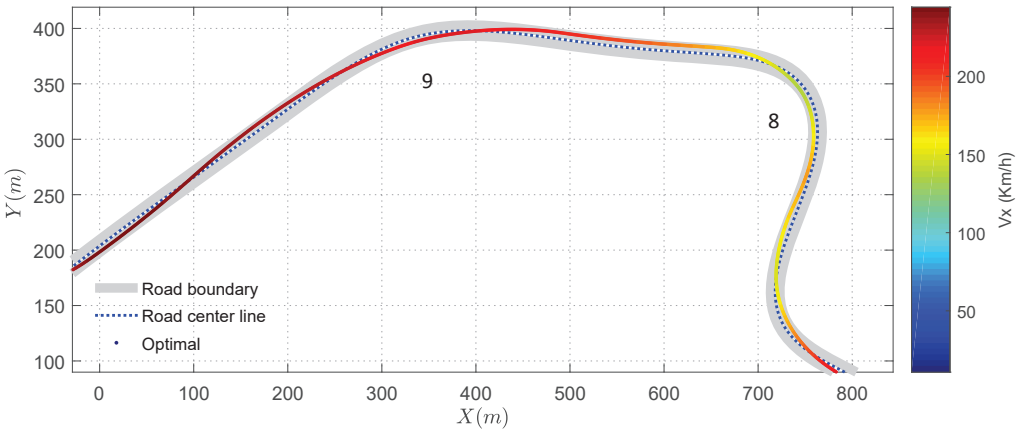

Figure B.14: *The optimal racing line of 2-motor driving topology: corner 8,9*

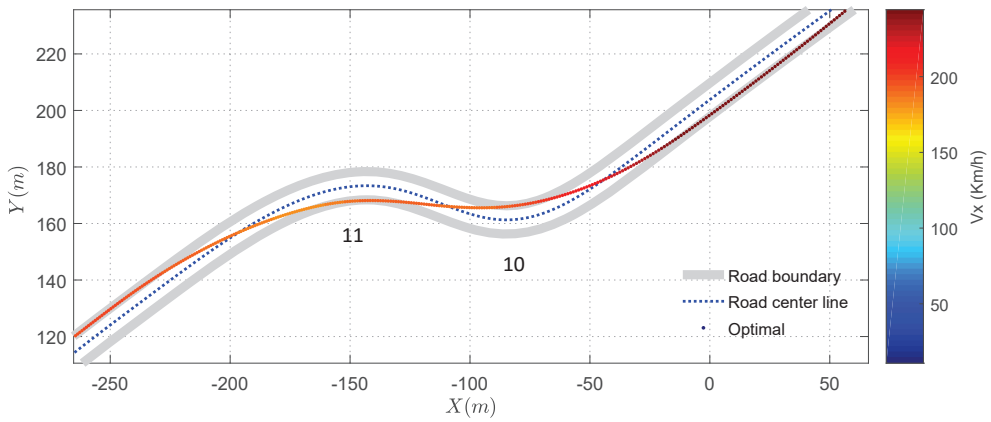

Figure B.15: *The optimal racing line of 2-motor driving topology: corner 10,11*

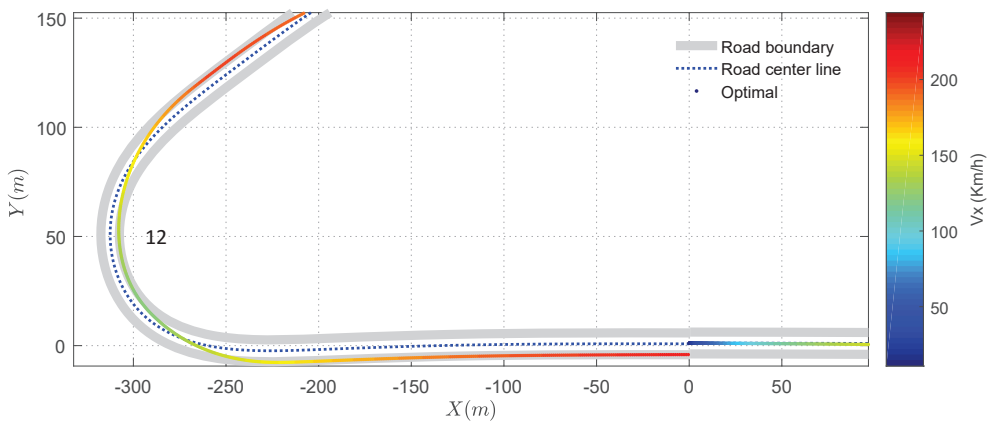

Figure B.16: *The optimal racing line of 2-motor driving topology: corner 12*

The steering wheel angle, orientation angle, braking torque and propulsion torque are demonstrated separately as Figure B.17, Figure B.18, Figure B.19 and Figure B.20.

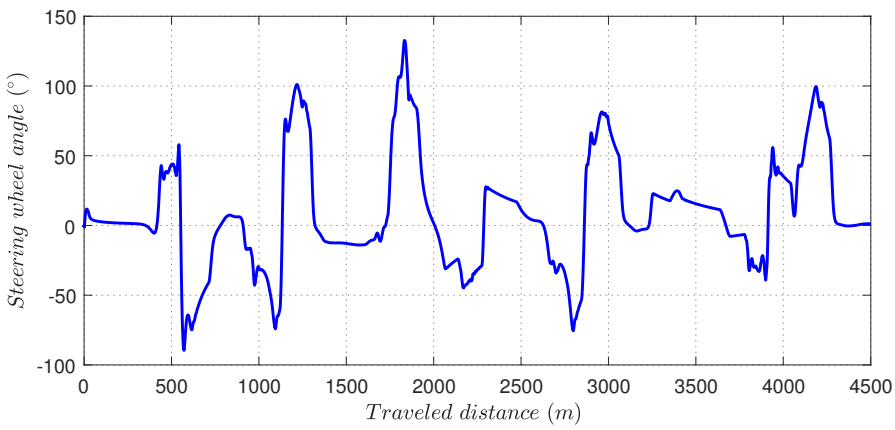

Figure B.17: *Steering wheel angle of the 2-motor driving topology*

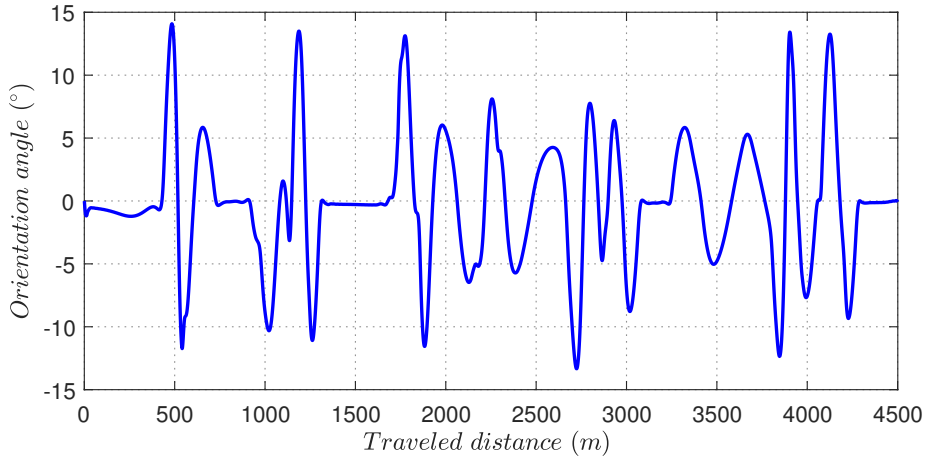

Figure B.18: *The orientation angle of the 1-motor driving topology*

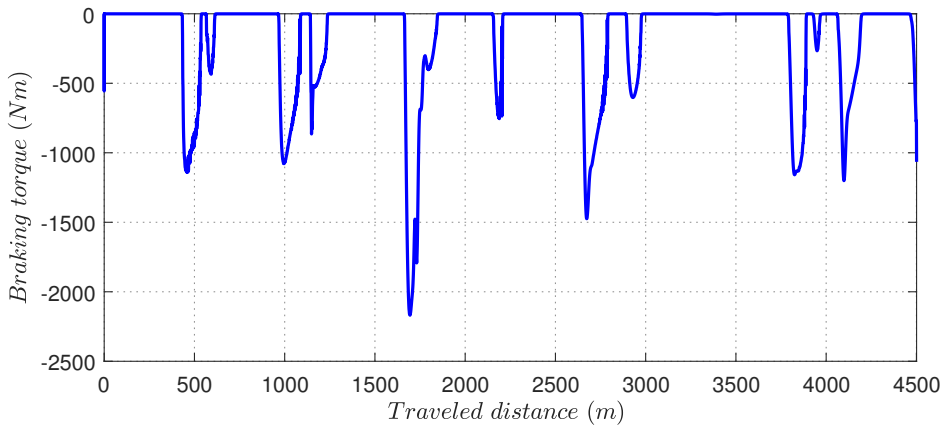

Figure B.19: *Braking torque of the 2-motor driving topology*

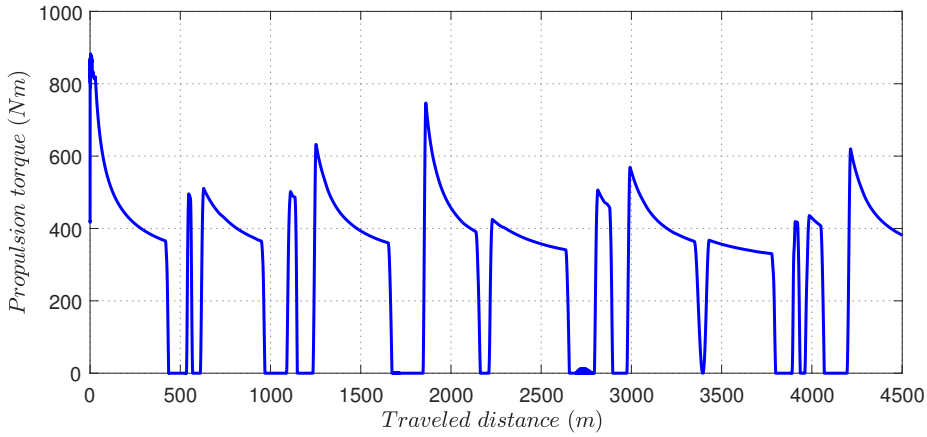

Figure B.20: *Propulsion torque on the rear right wheel of the 2-motor driving topology*

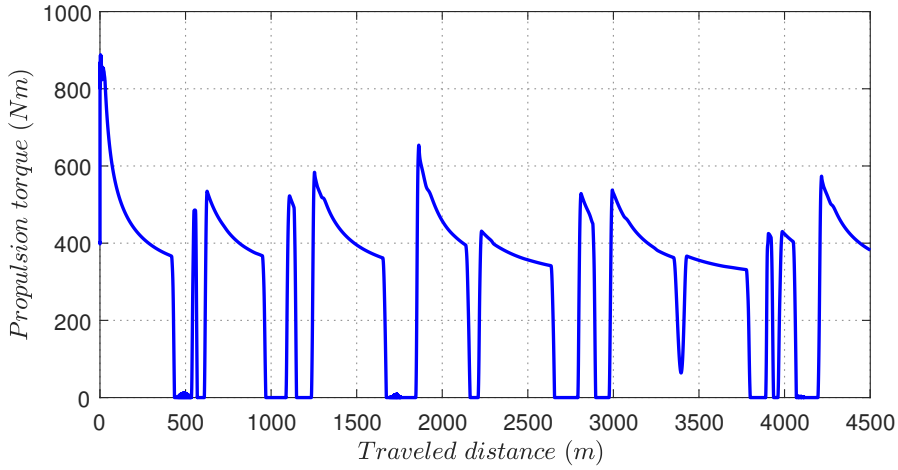

Figure B.21: *Propulsion torque on the rear left wheel of the 2-motor driving topology*

The longitudinal tire forces and lateral tire forces of the 2-motor driving topology are demonstrated as Figure B.22 and B.22 respectively.

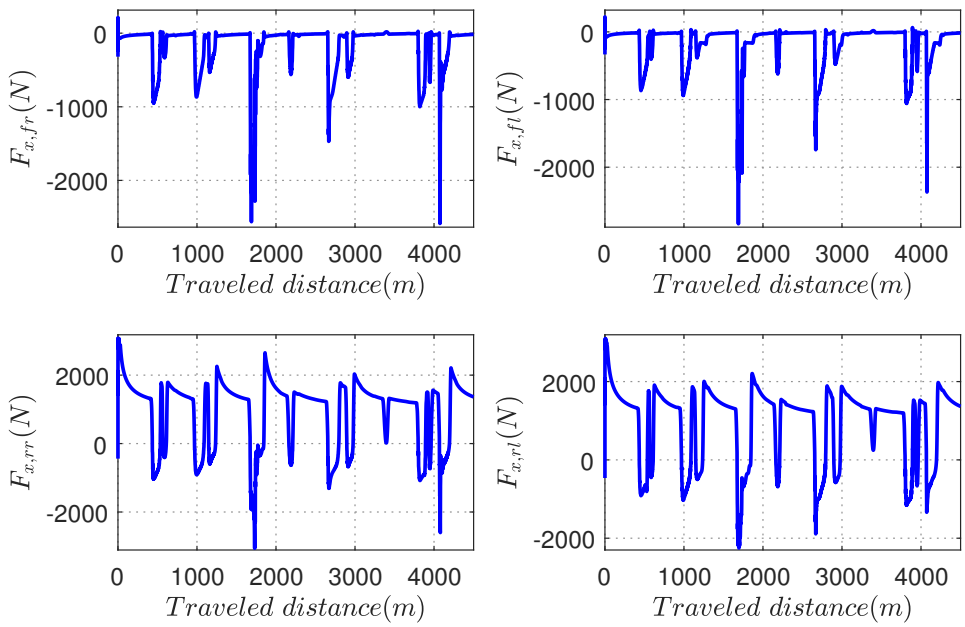

Figure B.22: *Longitudinal tire forces of the 2-motor driving topology*

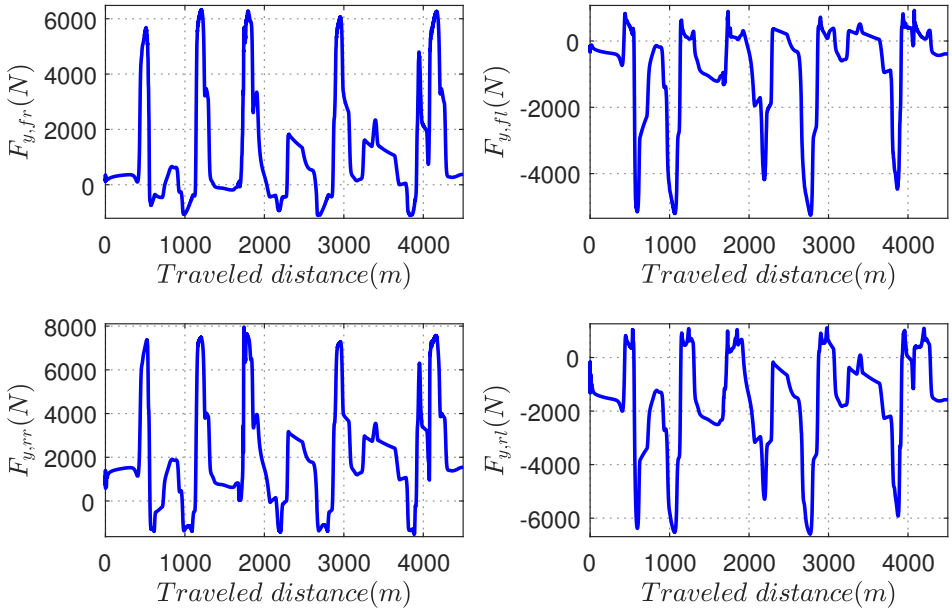

Figure B.23: *Lateral tire forces of the 2-motor driving topology*

B.3 4-Motor driving topology

For the 4-motor driving electric race car, the zoomed out trajectory at each corner of Figure 5.8 are demonstrated in Figure B.24-Figure B.28.

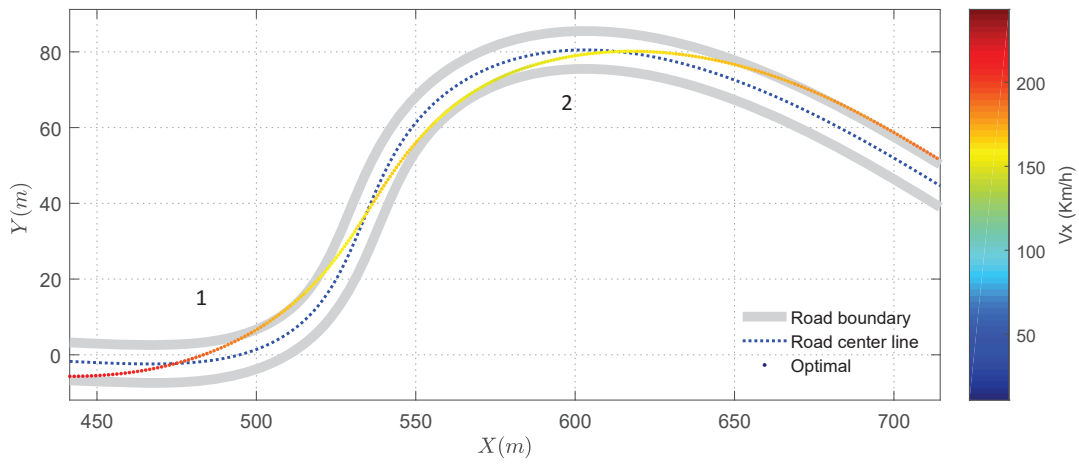

Figure B.24: *The optimal racing line of 4-motor driving topology: corner 1,2*

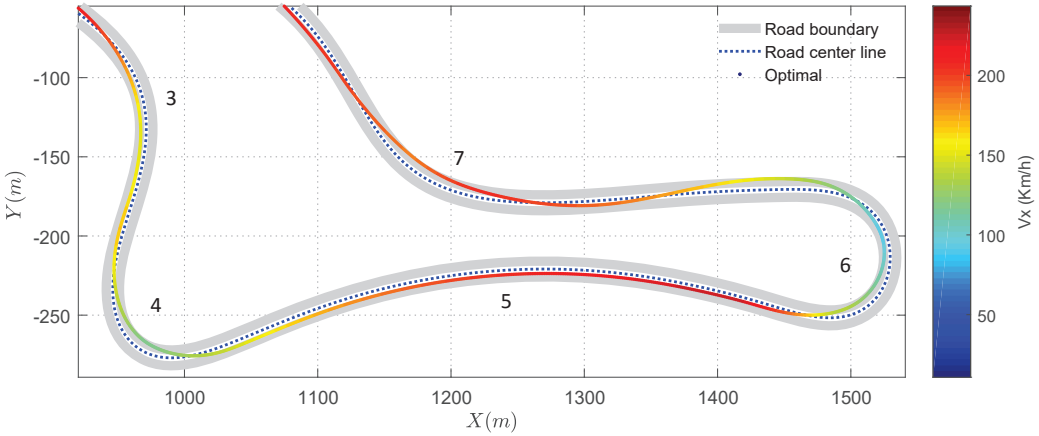

Figure B.25: *The optimal racing line of 4-motor driving topology: corner 3-7*

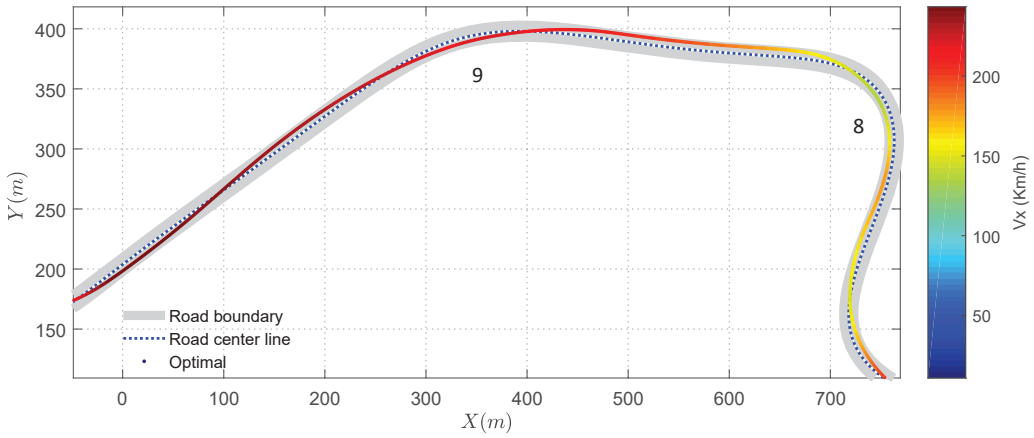

Figure B.26: *The optimal racing line of 4-motor driving topology: corner 8,9*

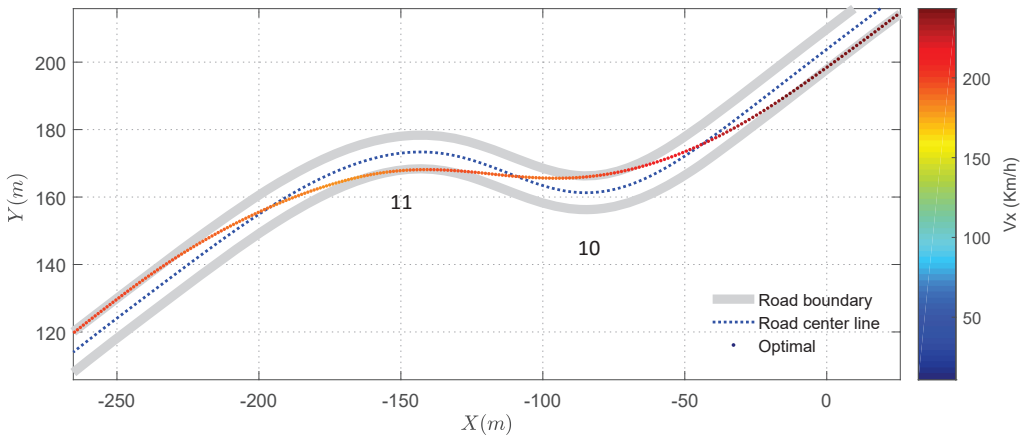

Figure B.27: *The optimal racing line of 4-motor driving topology: corner 10,11*

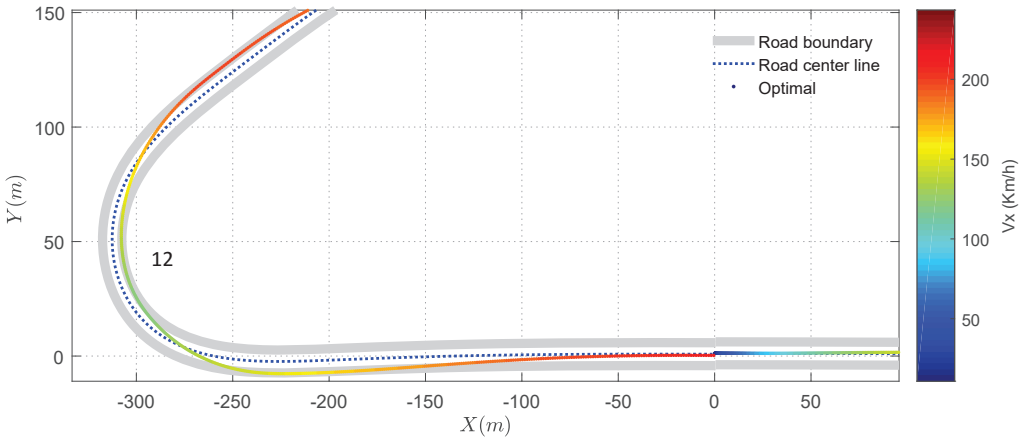

Figure B.28: *The optimal racing line of 4-motor driving topology: corner 12*

The steering wheel angle, orientation angle, braking torque and driving torque are demonstrated separately as Figure B.29, Figure B.30, Figure B.31.

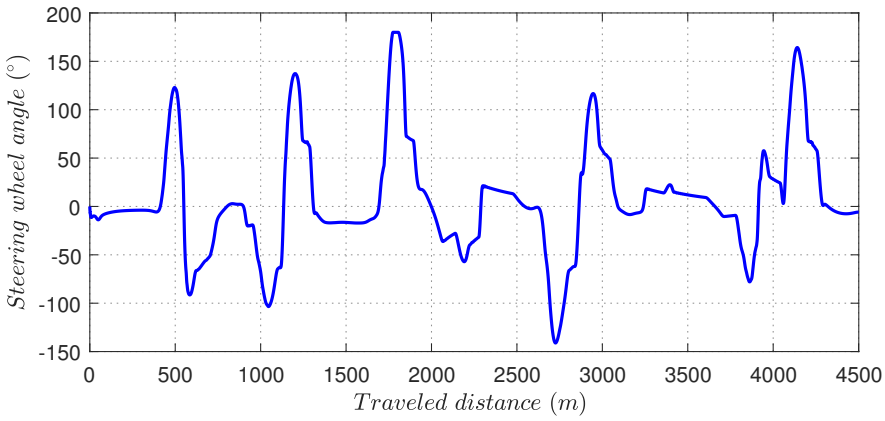

Figure B.29: *Steering wheel angle of the 4-motor driving topology*

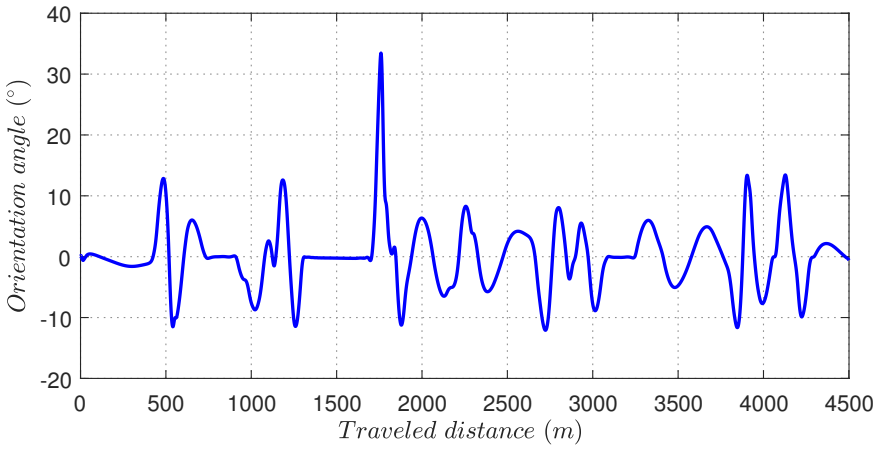

Figure B.30: *The orientation angle of the 4-motor driving topology*

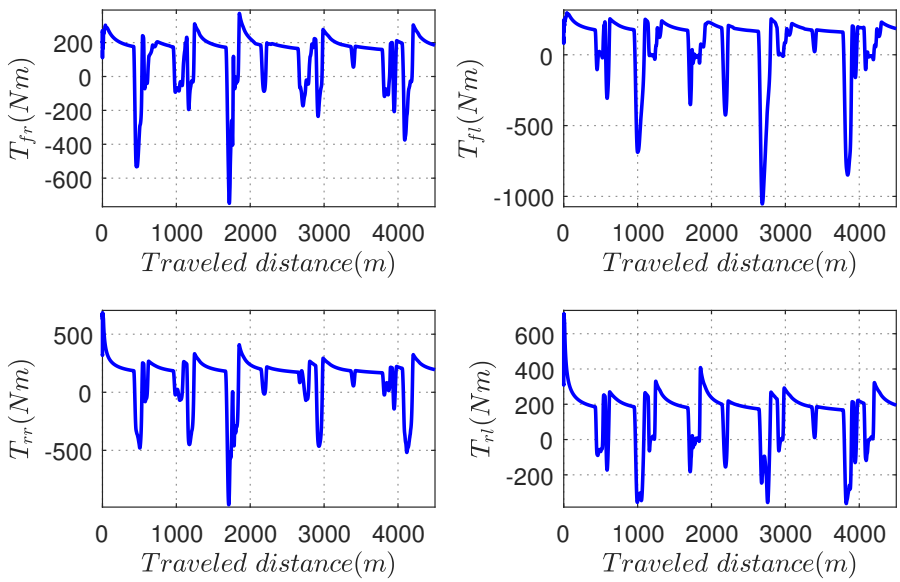

Figure B.31: *Torque on each wheel of the 4-motor driving topology*

The longitudinal tire forces and lateral tire forces of the 4-motor driving topology are given as Figure B.32 and Figure B.33 respectively.

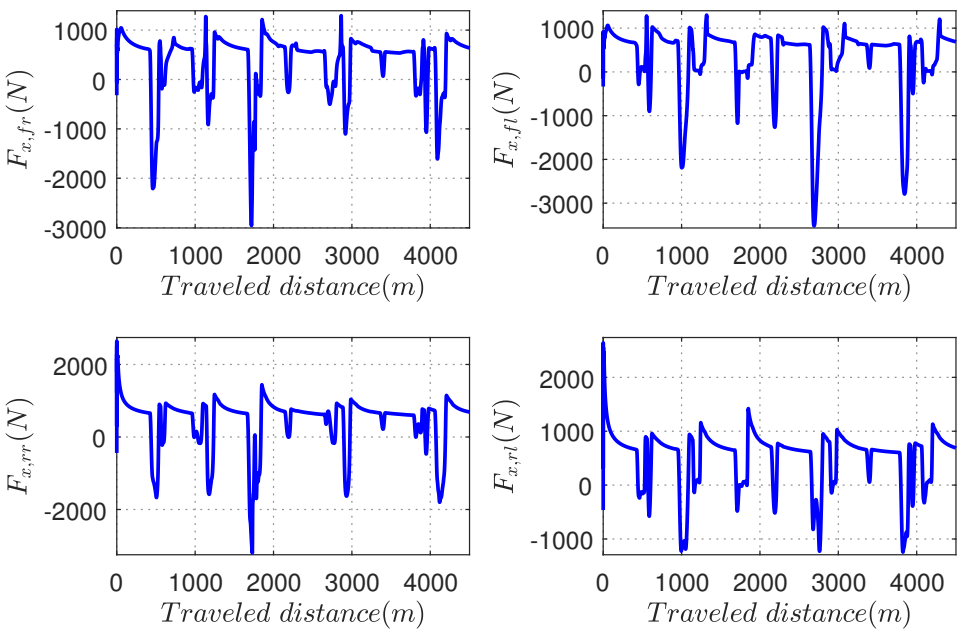

Figure B.32: *Longitudinal tire forces of the 4-motor driving topology*

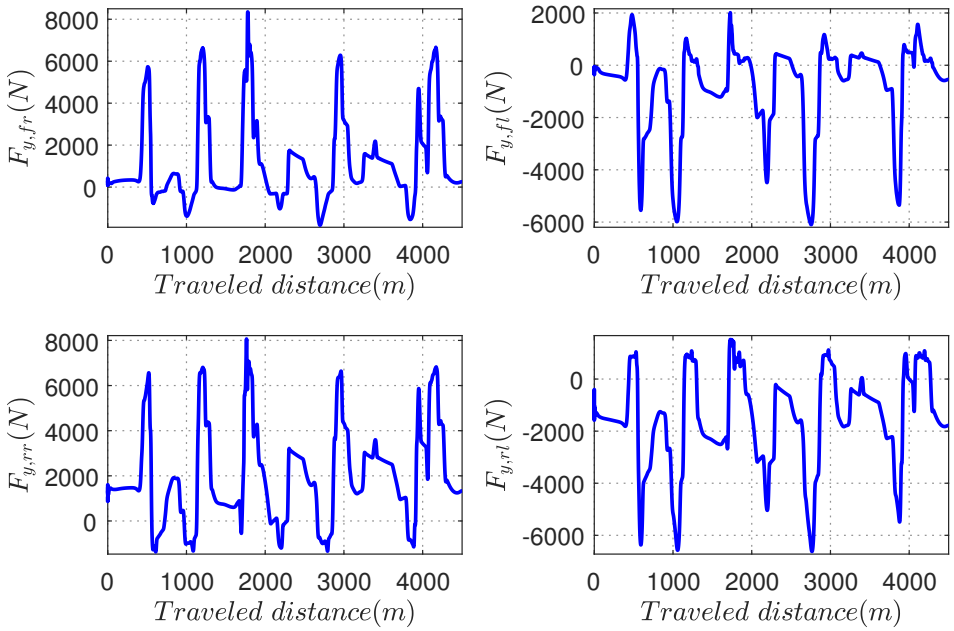

Figure B.33: *Lateral tire forces of the 4-motor driving topology*

Bibliography

[1] C. C. Chan. The state of the art of electric, hybrid, and fuel cell vehicles. *Proceedings of the IEEE*, 95(4):704–718, 2007.

[2] Samuel E. de Lucena. A Survey on Electric and Hybrid Electric Vehicle Technology. In *Electric Vehicles - The Benefits and Barriers*. InTech, sep 2011.

[3] John W Fairbanks. Automotive thermoelectric generators and hvac. In *Proceedings of DEEE Conference, Dearborn, MI, USA*, volume 1619, 2012.

[4] Jeff Cobb. Renault-Nissan And Leaf Lead All In Global EV Proliferation, 2015.

[5] Zachary Shahan. 1 Million Pure EVs Worldwide: EV Revolution Begins. *Clean Technica*, pages The Kidwind Project : Using Mini–Supercapacitors, 2016.

[6] D.B. Karner. State of the art in electric race car power electronics. *Power Electronics in Transportation*, pages 215–217, 1996.

[7] Formula e. FORMULA E, http://www.fia.com/events/formula-e-championship/season-2015-2016/formula-e, 2015.

[8] F. L. Mapelli, D. Tarsitano, D. Annese, M. Sala, and G. Bosia. A study of urban electric bus with a fast charging energy storage system based on lithium battery and supercapacitors. In *2013 8th International Conference and Exhibition on Ecological Vehicles and Renewable Energies, EVER 2013*, 2013.

[9] Ferdinando Luigi Mapelli and Davide Tarsitano. Energy control for plug-in hev with ultracapacitors lithium-ion batteries storage system for fia alternative energy cup race. In *Vehicle Power and Propulsion Conference (VPPC), 2010 IEEE*, pages 1–6. IEEE, 2010.

[10] Alireza Khaligh and Zhihao Li. Battery, ultracapacitor, fuel cell, and hybrid energy storage systems for electric, hybrid electric, fuel cell, and plug-in hybrid electric vehicles: State of the art. *IEEE Transactions on Vehicular Technology*, 59(6):2806–2814, 2010.

[11] Ali Emadi, Alireza Khaligh, Claudio H Rivetta, and Geoffrey A Williamson. Constant power loads and negative impedance instability in automotive systems: definition, modeling, stability, and control of power electronic converters and motor drives. *IEEE Transactions on Vehicular Technology*, 55(4):1112–1125, 2006.

[12] Yee-Pien Yang Yee-Pien Yang, Yih-Ping Luh Yih-Ping Luh, and Cheng-Huei Cheung Cheng-Huei Cheung. Design and control of axial-flux brushless DC wheel motors for electric Vehicles-part I: multiobjective optimal design and analysis. *IEEE Transactions on Magnetics*, 40(4):1873–1882, 2004.

[13] C M Liaw, K W Hu, Y Lin, and T Yeh. An Electric Vehicle IPMSM Drive with Interleaved Front-end DC/DC Converter, 2015.

[14] Li Zhai, Tianmin Sun, and Jie Wang. Electronic stability control based on motor driving and braking torque distribution for a four in-wheel motor drive electric vehicle. *IEEE Transactions on Vehicular Technology*, 65(6):4726–4739, 2016.

[15] Leonardo De Novellis, Aldo Sorniotti, and Patrick Gruber. Wheel torque distribution criteria for electric vehicles with torque-vectoring differentials. *IEEE Transactions on Vehicular Technology*, 63(4):1593–1602, 2014.

[16] Leonardo De Novellis, Aldo Sorniotti, Patrick Gruber, and Andrew Pennycott. Comparison of feedback control techniques for torque-vectoring control of fully electric vehicles. *IEEE Transactions on Vehicular Technology*, 63(8):3612–3623, 2014.

[17] Dhanaraja Kasinathan, Alireza Kasaiezadeh, Andy Wong, Amir Khajepour, Shih-Ken Chen, and Bakhtiar Litkouhi. An optimal torque vectoring control for vehicle applications via real-time constraints. *IEEE Transactions on Vehicular Technology*, 65(6):4368–4378, 2016.

[18] Mehrdad Ehsani, Yimin Gao, and Ali Emadi. *Modern electric, hybrid electric, and fuel cell vehicles: fundamentals, theory, and design*. CRC press, 2009.

[19] K Grider and G Rizzoni. Design of the Ohio State University Electric Race Car.PDF. *1996 Motorsports Engineering Conference Proceedings, Vol 2: Engines & Drivetrains*, pages 89–99, 1996.

[20] Seref Soylu. Electric vehicles–modelling and simulations. *InTech Europe, Rijeka, Croatia*, 2011.

[21] Fabrizio Marignetti, Damiano D'Aguanno, Riccardo Arcese, and Salvatore Ivagnes. Design of a drivetrain for a light urban electric vehicle powered by fuel cells. In *Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), 2015 IEEE 1st International Forum on*, pages 433–438. IEEE, 2015.

[22] Alexei Morozov, Kieran Humphries, Ting Zou, Sudarshan Martins, and Jorge Angeles. Design and optimization of a drivetrain with two-speed transmission for electric delivery step van. In *Electric Vehicle Conference (IEVC), 2014 IEEE International*, pages 1–8. IEEE, 2014.

[23] W Martinez, C A Cortes, L E Munoz, and M Yamamoto. Design of a 200 kW electric powertrain for a high performance electric vehicle [Diseño de un tren de potencia eléctrico de 200 kW para un vehículo eléctrico de alto desempeño]. *Ingenieria e Investigacion*, 36(3):66–73, 2016.

[24] Meilan Zhou, Liping Zhao, Yu Zhang, Zhaoming Gao, and Rongjie Pei. Pure electric vehicle power-train parameters matching based on vehicle performance. *International Journal of Control and Automation*, 8(9):53–62, 2015.

[25] Jiuyu Du, Minggao Ouyang, and Hewu Wang. Battery electric vehicle parameters design targeting to cost-benefit objective. *2012 IEEE Vehicle Power and Propulsion Conference, VPPC 2012*, pages 1160–1164, 2012.

[26] G. N. Reddy and Mathew Ademola Drayer. A Visual C++ based modeling & simulation package for electric vehicle design. *2011 IEEE Vehicle Power and Propulsion Conference*, pages 1–5, 2011.

[27] I. J. M. Besselink, P. F. van Oorschot, E. Meinders, and H. Nijmeijer. Design of an efficient, low weight battery electric vehicle based on a VW Lupo 3L. *The 25th World Battery, Hybrid and Fuel Cell Electric Vehicle Symposium & Exhibition*, 2010.

[28] Francesco Braghin and Edoardo Sabbioni. A 4ws control strategy for improving race car performances. In *ASME 8th Biennial Conference on Engineering Systems Design and Analysis*, pages 297–304. American Society of Mechanical Engineers, 2006.

[29] Federico Cheli, Stefano Melzi, Edoardo Sabbioni, and Michele Vignati. Torque Vectoring Control of a Four Independent Wheel Drive. *ASME 2013 International Design Engineering Technocal Conferences and Computers aand Information in Engineering Conference*, pages 1–8, 2013.

[30] Ricardo de Castro, Mara Tanelli, Rui Esteves Araújo, and Sergio M. Savaresi. Minimum-time manoeuvring in electric vehicles with four wheel-individual-motors. *Vehicle System Dynamics*, 52(6):824–846, 2014.

[31] Leonardo De Novellis, Aldo Sorniotti, and Patrick Gruber. Wheel torque distribution criteria for electric vehicles with torque-vectoring differentials. *IEEE Transactions on Vehicular Technology*, 63(4):1593–1602, 2014.

[32] R. Lot and F. Biral. A curvilinear abscissa approach for the lap time optimization of racing vehicles. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 19:7559–7565, 2014.

[33] Valentin Ivanov, Dzmitry Savitski, and Barys Shyrokau. A Survey of Traction Control and Antilock Braking Systems of Full Electric Vehicles with Individually Controlled Electric Motors. *IEEE Transactions on Vehicular Technology*, 64(9):3878–3896, 2015.

[34] Andreas Wächter and Lorenz T Biegler. *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*, volume 106. Springer, 2006.

[35] A V Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135(1):497–528, 2009.

[36] R.W.H. Sargent. Optimal control. *Journal of Computational and Applied Mathematics*, 124(1):361–371, 2000.

[37] Marco Frego. Numerical Methods for Optimal Control Problems with application to autonomous vehicles. *PhD dissertation*, pages 1–84, 2014.

[38] John T Betts. *Practical methods for optimal control and estimation using nonlinear programming*. SIAM, 2010.

[39] Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. CRC Press, 1987.

[40] Moritz Diehl and KU Leuven. Numerical Optimal Control. *Course manuscript*, 2011.

[41] Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, 1995.

[42] Benoit Chachuat. Nonlinear and dynamic optimization: from theory to practice. Technical report, 2007.

[43] Hans Georg Bock and Karl-Josef Plitt. A multiple shooting algorithm for direct solution of optimal control problems. PROCEEDINGS OF THE IFAC WORLD CONGRESS, 1984.

[44] Lorenz T. Biegler. An overview of simultaneous strategies for dynamic optimization. *Chemical Engineering and Processing: Process Intensification*, 46(11):1043–1053, 2007.

[45] Victor M Becerra. Psopt optimal control solver user manual. *United Kingdom, University of Reading*, 2009.

[46] Paola Falugi, Eric Kerrigan, and Eugene Van Wyk. Imperial college london optimal control software user guide (iclocs). *Department of Electrical and Electronic Engineering, Imperial College London. Available from: http://www.ee.ic.ac.uk/ICLOCS*, 2010.

[47] Gamal Elnagar, Mohammad A Kazemi, and Mohsen Razzaghi. The pseudospectral legendre method for discretizing optimal control problems. *IEEE transactions on Automatic Control*, 40(10):1793–1796, 1995.

[48] Gamal N Elnagar and Mohammad A Kazemi. Pseudospectral chebyshev optimal control of constrained nonlinear dynamical systems. *Computational Optimization and Applications*, 11(2):195–217, 1998.

[49] Paul Williams. Jacobi pseudospectral method for solving optimal control problems. *Journal of Guidance, Control, and Dynamics*, 27(2):293–297, 2004.

[50] David Benson. *A Gauss pseudospectral transcription for optimal control*. PhD thesis, Massachusetts Institute of Technology, 2005.

[51] Geoffrey Todd Huntington. *Advancement and analysis of a Gauss pseudospectral transcription for optimal control problems*. PhD thesis, Citeseer, 2007.

[52] Fariba Fahroo and I Michael Ross. Pseudospectral methods for infinite-horizon nonlinear optimal control problems. *Journal of Guidance, Control, and Dynamics*, 31(4):927–936, 2008.

[53] Michael A. Patterson and Anil V. Rao. GPOPS-II. *ACM Transactions on Mathematical Software*, 41(1):1–37, oct 2014.

[54] Jie Shen, Tao Tang, and Li-Lian Wang. *Spectral Methods*, volume 41 of *Springer Series in Computational Mathematics*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[55] William Karush. Minima of functions of several variables with inequalities as side conditions. *Master thesis, University of Chicago*, 1939.

[56] Edgardo D Castronuovo, Jorge M Campagnolo, and Roberto Salgado. New versions of interior point methods applied to the optimal power flow problem. *Optmization [Online] Digest*, 2001.

[57] Edgardo D. Castronuovo, Jorge M. Campagnolo, and Roberto Salgado. On the application of high performance computation techniques to nonlinear Interior Point methods. *IEEE Transactions on Power Systems*, 16(3):325–331, 2001.

[58] AS El-Bakry, Richard A Tapia, T Tsuchiya, and Yin Zhang. *On the formulation and theory of the Newton interior-point method for nonlinear programming*, volume 89. Springer, 1996.

[59] Philip E. Gill, Walter Murray, and Michael A. Saunders. SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Journal on Optimization*, 12(4):979–1006, 2002.

[60] Anil V Rao, David A Benson, Christopher Darby, Michael A Patterson, Camila Francolin, Ilyssa Sanders, and Geoffrey T Huntington. Algorithm 902: Gpops, a matlab software for solving multiple-phase optimal control problems using the gauss pseudospectral method. *ACM Transactions on Mathematical Software (TOMS)*, 37(2):22, 2010.

[61] Kenneth Holmström. The tomlab optimization environment in matlab. 1999.

[62] Andreas Wiegand et al. Astos user manual. *Unterkirnach, Germany: Astos Solutions GmbH*, 17, 2010.

[63] D. Metz and D. Williams. Near time-optimal control of racing vehicles. *Automatica*, 25(6):841–857, 1989.

[64] T Fujioka and T Kimura. Numerical simulation of minimum-time cornering behavior. *JSAE Review*, 13(1), 1992.

[65] AK Wu and Angelo Miele. Sequential conjugate gradient-restoration algorithm for optimal control problems with non-differential constraints and general boundary conditions, part i. *Optimal Control Applications and Methods*, 1(1):69–88, 1980.

[66] AK Wu and A Miele. Sequential conjugate gradient-restoration algorithm for optimal control problems with non-differential constraints and general boundary conditions, part 2. *Optimal Control Applications and Methods*, 1(2):119–130, 1980.

[67] JPM Hendrikx, TJJ Meijlink, and RFC Kriens. Application of optimal control theory to inverse simulation of car handling. *Vehicle System Dynamics*, 26(6):449–461, 1996.

[68] Vittore Cossalter, Mauro Da Lio, Roberto Lot, and Lucca Fabbri. A general method for the evaluation of vehicle manoeuvrability with special emphasis on motorcycles. *Vehicle system dynamics*, 31(2):113–135, 1999.

[69] Francesco Biral, Stefano Garbin, and Roberto Lot. Enhancing the performance of high powered motorcycles by a proper definition of geometry and mass distribution. *SAE Technical Paper*, 2002.

[70] V Cossalter, M Da Lio, R Lot, and L Fabbri. Simulation and performance evaluation of race motorcycle dynamics based on parts of real circuit. In *Power Two Wheels International Conference*, pages 14–15, 1998.

[71] Fabiano Maggio, Francesco Biral, and Mauro Da Lio. How gearbox ratios influence lap time and driving style. an analysis based on time-optimal maneuvers. Technical report, SAE Technical Paper, 2003.

[72] E. Bertolazzi, F. Biral, and M. Da Lio. Symbolic-Numeric Indirect Method for Solving Optimal Control Problems for Large Multibody Systems. *Multibody System Dynamics*, 13(2):233–252, 2005.

[73] Francesco Biral, Roberto Lot, Stefano Rota, Marco Fontana, and Véronique Huth. Intersection support system for powered two-wheeled vehicles: Threat assessment based on a receding horizon approach. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):805–816, 2012.

[74] Vittore Cossalter, Roberto Lot, and Davide Tavernini. Optimization of the centre of mass position of a racing motorcycle in dry and wet track by means of the âĂIJoptimal maneuver methodâĂİ. In *Mechatronics (ICM), 2013 IEEE International Conference on*, pages 412–417. IEEE, 2013.

[75] Enrico Bertolazzi, Francesco Biral, and Mauro Da Lio. Symbolic-numeric efficient solution of optimal control problems for multibody systems. *Journal of Computational and Applied Mathematics*, 185(2):404–421, 2006.

[76] Davide Tavernini, Matteo Massaro, Efstathios Velenis, Diomidis I. Katzourakis, and Roberto Lot. Minimum time cornering: the effect of road surface and car transmission layout. *Vehicle System Dynamics*, 51(10):1533–1547, 2013.

[77] Roberto Lot and S Evangelou. Lap time optimization of a sports series hybrid electric vehicle. In *2013 World Congress on Engineering*, 2013.

[78] R. Lot and N. Dal Bianco. The significance of high-order dynamics in lap time simulations. *The Dynamics of Vehicles on Roads and Tracks - Proceedings of the 24th Symposium of the International Association for Vehicle System Dynamics, IAVSD 2015*, pages 553–562, 2016. cited By 0.

[79] Roberto Lot and Nicola Dal Bianco. Lap time optimisation of a racing go-kart. *Vehicle System Dynamics*, 3114(January):1–21, 2015.

[80] Daniele Casanova. On Minimum Time Vehicle Manoeuvring: The Theoretical Optimal Lap, 2000.

[81] R.S. S Sharp, D. Casanova, P. Symonds, R.S. S Sharp, and D. Casanova. A Mathematical Model for Driver Steering Control , with Design, Tuning and Performance results. *Vehicle System Dynamics*, 33(768414408):289–326, 2000.

[82] Daniele Casanova. On Minimum Time Vehicle Manoeuvring: The Theoretical Optimal Lap, 2000.

[83] Daniele Casanova, Robin S Sharp, Mark Final, Bruce Christianson, and Pat Symonds. Application of automatic differentiation to race car performance optimisation. In *Automatic differentiation of algorithms*, pages 117–124. Springer, 2002.

[84] H B Pacejka and I J M Besselink. Magic Formula Tyre Model with Transient Properties. *Vehicle System Dynamics*, 27(Suppl):234–249, 1997.

[85] San Diego, La Jolla, Philip E. Gill, Laurent O. Jay, Michael W. Leonard, Linda R. Petzold, and Vivek Sharma. An SQP method for the optimal control of large-scale dynamical systems. *Journal of Computational and Applied Mathematics*, 120:197–213, 2000.

[86] D Casanova, RS Sharp, and P Symonds. Sensitivity to mass variations of the fastest possible lap of a formula one car. *Vehicle System Dynamics*, 35(2):119–134, 2001.

[87] Daniel Patrick Kelly. Lap time simulation with transient vehicle and tyre dynamics. 2008.

[88] Giacomo Perantoni and David J.N. Limebeer. Optimal control for a Formula One car with variable parameters. *Vehicle System Dynamics*, 00(March):1–26, 2014.

[89] D.J.N. Limebeer, G. Perantoni, and a.V. Rao. Optimal control of Formula One car energy recovery systems. *International Journal of Control*, 7179(October):1–16, 2014.

[90] David J.N. Limebeer and Anil V. Rao. Faster, Higher, and Greener: Vehicular Optimal Control. *IEEE Control Systems*, 35(2):36–56, 2015.

[91] D. J. N. Limebeer and G. Perantoni. Optimal Control of a Formula One Car on a Three-Dimensional TrackâĂŤPart 2: Optimal Control. *Journal of Dynamic Systems, Measurement, and Control*, 137(5):051019, may 2015.

[92] David J. N. Limebeer and Anil V. Rao. Faster, Higher, and Greener: Vehicular Optimal Control. *IEEE Control Systems*, 35(2):36–56, apr 2015.

[93] Divya Garg, William W Hager, and Anil V Rao. Pseudospectral methods for solving infinite-horizon optimal control problems. *Automatica*, 47(4):829–837, 2011.

[94] Michael A Patterson and Anil V Rao. Gpops-ii: A matlab software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming. *ACM Transactions on Mathematical Software (TOMS)*, 41(1):1, 2014.

[95] Ricardo De Castro, Rui Esteves Araujo, and Diamantino Freitas. Wheel slip control of EVs based on sliding mode technique with conditional integrators. *IEEE Transactions on Industrial Electronics*, 60(8):3256–3271, 2013.

[96] William F Milliken, Douglas L Milliken, et al. *Race car vehicle dynamics*, volume 400. Society of Automotive Engineers Warrendale, 1995.

[97] M. Doumiati, a. Victorino, D. Lechner, G. Baffet, and a. Charara. Observers for vehicle tyre/road forces estimation: experimental validation. *Vehicle System Dynamics*, 48(11):1345–1378, 2010.

[98] F Cheli, E Leo, S Melzi, and F Mancosu. A 14dof model for the evaluation of vehicleâĂŹs dynamics: numerical–experimental comparison. *International Journal of Mechanics and Control*, 6(2):19–30, 2006.

[99] Hans B. Pacejka. *Tyre and vehicle dynamics: Second Edition*. Butterworth-Heinemann, 2006.

[100] I. J.M. Besselink, a. J.C. Schmeitz, and H. B. Pacejka. An improved Magic Formula/Swift tyre model that can handle inflation pressure changes. *Vehicle System Dynamics*, 48(1):337–352, 2010.

[101] Juha Pyrhonen, Tapani Jokinen, and Valeria Hrabovcova. *Design of rotating electrical machines*. John Wiley & Sons, 2009.

[102] Stephen P Radzevich. *Dudley's handbook of practical gear design and manufacture*. CRC Press, 2012.

[103] A. Wachter Biegler and L. IPOPT - an Interior Point OPTimizer.https://projects.coinor. org/Ipopt, 2009.

[104] William Squire and George Trapp. Using complex variables to estimate derivatives of real functions. *SIAM Review*, 40(1):110–112, 1998.

[105] Kok-Lam Lai, John Crassidis, Yang Cheng, and Jongrae Kim. New Complex-Step Derivative Approximations with Application to Second-Order Kalman Filtering. *AIAA Guidance, Navigation, and Control Conference and Exhibit*, pages 1–17, 2005.

[106] Giacomo Perantoni and David J.N. Limebeer. Optimal control for a Formula One car with variable parameters. *Vehicle System Dynamics: International Journal of Vehicle Mechanics and Mobility*, 52(5):653–678, 2014.

[107] F. Topputo and C. Zhang. Survey of direct transcription for low-thrust space trajectory optimization with applications. *Abstract and Applied Analysis*, 2014, 2014.

[108] Mallory Daly and Anil V. Rao. Demonstration of GPOPS-II for Optimal Configuration of a Tetrahedral Spacecraft Formation. (April), 2015.

[109] Victor M Becerra. PSOPT Optimal Control Solver User Manual for PSOPT. *Psopt*, 2010.

[110] Roger A Horn. The hadamard product. In *Proc. Symp. Appl. Math*, volume 40, pages 87–169, 1990.