CHALMERS

Model predictive control for vehicle steering using road information in the local frame

Master's thesis in Systems, Control and Mechatronics

LOVE MOWITZ
NAM VU

Department of Mechanics and Maritime Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2019

MASTER'S THESIS IN SYSTEMS, CONTROL AND MECHATRONICS

Model predictive control for vehicle steering using road information in the local frame

LOVE MOWITZ
NAM VU

Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2019

Model predictive control for vehicle steering using road information in the local frame
LOVE MOWITZ
NAM VU

Master's thesis 2019:81
Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31-772 1000

Chalmers Reproservice
Göteborg, Sweden 2019

Model predictive control for vehicle steering using road information in the local frame
Master's thesis in Systems, Control and Mechatronics
LOVE MOWITZ
NAM VU
Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology

Abstract

This thesis was motivated by *Chalmers formula student driverless 2019* (CFSD19) project, with the aim to deliver a self-driving formula race car and compete in the *Formula Student Czech* competition. In this thesis, a control strategy to control the lateral motion of the CFSD19 car was designed and simulated. The control algorithm is based on the model predictive control (MPC) framework. The simulation used a single track dynamic model (also known as the bicycle model) for simulating the car's motion, and a kinematic model, linearized around the currently sampled state at every sampling update, for the controller. The goal of the controller is to steer the car, so that it can track a given path in the global frame, using only track information observed by the car in its local frame. The performance of the controller is evaluated by measuring the lateral deviation of the car from the track. Simulation shows that the car is able to track the path with acceptable lateral deviation, and the control scheme can be executed fast enough for real-time application.

Keywords: Model predictive control, MPC, Successive linearization

Acknowledgements

First and foremost, we would like to express our sincere gratitude towards our examiner, Ola Benderius, for his guide and support during both the project and thesis. Our gratitude also extends to all the members of CFSD19 team who have worked with us in the project. In addition, we would also like to thank the members of the CFSD18 and CFS17 team, our predecessors, who built the car and laid the foundation for the current project.

Last but not least, we also want to thank the staff of Revere lab, including Arpit Karsolia, Fredrik von Corswant, Christian Berger and many others, for sharing their resources and offering valuable help throughout the project.

Thesis examiner: Ola Benderius

Contents

1 Introduction

Autonomous driving technology is currently an active field of study, and is expected to change the automotive industry in the near future. Sharing the same vision, *Formula Student Germany* (FSG), an engineering design competition where students build their own formula racing cars to compete with others, has also adopted *Formula Student Driverless* as a new competition class since 2017, and by 2021, "all vehicles participating in FSG are supposed to have driverless technology on board" [1]. Given the rising interest in autonomous vehicles in this area, it is of interest to investigate different controller performances on an autonomous race vehicle.

In 2017, the same year as the first FSG competition with a driverless class, Chalmers University of Technology set up a *Formula Student Driverless* team. *Chalmers Formula Student Driverless* (CFSD) thus had it's first season in 2018 where the members converted a manually driven formula race car into a driverless vehicle. At the competition the vehicle races in four different dynamic events meant to test the vehicle's ability to navigate through the specified track environment. The event tracks are marked with colored cones and are designed to test the full capabilities of the vehicles.

This thesis investigates the lateral control of a formula car developed for use in the CFSD19 car and as such takes into account the dynamic events of driverless competitions when evaluating the controller. The proposed lateral controller utilises decoupled model predictive control (MPC) with two different system controller models and its performance is simulated using a two-track vehicle model. The decoupled MPC controllers using different controller plant models are simulated using test cases to provide insight on the controller performance in conditions comparable to the *Formula Student* competition dynamic events. To provide a simulation of the lateral controller in the context of the real system, the path planning and longitudinal control planned for the vehicle is additionally part of the simulation environment.

1.1 Aim

The aim of this thesis is to investigate the feasibility of a decoupled MPC for the steering of a *Formula Student* car. Specifically, this thesis aims to answer the following research questions:

- What is the performance of a decoupled MPC for steering in the specified driving environment?
- What is the performance effect of different controller plant models for the decoupled MPC in the specified driving environments?

1.2 Scope

This thesis will not investigate the implications coupled lateral and longitudinal control and will instead only consider decoupled lateral control at varying longitudinal speeds using different plant models. The vehicle model will also be tailored to the specific *Formula Student* car developed in 2017 by *Chalmers Formula Student* which was made autonomous in 2018 by *Chalmers Formula Student Driverless*. The *Formula Student Driverless* competition tracks and track setup will also be taken into consideration when testing and evaluating the controller.

The proposed controller sits between the perception and low-level controllers that interface with the steering actuation. Figure 1.1 shows an overview of the system architecture and where the focus of this thesis lies.

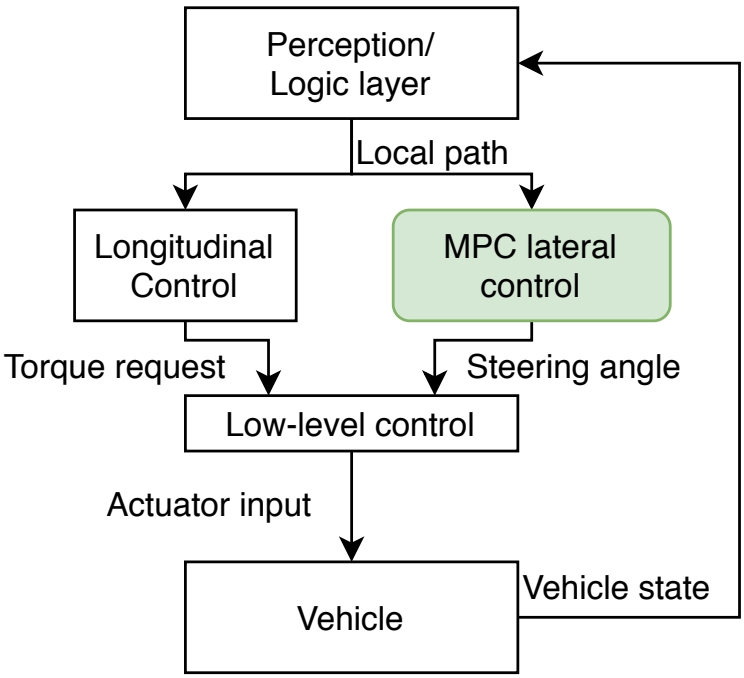

Figure 1.1: Overview of the information flow of the control system in the autonomous race car.

2 Background

Model predictive control is an advanced method for process control able to satisfy given input and output constraints and can therefore be useful for safety critical autonomous design [2]. MPC was originally developed in the context of power plants and petroleum refineries but can today be found in a wide variety of areas, including the automotive industry [3][4]. This can be attributed to the growing processing power increasing the possibility of real time application in some areas. This has also made it possible for non-linear MPC solutions to be feasible in real time in some cases [5]. Since MPC is model-based control, it is similar to a linear-quadratic regulator (LQR). MPC, however, differs from LQR in that a finite time-horizon is optimized but only the first input is used from the optimal control sequence and then the process is optimized again at the next time step.

One specific area of MPC application in automotive control is called Active Front Steering (AFS). In this application, MPC is used to adjust the front steering angle of vehicles, without changing the position of steering wheel. This is used to assist the driver around difficult cornering in non-autonomous driving. Using MPC allows the controller to account for physical constraints, such as maximum steer angle or maximum steering rate. This was demonstrated by Yoon et al. [6], or Falcone [7]. The positive results as well as the simple setup for these controllers gave motivation of the this thesis.

Since MPC is a model-based control scheme, it is therefore important to choose the right model to predict the vehicle motion. There are generally three types of models for vehicle control: kinematic model which describes vehicle motion by only examining its velocities and orientation; single-track (or bicycle) dynamic model which collapses a four-wheel vehicle into only two wheels in a single axis, and describes vehicle motions using forces and inertia (linear or rotational) of vehicle; and two-track model which is similar to single-track model but now examine all four wheels and also account for the vehicle's rolling motion. Each model has its own advantages and drawbacks, and the right model is the best compromise between its accuracy in describing motion, and its complexity. The first class of model, kinematic model is considered to be simplest and would require less computational power, which is beneficial for real-time application. This model also avoids the singular problem in tire modelling, namely that tire model have a tire slip angle equation which has the longitudinal velocity in the denominator (thus making tire model unusable at zero or low velocity). However, the biggest drawback of kinematic model is its inaccuracy particularly under high longitudinal speed or high lateral acceleration. In one report [8], a comparison in the accuracy of predicting a vehicle position between a kinematic and a dynamic-based MPC steering was done. The result showed that a kinematic model gives worse prediction over time compared to a dynamic one, particularly in high speed. Thus in high dynamic application such as controlling a racing car, it is deemed better to use a dynamic model for controller. In another report [9], an extensive comparison of accuracy between different car dynamic models were done: namely a single track model with three different tire models (linear, nonlinear magic formula, and tire force lag), and a two-track model. The outputs of these models (front side slip angle, yaw rate) were compared against real measurements in a double lane change test, where the test car was driven at three different speeds (36, 49, and 59 km/h). It was concluded that a single track model with tire for lag model would perform equally well as a two-track model, and the single-track, linear-tire model would perform well only up to a lateral acceleration of 0.5G. Thus the conclusion from this is that it is good enough to use a single track model, as long as the tire model is good enough for the use case.

In case of dynamics-model-based controller, one usually consider whether to couple or decouple the control strategy. A decoupled dynamic controller is a control strategy where the longitudinal and lateral dynamics of vehicles are separately controlled by two separate controller; whereas in coupled controllers both dynamics are controlled by a single controller. The industry practice for path tracking MPC is to decouple longitudinal and lateral dynamics, in which a speed profile (that aims at maximize acceleration capability for example) is generated first; and the longitudinal controller (MPC or PID) would make sure the car drives at the set speed. After that, another separate controller would use the longitudinal set speed as a parameter, and make sure the car follow a desired lateral trajectory (lane keeping or collision avoidance for example). Coupled controllers are generally more complex to implement, but they give better performance in theory. Several attempts have been made to implement and study coupled dynamic controllers. Both coupled and decoupled dynamic model based controllers have been discussed prominently in other reports [10][11]. One proposed design was a coupled controller, where longitudinal and lateral dynamics were controlled separately but they exchange information together while having to satisfy the same constraint on tire slip angle [11]. Extensive work

regarding comparing coupled and decoupled controllers in terms of both performance, complexity and execution time has likewise been done before. In one such report [12], both controllers used a single track dynamic model, with the decoupled controller controlling only the steering angle, while the coupled one controlling both the steering angle and longitudinal acceleration. The author concluded that the coupled controller only performed slightly better than the decoupled one in terms of lateral tracking (the difference is only 1 cm lateral deviation less for the coupled one). However, the coupled one is more complex and harder to tune, and it also takes quite more time to execute (the average control loop of the coupled one being 11 ms, 2 ms more than the decoupled one, and this violated a requirement of 10 ms control loop in this work). Figure 2.1 illustrates the comparison of coupled and decoupled controllers done in the previously mentioned report [12]. Based on these results, the decoupled control scheme was chosen in this thesis.

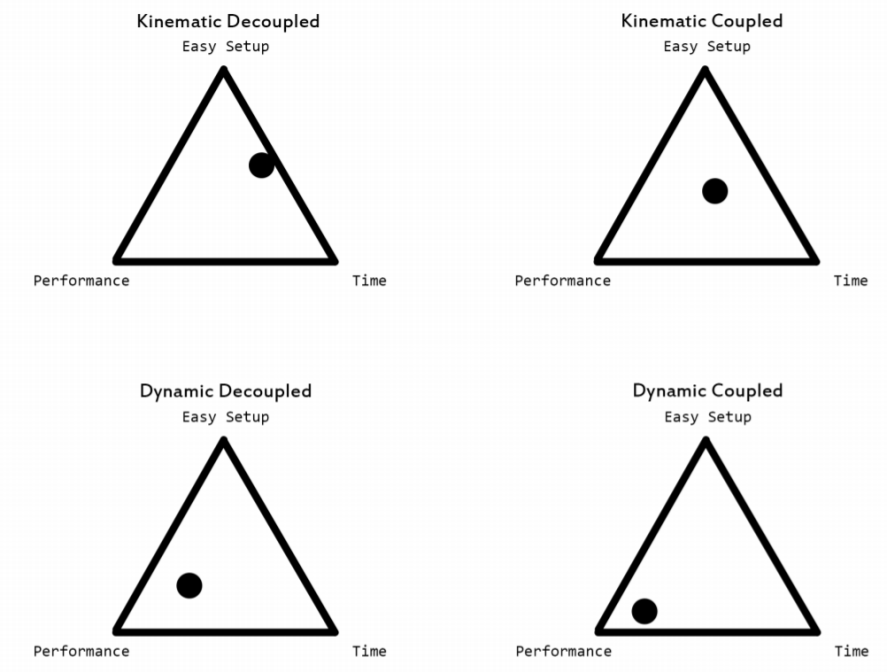

Figure 2.1: Comparison between coupled and decoupled controller [12].

In a model predictive control scheme, at each control step an optimization problem is solved based on some performance index (minimising or maximizing a cost function), and this is repeated for every control step. The optimization problem can be based on a linear or non-linear model with respect to the optimizing variables. Even though a non-linear model is usually more accurate than a linear one, it often results in a more complex nonlinear programming problem, and takes more to to solve. This would contradict the real-time requirement of most control problems. Therefore, many MPC application in autonomous driving would linearize the non-linear control models at the current operating conditions, to get an approximation of the nonlinear model, and use that linearized model for the MPC-solver. This is called successive linearization, and the optimization problem of this kind is called Linear-Time Varying model MPC, or LTV-MPC. LTV-MPC are commonly used in autonomous driving application, due to most dynamic models for lateral control being nonlinear with respect to both states and control inputs.

Other authors have previously proposed an LTV-MPC for active front steering, where a discrete-time, linear system was obtained by successive online linearization of a nonlinear vehicle mode [7][13]. Also in these works, the stability conditions for LTV-MPC controllers were presented. In a similar report the authors have also demonstrated that successive linearizations of a nonlinear model over the prediction horizon would improve the accuracy of the LTV prediction model and thus the performance of the controller [14]. Outside of applications for automotive autonomous control, successive linearization are also widely applied in other fields, such as robotics or in marine vessel tracking [15][16]. The results from the aforementioned works showed that LTV-MPC are robust for active front steering control, and thus an LTV-MPC obtained from successful linearization was chosen in this thesis.

After the controller is designed, it is also important to decide how it would be tested. Controller validation depends greatly on the selected test scenarios. For a racing car, it is desirable to examine the controller behaviour at its limiting conditions. One report investigates different test scenarios and their benefits [17]. Based on this work, the following tests were chosen in this thesis:

- Steady state cornering: The vehicle drives in a constant radius turn. The radius is the same as the turning radius in the skidpad dynamic event. The vehicle would drive at different constant longitudinal velocities. Lateral deviation from middle line of the test track would be recorded. This test would reveal information about the vehicle's under or over-steering properties.
- Double lane change: The vehicle drives in a first straight lane for 50 m, then steer sideways and change to a second lane. It continues on that lane for 12 m, then steer back to the first lane and continue driving on the first lane until the end. The lanes are three meters apart. This test is commonly used for testing electronic control systems (ECS).
- Track drive: The vehicle drives in a track following the *Formula Student Driverless* track drive rules which includes straights, constant turns and hairpin turns. The lanes are three meters apart. The test is used to provide insight to the controller performance in the intended application environment.

Last but not least, time complexity is a major consideration when implementing MPC. Due to the fast dynamic nature of vehicle dynamics, any controller must be able to execute control loops fast enough. In most of the works mentioned above, MPC controllers were able to run at a relatively high frequency, from 50 hz and above. Thus, in this thesis, it is of importance that the controller design enables a high control frequency in terms of complexity. It is however important to keep in mind that the intended execution environment on the formula car will have limited resources that have to be shared among multiple programs and that the controller should also be evaluated at lower control loop frequencies.

3 Theory

This chapter provides the necessary background for understanding the controller formulation and design. Theory about LTV MPC is presented through a general MPC formulation with the linearization and discretization of non-linear system dynamics.

3.1 Model predictive control

Model predictive control is well suited for system where safety requirements have to be considered, such as lane keeping for vehicle control. Some of the advantages of MPC is constraint handling, explicit use of a model and well defined tuning parameters together with its ability to take future system behaviour into consideration. By this it is possible to, for example, take wheel angle and road lane limits into consideration when formulating the problem, which is also taken into consideration when evaluating future system behavior.

MPC is based on a finite-horizon optimization of a dynamic system by utilizing a mathematical model of the plant. The current state of the system is sampled at time t and minimizes a cost function based on the system states and inputs between the current sampled time until some point in the future $t + T$ divided into N steps. A basic MPC formulation can be expressed as

$$\begin{aligned} &\min_{u} J(x_k, u_k) \\ &\text{subject to } x_{k+1} = f(x_k, u_k) \\ &x_k \in \mathbb{X} \\ &u_k \in \mathbb{U} \end{aligned} \tag{3.1}$$

where x are the system states, u are control inputs, J is the cost function to optimize, $f(x_k, u_k)$ captures the system dynamics and $\mathbb{X}$ and $\mathbb{U}$ are constraint sets for the system states and inputs respectively. Since the prediction is made in N steps over a given time T these are parameters that affects the complexity of the problem as well as the accuracy of the solution. A large time horizon T with a small number of steps N give a less accurate approximation of the system evolution and a small time horizon with a high number of steps might not look far enough into the future to be able to stabilize the system.

The cost function J will affect the goal of the controller and is often set up as a quadratic cost function of the system states and inputs as

$$J(x_k, u_k) = x_N^T Q x_N + \sum_{k=1}^{N-1} (x_k^T Q x_k + u_k^T R u_k) \tag{3.2}$$

where Q is a positive-semidefinite weight matrix for the system states and R is a positive-definite weight matrix for the control inputs. By adjusting these weight matrices, more or less importance can be given to individual states or inputs. By solving the problem given in equation (3.1) the optimal input solution, u_k^*, for the horizon is given. Only the first control input is then implemented on the actual system and the optimization problem is solved again at the next time step.

One of the drawbacks of MPC is that the problem complexity grows quickly with the system model and prediction horizon which results in the control scheme being computationally heavy. One way to reduce the complexity is to linearize the system dynamics around the sampled states over the horizon and using a linear MPC formulation. By utilizing a linear system formulation together with linear and convex constraints the complexity of the system is greatly reduced at the cost of some accuracy.

3.2 Successive linearization of controller model

Since the vehicle models are nonlinear they need to be linearized in order to be used in the quadratic programming formulation. At every update step when a new state is obtained, the model is linearized around state. This is called successive linearization. From the state evolution equations, the states and outputs of system can be summarised as

$$\dot{\mathbf{x}} = F(\mathbf{x}, \mathbf{u}) \tag{3.3}$$

$$\mathbf{y} = Z(\mathbf{x}, \mathbf{u}) \tag{3.4}$$

where F and Z are any nonlinear function of states and inputs. The linearization around the current states and inputs can be done by calculaing the Jacobian matrices and substitute the value of current states and inputs to these matrices. For example

$$A_L = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_{nx}} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_{nx}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_{nf}}{\partial x_1} & \frac{\partial f_{nf}}{\partial x_2} & \cdots & \frac{\partial f_{nf}}{\partial x_{nx}} \end{bmatrix} \tag{3.5}$$

or for short

$$A_L(i,j) = \frac{\partial f_i}{\partial x_j} \tag{3.6}$$

where i is the index of the number of nonlinear functions, and j is the index of the number of states (or control inputs). Similarly, one can also write $B_L(i,j) = \frac{\partial f_i}{\partial u_j}$, $C_L(i,j) = \frac{\partial z_i}{\partial x_j}$ and $D_L(i,j) = \frac{\partial z_i}{\partial u_j}$. The state and control input deviations from the chosen linearized point $\mathbf{x_0}$ and $\mathbf{u_0}$ is

$$\delta\dot{\mathbf{x}} = A_L\delta\mathbf{x} + B_L\delta\mathbf{u} \tag{3.7}$$

with $\delta\dot{\mathbf{x}} = \dot{\mathbf{x}}_\mathbf{L} - \dot{\mathbf{x}}_\mathbf{0}$, $\delta\mathbf{x} = \mathbf{x_L} - \mathbf{x_0}$, and $\delta\mathbf{u} = \mathbf{u_L} - \mathbf{u_0}$. Rewriting equation (3.7), one gets

$$\dot{\mathbf{x}} = \dot{\mathbf{x}}_\mathbf{0} + A_L(\mathbf{x_L} - \mathbf{x_0}) + B_L(\mathbf{u_L} - \mathbf{u_0}) \tag{3.8}$$

$\mathbf{x_L}$ and $\mathbf{u_L}$ are state and future input variables, while $\mathbf{x_0}$ and $\mathbf{u_0}$ are currently measured states (thus are constants). Grouping all constant terms into one term, K, and rearranging the above equation yields

$$\dot{\mathbf{x}}_\mathbf{L} = A_L\mathbf{x_L} + B_L\mathbf{u_L} + K \tag{3.9}$$

$$\mathbf{y_L} = C_L\mathbf{x_L} + D_L\mathbf{u_L} \tag{3.10}$$

where

$$K = \dot{\mathbf{x}}_\mathbf{0} - A_L\mathbf{x_0} - B_L\mathbf{u_0} \tag{3.11}$$

3.2.1 Discretization

The continuous state evolution equations further have to be discretized after linearization. Equation (3.9) can be rewritten as

$$e^{-A_L t}\dot{\mathbf{x}}(t) = e^{-A_L t}\mathbf{x_L}(t) + e^{-A_L t}\big(B_L\mathbf{u_L} + K\big) \tag{3.12}$$

by multiplying both sides with $e^{-A_L t}$. Rearranging equation (3.12) yields

$$\frac{d}{dt}(e^{-A_L t}\mathbf{x}(t)) = e^{-A_L t}\big(B_L\mathbf{u_L} + K\big) \tag{3.13}$$

Integrating and then multiplying both sides by $e^{A_L t}$ yields

$$\mathbf{x}(t) = e^{-A_L t}\mathbf{x}(0) + \int_0^t e^{-A_L(t-\tau)}\big(B_L\mathbf{u_L} + K\big)d\tau \tag{3.14}$$

Denote $\mathbf{x_L}[k] = \mathbf{x_L}(kT_s)$, T_s is the sampling time; and substitute t with kT_s yields

$$\mathbf{x_L}[k+1] = e^{A_L T_s}\mathbf{x_L}[k] + \int_{kT_s}^{kT_s+T_s} e^{A_L(kT_s+T_s-\tau)}\big(B_L\mathbf{u_L}(\tau) + K\big)d\tau \tag{3.15}$$

Assuming the control input is constant during each update step, $\mathbf{u_L} = \mathbf{u_L}[k]$. Additionally, if A_L is invertible the above equation can be rewritten into

$$\mathbf{x_L}[k+1] = e^{A_L T_s}\mathbf{x_L}[k] + A_L^{-1}\big(e^{A_L T_S} - I\big)\big(B_L\mathbf{u_L}[k] + K\big) \tag{3.16}$$

Denote $A_d = e^{A_L T_s}$, $Bd = A_L^{-1}\big(e^{A_L T_s} - I\big)$, $C_d = C_L$, $D_d = D_L$ and $\mathbb{K} = A_L^{-1}\big(e^{A_L T_s} - I\big)K$, one gets

$$\mathbf{x_L}[k+1] = A_d\mathbf{x_L}[k] + B_d\mathbf{u_L}[k] + \mathbb{K} \tag{3.17}$$

and

$$\mathbf{y_L}[k+1] = C_d\mathbf{x_L}[k] + D_d\mathbf{u_L}[k] \tag{3.18}$$

which are the discretized state update and output equations.

3.3 Quadratic programming formulation

To solve the linear MPC problem fast it can be formulated as a quadratic programming (QP) problem. The general QP problem has the form

$$\begin{aligned} \min_u J &= \frac{1}{2}u^T H u + f^T u \\ \text{subject to } & D_{in}u \le b_{in} \\ & D_{eq}u = b_{eq} \\ & lb \le u \le ub \end{aligned} \tag{3.19}$$

where u is a vector that minimizes the cost function, H is a positive definite matrix, f^T is a real valued vector while D_{in} and b_{in} are inequality constraints, D_{eq} and b_{eq} are equality constraints and lb and ub are lower and upper bounds of u.

Denote $\mathbf{x}_L[k]$ as $\mathbf{x}_D^k$, $\mathbf{u}_L[k]$ as $\mathbf{u}_D^k$, and $\mathbf{y}_L[k]$ as $\mathbf{y_D^k}$. The cost function of the QP problem usually depends on the error between the state and the reference signal, $\mathbf{e}^k = \mathbf{y}_D^k - \mathbf{r}_k$. For a given system with n_x states, n_u control inputs and n_o measured states, the change in the tracking error can be calculated over the prediction horizon, n_p as

$$\mathbf{e}^k = C_d\mathbf{x}_D^k + D_d\mathbf{u}_D^k - \mathbf{r}^k \tag{3.20a}$$

$$\mathbf{e}^{k+1} = C_d\mathbf{x}_D^{k+1} + D_d\mathbf{u}_D^{k+1} - \mathbf{r}^{k+1} = C_dA_d\mathbf{x}_D^k + C_dB_d\mathbf{u}_D^k + C_d\mathbb{K} + D_d\mathbf{u}_D^{k+1} - r^{k+1} \tag{3.20b}$$

$$\vdots$$

$$\mathbf{e}^{k+n_p-1} = C_d\mathbf{x}_D^k + D_d\mathbf{u}_D^k - \mathbf{r}^k \tag{3.20c}$$

which can be expanded to any arbitrary error within the horizon. Thus, all error states within the horizon can be calculated as

$$\underbrace{\begin{bmatrix}\mathbf{e}^k\\ \mathbf{e^{k+1}}\\ \mathbf{e^{k+2}}\\ \vdots\\ \mathbf{e^{k+n_p-1}}\end{bmatrix}}_{\overrightarrow{\mathbf{e}}_k} = \underbrace{\begin{bmatrix}C_d\\ C_dA_d\\ C_dA_d^2\\ \vdots\\ C_dA_d^{n_p-1}\end{bmatrix}}_{P\in\mathbb{R}^{n_pn_o\times n_x}}\mathbf{x}_D^k + \underbrace{\begin{bmatrix}D_d & 0 & 0 & \dots\\ C_dB_d & D_d & 0 & \dots\\ C_dA_dB_d & C_dB_d & D_d & \dots\\ \vdots & \vdots & \vdots & \ddots\\ C_dA_d^{n_p-2}B_d & C_dA_d^{n_p-3}B_d & C_dA_d^{n_p-4}B_d & \dots\end{bmatrix}}_{H\in\mathbb{R}^{n_pn_o\times n_pn_u}}\underbrace{\begin{bmatrix}\mathbf{u}_D^k\\ \mathbf{u}_D^{k+1}\\ \mathbf{u}_D^{k+2}\\ \vdots\\ \mathbf{u}_D^{k+n_p-1}\end{bmatrix}}_{\overrightarrow{\mathbf{u}}_k} +$$
$$\underbrace{\begin{bmatrix}0\\ C_d\\ C_d(I+A_d)\\ \vdots\\ C_d(I+\sum_{i=1}^{n_p-2}A_d^i)\end{bmatrix}}_{E\in\mathbb{R}^{n_pn_o\times n_x}}\mathbb{K} - \underbrace{\begin{bmatrix}\mathbf{r}^k\\ \mathbf{r}^{k+1}\\ \mathbf{r}^{k+2}\\ \vdots\\ \mathbf{r}^{k+n_p-1}\end{bmatrix}}_{\overrightarrow{\mathbf{r}}_k} \tag{3.21}$$

where P, H and E are output prediction matrices and $\overrightarrow{\mathbf{e}}_k$, $\overrightarrow{\mathbf{u}}_k$ and $\overrightarrow{\mathbf{e}}_k$ are the combined error, control input and reference vectors for the horizon. Equation (3.21) can then compactly be written as

$$\overrightarrow{\mathbf{e}}_k = P\mathbf{x}_D^k + H\overrightarrow{\mathbf{u}}_k + K \tag{3.22}$$

where $K = E\mathbb{K} - \overrightarrow{\mathbf{r}}_k$ contains all constant terms. The cost function for this problem could then be written as

$$J = \frac{1}{2}(\overrightarrow{\mathbf{e}}_k^T Q\overrightarrow{\mathbf{e}}_k + \overrightarrow{\mathbf{u}}_k^T R\overrightarrow{\mathbf{u}}_k) \tag{3.23}$$

where $Q \in \mathbb{R}^{n_pn_o\times n_pn_o}$ and $R \in \mathbb{R}^{n_pn_u\times n_pn_u}$ are weighting matrices for the combined error and control input vectors respectively. By expanding equation (3.21) in (3.23) one gets the following expression for the cost function

$$J = \frac{1}{2}((P\mathbf{x}_D^k + H\overrightarrow{\mathbf{u}}_k + K)^T Q(P\mathbf{x}_D^k + H\overrightarrow{\mathbf{u}}_k + K) + \overrightarrow{\mathbf{u}}_k^T R\overrightarrow{\mathbf{u}}_k) \tag{3.24}$$

Some of the resulting terms in the cost function are independent of the control inputs and can therefore be removed without affecting the results of the optimization problem. It is also possible to group the linear and quadratic terms to gain the expression

$$J = \frac{1}{2}\overrightarrow{\mathbf{u}}_k^T \underbrace{(H^T Q H + R)}_{G} \overrightarrow{\mathbf{u}}_k + \underbrace{({\mathbf{x}_D^k}^T P^T + K^T) Q H}_{W^T} \overrightarrow{\mathbf{u}}_k \tag{3.25}$$

resulting in

$$J = \frac{1}{2}\overrightarrow{\mathbf{u}}_k^T G \overrightarrow{\mathbf{u}}_k + W^T \overrightarrow{\mathbf{u}}_k \tag{3.26}$$

which can be compared to the cost in the general QP formulation in equation (3.19). Since the cost function includes the whole input, the resulting minimum cost solution might introduce a steady state error. It is therefore beneficial to use the input change in the cost function instead of the full function. This can be solved by rewriting the input in terms of input increments as

$$\mathbf{u}_D^{k+n-1} = \mathbf{u}_D^{k-1} + \sum_{i=1}^{n} \delta \mathbf{u}_i \tag{3.27}$$

where $\delta \mathbf{u}_i$ are the input increments at time point i and $n \in [1, n_p]$. This can be represented in matrix form as

$$\underbrace{\begin{bmatrix} \mathbf{u}_D^k \\ \mathbf{u}_D^{k+1} \\ \vdots \\ \mathbf{u}_D^{k+n_p-1} \end{bmatrix}}_{\overrightarrow{\mathbf{u}}_k} = \underbrace{\begin{bmatrix} \mathbf{u}_D^{k-1} \\ \mathbf{u}_D^{k-1} \\ \vdots \\ \mathbf{u}_D^{k-1} \end{bmatrix}}_{\overrightarrow{\mathbf{u}}_{k-1}} + \underbrace{\begin{bmatrix} I & 0 & \dots \\ I & I & \dots \\ \vdots & \vdots & \ddots \\ I & I & \dots \end{bmatrix}}_{\Delta \in \mathbb{R}^{n_p n_u \times n_p n_u}} \underbrace{\begin{bmatrix} \delta \mathbf{u}_1 \\ \delta \mathbf{u}_2 \\ \vdots \\ \delta \mathbf{u}_{n_p} \end{bmatrix}}_{\overrightarrow{\delta \mathbf{u}}_k} \tag{3.28}$$

where Δ is a lower diagonal matrix where each element is identity $I \in \mathbb{R}^{n_u \times n_u}$ or a zero matrix of the same size. It is then possible to rewrite equation (3.22) using equation (3.28) as

$$\overrightarrow{\mathbf{e}}_k = P\mathbf{x}_D^k + H(\overrightarrow{\mathbf{u}}_{k-1} + \Delta \overrightarrow{\delta \mathbf{u}}_k) + K \tag{3.29}$$

Now $\overrightarrow{\mathbf{u}}_{k-1}$ can be included into the constant vector K since it is constant during the whole horizon. This results in the new cost function

$$J_\Delta = \frac{1}{2}\overrightarrow{\delta \mathbf{u}}_k^T G_\Delta \overrightarrow{\delta \mathbf{u}}_k + W_\Delta^T \overrightarrow{\delta \mathbf{u}}_k \tag{3.30}$$

where $G_\Delta = \Delta^T G \Delta$ and $W_\Delta^T = \overrightarrow{\mathbf{u}}_{k-1} G \Delta + W^T \Delta$.

The last piece of the general QP formulation is to be able to formulate linear constraints on the system states and inputs in terms of the input increments, $\overrightarrow{\delta \mathbf{u}}_k$. Firstly, the state evolution within the horizon can be

determined as

$$\underbrace{\begin{bmatrix} \mathbf{x}_D^k \\ \mathbf{x}_D^{k+1} \\ \mathbf{x}_D^{k+2} \\ \vdots \\ \mathbf{x}_D^{k+n_p} \end{bmatrix}}_{\overrightarrow{\mathbf{x}}_k \in \mathbb{R}^{(n_p+1)n_x}} = \underbrace{\begin{bmatrix} I \\ A_d \\ A_d^2 \\ \vdots \\ A_d^{n_p} \end{bmatrix}}_{P_x \in \mathbb{R}^{(n_p+1)n_x \times n_x}} \mathbf{x}_D^k + \underbrace{\begin{bmatrix} 0 & 0 & 0 & \dots \\ B_d & 0 & 0 & \dots \\ A_d & B_d & 0 & \dots \\ \vdots & \vdots & \vdots & \ddots \\ A_d^{n_p-1}B_d & A_d^{n_p-2}B_d & A_d^{n_p-3}B_d & \dots \end{bmatrix}}_{H_x \in \mathbb{R}^{(n_p+1)n_x \times n_p n_u}} \begin{bmatrix} \mathbf{u}_D^k \\ \mathbf{u}_D^{k+1} \\ \mathbf{u}_D^{k+2} \\ \vdots \\ \mathbf{u}_D^{k+n_p-1} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ I \\ I + A_d \\ \vdots \\ I + \sum_{i=1}^{n_p-1} A_d^i \end{bmatrix}}_{E_x \in \mathbb{R}^{(n_p+1)n_x \times n_x}} \mathbb{K} \tag{3.31}$$

where $\overrightarrow{\mathbf{x}}_k$ contains the state predictions for the horizon while P_x, H_x and E_x are state prediction matrices similar to the error output prediction matrices in equation (3.22). More concisely, equation (3.31) can be written as

$$\overrightarrow{\mathbf{x}}_k = P_x \mathbf{x}_D^k + H_x \overrightarrow{\mathbf{u}}_k + E_x \mathbb{K} \tag{3.32}$$

With the given formulation, constraints on the states can be given in the form

$$\mathbf{x}_l \leq V \overrightarrow{\mathbf{x}}_k \leq \mathbf{x}_u \tag{3.33}$$

where $\mathbf{x}_l \in \mathbb{R}^{(n_p+1)n_c}$ and $\mathbf{x}_u \in \mathbb{R}^{(n_p+1)n_c}$ are the lower and upper bound state constraints respectively and $V \in \mathbb{R}^{(n_p+1)n_c \times (n_p+1)n_x}$ is a matrix to select a given number of states, n_c, with active constraints. If all states have individual active constraints, V would be identity of size $(n_p+1)n_x \times (n_p+1)n_x$. Equation (3.33) can be written in terms of the input $\overrightarrow{\mathbf{u}}_k$ as

$$\mathbf{x}_l - VP_x\mathbf{x}_D^k - VE_x\mathbb{K} \leq VH_x\overrightarrow{\mathbf{u}}_k \leq \mathbf{x}_u - VP_x\mathbf{x}_D^k - VE_x\mathbb{K} \tag{3.34}$$

Equation (3.34) can in turn instead be expressed in terms of input increments $\delta\overrightarrow{\mathbf{u}}_k$ instead of the full input vector as

$$\mathbf{U}_l \leq U\delta\overrightarrow{\mathbf{u}}_k \leq \mathbf{U}_u \tag{3.35}$$

where $U = VH_x\Delta$, $\mathbf{U}_l = x_l - \mathbf{U}^k$, $\mathbf{U}_u = x_u - \mathbf{U}^k$ and $\mathbf{U}^k = VP_x\mathbf{x}_D^k + VE_x\mathbb{E} + VH_x\overrightarrow{\mathbf{u}}_{k-1}$.

Lastly, constraints on the system input can be written as

$$\mathbf{u}_l \leq \overrightarrow{\mathbf{u}}_k \leq \mathbf{u}_u \tag{3.36}$$

where $\mathbf{u}_l$ and $\mathbf{u}_u$ are the upper and lower input constraints respectively. These input constraints can be written in terms of the input increments as

$$u_l \leq \delta\overrightarrow{\mathbf{u}}_k \leq u_u \tag{3.37}$$

where $u_l = \Delta^{-1}(\mathbf{u}_l - \overrightarrow{\mathbf{u}}_k)$ and $u_u = \Delta^{-1}(\mathbf{u}_u - \overrightarrow{\mathbf{u}}_k)$. This results in the final QP formulation

$$\begin{aligned} \min_{\overrightarrow{\delta \mathbf{u}}_k} J &= \frac{1}{2}\overrightarrow{\delta \mathbf{u}}_k^T G_\Delta \overrightarrow{\delta \mathbf{u}}_k + W_\Delta \overrightarrow{\delta \mathbf{u}}_k \\ \text{subject to } \mathbf{U}_l &\leq U\overrightarrow{\delta \mathbf{u}}_k \leq \mathbf{U}_u \\ u_l &\leq \overrightarrow{\delta \mathbf{u}}_k^T \leq u_u \end{aligned} \tag{3.38}$$

where the full problem is expressed using only the input increments as the optimization variable. The QP problem in (3.38) can then be solved at every update while only keeping the first of the optimal inputs for the horizon.

4 Method

In this chapter the specifics of the simulation setup and simulation model together with the controller designs and references are presented.

4.1 Simulation setup

In this section a detailed explanation of the simulation is presented. An overview of the simulation, which includes different function blocks and information flow, is shown in Figure 4.1. The simulation runs in discrete time steps, in each time step the function blocks are executed. At the end of each update step, an optimal control action is calculated and fed to the plant model. The plant model calculates a new state based on this control action, and the calculated new state is used for the next update step.

The first step in simulation is to obtain the local path information for the vehicle to follow. This is done by the driver model. The obtained local path is a list of Cartesian coordinate (X, Y) in the vehicle's frame. From this, a new list of reference coordinates is calculated for the lateral control block. An acceleration request is also calculated for the longitudinal control, which would calculate a torque request for the vehicle's motors, based on a model of the vehicle's powertrain.

For lateral control, a control strategy based on model predictive control is used, which calculates an optimal steering angle that would result in the vehicle tracking the given local path as closely as possible. Two different models were investigated: a kinematic model, and dynamic single-track model. Both models assume a constant longitudinal speed, to decouple longitudinal and lateral motions:

- The kinematic model was used which calculates the car's position only based on the heading and longitudinal velocity. This model was chosen with the assumption that the longitudinal velocity is not high enough so that the car would have any significant lateral velocity. The chosen kinematic model, though simple, is still nonlinear, and thus needs to be linearized around the current states at every update step. This is because the MPC lateral control needs a linear model in order to formulate a quadratic programming problem, which is then solved to obtain the optimal steering angle.
- The dynamic model used is a single track model. It is an extension of the kinematic model above. This model takes into account the yaw inertia of the car to predict its heading, and is more accurate during steering at higher speed.

For longitudinal control a simple velocity reference is generated based on the path curvature which is used to calculate the required acceleration to reach the specified speed at a given point. The acceleration request can then be used to calculate the required motor torque of the vehicle.

The obtained linear model, along with currently measured (or estimated) states and the local reference path, are passed to the MPC lateral control block. This block formulates the lateral control task into a quadratic programming problem, with the optimization variable being the steering angle. The cost function would be the predicted position of the car, and the local path coordinates that the car would follow. Additional constraints are added to account for the physical limitations of the real system (path boundary, steering angle or steering rate limits). After a feasible optimal control sequence is calculated (steering angle in this case), the first control action in this sequence is applied to the plant model block. The plant model would calculate the vehicle states in the next update step, and in the next update step the whole procedure of simulation is repeated.

The plant model is a two-track dynamic model to capture some of the more important high speed dynamics of the vehicle.

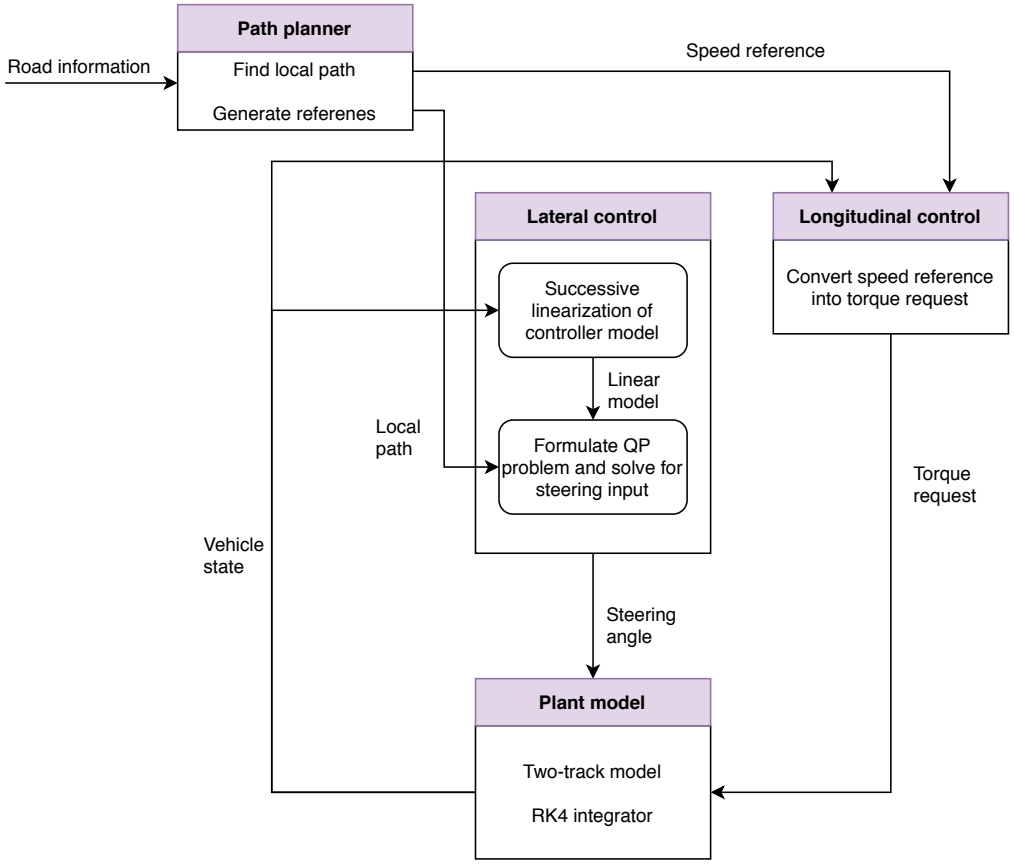

Figure 4.1: Flow chart of the simulation process.

4.2 Simulation of vehicle motion

The simulation model described in this section is a two-track model with eight degrees of freedom. The model is designed to try to accurately capture the vehicle dynamics of the CFS17 car. It is assumed that the car is driving on a flat surface without any inclination. The basis for the simulation model is therefore a dynamic model with load transfer around the longitudinal axis taken into account.

4.2.1 Equations of motion

Two coordinate frames are used in the simulation process of the vehicle. The coordinates X and Y represent the global position of the vehicle together with its heading θ in the inertial frame of reference. Similarly, x and y denotes the local position of the vehicle in the vehicle fixed coordinate frame. Figure 4.2 illustrates the free body diagram of the vehicle with the two frames and forces acting upon the vehicle. The longitudinal and lateral velocities in the vehicle frame is given by $\dot{x}$ and $\dot{y}$. The angular rotation around the center of gravity is denoted r.

In the figure, F is the force acting on the vehicle tires, α is the slip angles, δ is the steering angle, l is the distance to the center of gravity from the wheel centers and w is the track width. The subscript $(.)_{F.}$ denotes the front of the vehicle, $(.)_{R.}$ the rear, $(.)_{.l}$ the left side and $(.)_{.r}$ the right side. The rate of change of the vehicle position in the inertial frame is given by

$$\dot{X} = v_x \cos(\theta) - v_y \sin(\theta) \tag{4.1}$$

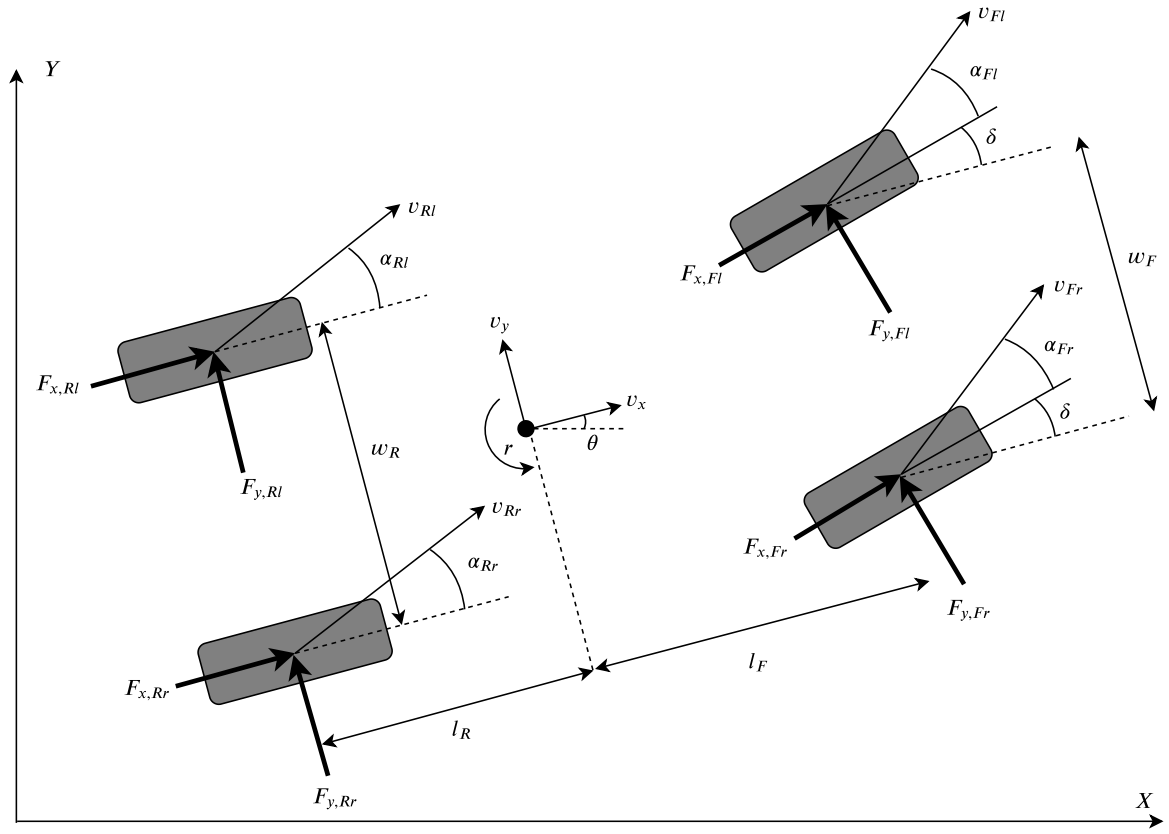

Figure 4.2: Free body diagram of the two track model.

$$\dot{Y} = v_x \sin(\theta) + v_y \cos(\theta) \tag{4.2}$$

The forces acting on vehicle in the longitudinal and lateral direction can be summed up as

$$F_x = F_{x,Fl}\cos(\delta) + F_{x,Fr}\cos(\delta) - F_{y,Fl}\sin(\delta) - F_{y,Fl}\sin(\delta) + F_{x,Rl} + F_{x,Rr} \tag{4.3}$$

$$F_y = F_{x,Fl}\sin(\delta) + F_{x,Fr}\sin(\delta) + F_{y,l}\cos(\delta) + F_{Fr}\cos(\delta) + F_{y,Rl} + F_{y,Rr} \tag{4.4}$$

External forces acting on the vehicle are air resistance and rolling resistance. Assuming negligible wind velocity, air resistance is given by

$$F_a = \frac{1}{2}\rho C_d A v_x^2 \tag{4.5}$$

where ρ is the density of air, C_d is the drag coefficient, and A is the frontal area of the vehicle. Roll resistance is given by

$$F_r = f_r m g \tag{4.6}$$

under the assumption of flat road. Here f_r is the rolling resistance coefficient, m is the total vehicle mass and g is the gravitational acceleration.

Through force and momentum equilibrium the equations of motion in the vehicle frame can be obtained as

$$m\dot{v}_x = F_x - F_a - F_r \tag{4.7}$$

$$m\dot{v}_y = F_y \tag{4.8}$$

$$\begin{aligned} I_z\dot{r} = (F_{x,Fl}\cos(\delta) - F_{x,Fr}\cos(\delta) - F_{y,Fl}\sin(\delta) - F_{y,Fr}\sin(\delta))\frac{w_F}{2} + (F_{x,Rl} - F_{x,Fr})\frac{w_R}{2} + \\ (F_{x,Fr}\sin(\delta) + F_{x,Fr}\sin(\delta) + F_{y,Fl}\cos(\delta) + F_{y,Fr}\cos(\delta))l_F - (F_{y,Rl} + F_{y,Rr})l_R \end{aligned} \tag{4.9}$$

4.2.2 Lateral tire forces

For the lateral forces, measurements for lateral slip angles had been collected by the team that designed the vehicle. As with the longitudinal road force the lateral tire forces are a function of the current load and road friction but also the slip angles, α_i. The slip angles can be calculated as

$$\alpha_{Fl} = \delta - \arctan\left(\frac{v_y + l_F r}{v_x + w_F r}\right) \tag{4.10a}$$

$$\alpha_{Fr} = \delta - \arctan\left(\frac{v_y + l_F r}{v_x - w_F r}\right) \tag{4.10b}$$

$$\alpha_{Rl} = -\arctan\left(\frac{v_y + l_R r}{v_x + w_R r}\right) \tag{4.10c}$$

$$\alpha_{Rr} = -\arctan\left(\frac{v_y + l_R r}{v_x - w_R r}\right) \tag{4.10d}$$

The lateral road force is then estimated by linear 2D-interpolation from the wheel load and slip angles. Figure 4.3 illustrates the captured data for the lateral force as a function of the slip angle at varying loads.

4.2.3 Longitudinal tire forces

The non-linear interaction between the tires and ground can be modelled as a function of slip and vertical load. For the tire-road friction force there was no empirical data for the research vehicle available and therefore a semi-empirical model similar to Pacejka's magic tire formula was used. The longitudinal force was calculated as

$$F_{x,i} = \mu_i F_{z,i}\sin(C\arctan(Bs_{x,i} - E(Bs_{x,i} - \arctan(Bs_{x,i})))) \tag{4.11}$$

for $i = \{FL, Fr, Rl, Rr\}$. Here μ_i is the friction coefficient, $F_{z,i}$ is the tire load, B, C, and E are fitting parameters and $s_{x,i}$ is the longitudinal slip ratio of each tire. The slip is in turn given by

$$s_{x_i} = \frac{R_w \cdot \omega_i - v_x}{v_x} \tag{4.12}$$

where R_w is the radius of the wheel and ω_i is the rotational speed of each wheel.

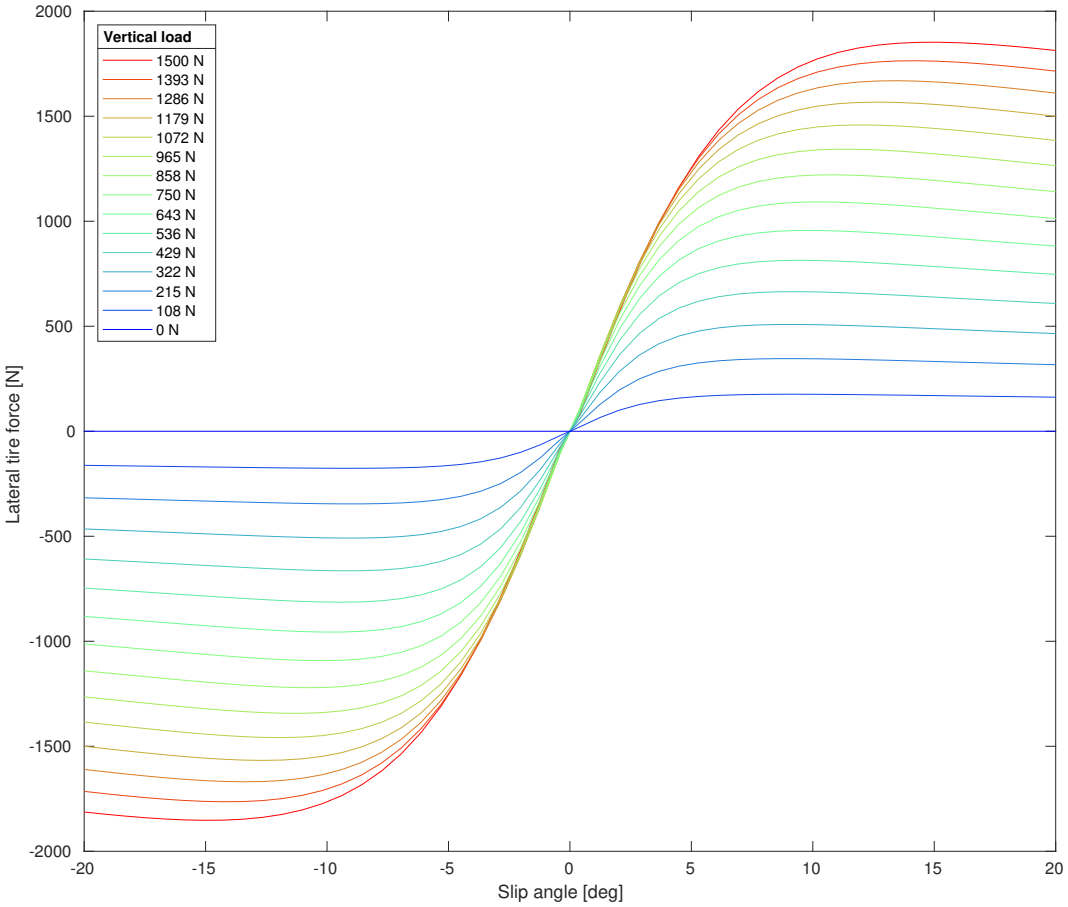

Figure 4.3: Lateral road forces at different slip angles and vertical loads.

The rotational velocity of each wheel depends on the wheel torque $T_{d,i}$, and road force by

$$\dot{\omega}_i = \frac{T_{d,i} - F_{x,i} R_w}{I_w} \tag{4.13}$$

where $\dot{\omega}_i$ is the rotational acceleration of the wheel and I_w is the rotational inertia of the wheel. The forces acting on the wheels are illustrated in Figure 4.4. The wheel torque $T_{d,i}$ is positive when driving the wheel and negative when decelerating it.

Since the rate of change for the driving torque at the rear wheels is not instantaneous this action need to be modeled for an accurate representation of the vehicle driving force. The vehicle is driven by two separate motors at the rear which are electrically powered. By investigation of the team that manufactured the vehicle the maximum rate of change of the driving torque was estimated. The driving torque in equation (4.13) is therefore simply modeled as

$$T_{d,i_{k+1}} = T_{d,i_k} + \Delta T_d \tag{4.14}$$

in discrete time where k is the time step and ΔT_d is the saturated torque request to comply with the maximum and minimum rate of change of the motor torque.

The motors are also restricted in the maximum output by being current limited and by the motor speed. The result of this is that the maximum torque is constant at lower speeds but that the maximum limit drops at high motor speeds.

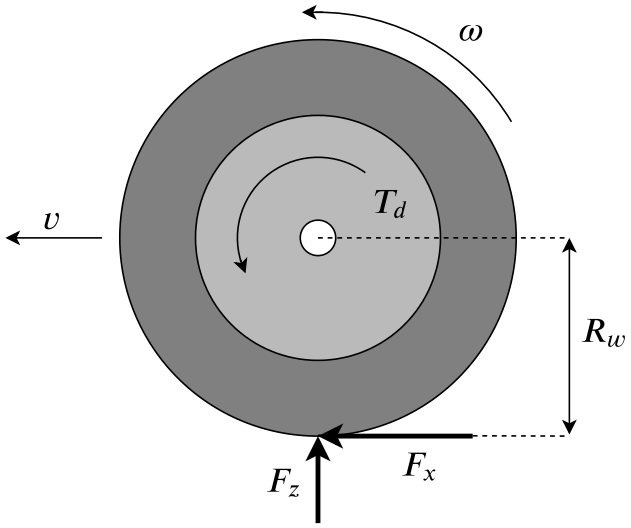

Figure 4.4: Wheel torque and tire-road forces at contact patch.

4.2.4 Load transfer

When the car is accelerating fast in the longitudinal or lateral direction the load of the vehicle mass will shift between the four wheels. This load transfer has a direct effect on the lateral and longitudinal wheel forces and is therefore modelled. The total load transfer on each wheel can be divided into the different affecting sources as

$$F_{z,i} = F_{0,i} + F_{L,i} + F_{C,i} + F_{A,i} \tag{4.15}$$

where i indicates the wheel, $F_{0,i}$ is the static load transfer, $F_{L,i}$, $F_{C,i}$ and $F_{A,i}$ are load transfer from longitudinal, lateral and aerodynamic effects respectively.

The static load on each wheel under the assumption of symmetry along the longitudinal direction of the vehicle can be calculated by

$$F_{0,Fl,Fr} = \frac{m_s g l_R}{2L} \tag{4.16a}$$

$$F_{0,Rl,Rr} = \frac{m_s g l_F}{2L} \tag{4.16b}$$

where m_s is the sprung mass of the vehicle and $L = l_F + l_R$ is its wheel base. When the vehicle is accelerating the load on the front and rear wheels will shift. Under the same assumption of symmetry as with the static load and assuming negligible pitch dynamics, the load transfer from a longitudinal acceleration is given by

$$F_{L,\{Fl,Fr\}} = -\frac{\dot{v}_x h_g}{L} \tag{4.17a}$$

$$F_{L,\{Rl,Rr\}} = \frac{\dot{v}_x h_g}{L} \tag{4.17b}$$

where h_g is the height from the ground to the vehicle center of gravity. Positive longitudinal acceleration results in an increase of the load on the rear wheels and a reduction on the front wheels and vice versa for negative acceleration.

For the lateral load the roll dynamics are taken into account in the model under the assumption of steady state and that the roll angles are small. In steady state the roll angle of the vehicle is given by

$$c_\phi \phi = m_s \dot{v}_y h_\phi \cos(\phi) + m_s g h_\phi \sin(\phi) \tag{4.18}$$

where ϕ is the roll angle, c_ϕ is the vehicle roll stiffness, h_ϕ is the height difference between the roll axis and center of gravity. Since the roll angle is assumed to be small equation (4.18) can be solved as

$$\phi = \frac{m_s \dot{v}_y h_\phi}{c_\phi - m_s h_\phi g}. \tag{4.19}$$

By moment equilibrium around the roll axis' the lateral load transfer is obtained as

$$F_{C,\{Fl,Fr\}} = \frac{1}{2w_F}(c_{\phi,F}\phi + \frac{l_R}{L} m_s \dot{v}_y) \tag{4.20a}$$

$$F_{C,\{Rl,Rr\}} = \frac{1}{2w_R}(c_{\phi,R}\phi + \frac{l_F}{L} m_s \dot{v}_y) \tag{4.20b}$$

where $c_{\phi,F}$ and $c_{\phi,R}$ are the effective cornering stiffness at the front or rear roll axles.

Since the vehicle is designed for high speed cornering the aerodynamic packages have a noticeable effect on the longitudinal load transfer. Assuming all aerodynamic lift and drag forces can be summed up at a single point with height h_p and longitudinal distance l_p the load on each wheel can be calculated as

$$F_{A,\{Fl,Fr\}} = \frac{1}{4L}\rho A(C_l v_x^2(l_R - l_p) - C_d h_p) \tag{4.21a}$$

$$F_{A,\{Rl,Rr\}} = \frac{1}{4L}\rho A(C_l v_x^2(l_R + l_p) + C_d h_p). \tag{4.21b}$$

4.3 Reference path generation

For the lateral control to work, a local path is needed. This path would be the output of the path planner module given as a list of coordinates relative to the car's local frame. This path is updated at the same frequency as the lateral controller.

4.3.1 Generating reference path from local path

To use the local path as reference trajectory for the lateral controller, the path needs to be redefined. Assuming if there is an optimal steering angle sequence such that the car could track the local path perfectly, then one could divide the local path into a number of segments; each segment is the distance covered by the car in one sampling time. Then the first n_p segments, where n_p is the prediction horizon for the MPC lateral controller, would be chosen as the reference local path. Figure 4.5 illustrates the path with its segments.

Assume the longitudinal speed of the car, v_x, is constant within the prediction horizon. n_p is the prediction horizon, $n_{p,max}$ is the maximum prediction horizon, L_p is the total local path length, T_s is the sampling time. Then n_p would be

$$N = \min\left(ceil\left(\frac{L_p}{v_x T_s}\right), n_{p,max}\right) \tag{4.22}$$

where $ceil()$ would return the nearest rounded integer greater than or equal to the argument.

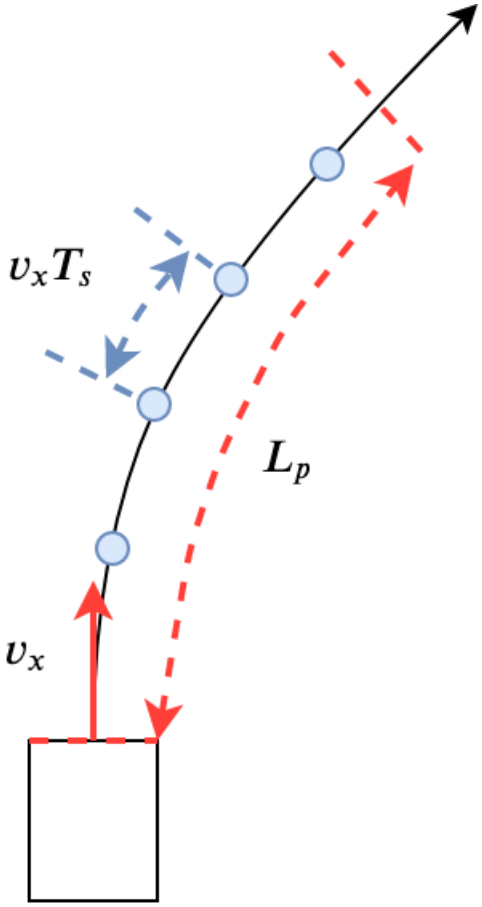

Figure 4.5: Generation of prediction horizon with n_p segments.

4.4 Longitudinal control

This section provides an insight to how the longitudinal speed of the vehicle is set during simulation and is what was implemented on the formula race car and how the decoupled lateral control works together with the longitudinal control of the vehicle.

The lateral acceleration of the vehicle can be estimated based on the curvature of the road as

$$\dot{v}_y = \frac{v_x^2}{R} \tag{4.23}$$

where v_x is the speed of the vehicle and R is represents the path radius at some given part of the track. If the curvature of the track is known a reference velocity can be calculated based on the vehicles maximum lateral acceleration as

$$v_x = \sqrt{a_{y,max} R} \tag{4.24}$$

where v_x is the velocity reference at a given point and $a_{y,max}$ is the maximum lateral acceleration. By calculating a velocity reference for all points in the given local path it is possible to trace back from the last point to the first to ensure that the maximum longitudinal acceleration is not exceeded. If it is, the velocities are recalculated with the maximum longitudinal acceleration to ensure that all reference velocities can be reached

by the vehicle. Based on these velocities the required acceleration can be calculated by differentiation as

$$a_{x,k} = \frac{v_{x,k+1} - v_{x,k}}{t_k} \tag{4.25}$$

where $a_{x,k}$ is the longitudinal acceleration at point k, v_k is the reference velocity at point k and t_k is the time it takes to move from point k to $k+1$. The time between points, t_k is estimated as

$$t_k = \frac{2s_k}{v_{x,k+1} + v_{x,k}} \tag{4.26}$$

where s_k is the Euclidean distance between point k and $k+1$.

This acceleration is then used to calculate the force to accelerate the vehicle

$$F_{x,k} = m(\dot{v}_{x,k} + gf_r) + \frac{1}{2}C_d A \rho v_{x,k}^2 \tag{4.27}$$

where $m\dot{v}_{x,k}$ is the inertial force to accelerate the vehicle, mgf_r is the approximated roll resistance of the wheels and $\frac{1}{2}C_d A \rho v_{x,k}^2$ is the air resistance with C_d being the drag coefficient, A being the frontal area of the vehicle and ρ being the air density. The required motor torque can then be calculated as

$$T_{m,k} = \frac{F_{x,k} R_w}{K} \tag{4.28}$$

where T_m is the motor torque, R_w is the wheel radius and K is the gear ratio.

4.5 Lateral control

Two models for lateral control were investigated for the control model: a kinematic model, and a dynamic model. The kinematic model is simple and therefore not computationally demanding for the controller. However, it does not model the vehicle state change at higher velocities as well as compared to the dynamic model. There is therefore a trade off between the accuracy of the vehicle model for the controller and the computational effort. An addition is made to the dynamic model to include the vehicle motor differentials to enable torque vectoring.

4.5.1 Kinematic model

A simple kinematic model is used as the lateral controller model, with three states: the car's coordinate in Cartesian coordinate system and the heading $\mathbf{x} = \begin{bmatrix} x, y, \theta \end{bmatrix}^T$; and one control input: the steering angle $\mathbf{u} = \begin{bmatrix} \delta \end{bmatrix}$. The assumption for choosing this model is that the car does not run at sufficiently high speed while taking curves, so that the lateral forces and side slips are small. The state evolution equations are

$$\dot{x} = v_x \cos\theta \tag{4.29}$$

$$\dot{y} = v_x \cos\theta \tag{4.30}$$

$$\dot{\theta} = v_x \frac{\tan(\delta)}{L} \tag{4.31}$$

where L and v_x is the distance between front and rear wheel axles (wheel base), and the longitudinal velocity. An illustration of the kinematic model is shown in Figure 4.6

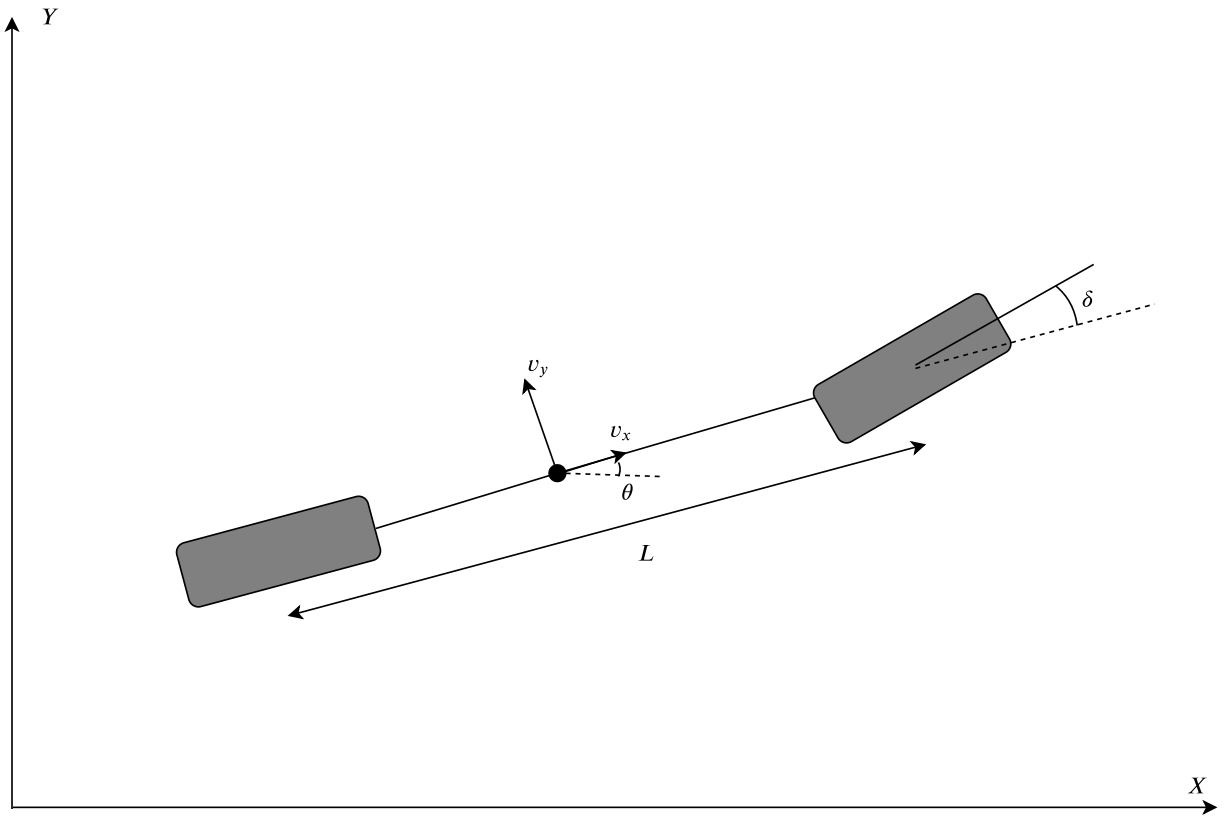

Figure 4.6: Free body diagram of the kinematic single track model.

4.5.2 Dynamic model

The dynamic single-track model is an extension of the kinematic model mentioned above. It also describes the car's velocities in a Cartesian coordinate system, however the yaw rate and lateral velocity are now incorporated into model as well. Firstly, the state variables and control input were chosen as:

$$\mathbf{x} = [x\ y\ v_y\ \theta\ \dot{\theta}]^T$$
$$\mathbf{u} = \delta$$

Figure 4.7 illustrates the dynamic single track model of the vehicle. v_x and v_y represent the longitudinal and lateral velocities respectively.

The lateral acceleration of the vehicle can be found to be

$$m\dot{v}_y = F_{fy} + F_{ry} - mrv_x \tag{4.32}$$

where v_y is the lateral speed, F_{fy} and F_{ry} are the lateral forces acting on the front and rear wheels, and r is

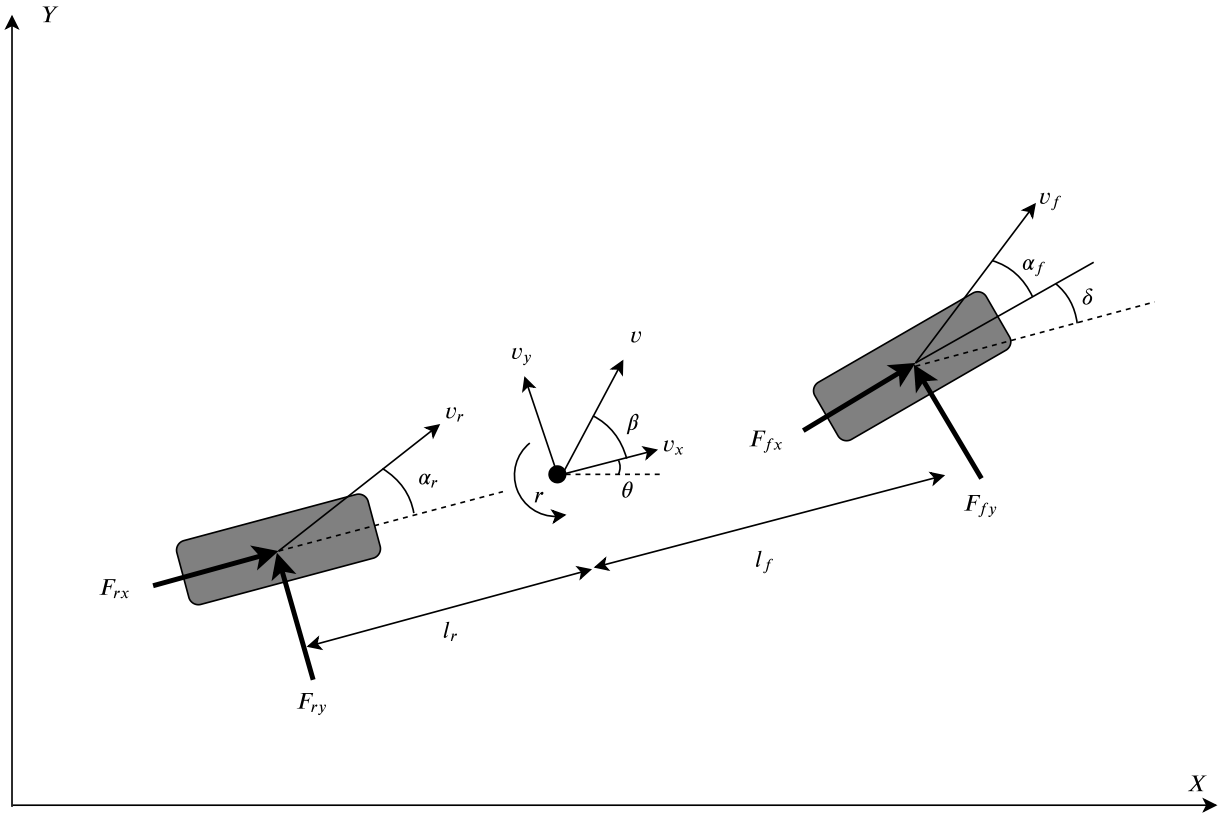

Figure 4.7: Free body diagram of the single track model.

the rotational speed around the vehicle CoG. The rotational acceleration around the CoG is given by

$$I_z\dot{r} = I_z\ddot{\theta} = l_f F_{fy} - l_r F_{ry} \tag{4.33}$$

where l_f and l_r is the distance to the front and rear axle respectively.

The lateral tire forces F_{fy} and F_{ry} are approximated as a linear function of tire slip angle:

$$F_{ry} = C_a a_r = C_a \left(\delta - \beta - \frac{l_f\dot{\theta}}{v_y}\right) \tag{4.34}$$

and

$$F_{fy} = C_a a_f = \left(-\beta + \frac{l_r\dot{\theta}}{v_y}\right) \tag{4.35}$$

where a_r and a_f are the rear and front lateral wheel slips and C_a is the vehicle cornering stiffness, that is assumed here to be the same for both front and rear wheels. β is the car's body slip angle, which can be expressed in terms of longitudinal and lateral velocities (assuming β is small):

$$\beta = \frac{v_y}{v_x} \tag{4.36}$$

Combining the above three equation gives:

$$F_{ry} = C_a \left(\delta - \frac{v_y}{v_x} - \frac{l_f \dot{\theta}}{v_y} \right) \tag{4.37}$$

$$F_{fy} = \left(-\frac{v_y}{v_x} + \frac{l_r \dot{\theta}}{v_y} \right) \tag{4.38}$$

States of interest are $\dot{v_y}$ and $\ddot{\theta}$ which can be given by:

- Substituting 4.37 into 4.33 and 4.32.
- Expressing $\dot{v_y}$ and $\ddot{\theta}$ as functions of other state variables and control input

This would give

$$\dot{v}_y = \left(\frac{-2C_a}{v_x m} \right) v_y + \left(-v_x - \frac{C_a(l_f + l_r)}{v_x m} \right) \dot{\theta} + \left(\frac{C_a}{m} \right) \delta \tag{4.39}$$

$$\ddot{\theta} = \left(\frac{C_a(l_r - l_f)}{I_z v_x} \right) v_y + \left(-\frac{C_a(l_f^2 + l_r^2)}{v_x m} \right) \dot{\theta} + \left(\frac{l_f C_a}{I_z} \right) \delta \tag{4.40}$$

and v_y, θ and $\dot{\theta}$ can be found by integrating $\dot{v_y}$ and $\ddot{\theta}$.
The change of position of the car in the inertial frame is given by

$$\dot{X} = v_x \cos(\theta) - v_y \sin(\theta) \tag{4.41}$$

and

$$\dot{Y} = v_x \sin(\theta) + v_y \cos(\theta). \tag{4.42}$$

Equation 4.39, 4.40, 4.41, and 4.42 describe the state evolution of the dynamic model.

4.5.3 Torque vectoring

The formula car uses two separate motors to drive the rear wheels which enables the possibility to provide different amount of torque to each wheel. The difference in wheel speed results in a yaw moment around the vehicle CoG. The dynamic plant model can therefore be expanded to include torque vectoring as a control input by modifying equation (4.40) as

$$\ddot{\theta} = \left(\frac{C_a(l_r - l_f)}{I_z v_x} \right) v_y + \left(-\frac{C_a(l_f^2 + l_r^2)}{v_x m} \right) \dot{\theta} + \left(\frac{l_f C_a}{I_z} \right) \delta + \left(\frac{1}{I_z} \right) M_z \tag{4.43}$$

where M_z is the yaw moment. The yaw moment is in turn a function of the motor torque difference as

$$M_z = \frac{K w_R}{R_w} \Delta T_m \tag{4.44}$$

where K is the gear ratio, W_R is the distance between the rear wheels and ΔT_m is the motor torque difference. The torque to each motor is then given by

$$T_{m,r} = T_m + \frac{\Delta T_m}{2}$$

$$T_{m,l} = T_m - \frac{\Delta T_m}{2}$$

where T_m is the motor torque provided by the longitudinal speed controller, $T_{m,r}$ is the right motor torque and $T_{m,l}$ is the left motor torque. The system inputs are then the front wheel steering angle δ and the motor torque difference ΔT_m. The state variables and control inputs are therefore given by

$$\mathbf{x} = [x\ y\ v_y\ \theta\ \dot{\theta}]^T$$
$$\mathbf{u} = [\delta\ \Delta T_m].$$

5 Results

This chapter covers the final controller parameters and result of the controller simulations for the different test cases: steady state cornering, double lane change, and track drive. The difference in performance between the controllers using different controller plant models is shown through lateral track deviation at varying longitudinal velocities. All simulations where done in Matlab, using a Intel quad core i5-8250U processor.

5.1 Controller specifications

This section present the different controller parameters used in simulation unless otherwise specified.

1. Frequency: The controller frequency was 10 Hz when simulating and was set after the expected velocities for the test cases. An effect of the relatively low frequency is that the controller performances would be noticeably affected at high velocities when performing challenging actions such as sharp turns.
2. Horizon: The horizon was set to $n_p = 40$. Since the different test scenarios have fast changing environment it is beneficial for the controller to be able to react early. This also reflects the intended application environment of the lateral controller.
3. Constraints: Input constraints of the controllers are given by the mechanical limitations of the vehicle. The steering angle of the wheels are for example limited to 24^o both directions and are as such defined in the controller constraints. All constraints imposed on the QP problem are presented in Table 5.1.
4. Weights: The state and input weights used in the different controllers are presented in Table 5.2. By tuning these weights the desired behavior of the controllers can be achieved. Here more importance is given to the heading error, θ, to keep the vehicle on track. Additionally the lateral position, y, has to be kept but is given less importance so as to not cause the controllers to turn too quickly at high velocities.

Table 5.1: Input constraints

$\lvert\delta\rvert$	0.419 rad
$\lvert\dot{\delta}\rvert$	0.873 rad/s
$\lvert\Delta T_m\rvert$	5.0 Nm
$\lvert\Delta \dot{T}_m\rvert$	50.0 Nm/s

Table 5.2: Controller state and input weights

	Kinematic	Dynamic	Torque vectoring
Q_x	0	0	0
Q_y	1	1	1
Q_θ	6	6	6
Q_{v_y}	-	0	0
$Q_{\dot{\theta}}$	-	0	0
R_δ	30	30	30
$R_{\Delta T_m}$	-	-	1

5.2 Steady state cornering

To assess the performance of the car during turns, a steady-cornering test was done, in which the car drove in a U-turn at a constant velocity. The turning radius was set to be the same as that in a skidpad event (about 9.125 m). The car starts at position (0,0).

After several runs, it was found that the maximum velocity that the car could make the turn without hitting any cone is 40 km/h. Any velocity higher than this would result in driving off the track. Figure 5.1 shows the position trace of the car using the dynamic model (with and without torque vectoring) and kinematic model for lateral controller. In general, all three models behaved similarly. Nevertheless, there is a slight difference: the dynamic models were slightly more under steering, compared to the kinematic model.

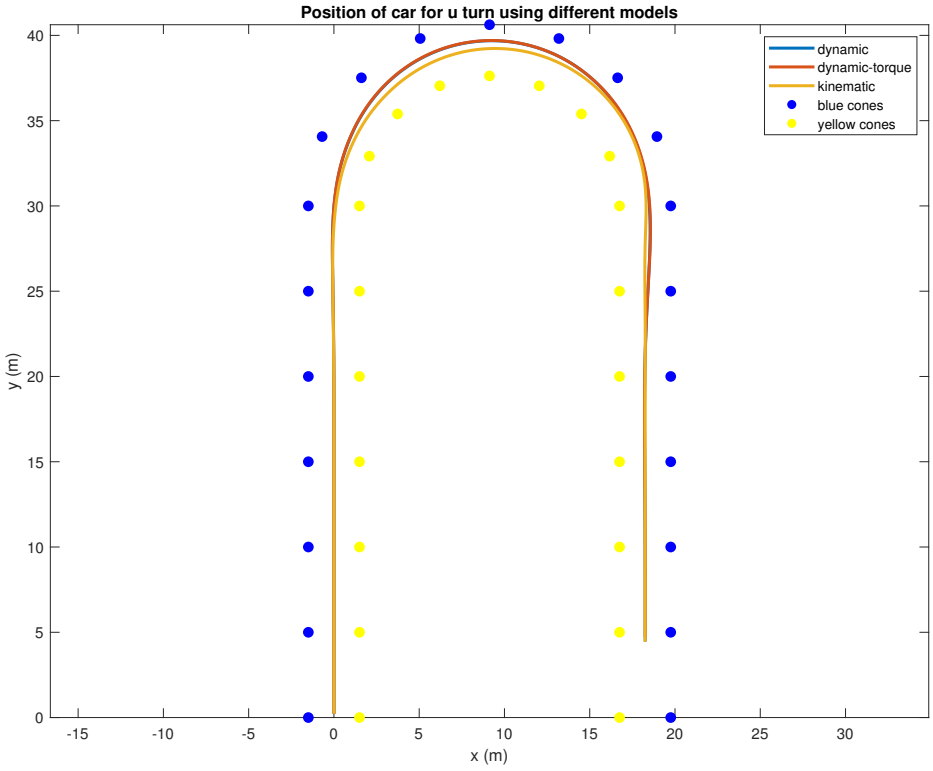

Figure 5.1: U-turn with position of the car using different lateral controller models.

5.3 Double lane change

Figure 5.2–5.5 illustrate the position traces of the car using different models of controller, for the double lane change test case. Due to the constant-speed assumption while designing the controller, the lane change test were done with four different constant speeds: 40, 50, 60 and 70 km/h.

From the four speed settings, it can be seen that at low speeds (40 and 50 km/h), all controller models could handle well, albeit the kinematic model performs slightly better. However, at higher speed (70 and 80 km/h), the dynamic models (both with and without torque vectoring) are more stable: at 70 km/h, the maximum lateral deviation is about 0.3 m for all models, but the kinematic model started to exhibit oscillatory behaviour. At 80 km/h, both dynamic models can still keep the car follow the lane, albeit with larger lateral deviation, whereas the kinematic model completely drove the car outside the lane with large, oscillatory lateral error. At higher velocities than 80 km/h, no model could keep the car in lane.
The controller frequency also has noticeable impact on the steering performance. A similar test for lane changing at 70km/h for all three controller models was also run, but now the controller was set to run at 50 Hz. It was observed that at 50 Hz, the dynamic models (both with and without torque vectoring) performed better than at 10 Hz, the difference can be seen when the car reaches the 60 m mark, where the deviation from path centerline was 1.3 m at 10 Hz, and 0.4 m at 50 Hz,a 70 percent reduction in lateral deviation. For the kinematic

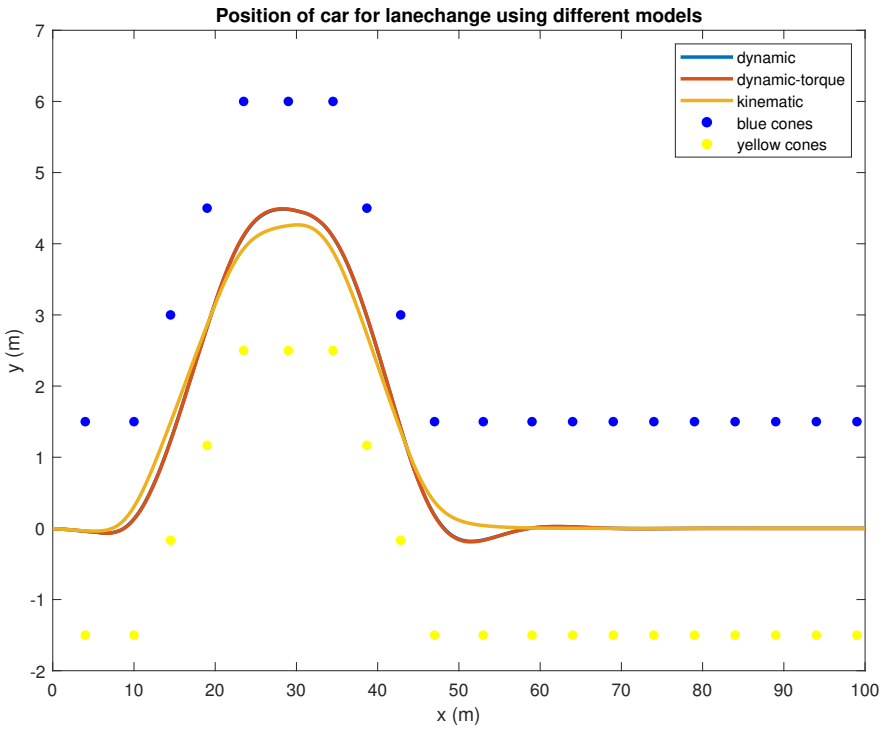

Figure 5.2: Double lane change at 40 km/h entrance velocity.

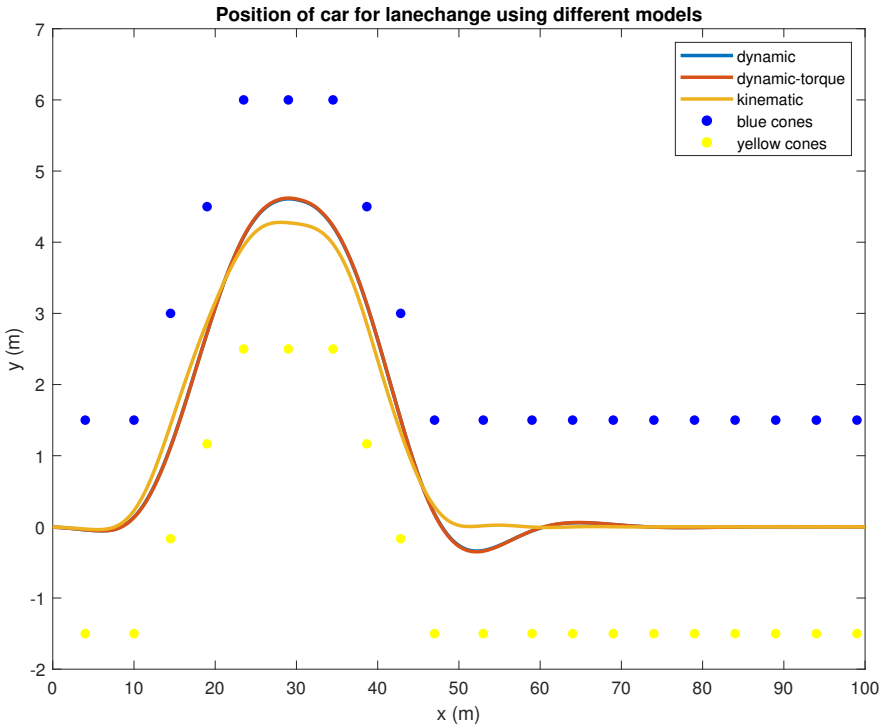

Figure 5.3: Double lane change at 50 km/h entrance velocity.

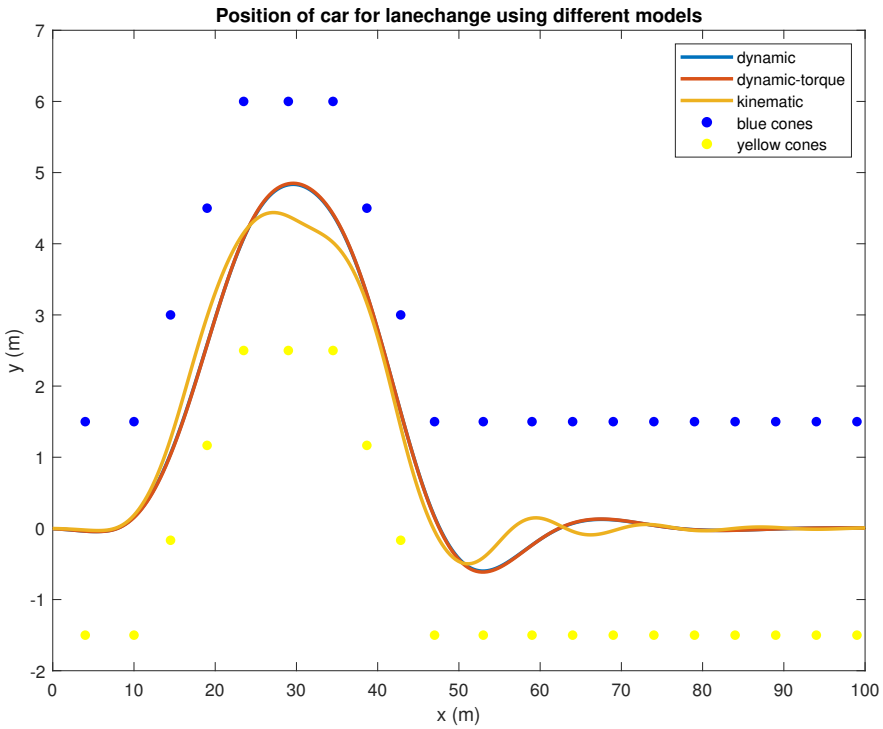

Figure 5.4: Double lane change at 60 km/h entrance velocity.

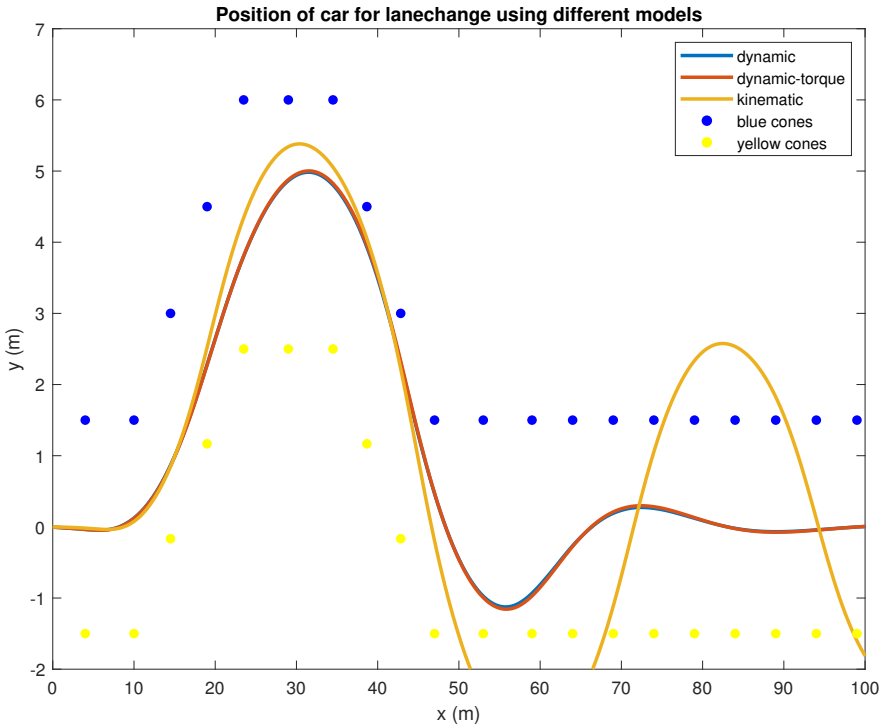

Figure 5.5: Double lane change at 70 km/h entrance velocity.

model, the performance was slightly better when the car was still inside the track, but became much worse once it went off the track.

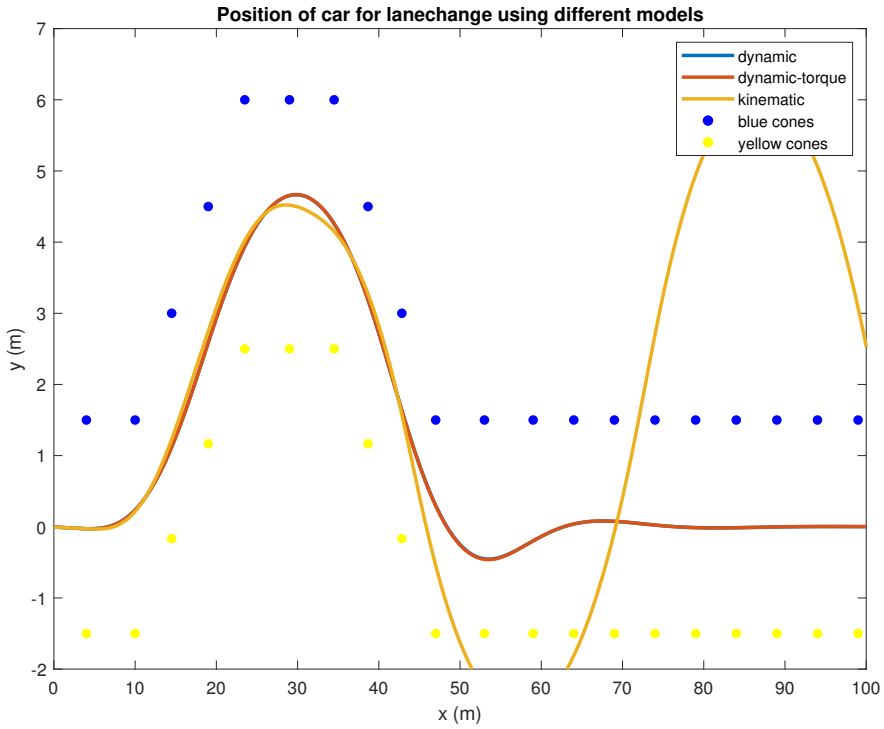

Figure 5.6: Double lane change at 70 km/h with controller frequency at 50 Hz.

5.4 Track drive

Figure 5.7 illustrates the layout of track drive, along with the track length covered by the car. The car starts at the bottom left of the map.

The controller using the dynamic model without torque vectoring performs better than the kinematic and the dynamic, torque vectoring ones. This can be seen when the car enters curves at high speed, where the lateral deviation for dynamic model is noticeably lower than the other two models. Given that the car's width is 1.25 m, and track width is 3 m, then as long as the car stays within 0.875 m from the track's middle path, then it would not hit the cones. For the dynamic model, there were only two curves: at 550 m (green part) and 950 m (the last curve) where the lateral deviation exceeded this 0.875 m.

The calculation using quadprog in Matlab takes a short time (around 0.015–0.02 s for a horizon of 40 steps), thus it is reasonable to assume that it is possible to implement this controller in a live system. It should also be noted that a QP solver implemented in a faster and compiled language such as C++ could have a much faster computation time compared to Matlab.

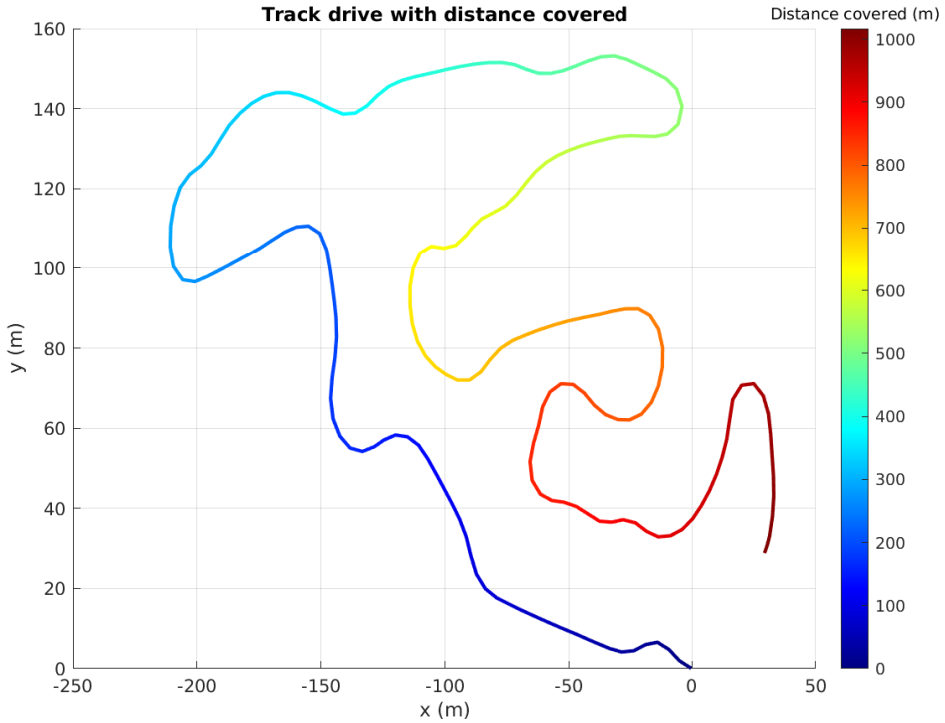

Figure 5.7: Track drive with track length.

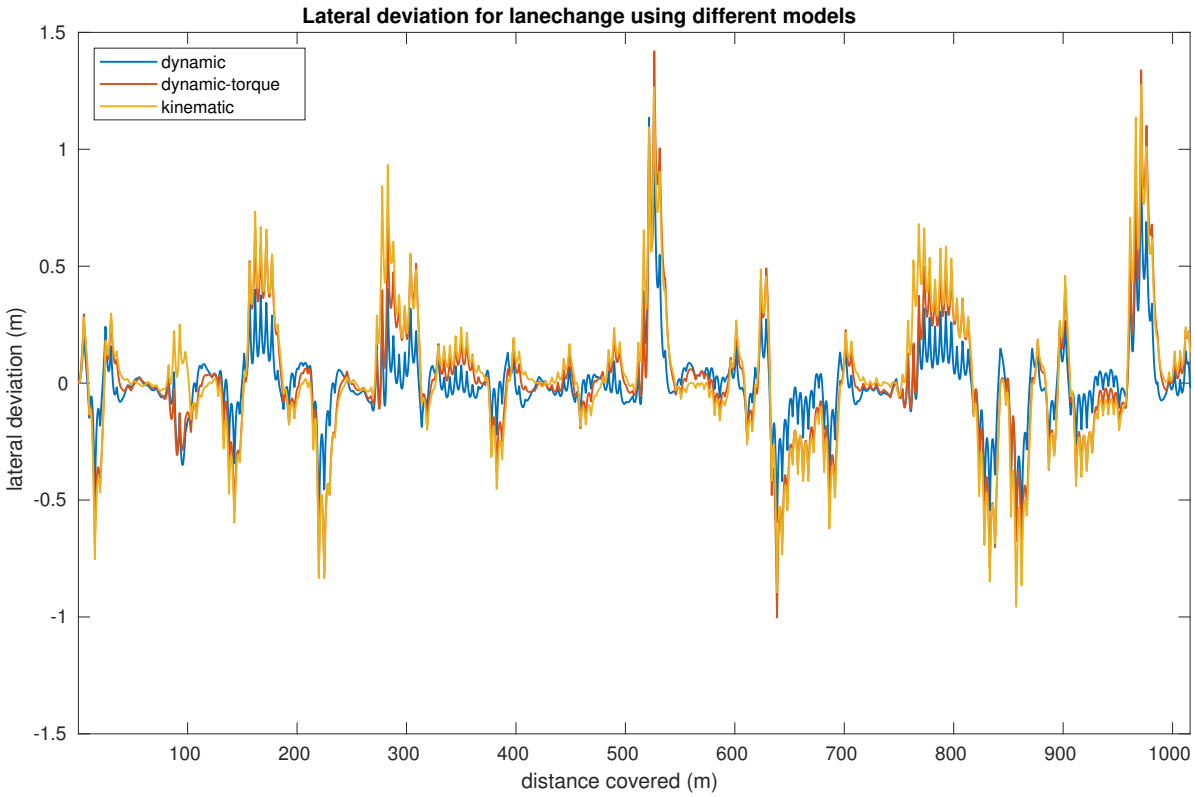

Figure 5.8: Lateral deviation for track drive with different controller models.

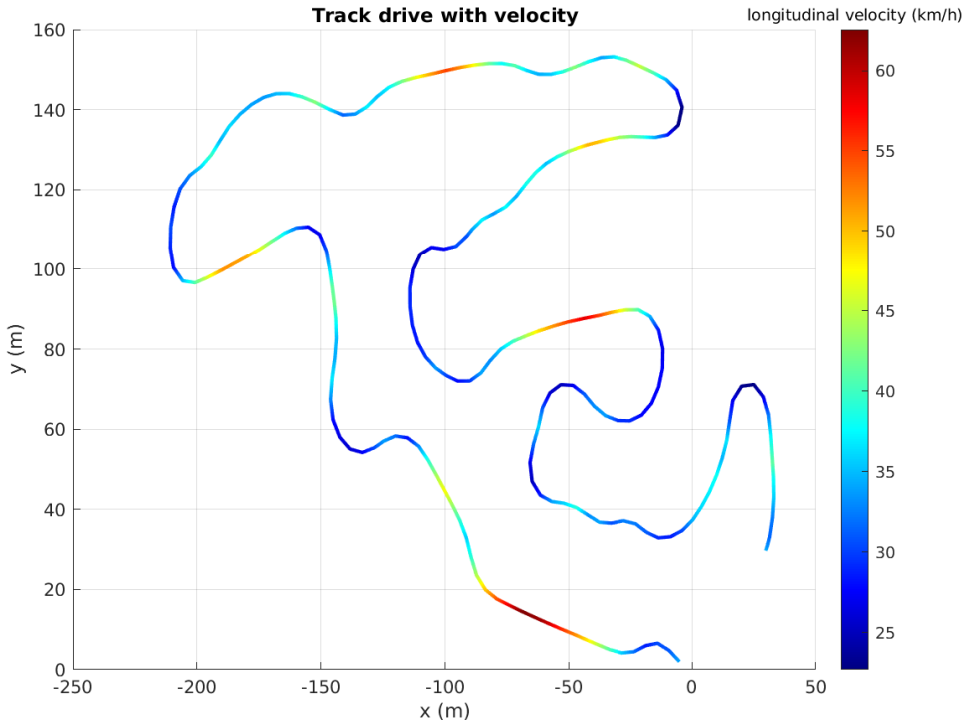

Figure 5.9: Longitudinal velocity for track drive with different controller models. All controller models resulted in nearly identical velocity profiles.

6 Discussion

In this thesis the design and implementation of linear time variant model predictive steering controllers have been presented together with the simulated result with a given vehicle model. Two different control models were used: a kinematic pointmass model, and a dynamic single track model with decoupled longitudinal and lateral dynamics. The different lateral controllers were tested in three test scenarios: a steady state cornering test, a double lane change test, and a track drive test suing the track layout of Formula Student Germany. The results of using different controller models have also been presented and a comparison between their performances can be made.

In steady state cornering test, the car drives in a u-turn with a constant turning radius at constant speed. This test will reveal a vehicle's oversteering and understeering properties. From the test results, it shows that the kinematic model controller appears to be more responsive than the dynamic model controllers in steady-state conditions. This could be due to the reactive nature of the kinematic model, where a control input is calculated directly from the local path information, without taking into account the dynamic properties of the vehicle.

In double lane change test, the car will perform two lane-change maneuver while trying to stay as close to the lane centerline as possible. These maneuvers are commonly used to test the vehicle's ability in handling high dynamic situations. The test results show that in high dynamic situations, the dynamic model controllers perform better than the kinematic controller, especially at high speeds. In addition, the controllers were also tested at two different frequencies (10 Hz and 50 Hz) at 70 km/h longitudinal speed, and as expected the lateral control performance was much better at the higher frequency: the car managed to stay 70 percent closer to the lane centerline, and the increased controller frequency did not result in any significant computational time increase.

Lastly, a track drive test was carried out for all three controllers, to assess their performance in an open environment. In this test a full racing lap simulation was done, where the car would drive through a racing track. The track is composed of straight lines, constant radius turns, hairpin turns. This combination of different maneuvers will test the overall performances of the controllers. Lateral deviation from the track centerline was recorded for all controllers, in order to judge their lateral control performance. A speed profile along the track of all controllers was also recorded, to judge the longitudinal controller performance. From 5.8, it shows that the lateral controllers are good enough to keep the car inside the track most of the time, there are only two places where the lateral deviation are higher than 0.8 meter which were considered to be outside of the safe zone. These places correspond to two sharp turns after a straight line, where the car has to slow down fast to take the turn. As the longitudinal speed has to change quickly, the lateral controllers which assumed a constant longitudinal speed in the dynamic model, would become less accurate, and thus results in a poor performance. The longitudinal speed profile shows that the top speed reached was only 63 km/h, a rather slow speed for racing car. Nevertheless, this speed is acceptable, as it is easier to keep the car stay within the 3 meter wide track.

Using Matlab's quadratic programming solver **quadprog** the mean execution times on a Intel quad core i5-8250U are around 4 ms for all controllers. This provides a good indication that the controller designs would have feasible execution times if implemented in the real world on the formula car. If the controller would be implemented on the formula car it would also be written in a compiled language such as C++ which might improve the execution time further. However, the MPC controller would also run alongside many other programs on the vehicle and as such it is not possible to make any estimation on the execution time of the proposed controller designs on the real system.

From the results it is apparent that the different controller performance are similar. In the case of the U-turn the different controllers can keep the vehicle close enough to the center line at roughly the same maximum longitudinal speed. The only noticeable difference is that the dynamic models were understeering more when compared to the kinematic model. The understeering of the dynamic model controller could be cased by inaccuracies in the lateral and rotational dynamics of the control model. In the double lane change test the controller performances were once again similar at lower speeds. At higher speeds however, the kinematic

controller loses control of the vehicle completely. This test shows a clear difference between the controllers at higher longitudinal speed indicates that the dynamic model controller might be preferable in the actual vehicle. During the track drive the dynamic based controller generally performs better in keeping the vehicle close to the track centerline. The difference during sharp turns is also large enough to possibly make the difference between hitting a cone or not during competition.

From the controller performances there is a minor difference between the controllers based on the kinematic and dynamic models while the calculation time remains roughly the same. The controller using the kinematic model would appear to be sufficient at lower longitudinal speeds but, as could be seen in the double lane change test, it performs worse at higher longitudinal and lateral speeds. The dynamic model controller also generally performs better at the track drive where the longitudinal speed is constantly changing. Since the calculation time is also the same between models, the dynamic model based controller might be favourable to use in the real vehicle.

Since the proposed lateral controllers are decoupled from the longitudinal control they perform better at constant speed. Cornering performance is worsened since the speed is assumed to be constant during linearization of the controller plant model while it is changing during simulation. This can be seen during the track drive testing in results where the vehicle slows down when approaching a turn which leads to worsened lateral performance. Better performance could be achieved if the change in speed is taken into account when planning the steering action of the vehicle. However, literature study about performance of coupled and decoupled steering control has been mentioned, which showed that the a coupled dynamic model will have higher complexity and thus is harder to tune, while the performance improvement is little. In addition, the increased complexity will also result in longer execution time of the controller. From the double lane change test result, it can be seen that much better steering performance can be achieved with only a decoupled model but at higher frequency (50 Hz instead of 10 Hz). Thus, it can be concluded that the decoupled MPC steering controller has performed well in this specified environment.

The weight matrices Q (state weights) and R (control input weights) were carefully tuned during each test scenario to maximize the performance. For all tests, the weights were chosen as (Q_x, Q_y, Q_{v_y}, Q_θ, $Q_{\dot{\theta}}$) = (0, 1, 6, 0, 0) for Q, and ($\dot{\theta}$) = (30) for R (where x, y are the car position, v_y is lateral velocity, $\dot{\theta}$ is yaw rate of the car, and $\dot{\theta}$ is the steer angle rate in radians per second, all are in the car's local frame). It could be seen that the lateral velocity has a big impact on the steering performance amongst all states; the steering rate weight does not have a big impact because it was in radians and not degree. Tuning the weights require a lot of trials, and one has to account for how each state affects another. The tuning strategy was as follow: first set all weights to zero, then increase the control input weight until the car could somehow follow the track, then increase the state weights, on by one, until the desired performance is obtained. As mentioned before, tuning requires a lot of trials and depends a lot on the actual dynamic of the system. This set of weights was the most optimized, and further tuning did not result in any performance improvement.

With the proposed controller designs the steering angle and steering rate are kept at all times. This provides a way to take the constraints of the physical system into account when planning the steering action of the vehicle. Due to the physical limitations of the steering system on the vehicle the steering rate is low and thus it is beneficial to take into account in the controller.

In the controllers, Cartesian coordinates were used to represent the local path. Since the local path is the center line of the tracks, a curvilinear coordinate system might have provided a better representation of the system since it would have been possible to represent the change in curvature in the model. In the current system representation the model does not take the turn rate of the path into account when modeling the vehicle dynamics which could possibly have a noticeable effect on the controller performances. With a curvilinear coordinate system and using the path deviation as a state it would also be possible to add constraints on the path deviation in the controller design. One could additionally explore the use of constraints on system states such as lateral acceleration and yaw rate.

7 Conclusions

Satisfactory steering behavior could be achieved in simulation for decoupled lateral control using MPC. All three controllers performed well in the different test cases at lower velocities and could be considered for testing in the real world on the formula car. While the different controllers performed similarly, using a dynamic vehicle model had slightly better performance at higher velocities. The controller utilizing a kinematic vehicle model did perform better at the cornering test due to less understeering. However, since the controllers using a dynamic model did perform better overall while not increasing the computational complexity by much, it should be considered firsthand if tested on the real car.

It can be concluded that the controllers all performed sufficiently within the simulation environment and could be considered for testing on the real vehicle. A difference in the performance of the kinematic and dynamic based controllers could be observed in simulation. It is however not possible to make any conclusions on their actual performance in the real vehicle due to the limitations of the simulation environment.

7.1 Future work

The proposed controllers all have acceptable performance at somewhat high speeds. There are however aspects to the controllers that could be investigated for further improvements. A possibility would be to investigate coupled longitudinal and lateral control to provide insight to the benefits of coupled control in the application environment. One could also change the path representation to a curvilinear coordinate system which would provide the possibility to constrain the path deviation in the MPC formulation. With coupled longitudinal and lateral control this could improve the car's ability to stay on track during sharp turns.

No verification data of the vehicle movement could be collected during this work which would have provided better insight on the performance of the controller on the actual formula car. As it stands right now nothing can be said about the performance of the designed controllers outside of the simulation environment.

Very little can be said about the controller execution time outside of the simulation environment in Matlab. A formal analysis of the execution time to verify real-time capabilities of the controller could additionally be done. To implement the controllers on the formula car the current controller code would have to be ported to code that can run in the formula car execution environment. This would most likely also serve to improve the execution time of the MPC calculations.

References

[1] F. S. Germany. *An outlook on FSG 2021 and the following seasons*. 2018. URL: https://n.fs-g.org/1064 (visited on 12/28/2018).

[2] E. F. Camacho and C. B. Alba. *Model Predictive Control*. 2nd ed. 1-29. Springer Verlag, 2004. ISBN: 9780857293985.

[3] S. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice* **11**.7 (2003), 733–764. ISSN: 0967-0661. DOI: https://doi.org/10.1016/S0967-0661(02)00186-7. URL: http://www.sciencedirect.com/science/article/pii/S0967066102001867.

[4] M. Zanon et al. "Model Predictive Control of Autonomous Vehicles". Vol. 455. Mar. 2014. DOI: 10.1007/978-3-319-05371-4_3.

[5] R. Verschueren et al. "Towards time-optimal race car driving using nonlinear MPC in real-time". *53rd IEEE Conference on Decision and Control*. 2014, pp. 2505–2510. DOI: 10.1109/CDC.2014.7039771.

[6] Y. Yoon et al. Model-predictive active steering and obstacle avoidance for autonomous ground vehicles. *Control Engineering Practice* **17** (July 2009), 741–750. DOI: 10.1016/j.conengprac.2008.12.001.

[7] P. Falcone et al. Predictive Active Steering Control for Autonomous Vehicle Systems. *Control Systems Technology, IEEE Transactions on* **15** (June 2007), 566–580. DOI: 10.1109/TCST.2007.894653.

[8] J. Kong et al. "Kinematic and dynamic vehicle models for autonomous driving control design". *2015 IEEE Intelligent Vehicles Symposium (IV)*. June 2015, pp. 1094–1099. DOI: 10.1109/IVS.2015.7225830.

[9] K. Lundahl, J. Aslund, and L. Nielsen. Investigating Vehicle Model Detail for Close to Limit Maneuvers Aiming at Optimal Control (May 2012).

[10] R. Attia, R. Orjuela, and M. Basset. "Coupled longitudinal and lateral control strategy improving lateral stability for autonomous vehicle". *2012 American Control Conference (ACC)*. June 2012, pp. 6509–6514. DOI: 10.1109/ACC.2012.6315130.

[11] R. Attia, R. Orjuela, and M. Basset. Combined longitudinal and lateral control for automated vehicle guidance. *Vehicle System Dynamics: International Journal of Vehicle Mechanics and Mobility* **52** (Jan. 2014), 261–279. DOI: 10.1080/00423114.2013.874563.

[12] C. Olsson. "Model Complexity and Coupling of Longitudinal and Lateral Control in Autonomous Vehicles Using Model Predictive Control". PhD thesis. May 2015. URL: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-175389.

[13] P. Falcone et al. "A linear time varying model predictive control approach to the integrated vehicle dynamics control problem in autonomous systems". *2007 46th IEEE Conference on Decision and Control*. Dec. 2007, pp. 2980–2985. DOI: 10.1109/CDC.2007.4434137.

[14] A. Katriniok and D. Abel. LTV-MPC approach for lateral vehicle guidance by front steering at the limits of vehicle dynamics. *In Proceedings of the IEEE Decision and Control and IEEE European Control Conference* **52** (2011), 6828–6833.

[15] A. Zhakatayev et al. Successive linearization based model predictive control of variable stiffness actuated robots (July 2017), 1774–1779. ISSN: 2159-6255. DOI: 10.1109/AIM.2017.8014275.

[16] H. Zheng. Trajectory Tracking of Autonomous Vessels Using Model Predictive Control (Aug. 2014), 8812–8818. DOI: 10.3182/20140824-6-ZA-1003.00767.

[17] A. Karlsson. Test Procedures and Evaluation Tools for Passenger Vehicle Dynamics (2014).